Design and Performance Guarantees in Cloud Computing: Challenges and Opportunities

Papadopoulos, Alessandro Vittorio

2015

Design and Performance Guarantees in Cloud Computing: Challenges and Opportunities *

[Challenge Paper]

Alessandro Vittorio Papadopoulos
Department of Automatic Control, Lund University
22363 Ole Römers väg 1
Lund, Sweden
alessandro.papadopoulos@control.lth.se

ABSTRACT

In the last years, cloud computing received an increasing attention both from academia and industry. Most of the solutions proposed in the literature strive to limit the effect of uncertain and unpredictable behaviors that may occur in cloud environments, like for example flash crowds or hardware failures. However, managing uncertainty in a cloud environment is still an open problem. In such a panorama, the service provider is not able to define suitable Service Level Objectives (SLO) that are easy to measure, and control. In this work we analyze two of the critical problems that are encountered in cloud environments, but seldom discussed or addressed in the literature: (1) how to reduce the uncertainty providing suitable control interfaces at different levels of the computing infrastructure; (2) how to assess performance evaluation in order to get probabilistic guarantees for the SLOs. We here briefly describe the two problems and envision some possible control-theoretical solutions.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Cloud computing, performance evaluation, uncertainty management

1. INTRODUCTION

The idea of using feedback control for managing complex computing systems functionalities has been very successful in the last two decades [1]–[5]. However, as suggested also in [6], while low level functionalities, such as task scheduling or memory management, can be easily modeled without much uncertainty, the higher layers are affected by different sources of uncertainty. As a result, high level functionalities are typically more difficult to control, and in some cases heuristic techniques might perform better than control-theoretical approaches [7].

Such an uncertainty can be linked to two facts. First, most of the high level functionalities are "closer to the environment" where the computing system operates. This means that many external factors are directly affecting the performance of the considered functionality. For example, in the case of a load-balancer, its performance will be affected by the incoming traffic, the enqueued requests, and by the type of request; all these factors will affect the response time of the single request from the user perspective[8]–[11], possibly yielding bad performance, thus loss of customers. More important, these factors are just external, i.e., they cannot be affected by any component, but their effect can be limited.

Second, the uncertainty may not come from environment, but from "internal phenomena" of the computing infrastructure, i.e., from the lower layers of the system. For example, it is almost impossible to track an incoming request down to the thread that is in charge of serving it; and the adopted scheduling policy may affect the response time for the single request. Therefore, an internal behavior, i.e., something that in principle we can control, will introduce an uncertainty that cannot be easily compensated. Indeed, whereas the process of abstraction from the lower layers while designing new high functionalities is a desirable feature, the interfaces provided by the lower layers to the higher ones practically limit the controllability also of high level quantities, e.g., the response time of a request.

Providing the "right interfaces" towards the other layers becomes particularly relevant whenever control theory comes into play, since the "right measurements" or the "right actuators" to control the desired quantities must be available in order to obtain good performance. For example, most of the research conducted on task scheduling was assuming to modify some parameters of the existing scheduling algorithm [1], [12], instead of redesigning as a new controller the scheduler itself as done in [13]. In [14] the authors prove that this approach gives better results, thanks to the fact that the right interfaces for the scheduler have been exposed, and the controller is not acting on some quantities that will indirectly affect the scheduling policies, like for example the nice number. Similar remarks can be found in [15]–[19].

Whatever is the source of uncertainty, either external or internal, it typically hinders the possibility of obtaining repeatable results, especially in cloud environments, which are somehow in the upper layers of the computing infrastructure. In cloud computing, the evaluation of different algorithms or policies is indeed a hard task due to non-repeatable

*This work was supported by the Swedish Research Council (VR) under contract number C0590801 (2012-5908) for the project Cloud Control and through the LCCC Linnaeus Center.

experiments. Many papers propose some novel and interesting techniques aimed at solving well known important problems in the context of cloud computing, e.g., admission control [20], [21], load balancing [11], auto-scaling [22]–[24], etc., but up to date, most of them show their effectiveness against other approaches in few specific cases. This is due to the lack of standardized benchmarking, and to the difficulty in managing stochastic or uncertain behaviors.

