Using Microservices to Modularize Components and Teaching Assistant Development Teams for a Robotics Design Project Computer System

Mr. Jared Dean Mitten, Ohio State University

Jared D. Mitten is a Computer Science and Engineering (CSE) major at The Ohio State University and is currently an Undergraduate Teaching Assistant with the Fundamentals of Engineering for Honors (FEH) program. He is a lead developer for several software systems used by the FEH program, including the robot course scoring system and the online robot part store. He will graduate in December 2019 with his B.S in CSE with a focus on data management in the cloud.

Andrew Phillips, Ohio State University

Andrew Phillips graduated summa cum laude from The Ohio State University in May 2016 with a B.S. in Electrical and Computer Engineering and with Honors Research Distinction and again in December 2018 with a M.S. in Electrical and Computer Engineering. He is currently pursuing a Ph.D. in Engineering Education at The Ohio State University. His engineering education interests include first-year engineering, active learning, learning theory, and teaching. His current dissertation topic is the training and skill development of teaching assistants. As a Graduate Teaching Associate for the Fundamentals of Engineering for Honors program, he is heavily involved with developing and teaching laboratory content, leading the maintenance of the in-house robotics controller, and managing the development of the robotics project.

Dr. Kathleen A. Harper, Ohio State University

Kathleen A. Harper is a senior lecturer in the Department of Engineering Education at The Ohio State University. She received her M. S. in physics and B. S. in electrical engineering and applied physics from Case Western Reserve University, and her Ph. D. in physics from The Ohio State University. She has been on the staff of Ohio State's University Center for the Advancement of Teaching, in addition to teaching in both the physics and engineering education departments. She is currently a member of the ASEE Board of Directors' Advisory Committee on P-12 Engineering Education and ASEE's Projects Board.

Dr. Richard J. Freuler, Ohio State University

Richard J. (Rick) Freuler is a Professor of Practice and the Director for the Fundamentals of Engineering for Honors (FEH) Program in Ohio State's Department of Engineering Education in the College of Engineering. He teaches the two-semester FEH engineering course sequence and is active in engineering education research. He is also affiliated with the Mechanical and Aerospace Engineering Department and conducts scale model investigations of gas turbine installations for jet engine test cells and for marine and industrial applications of gas turbines at the Aerospace Research Center at Ohio State. Dr. Freuler earned his Bachelor of Aeronautical and Astronautical Engineering (1974), his B.S. in Computer and Information Science (1974), his M.S. in Aeronautical Engineering (1974), and his Ph.D. in Aeronautical and Astronautical Engineering (1991) all from The Ohio State University.

Using Microservices to Modularize Components and Teaching Assistant Development Teams for a Robotics Design Project Computer System

Abstract

This paper describes the use of a microservices architecture for separation of concerns in a complex hardware and software system which runs multiple components of a robotics design and build project. At The Ohio State University, this robotics project serves as a cornerstone design experience in a two-semester first-year engineering honors course sequence. Student teams must design, construct, and program autonomous robots to navigate and complete tasks on a themed course. This extensive robot course system is operated entirely electronically with a system of hardware sensors, outputs, boards, and several pieces of software that track the state and score of each task in real-time. The students operate their robot via a programmable robotics controller that communicates wirelessly with the course system. All the software and hardware components of the course computer system and robot controller are developed and maintained by teams of graduate and undergraduate teaching assistants with technical experience in these areas, and the components must all communicate efficiently so that students have the smoothest experience possible.

Managing the complexity of this large computer system is a challenge since several software and hardware components must interact. Microservices allow for a reduction in overall system complexity by structuring the system into a collection of loosely coupled components. The individual system components are thus smaller in scope and are more easily developed and maintained by specialized teams of teaching assistants. This microservices architecture allows several specialized teams to design, develop, and maintain different software and hardware pieces of the robot course simultaneously. By implementing principles of separation of concerns, the system is modularized, which aids with debugging by narrowing the scope of issues that can occur. Additionally, the reduced scope of the individual components makes it easier to explore and implement new innovative design features. Finally, this system allows for onboarding of new teaching assistant team members who may have little or no experience because they are only responsible for learning how a single component works. All these advantages of microservices and separation of concerns allow the teaching assistants to construct an entirely new course scenario every year in a timely and modular manner while strengthening their technical and team working skills.

