Concurrent Programming with the Actor Model in Elixir*

Conference Tutorial

Saverio Perugini[1] *and David J. Wright*[2]
[1]*Department of Computer Science*
[2]*Learning Teaching Center*
University of Dayton
Dayton, Ohio 45469
{saverio,dwight1}@udayton.edu

Project Site: http://sites.udayton.edu/operatingsystems

1 Tutorial Summary

This tutorial fosters and facilitates discussion and innovation around the topic of *teaching concurrency and synchronization* in an undergraduate *operating systems* (OS) course [2]. In particular, we lead participants through a variety of active-learning exercises for teaching a 'concurrent programming and synchronization' module of an undergraduate OS course using the Actor model of concurrency—a concurrent programming model that is increasing in popularity, especially in the Elixir programming language. The model naturally focus the programmer's locus of attention on the interaction between concurrent entities to collaboratively solve a problem—the primary learning outcome—rather than the level-low details of language syntax and the minutia of which individual data to protect/lock and how to protect/lock it. In this tutorial, classical (e.g., sleeping barber) and modern (e.g., chat server) problems of synchronization are demonstrated through the lens of the Actor model using in-class laboratory plans that attendees can adopt. Participants are exposed to concurrent programming in Elixir[1] through asynchronous communication via message

[1]http://elixir-lang.org/

passing and mailboxes. Participants should have working knowledge of concurrency/synchronization in a language such as C or Java and are encouraged to bring laptop computers, especially to follow the labs demonstrated.

2 Broadening Aspects

This tutorial is part of a three-year NSF-funded IUSE (Improving Undergraduate STEM Education) project titled "Engaged Student Learning: Reconceptualizing and Evaluating a Core Computer Science Course for Active Learning and STEM Student Success" (2017–20) whose goal is to foster innovation, in both content and delivery, in teaching OS through the development of a contemporary model for an OS course that aims to resolve significant issues of misalignment between existing OS courses and employee professional skills and knowledge requirements.

While an OS course is a nexus in a CS program (connecting the introductory programming sequence to upper-level electives), the typical pedagogical approach to it has become dated and stale—most of the recent focus has been on improving the introductory programming sequence primarily for retention—because it has not been responsive to the transformed landscape of modern computing platforms (e.g., from desktop to mobile; from single- to multi-core architectures), the job market, and the concomitant progress in active, student-centric learning (e.g., from a traditional, content-centric, purely lecture-based course to an active, learner-centric, hybrid lecture/lab-based format). Addressing this issue is the rationale for our NSF-funded project. Thus, a broader objective of this tutorial is to promote and facilitate use of our re-conceptualized course model for OS content and pedagogy. Specifically, we aim to help faculty at other institutions adopt this model in a systematic and simplified way.

The course model involves three progressive modules: 1) mobile OSs and Internet of Things, 2) concurrent programming and synchronization, and 2) cloud computing and big data processing. This tutorial focuses on the 'concurrent programming and synchronization' module of the model, which is the most easily adoptable of the three modules, because it is not dependent on the presence of mobile devices or external resources (e.g., Amazon Web Services). However, participants can also expect an introduction to our re-conceptualized course model; a demonstration of a set of re-usable, in-class laboratory plans; discussion of benefits and tradeoffs of employing the model; and an invitation to participate more actively in the project beyond the tutorial (see below).

2.1 Laboratory Manual

In a 2019 SIGCSE birds-of-a-feather discussion that we lead [2], faculty teaching OS found the ability to plug-and-play with the active-learning, laboratory

plans attractive and see significant value in making use of them in their courses and teaching activities, especially since developing real-world lab and project plans requires substantial effort and time. Tutorial attendees will be granted access to our Laboratory Manual, which contains these active-learning, exercises, each categorized into one of the three topic modules. Moreover, we intend to provide stipends from our grant to attendees who both adopt the model, or parts thereof, in Spring 2020 and collect pre- and post-evaluative data for a semester.

2.2 Community of Practice

We are also establishing an OS teaching community of practice to share both materials and experiences in the teaching of OS. To support the community in sharing expertise and perspective, we have developed an open-access, Git repository of items related to teaching OS The items collected in the repository are shared and accessed through our GitHub Pages portal site: `https://saverioperugini.github.io/Teaching-Operating-Systems-Community-of-Practice/`). The community includes members who attended our 2018 CCSC:MW tutorial [1] and 2019 SIGCSE birds-of-a-feather [2]. We plan to invite 2019 CCSC tutorial attendees to share both their perspective and materials to this repository.

2.3 Advisory Group

We are also establishing an *advisory group* of computer science faculty members for this project for an external perspective on the model and its adoption. Another goal of this tutorial is to expand the participation of regional faculty in the advisory group. Members of this group serve as advisors and are asked to provide an external perspective on both the module content and the laboratory plans.

3 Schedule of Activities

- Introduce attendees to the course model, the content modules, and the active-learning exercises in the Laboratory Manual.
- Work through lab plans of the Actor model of concurrency in Elixir.
- Share our experience in teaching the OS course using this model.
- How to adopt the course model and use the active-learning labs in the Laboratory Manual.

- Invite participation in our community of OS educators.
- Discussion: Questions and Answers.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Numbers 1712406 and 1712404. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] S. Perugini and D.J. Wright. Developing a contemporary operating systems course. *Journal of Computing Sciences in Colleges*, 34(1):155–158, 2018. Conference Tutorial.

[2] S. Perugini and D.J. Wright. Developing a contemporary and innovative operating systems course. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE)*, page 1248, New York, NY, 2019. ACM Press. Conference Birds-of-a-Feather; DOI: `http://doi.acm.org/10.1145/3287324.3293734`.