Designing Self-Adaptive Software Systems with Control Theory: An Overview

Alessandro V. Papadopoulos
Mälardalen University
Västerås, Sweden
alessandro.papadopoulos@mdu.se

***Abstract*—The complexity of modern software systems is continuously growing, as well as the amount of data that is produced on a daily basis. This calls for sound and scalable approaches that can be used to tame such an emerging complexity. This tutorial aims at introducing the basic concepts of control theory that can be used to design self-adaptive systems. The tutorial is divided into two main parts. The first part discusses the modeling of discrete-time systems, and the design of controllers in the discrete-time domain. The second part presents a number of successful examples of the application of control-based approaches for the design of self-adaptive software systems.**

***Index Terms*—self-adaptive systems, control theory, software design**

I. Introduction and Motivation

The number of Internet of Things (IoT) devices is increasing exponentially, and it is expected to reach 31.4 billion by 2023 [1]. Digitalization and the Industrial IoT (IIoT) have caused business models to change, and many companies are increasingly becoming data companies, as they collect, administer, and analyze massive amounts of data to support their primary operations. IoT has gained a lot of attention in several areas including smart houses, industrial automation, and robotics, and applications are becoming more and more distributed, posing a number of challenges for the **management of the computational resources** needed for handling such increasingly complex systems, and for the **design of scalable and efficient solutions**.

The design of automated decision-making strategies for computing systems is known in the literature as **self-adaptation**, i.e., computing systems dynamically adapt to changes. Several disciplines have been identified as potential contributors. Among these, machine learning provides additional knowledge of the system [2], and control theory provides a vast array of tools for designing robust adaptive physical systems with formally assured behavior [3]. The combination of knowledge, robustness and formal guarantees has led to increased interest in developing control-based approaches to various computing system problems [4], [5].

Self-adaptation has a key role in the development of software systems [3] and control theory has proven to be a useful tool to introduce adaptation in such systems [5]–[10]. Self-management techniques are also prominent in the cloud industry. For example, companies like IBM (see projects like the IBM Touchpoint Simulator, the K42 Operating System [11]), Oracle[1], and Intel[2].

The research area of applying control theory to computing systems is becoming quite vast, as also analyzed in the recent survey by Shevtsov et al. [12]. Developing accurate system models for computing systems is in fact hard [13]–[15]. Moreover, both expertise in the computing system to be modeled, along with mathematical modeling skills are needed to deal with real systems [16]. These difficulties usually lead to the design of controllers focused on particular operating regions or conditions and *ad hoc* solutions that address a specific computing problem using control theory, but do not generalize [17], [18]. For example, in [19] the specific problem of building a controller for a .NET thread pool is addressed.

This line of research is motivated by the belief that safe, reliable, and resource-efficient solutions will not be possible without a firm mathematical foundation. Being able to program and manage computing systems efficiently and autonomously, while providing guarantees on the runtime behavior is of fundamental importance. As a consequence, a deeper understanding of control theory, and how a control-based solution can be designed can be beneficial to software designers to build more efficient computing systems.

II. Tutorial structure

A. Introduction to Discrete-time Control Systems

In the first part, the basic principles of linear control theory for discrete-time systems are revised [20], [21], including

- The introduction of the basic notation used in control systems, for the control input $u(t)$, the measured output $y(t)$, and the state variable $x(t)$. The variable $t \in \mathbb{Z}$ counts the discrete time.
- The definition of a discrete-time linear time-invariant system in the state-space form

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases}$$

where the matrices have suitable sizes.

This work has been partially funded by the Swedish Research Council (VR), under the "PSI" project.

[1]Oracle Automatic Workload Repository: https://oracle-base.com/articles/10g/automatic-workload-repository-10g

[2]Intel RAS Technologies for Enterprise: http://intel.ly/1TMzPKG

- An introduction of the discrete-time PID (Proportional Integral Derivative) control law in the *velocity form*,

$$u(k) = u(k-1) + K_p\left(e(k) - e(k-1)\right) + K_i h e(k) + K_d \frac{e(k) - 2e(k-1) + e(k-2)}{h}$$

and a discussion on how this can be implemented as a simple function in a software component.
- An introduction to more advanced control techniques, such as the Model Predictive Control (MPC) (more extensively presented in [21]).