Thus, the goals of this paper are to describe the modular team structure for the development and maintenance of this large robotics course computer system, and to use data from recent years to show the benefits of this structure on the skill development of teaching assistants and on the design experience for the students.

Introduction

Developing a complex system of software and hardware components can be challenging when many components must interact. Large computer systems can be designed using a microservice architecture to form a system with loosely coupled components that are small and easy to maintain. The Fundamentals of Engineering for Honors program at The Ohio State University is using microservices to modularize the components of a course computer system for a robotics design project. This course computer system consists of several small software and hardware components that can be developed and maintained by specialized teams of teaching assistants. This paper describes how this separation of concerns benefits the skill developments of teaching assistants and contributes to the enhancement of the students' experiences as they conduct their design project.

Program and Project Background

The first-year engineering honors program is a two-semester course sequence. In the first semester, students learn problem solving techniques and programming languages, such as Excel, MATLAB, and C/C++. They also complete several hands-on laboratory experiences designed to introduce them to a breadth of engineering disciplines, improve their technical communication skills, and allow them to practice effective teamworking. The class employs an inverted classroom approach to increase student engagement with the content and instructional staff [1]. In the second semester, students learn hand-drawn and computer-aided design (CAD) graphics, and they participate in a large cornerstone design project.

The most popular of the second-semester cornerstone design projects is the robotics project. This robot project has many components which have been developed over the years [2]. Teams of four students are tasked to design, build, program, wire, budget, and document an autonomous robotic vehicle to complete tasks on a robot course. Students use the programing and design skills they have previously learned in the program along with new robotics skills to complete this large project over about ten weeks. During the project, student teams spend significant time testing their robot and interacting with the robot course.

The robot course is a 12' by 12' platform composed of four identical regions. Each region has several tasks the robot must complete, such as detecting lights, pushing buttons, flipping levers, and transporting objects. There are two identical course platforms, each with four individual regions. Robots earn points in a run based on the tasks they complete. In the last few years, these tasks have become fully electronically implemented such that the scoring can be done automatically. Additionally, each team is provided a programmable robot controller which interacts with the course software system through the Robot Positioning System (RPS). Thus, there are several hardware and software components which must be developed and maintained for this robot course to function properly. A CAD render of a robot course platform can be seen in Figure 1.

Approximately 50 Undergraduate Teaching Assistants (UTAs) and eight Graduate Teaching Associates (GTAs) are employed to assist with all aspects of the program, such as helping students learn content, grading student work, and preparing materials for the design projects. Most of these teaching assistants (TAs) work on the robot project, although some work on other projects. In order to break down all of the components of the robot course and to make sure they each work properly, there are several teams of TAs which are in charge of various aspects of the project. This paper focuses on the TAs and course components relating to hardware and software, although there are other teams and parts of the project. It is the modularity and separation of concerns of the robot course components and TA development teams which enable the whole project to go smoothly for the students.

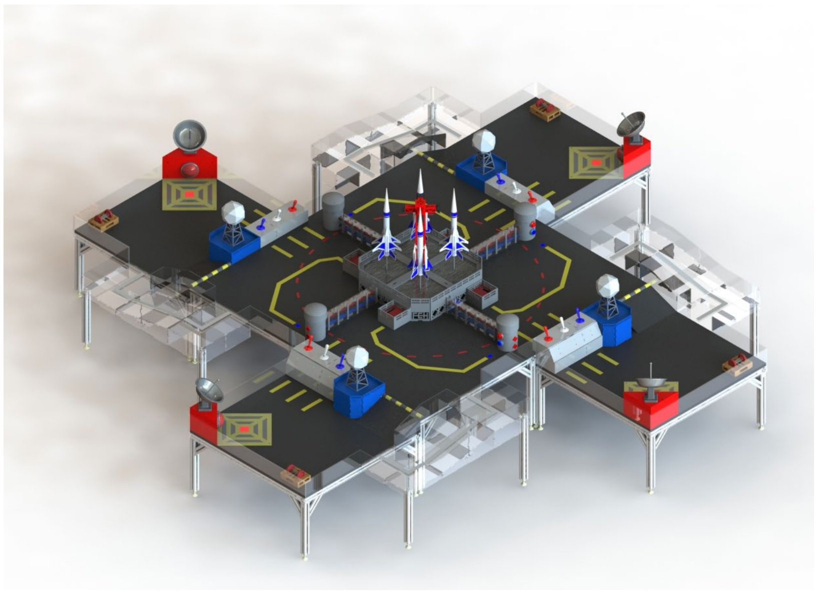
Figure 1: CAD render of a robot course platform with four regions.

