ECE750-Topic 37
Engineering Self-Adaptive Software Systems (ESASS)
Fall 2023

Ladan Tahvildari (ladan.tahvildari@uwaterloo.ca)

Calendar Description: Distributed software systems and microservices with complex architectures running on cloud have significantly changed the way applications are developed and managed throughout the software development life cycle. However, the increased robustness and dynamic nature of these systems has also led to additional operational complexities. Some of these complexities are partially resolved by auto-scaling and recovery services provided in the cloud. Nonetheless, a large portion of the challenges remain dependent on operational expertise. In response to this issue, self-adaptive mechanisms have proven to be one of the most promising approaches in achieving system versatility, flexibility, and dependability. The topic of self-adaptive systems has been studied in a variety of application areas, including robotics, control systems, fault-tolerant computing, and biological computing. This course focuses on the software engineering aspects, including the methods, architectures, algorithms, techniques, and tools support self-adaptive and self-managing behavior. The graduate students will also learn to build "Self-Adaptive Software Systems (SAS)" using open-source runtime technologies.

Course Outline: This course offers graduate students not only relevant theoretical knowledge for designing self-adaptive systems but also practical experience for self-adaptive software implementation. On the theoretical side, the course will cover a wide range of subjects, which include topics in the domains of software-intensive systems (static versus dynamic evolution), feedback control of computing systems, autonomic systems and architecture, and adaptive software systems. Weekly lectures will be scheduled to present the following topics:

Basic Principles of Self-Adaptation and Conceptual Model: This topic concentrates on explaining the basic principles of self-adaptation, describing the conceptual model of a self-adaptive system, and applying the conceptual model to a concrete self-adaptive application [3,6,10].

Automating Tasks: This topic focuses on shifting complex system management tasks away from error-prone human operators and towards software defined managers. The central aspect is the decomposition of the external manager into its basic functions, such as MAPE-K (Monitor Analyze Plan Execute-Knowledge) feedback loop [9,12].

Requirements-Driven Adaptation: This topic concentrates on the need for full consideration of the requirements that should be met by the managing system [11]. The focus here is on requirements elicitation, analysis, specification, operationalization, and maintenance.

Architecture-Based Adaptation: This topic focuses on addressing the development of self-adaptive systems in a systematic manner [4,8]. Architectural principles such as abstraction and separation of concerns are taught to offer perspective on the system and its environment to make effective adaptation decisions.

Control-Based Software Adaptation: This topic concentrates on applying mathematical frameworks from control theory for designing self-adaptive systems [2]. The quantification of adaptation goals, selection of an appropriate controller, and identification of a model for the managed system that is suitable for the controller are taught.

Runtime Models: This topic focuses on realizing concrete design models at runtime for self-adaptive systems [1]. The notion of models used during software development are extended to the runtime for providing various metrics that can be used to make adaptation decisions.

Guarantees Under Uncertainties: This topic concentrates on the need to mitigate uncertainties and provide guarantees for adaptations [7]. Evidence of adaptation goal satisfaction on how to provide trustworthiness for self-adaptive systems is taught. The other focus is on introducing machine learning techniques to facilitate the functional components of a self-adaptive system. For instance, a fuzzy-learning approach can be used to reduce a large search space of adaptation options [5].

It is also planned for some lectures to feature guest speakers from IBM to provide insights regarding specialized course material regarding the latest platforms and technologies.

Grading: The course will consist of three assignments (weighted equally), a project, and two quizzes. The grading scheme is: (i) assignments (45%), (ii) project proposal (5%), (iii) project presentation/demo (10%), (iv) project report (20%), and (v) two quizzes (20%).

Course Assignments: The course includes three assignments to be done by a group of two students. Assignments will guide students through process of developing a self-adaptive system using various advanced platforms and services. The primary runtime technologies for the assignments include Virtual Private Servers from IBM Cloud, metrics monitoring from IBM Observability, Java runtime based on Eclipse OpenJ9, and microservices deployed in the OpenShift Container Platform. In the context of self-adaptive systems, the OpenJ9 runtime, the Open Liberty web applications, and the OpenShift resources make up the managed elements.

The students are guided to construct the managing system and complete a self-adaptive software system as follows:

Assignment 1: The learning goal is to set up and monitor the managed elements. Students will be tasked with familiarizing with IBM Cloud, setting up their development tools, deploying a target system on the OpenShift platform, and configuring a monitoring agent using IBM Observability. In relation to the MAPE-K loop, the focus here is on the monitoring component. During this phase, sensors from IBM Observability are utilized for data collection, and the collected data are subsequently converted to behavioral patterns and symptoms.