On the other hand, the service providers need to guarantee certain Service Level Objectives (SLOs) to their customers. Sturm et al. [25] define some features that a SLO should fulfill. Among them, three of them reflect the discussion above, i.e., a SLO must be *controllable*, *repeatable*, and *measurable*. While the first is affected by internal phenomena, the last two properties, are affected both from internal and from external factors. In particular, the measurable part should include probabilistic guarantees on the obtainable performance, which, to the best of the author's knowledge, is seldom carried out in most of the proposed techniques.

The rest of the paper is thus focusing on two main research challenges, sketching some possible ideas on how they might be addressed:

1. How can one guarantee *controllability* of a cloud system, and to which extent.
2. How can one provide SLOs that are *repeatable*, and *measurable*.

2. ADDRESSING CONTROLLABILITY

One of the large challenges for feedback computing in cloud applications is that most of the components were not designed having controllability, or "self-adaptivity", in mind [16].

Since computing systems are moving toward dealing with continuously changing and unpredictable execution environments and user interactions, their design must allow for self-adaptation [26]. Controllability will require a paradigm shift in how computing systems at large are conceived, which has to come out with appropriate theories, design practices, and training to enable engineers creating software that is controllable by design. Such software would provide formally guaranteed self-adaptation capabilities off-the-box, overcoming the limitations of trying to control software not designed to be controlled.

2.1 Possible solutions

In order to address controllability issues, and, at the same time, to reduce the uncertainty present in the higher layers of the computing infrastructure, one should design an end-to-end integrated design of the computing hierarchy. In other words, one should modify, and possibly redesign all the layers of the computing infrastructure, accounting for control wherever is needed, and providing reasonable interfaces among the different layers. This must be done after having analyzed and decided which are the high level objectives that one wants to achieve.

In some sense, one can compare computing systems with power plants, and consider what has been done in the last decades [27], [28]. At the very beginning, most of the control systems were pretty rudimentary, and the overall plant was managed in a very inefficient way. However, the advances in control theory, and the availability of reliable physical models, enabled the possibility of automatizing many processes in power plants increasing their efficiency, and resilience [29]. As a consequence, starting from the lower layers of the plant – closer to the actual physical phenomenon that was to be controlled – control strategies for higher levels were designed aimed at coordinating and synchronizing the lower layers, and at the same time achieving higher efficiency, and increasing productivity. It is worth noticing that such a design paradigm permitted to reduce the uncertainty coming from the lower layers, also thanks to the adoption of robust control design techniques [30]. Hence, on top of these layers, high level functionalities of the plant were constructed with less difficulties, but keeping controllability of the plant in mind [31]. An interesting historical perspective on how control theory affected, and is still affecting different kind of industries can be found in [32].

On the other hand, computing systems were constructed on top of layers often based on heuristics, and, more important, that were not conceived having control in mind. As a consequence, obtaining reliable mathematical models of the lower layers is too difficult, and control capabilities are then limited. Sticking to the parallel example, it is interesting to notice how the uncertainty of the controlled variables in a power plant is decreasing while going towards the top layers of the hierarchy, while the exact opposite phenomenon can be observed in computing systems.

It is thus needed to define suitable control design patterns that can be used to design computing systems, and how they can be combined together. This is not an easy task since it will require to include in the design of cloud computing systems the integration of high level goals, functional and non-functional requirements, and to rethink and redesign how the different components interact and cooperate/compete together to achieve their own goals. The adoption of mathematical models of the computing phenomena would help in the development of control paradigms for computing systems.

There is some active research on these topics, especially in the software engineering community. They both consider top-down, e.g., [33], and bottom-up approaches, e.g., [34], [35], or a combination thereof in order to achieve better controllability of computing systems. Noteworthy, a holistic approach of a cloud computing system, and, more in general, of control-based computing infrastructure is definitely a large challenge [36].

Of course, much has been done over the last 30 years in computer science, and the challenge is not only how one can redesign the computing layers, but also how to deal with backward compatibility issues. Think, for example, to the introduction of IPv6 and all the backward compatibility problems that are still present with IPv4.