Description of Robot Course Hardware and Software Components

Six main hardware and software components make up the computer-related features of the robot course. These components are developed and maintained separately, but with regard to how they need to interface when coming together. The following paragraphs describe how each of these components work in context of the robot course and project. The six components are:

- Course electronics
- Course software
- RPS
- Controller hardware
- Controller software

Course electronics refers to all of the hardware elements which run the tasks on the robot course. This includes LEDs for the robot to detect, buttons and other switches for the robot to press or flip, switches for detecting when an object has been picked up or set down, and other sensors such as potentiometers and break beam sensors for detecting completion of other unique tasks. This also includes extra LEDs, motors, and other components which perform only aesthetic functions on the course, such as lighting up a panel when a button is pressed or moving an arm when a lever is flipped. All of these elements are embedded on top of or within the robot course and are controlled by boards located underneath the courses. Under each of the four course regions, an input/output board designed by TAs is used to control and monitor digital inputs and outputs, PWM signals, and analog input signals. An Arduino Uno board is used to control more power-intensive PWM signals, such as for strips of individually addressable LEDs. A Raspberry Pi board is used to control the other two boards and communicate with the course software framework and RPS.

The course software system is responsible for managing each course, determining the state of task completion, and controlling the visual elements on the courses. The system is designed using a client-server model. A course contains four Raspberry Pi clients that are each responsible for handling a region. Each Raspberry Pi is connected to a board with general purpose I/O, PWM, and analog ports [3]. These ports are used to connect to various sensors and visual elements on the courses such as buttons, switches, LEDs, and servos. Each Raspberry Pi also communicates over a serial connection with an Arduino to control individually addressable LEDs. These Raspberry Pi clients constantly send updates to the central course computer server via a TCP connection. The course computer is responsible for taking the information from each client and processing it to determine the progress of robots on the course while also keeping track of other properties such as the run duration. The server presents information on the course control panel, which is a touch screen, about task completion, randomized task decisions, and run time. Finally, the course software system is responsible for sending updates with this information to the Robot Positioning System (RPS) via a UDP connection.

The Robot Positioning System (RPS) is responsible for tracking the position of robots on the courses using a system of cameras that locate micro QR codes mounted on student robots [4]. Using a virtual grid system, the positions of robots can be determined with quarter-inch accuracy. Position coordinates and information about certain tasks received from the course computer are broadcasted to student robot controllers using wireless XBee radio modules. Finally, the RPS computer is responsible for making a copy of the data it receives from the course server computer. Additional information such as the team name encoded in the QR code is combined with the information from the course server and is transmitted to the course scoring computer via a UDP connection.

The course scoring system is responsible for calculating numeric scores for each robot on the course as it completes tasks in real time. This real-time scoring is done by processing the information received over the network from the RPS computers. Unlike the other systems, the scoring system consists of a single central computer. It processes the data received from the RPS computers at both course platforms and conducts scoring on all eight regions simultaneously. When runs are finished, the scoring system saves scores to an online Google spreadsheet and generates a PDF scoring sheet that is shared with students for their own documentation purposes. Scores are displayed in real-time using one screen for each course that shows team designations and current total scores. Throughout the semester, these displays are present for students to evaluate their robot performance as they are conducting tests. During the final public competition, the software also maintains a competition bracket that is displayed between matches on each of the two screens. As teams win in matchups of four robots, the bracket is automatically updated and displayed to the audience.