B. Application Examples

The tutorial will also explore in more detail different research results that have been developed over the years using such kinds of techniques. In particular, the presented examples are:

- Task scheduling in real-time systems [22], [23].
- Clock synchronization in wireless sensor networks [24]–[27].
- Control of cloud-based applications [28]–[31].
- Elastic data streaming applications [14], [15].

C. Automating the design process

The tutorial concludes with the analysis of current trends in the area of software engineering for the automated design of control systems for self-adaptive software [32], [33], and some remarks on future research directions.

REFERENCES

[1] Ericsson. (2018, June) Ericsson mobility report. [Online]. Available: https://www.ericsson.com/491e17/assets/local/reports-papers/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf

[2] N. Esfahani, A. Elkhodary, and S. Malek, "A learning-based framework for engineering feature-oriented self-adaptive software systems," *IEEE Trans. on Software Engineering*, vol. 39, no. 11, 2013.

[3] B. H. C. Cheng *et al.*, *Software Engineering for Self-Adaptive Systems*, 2009, ch. Software Engineering for Self-Adaptive Systems: A Research Roadmap.

[4] 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 *SEAMS*, 2012.

[5] E. C. Kerrigan, "Feedback and time are essential for the optimal control of computing systems," *IFAC-PapersOnLine*, vol. 48, no. 23, 2015.

[6] G. A. Moreno, A. V. Papadopoulos, K. Angelopoulos, J. Cámara, and B. Schmerl, "Comparing model-based predictive approaches to self-adaptation: CobRA and PLA," in *Int. Symp. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2017.

[7] S. Shevtsov and D. Weyns, "Keep it simplex: Satisfying multiple goals with guarantees in control-based self-adaptive systems," in *FSE*, 2016.

[8] M. Maggio, T. Abdelzaher, L. Esterle, H. Giese, J. O. Kephart, O. J. Mengshoel, A. V. Papadopoulos, A. Robertsson, and K. Wolter, *Self-adaptation for Individual Self-aware Computing Systems*, 2017.

[9] S. Cerf, M. Berekmeri, B. Robu, N. Marchand, and S. Bouchenak, "Cost function based event triggered model predictive controllers application to big data cloud services," in *CDC*, Dec 2016.

[10] K. Angelopoulos, A. V. Papadopoulos, V. E. S. Souza, and J. Mylopoulos, "Engineering self-adaptive software systems: From requirements to model predictive control," *ACM Trans. Aut. and Adapt. Sys.*, vol. 13, no. 1, 2018.

[11] O. Krieger, M. Auslander, B. Rosenburg, R. W. Wisniewski, J. Xenidis, D. Da Silva, M. Ostrowski, J. Appavoo, M. Butrico, M. Mergen, A. Waterland, and V. Uhlig, "K42: Building a complete operating system," *SIGOPS Oper. Syst. Rev.*, 2006.

[12] S. Shevtsov, M. Berekmeri, D. Weyns, and M. Maggio, "Control-theoretical software adaptation: A systematic literature review," *IEEE Transactions on Software Engineering*, vol. 44, no. 8, pp. 784–810, 2018.

[13] 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*, 2015.

[14] V. Gulisano, A. V. Papadopoulos, Y. Nikolakopoulos, M. Papatriantafilou, and P. Tsigas, "Performance modeling of stream joins," in *ACM Int. Conf. on Distr. and Event-based Syst. (DEBS)*, 2017.

[15] V. Gulisano, H. Najdataei, Y. Nikolakopoulos, A. V. Papadopoulos, M. Papatriantafilou, and P. Tsigas, "STRETCH: Virtual shared-nothing parallelism for scalable and elastic stream processing," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–18, 2022.