In any case, such a design could possibly benefit from the large experience that has been done in power systems, in order to address similar problems. One industrially relevant example is the management of alarms [37]. In power plants, indeed, there is a large amount of alarms that need to be quickly managed, in order to avoid damages to the plant or to the environment. Similar problems can be encountered in cloud computing, where alarms typically indicate a flash crowd, a hardware failure, or a malfunctioning of certain component of the infrastructure [24]. All these things must be managed promptly, in order to avoid the system to collapse, to lose customers, or to provide the service with limited capacity for long time [34].

3. ADDRESSING REPEATABILITY AND MEASURABILITY

Whatever is the considered computing infrastructure, even in the case of a control-theoretical integrated solution as proposed in the previous section, some sources of uncertainty still remain: flash crowds, hardware or software failures, etc will always affect the performance of the computing infrastructure. Interestingly, in some cases, uncertainty is also deliberately "injected" in the computing system as "approximate computing" in order to get higher computing performance; approximate computing can be both at the hardware level [38]–[40], and at the software level [41]–[43].

Independently of the source of uncertainty, the service provider should be able to offer some guarantees (possibly probabilistic), about some quantities of interest, in order to define understandable, repeatable and measurable SLOs. It is clear that cloud computing has so many diverse sources of uncertainty, that properly defining SLOs becomes a really hard task. Deterministic measurements of SLOs are practically impossible to obtain in a cloud infrastructure, due to many stochastic phenomena affecting the system. This hinders the possibility to have "repeatable experiments", at least in a strict sense.

The same problem comes when a new method is proposed in the literature and it should be evaluated against previous ones. As anticipated, the community has not agreed yet on a standard benchmark, or on standard procedures for carrying out such an evaluation. Thus, for example, determining what autoscaler is the best among the numerous ones proposed in the literature [44] is not so simple, and it depends on the specific stochastic realizations of different variables that were considered in the evaluation. This applies to virtually any functionality in a cloud environment, ranging from service admission control, to load balancing, up to VM placement [36]. Even though this is a well known issues, most of the work in the literature just limit the evaluation of the proposed methodologies to few experiments and providing no probabilistic bounds on the obtainable performance.

Therefore, the research question becomes: How can one properly define SLOs that can be measurable and repeatable in a cloud infrastructure?

3.1 Possible solutions

The only possibility to deal with the mentioned uncertain and stochastic behaviors is to adopt a probabilistic approach, as also suggested in [45]. However, in order to formally provide grounded probabilistic guarantees, one should follow these steps:

1. define an *ideal* (possibly deterministic) behavior for the SLO, $y^\circ(\cdot)$, independent of the sources of uncertainty;
2. set a prescribed risk ϵ that this ideal behavior is not met;
3. perform experiments over a sensible finite horizon $\mathcal{T}$ in order to evaluate which is the maximum *distance* ρ between the ideal behavior $y^\circ(\cdot)$, and the measured performance $y_m(\cdot)$ with a certain method $m \in \mathbb{M}$, where $\mathbb{M}$ is the set of all the possible methodologies. Notice that $y_m(\cdot)$ is a stochastic quantity, since it is affected by all the uncertain and unexpected behaviors described so far.

Formally, one can formulate the performance evaluation problem as a chance constrained optimization problem (CCP), i.e.,

$$CCP_m : \min_{\rho} \rho \qquad \text{subject to: } \mathbb{P}\{d_{\mathcal{T}}(y^\circ, y_m) \leq \rho\} \geq 1 - \epsilon. \tag{1}$$

for all the possible methodologies $m \in \mathbb{M}$, where $d_{\mathcal{T}}(\cdot,\cdot)$ computes the distance between the desired behavior of the system $y^\circ(t)$, and the actual behavior $y_m(t)$, over a finite horizon $\mathcal{T}$, and should be suitably chosen according to the specific application. The solution $\rho^\star_m$ for a given CCP_m is the maximum distance from the ideal behavior of the SLO for the considered method m. Among all the possible methodologies in the set $\mathbb{M}$, then, one should choose the methodology with minimum $\rho^\star_m$.

Unfortunately, solving a generic CCP is an NP-hard problem [46], [47]. However, randomized methods have been developed in the control community in order to obtain approximate solutions $\hat{\rho}^\star_m$ to the CCP (1) with a very high confidence [48]–[50], and with a limited number of experiments.