The robot controller used in this project was made in-house specifically for this project, although it was designed with versatility in mind [5]. The controller hardware includes 32 general-purpose I/O ports, 8 servo motor ports, 4 DC motor ports, a touch LCD, an accelerometer, an XBee radio transmitter for communication with the wireless RPS system, and a Micro SD card port for uploading programs. It is powered by a 12-volt lithium-ion battery, and the whole controller is embedded in an acrylic case. In this way, the low-level development board electronic interfacing is abstracted away into a higher-level controller design which can be used much more easily by the students. All of the electrical components of a team's robot are operated through this controller, and the robot also communicates with the RPS through the XBee on the controller. Some parts or components on the controller are susceptible to improper handling by the user, and thus, they require maintenance in order to keep an adequate supply on hand.

Students program the robot controller using C++ with a program-specific library of functions designed to abstract away some of the lower-level difficulties associated with getting programs to interact with hardware on the controller. This code is composed and compiled in the open-source QtCreator environment, loaded onto a micro SD card, and then loaded onto the controller from the SD card. Students can use specific functions to get certain information from the RPS to use during a run on the course, including positional, heading, and task objective information. Some of the task objective information may change from year to year based on the changing tasks on the course. In order for all of the above to work, the firmware of the controller must also be up to date. Functions have to be updated for the tasks objectives, and when the RPS is updated, the firmware is similarly updated to remain consistent. Additionally, the QtCreator environment must be maintained to work with the SD card and controller as updates are made.

Each software and hardware component of the system interacts with the others using various forms of communication protocols. Figure 2 provides an overview diagram of the course computer system architecture and how each component connects with the other components.

Teaching Assistant Teams

Each of the course computer system hardware and software components described above is under the supervision of a team of teaching assistants. These TA teams are in charge of implementing required elements of the component for the course each year, developing new ideas and more efficient ways of achieving functionalities, and maintaining the components throughout the duration of the project. Each team is led by one or more GTAs or experienced UTAs. These team leads communicate with each other and with the program instructors in order to ensure the needs of the students and of the other teams are met. Underneath the lead TAs, each team has five or more other TAs involved with it. It is up to the lead TAs to assign tasks to the team's members and make sure that all tasks are being completed on time and with the students in mind. TAs are encouraged to join more than one team based on their prior experience, skill, and interest, and thus, there is some overlap in TAs on each of the computer system teams in any given year. TAs who are involved with a team over multiple years may take on larger leadership roles and more complicated tasks within the team.

This TA team structure has several benefits to the TAs themselves, which help to empower them to improve the system overall [6]. The modularity of the teams allows TAs to specialize in areas which interest them, and this allows them to get more hands-on experience in these electronics areas. Most of these skills are not gained during normal class work, so, these TAs are gaining valuable skills which can be translated to job interviews and industry work, as well as future academic work. Much of this skill development occurs due to the mentoring structure inherent in the upkeep of these teams. Lead TAs must pass on information to newer TAs so that they can become more involved and knowledgeable to be more productive for the robot project, but also so that they can take on those lead roles in the future.

