CS540 Advanced Software Engineering

COURSE DETAILS

Classes are held MWF 9:45AM-11:00AM in MK622 and online Zoom bridge 276-644-8345

Contact Instructor: Audris Mockus, audris@utk.edu

Course Web Site: `https://github.com/cs540-25/news`

Please provide me with your GitHub ID (in case I don't have it already) so that I could add you as a member of the group.

IN A NUTSHELL

We will read and present papers

We will work on major big data projects

WHO SHOULD TAKE THIS COURSE

Students who want to get familiar with contemporary software development technologies used to build data-intensive software systems: systems that rely on operational data to perform their basic functions. Recommendation, rating, search, and social networking systems or systems that rely on crowd-sourcing are representative examples. In particular, cloud-first software relies on usage data via, for example, AB testing, to define its functionality and requires practices such as DevOps to improve quality and to accelerate delivery schedules.

Students who can work independently and are highly motivated to learn how to effectively use cloud technologies and other tools necessary to build contemporary data-intensive systems.

EXPECTED OUTCOMES

Upon completion, students will have familiarity with the main principles of how to build effective data-intensive systems.

WHY TAKE THIS COURSE?

New generation of software technologies enables an order of magnitude faster delivery cycles, but the traditional development skills and practices in most mature organizations hold back this transformational change. The course will provide some of the critical, but presently rare skills of how to translate the abundant data streams into the evidence that defines system's functionality, quality, and performance. With the data revolution just beginning, the strong need for such skills will extend into the foreseeable future.

OVERVIEW

With the costs of computing plummeting a billion times over half a century, the primary objective for software engineering has followed a trajectory from techniques that minimize computing cycles, to improving developer productivity, and, as of late, to creating engaging and productive user experiences in a wide range of software such as search, recommendation, social and professional networks and other areas. As computing becomes a commodity, software is omnipresent in all parts of life. Because most users are not able to understand software systems or articulate their needs, software systems have both to collect massive amounts of operational data related to user activities and to analyze and use that data to provide user experiences that lead to desired outcomes, e.g., increasing sales revenue or the quality of software (if the user happens to be a software developer).

It no longer suffices to deliver software that requires, for example, an entry field for a specific piece of data. Instead, the software has to ensure that users can and will enter the relevant data or, more often, it has to derive the necessary information from observed user behavior. Moreover, the software has to ensure that the resulting data reflects the intended quantities, and that the quality of that data is sufficient to make important decisions. Unlike in experimental data, where measures are carefully designed and accurate, events in operational data, such as sensor logs, may be generated in distinct contexts, the relevant events may be unobserved, and the data may be not accurate, filtered, or even tampered with.

Such software needs to be engineered to provide accurate and actionable data (evidence) and, therefore, it requires novel approaches to design, implement, test, and operate it (evidence engineering). The criteria for success demand much more than performing a well-defined task according to specification. Software has to provide data that is both accurate and also leads to the intended user behavior.

In contexts where the desired user behaviors are relatively well defined, some existing software systems achieve these goals through detailed measurement of behavior and an extensive use of AB testing (in which two samples of users provided slightly different versions of software in order to estimate the effect these differences have on user activity).

As operation and measurement are becoming increasingly a part of software development, the separation between the software tools and end-user software are increasingly blurred. Similarly, the measurement associated with testing and use of software is increasingly becoming an integral part of the software delivered to users.

Definition Evidence is accurate data, properly interpreted.

Definition Evidence Engineering: methods and principles to produce actionable evidence from operational data

Primary aim of Evidence Engineering is to discover approaches that increase the integrity of data.

COURSE REQUIREMENTS

Students are expected to be proficient in at least one language, such as shell, python, or R. The need to be familiar with basic software engineering practices such as use of version control systems and issue trackers and basic test strategies. Some familiarity with contemporary cloud technologies, such as containers, virtual machines and related frameworks and languages would be beneficial. Basic understanding of quantitative concepts such as probability, random variables, and regression would be needed.

Everyone is expected, however, to be willing and highly motivated to catch up in the areas where they have gaps in the relevant skills.

Students will be evaluated using the following criteria:

1. Classroom participation (15%): students are expected to read all material covered in a week and come to class prepared to take part in the classroom discussions and present their work.

2. Assignments (40%): Assignments may involve a programming task, a presentation of an academic paper, and an implementation of the approach described in a paper.

3. Project (45%): one original project done alone or in a group of 2 or 3 students. The project will explore one or more of the themes covered in the course that students find particularly compelling. The group needs to submit a project proposal (2 pages IEEE format) approximately 1.5 months before the end of term. The proposal should provide a brief motivation of the project, detailed discussion of the data that will be obtained or used in the project, along with a time-line of milestones, and expected outcome.