From the service provider perspective, such an approach offers a way to define specific measurable quantities, and confidence intervals with probabilistic guarantees that can be used to properly define new SLO. In addition, it is also defining procedures to evaluate and formally assess the performance of different functionalities. According to this perspective, the repeatability would not be in terms of obtaining the very same quantitative results, but obtaining similar probabilistic properties for the considered functionality.

4. CONCLUSION

Cloud computing received a lot of attention in the last years. However, managing uncertainty in a cloud environment is still an open problem. On one side, this paper discussed the problem of using control theory to decrease the uncertainty in the higher levels of the computing infrastructure, ranging from the hardware level, scaling up to the cloud. On the other hand, service providers should be aware of such uncertainty while defining their SLOs, since it is mining the possibility of having measurable and repeatable performance. This was related to the performance evaluation of different methodologies, which is a problem that, to date, was not addressed in a general way in the cloud community, and that could benefit by some results present in the control one.

References

[1] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems.* John Wiley & Sons, 2004.

[2] T. Abdelzaher, K. Shin, and N. Bhatti, "Performance guarantees for web server end-systems: a control-theoretical approach", *IEEE Trans. on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 80–96, 2002. DOI: 10.1109/71.980028.

[3] M. Kihl, G. Cedersjö, A. Robertsson, and B. Aspernäs, "Performance measurements and modeling of database servers", in *6th Int. Workshop on Feedback Control Implementation and Design in Computing Systems and Networks*, ser. FeBID, 2011.

[4] A. Leva, M. Maggio, A. V. Papadopoulos, and F. Terraneo, *Control-based operating system design.* IET, 2013. DOI: 10.1049/PBCE089E.

[5] E. Bini *et al.*, "Resource management on multicore systems: the ACTORS approach", *IEEE Micro*, vol. 31, no. 3, pp. 72–81, 2011. DOI: 10.1109/MM.2011.1.

[6] A. V. Papadopoulos, M. Maggio, F. Terraneo, and A. Leva, "A dynamic modelling framework for control-based computing system design", *Mathematical and Computer Modelling of Dynamical Systems*, 2014. DOI: 10.1080/13873954.2014.942785.

[7] H. C. Lim, S. Babu, J. S. Chase, and S. S. Parekh, "Automated control in cloud computing: challenges and opportunities", in *Proc. 1st Workshop on Automated Control for Datacenters and Clouds*, ser. ACDC, 2009, pp. 13–18. DOI: 10.1145/1555271.1555275.

[8] A. Ali-Eldin *et al.*, "How will your workload look like in 6 years? analyzing wikimedia's workload", in *Proc. IEEE Int. Conf. on Cloud Engineering*, ser. IC2E, 2014, pp. 349–354. DOI: 10.1109/IC2E.2014.50.

[9] P. Bodik *et al.*, "Characterizing, modeling, and generating workload spikes for stateful services", in *Proc. 1st ACM Symposium on Cloud Computing*, ser. SoCC, 2010, pp. 241–252. DOI: 10.1145/1807128.1807166.

[10] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn, "Self-adaptive workload classification and forecasting for proactive resource provisioning", in *Proc. 4th ACM/SPEC Int. Conf. on Performance Engineering*, ser. ICPE, 2013, pp. 187–198. DOI: 10.1145/2479871.2479899.

[11] C. Klein *et al.*, "Improving cloud service resilience using brownout-aware load-balancing", in *IEEE 33rd Int. Symposium on Reliable Distributed Systems*, ser. SRDS, 2014, pp. 31–40. DOI: 10.1109/SRDS.2014.14.

[12] T. Patikirikorala, A. Colman, J. Han, and L. Wang, "A systematic survey on the design of self-adaptive software systems using control engineering approaches", in *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS, 2012, pp. 33–42.

[13] A. Leva and M. Maggio, "Feedback process scheduling with simple discrete-time control structures", *IET Control Theory & Applications*, vol. 4, pp. 2331–2342, 11 2010. DOI: 10.1049/iet-cta.2009.0260.

[14] M. Maggio, F. Terraneo, and A. Leva, "Task scheduling: a control-theoretical viewpoint for a general and flexible solution", *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 4, 76:1–76:22, 2014. DOI: 10.1145/2560015.