[16] X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, P. Padala, and K. Shin, "What does control theory bring to systems research?" *SIGOPS Oper. Syst. Rev.*, vol. 43, no. 1, 2009.

[17] C. Lu, Y. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son, "Feedback control architecture and design methodology for service delay guarantees in web servers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 9, 2006.

[18] M. Tanelli, D. Ardagna, and M. Lovera, "LPV model identification for power management of web service systems," in *IEEE Conf. on Control Applications (CCA)*, 2008.

[19] J. L. Hellerstein, V. Morrison, and E. Eilebrecht, "Applying control theory in the real world: Experience with building a controller for the .net thread pool," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 3, 2010.

[20] B. Wittenmark, K. J. Åström, and K.-E. Årzén, "Computer control: An overview," *IFAC Professional Brief*, 2002.

[21] A. Leva, M. Maggio, A. V. Papadopoulos, and F. Terraneo, *Control-based operating system design*. IET, 2013.

[22] A. V. Papadopoulos, M. Maggio, A. Leva, and E. Bini, "Hard real-time guarantees in feedback-based resource reservations," *Real-Time Systems*, vol. 51, no. 3, 2015.

[23] A. V. Papadopoulos, E. Bini, S. Baruah, and A. Burns, "AdaptMC: A control-theoretic approach for achieving resilience in mixed-criticality systems," in *Euromicro Conf. on Real-Time Systems (ECRTS)*, 2018.

[24] F. Terraneo, L. Rinaldi, M. Maggio, A. V. Papadopoulos, and A. Leva, "FLOPSYNC-2: efficient monotonic clock synchronisation," in *IEEE Real-Time Systems Symp. (RTSS)*, 2014.

[25] F. Terraneo, A. Leva, S. Seva, M. Maggio, and A. V. Papadopoulos, "Reverse flooding: exploiting radio interference for efficient propagation delay compensation in wsn clock synchronization," in *IEEE Real-Time Systems Symp. (RTSS)*, 2015.

[26] A. V. Papadopoulos, F. Terraneo, A. Leva, and M. Prandini, "Switched control for quantized feedback systems: invariance and limit cycles analysis," *IEEE Trans. Aut. Contr.*, vol. 63, no. 11, 2018.

[27] F. Terraneo, A. V. Papadopoulos, A. Leva, and M. Prandini, "Flopsync-qacs: Quantization-aware clock synchronization for wireless sensor networks," *Journal of Systems Architecture*, vol. 80, 2017.

[28] C. Klein, M. Maggio, K.-E. Årzén, and F. Hernández-Rodriguez, "Brownout: Building more robust cloud applications," in *International Conference on Software Engineering (ICSE)*, 2014, p. 700–711.

[29] C. Klein, A. V. Papadopoulos, M. Dellkrantz, J. Dürango, M. Maggio, K.-E. Årzén, F. Hernández-Rodriguez, and E. Elmroth, "Improving cloud service resilience using brownout-aware load-balancing," in *IEEE Int. Symp. on Reliable Distributed Systems (SRDS)*, 2014.

[30] A. V. Papadopoulos, C. Klein, M. Maggio, J. Dürango, M. Dellkrantz, F. Hernández-Rodriguez, E. Elmroth, and K.-E. Årzén, "Control-based load-balancing techniques: Analysis and performance evaluation via a randomized optimization approach," *Control Engineering Practice*, vol. 52, 2016.

[31] E. B. Lakew, A. V. Papadopoulos, M. Maggio, C. Klein, and E. Elmroth, "KPI-agnostic control for fine-grained vertical elasticity," in *IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Comp.*, ser. CCGrid, 2017.

[32] M. Maggio, A. V. Papadopoulos, A. Filieri, and H. Hoffmann, "Automated control of multiple software goals using multiple actuators," in *ACM SIGSOFT Symp. on the Foundations of Soft. Eng.*, ser. FSE, 2017.

[33] A. Filieri, H. Hoffmann, and M. Maggio, "Automated multi-objective control for self-adaptive software design," in *Foundations of Software Engineering (FSE)*, 2015, p. 13–24.