Assignment 2: The second assignment will be in continuation of the first assignment. The students will focus on processing the collected runtime metrics of a target system. The learning goal for this assignment is to have students gain familiarity with certain self-* properties of self-adaptive systems as well as how to analyze microservice metrics to achieve these properties. In relation to the adaptation loop, the focal point here is the analyze component.

Assignment 3: For the last assignment, students are expected to integrate prior assignment solutions together to complete the closed adaptation loop. By the conclusion of the third assignment, students will have touched on all parts of the MAPE-K loop. Furthermore, they will have gained some of the skills and knowledge for implementing a self-adaptive system. In terms of the MAPE-K loop, the focus here is on the plan and execute components, which determines how to achieve the optimal outcome and carry out the actions through actuators.

Course Project: Overall, the three assignments offer a relatively guided path for students in the practical implementation of a self-adaptive software system. In contrast, the course project is where students are given free rein in terms of domain and application to build a self-adaptive system of their interest. The course project should also be done by a group of two students. Prior beginning the course project, each group is expected to submit a proposal outlining their project goals with respect to self-adaptivity and a preliminary high-level design of the system. As the main deliverable for the end of the course, each group is to submit their final implementation as well as a report documenting their work in detail. Moreover, a short presentation and demonstration of their project is also to be delivered.

Intended Learning Outcomes: Upon completion of the course, graduate students will learn theoretical knowledge for designing self-adaptive systems, gain hands-on experience of runtime technologies, understand the current outstanding problems, and be familiar with the state-of-the-art solutions of self-adaptive software systems.

Required Background: Undergraduate degree in software engineering or computer engineering or computer science or electrical engineering or related degree with software development experience in design and implementation.

Prerequisites: While not required, it is recommended that students have taken courses such as ECE 606, ECE 651, ECE 653 prior to enrolling in this course.

References: Selection of articles, journal papers, and book chapters from technical literature; just to name a few:

[1] N. Bencomo, R. France, B. Cheng, and U. Aßmann, *Models@run.time: Foundations, Applications, and Roadmaps*, Lecture Notes in Computer Science (LCSN) 8378, 2014.

[2] Y. Brun, G. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, "Engineering Self-Adaptive Systems Through Feedback Loops", *In B. Cheng, R. Lemos, H. Giese, P. Inverardi, and J. Magee (editors), Software Engineering for Self-Adaptive Systems*, pages 48–70, 2009

[3] T. Chen, R. Bahsoon, and X. Yao, "A Survey and Taxonomy of Self-Aware and Self-Adaptive Cloud Autoscaling Systems", *ACM Computing Survey*, 51(3):1–40, 2018.

[4] J. Floch, S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, and E. Gjorven, "Using Architecture Models for Runtime Adaptability", IEEE Software 23(2):62–70, 2006.

[5] P. Jamshidi, C. Pahl, N. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The Journey So Far and Challenges Ahead", *IEEE Software*, 35(3):24–35, 2018.

[6] J. Kephart and D. Chess, "The Vision of Autonomic Computing", *IEEE Computer*, 36 (1):41–50, 2003.

[7] D. Perez-Palacin and R. Mirandola, "Uncertainties in the Modeling of Self-Adaptive Systems: A Taxonomy and an Example of Availability Evaluation", In *Proceedings of the International Conference on Performance Engineering (ICPE)*, pp. 3–14, 2014.

[8] P. Oreizy, M. Gorlick, R. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. Rosenblum, and A. Wolf, "An Architecture-Based Approach to Self-Adaptive Software", *IEEE Intelligent Systems,* pp. 54–62, 1999.

[9] M. Rodriguez and R. Buyya, "Container-Based Cluster Orchestration Systems: A taxonomy and Future Directions", *Software: Practice and Experience*, 49(5):698–719, 2019.

[10] M. Salehie and L. Tahvildari, "Adaptive Software Systems: Emerging Trends and Open Problems", *ACM Transactions on Autonomous and Adaptive Systems*, 4(2):1–42, 2009.

[11] V. Souza, A. Lapouchnian, W. Robinson, and J. Mylopoulos. "Awareness Requirements for Adaptive Systems", In *Proceedings of the International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pp. 60–69, 2011.

[12] D. Weyns, *An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective*, Wiley IEEE Press, 2021.