[15] J. O. Kephart and D. M. Chess, "The vision of autonomic computing", *Computer*, vol. 36, no. 1, pp. 41–50, 2003. DOI: 10.1109/MC.2003.1160055.

[16] Y. Brun *et al.*, "Engineering self-adaptive systems through feedback loops", in *Software Engineering for Self-Adaptive Systems*, ser. Lecture Notes in Computer Science, vol. 5525, Springer Berlin Heidelberg, 2009, pp. 48–70. DOI: 10.1007/978-3-642-02161-9_3.

[17] M. Maggio, A. V. Papadopoulos, and A. Leva, "On the use of feedback control in the design of computing system components", *Asian Journal of Control*, vol. 15, no. 1, pp. 31–40, 2013, (invited paper). DOI: 10.1002/asjc.509.

[18] M. Maggio *et al.*, "Comparison of decision-making strategies for self-optimization in autonomic computing systems", *ACM Trans. on Autonomous and Adaptive Systems*, vol. 7, no. 4, 36:1–36:32, 2012. DOI: 10.1145/2382570.2382572.

[19] A. Filieri, C. Ghezzi, A. Leva, and M. Maggio, "Reliability-driven dynamic binding via feedback control", in *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS, 2012, pp. 43–52. DOI: 10.1109/SEAMS.2012.6224390.

[20] L. Wu, S. K. Garg, and R. Buyya, "SLA-based admission control for a Software-as-a-Service provider in cloud computing environments", *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1280–1299, 2012. DOI: 10.1016/j.jcss.2011.12.014.

[21] L. Tomás and J. Tordsson, "Improving cloud infrastructure utilization through overbooking", in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conf.*, ser. CAC, 2013, 5:1–5:10. DOI: 10.1145/2494621.2494627.

[22] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth, "Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control", in *Proc. 3rd Workshop on Scientific Cloud Computing Date*, ser. ScienceCloud 12, 2012, pp. 31–40. DOI: 10.1145/2287036.2287044.

[23] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch, "AutoScale: dynamic, robust capacity management for multi-tier data centers", *ACM Trans. Comput. Syst.*, vol. 30, no. 4, 14:1–14:26, 2012. DOI: 10.1145/2382553.2382556.

[24] H. Nguyen *et al.*, "AGILE: elastic distributed resource scaling for Infrastructure-as-a-Service", in *Proc. 10th Int. Conf. on Autonomic Computing*, ser. ICAC, 2013, pp. 69–82.

[25] R. Sturm, W. Morris, and M. Jander, *Foundations of Service Level Management.* SAMS, 2000.

[26] B. H. Cheng *et al.*, "Software engineering for self-adaptive systems", in *Software Engineering for Self-Adaptive Systems*, B. H. Cheng *et al.*, Eds., Berlin, Heidelberg: Springer-Verlag, 2009, ch. Software Engineering for Self-Adaptive Systems: A Research Roadmap, pp. 1–26. DOI: 10.1007/978-3-642-02161-9_1.

[27] R. Miller and J. Malinowski, *Power System Operation.* McGraw-Hill Education, 1994.

[28] B. Galloway and G. Hancke, "Introduction to industrial control networks", *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 860–880, 2013. DOI: 10.1109/SURV.2012.071812.00124.

[29] K. J. Åström, "Modelling and identification of power system components", in *Real-time Control of Electric Power Systems*, Elsevier, 1972.

[30] M. Morari and E. Zafiriou, *Robust Process Control.* Prentice Hall, Englewood Cliffs, 1989.

[31] R. Scattolini, "Architectures for distributed and hierarchical Model Predictive Control–a review", *Journal of Process Control*, vol. 19, no. 5, pp. 723–731, 2009. DOI: 10.1016/j.jprocont.2009.02.003.

[32] K. J. Åström and P. R. Kumar, "Control: a perspective", *Automatica*, vol. 50, no. 1, pp. 3–43, 2014. DOI: http://dx.doi.org/10.1016/j.automatica.2013.10.012.

[33] A. Filieri, H. Hoffmann, and M. Maggio, "Automated design of self-adaptive software with control-theoretical formal guarantees", in *Proceedings of the 36th Int. Conf. on Software Engineering*, ser. ICSE, Hyderabad, India, 2014, pp. 299–310. DOI: 10.1145/2568225.2568272.

[34] C. Klein, M. Maggio, K.-E. Årzén, and F. Hernández-Rodriguez, "Brownout: building more robust cloud applications", in *Proceedings of the 36th Int. Conf. on Software Engineering*, ser. ICSE, Hyderabad, India, 2014, pp. 700–711. DOI: 10.1145/2568225.2568227.

[35] J. Dürango *et al.*, "Control-theoretical load-balancing for cloud applications with brownout", in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, Los Angeles, CA, USA, 2014, pp. 5320–5327. DOI: 10.1109/CDC.2014.7040221.

[36] M. Kihl *et al.*, "The challenge of cloud control", in *8th Int. Workshop on Feedback Computing*, 2013.

[37] D. Rothenberg, *Alarm Management for Process Control: A Best-practice Guide for Design, Implementation, and Use of Industrial Alarm Systems*. Momentum Press, 2009.

[38] S. Narayanan, J. Sartori, R. Kumar, and D. L. Jones, "Scalable stochastic processors", in *Proceedings of the Conf. on Design, Automation and Test in Europe*, ser. DATE, Dresden, Germany, 2010, pp. 335–338. DOI: 10.1109/DATE.2010.5457181.

[39] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs", in *Proceedings of the 2012 45th Annual IEEE/ACM Int. Symposium on Microarchitecture*, ser. MICRO-45, Vancouver, B.C., CANADA, 2012, pp. 449–460. DOI: 10.1109/MICRO.2012.48.

[40] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: saving dram refresh-power through critical data partitioning", in *Proceedings of the 16th Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS, Newport Beach, California, USA, 2011, pp. 213–224. DOI: 10.1145/1950365.1950391.

[41] S. Misailovic, S. Sidiroglou, H. Hoffmann, and M. Rinard, "Quality of service profiling", in *Proceedings of the 32Nd ACM/IEEE Int. Conf. on Software Engineering*, ser. ICSE, Cape Town, South Africa, 2010, pp. 25–34. DOI: 10.1145/1806799.1806808.

[42] Z. A. Zhu, S. Misailovic, J. A. Kelner, and M. Rinard, "Randomized accuracy-aware program transformations for efficient approximate computations", in *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL, Philadelphia, PA, USA, 2012, pp. 441–454. DOI: 10.1145/2103656.2103710.

[43] M. Samadi, D. A. Jamshidi, J. Lee, and S. Mahlke, "Paraprox: pattern-based approximation for data parallel applications", in *Proceedings of the 19th Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS, Salt Lake City, Utah, USA, 2014, pp. 35–50. DOI: 10.1145/2541940.2541948.

[44] T. Lorido-Botrán, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments", *Journal of Grid Computing*, pp. 1–34, 2014. DOI: 10.1007/s10723-014-9314-7.

[45] A. Filieri *et al.*, "Software engineering meets control theory", in *10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, ser. SEAMS 15, Florence, Italy, 2015.

[46] A. Prékopa, "Probabilistic programming", in *Stochastic Programming*, ser. Handbooks in Operations Research and Management Science, vol. 10, 2003, pp. 267–351. DOI: 10.1016/S0927-0507(03)10005-9.

[47] A. Nemirovski and A. Shapiro, "Scenario approximations of chance constraints", in *Probabilistic and Randomized Methods for Design under Uncertainty*, Springer London, 2006, pp. 3–47. DOI: 10.1007/1-84628-095-8_1.

[48] G. C. Calafiore and M. Campi, "Uncertain convex programs: randomized solutions and confidence levels", *Mathematical Programming*, vol. 102, no. 1, pp. 25–46, 2005. DOI: 10.1007/s10107-003-0499-y.

[49] M. C. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design", *Annual Reviews in Control*, vol. 33, no. 2, pp. 149–157, 2009. DOI: 10.1016/j.arcontrol.2009.07.001.

[50] M. C. Campi and S. Garatti, "A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality", *Journal of Optimization Theory and Applications*, vol. 148, no. 2, pp. 257–280, 2011. DOI: 10.1007/s10957-010-9754-6.