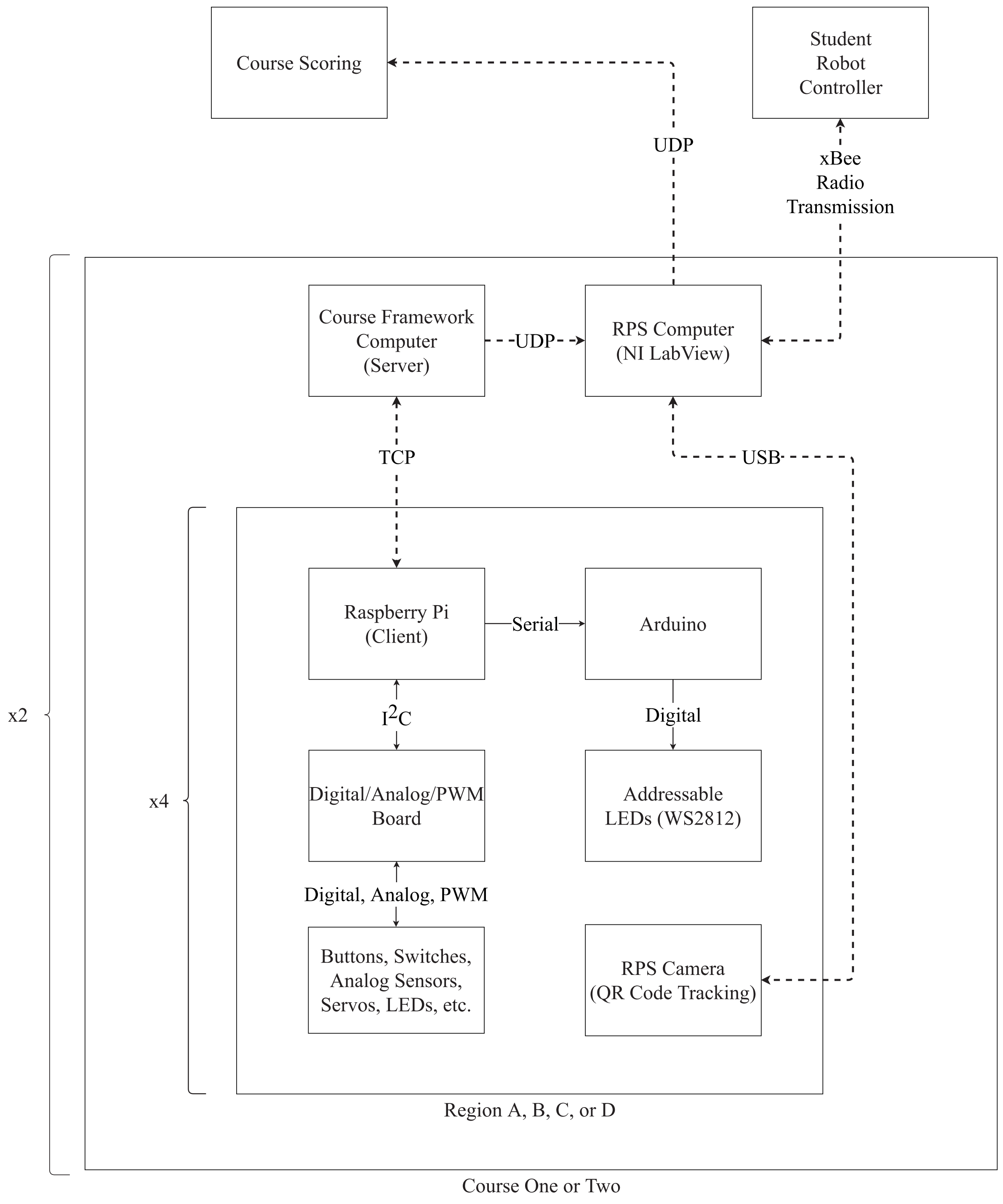

Figure 2: Robot course computer system architecture diagram.

Additionally, this TA team structure has benefits as it relates to the students. The modularity of the teams means that if a component fails in some way during the project, often in the moment while students are actively using that component in class or open lab, TAs who are specialized in that component can work to fix the issues much more efficiently and quickly. This helps reduce the impact on students. Also, the enhanced empowerment of the TAs often translates to greater interest in the success of the students, which leads to innovation in future component design choices. At the end of each project term, each team makes a list of future improvements based on the experience of the TAs and the students.

Student Experience

The modular design of the software system assists the instructional team with enhancing the experience students have during the design project. The ability for several development teams to work simultaneously ensures that the course is ready to be revealed to students early in the spring semester. Having a system of smaller hardware and software components makes it easy for new teaching assistants to join development teams and learn new technical skills. This has resulted in new ideas being integrated each year to improve the design project. When technical issues occur, development teams are more easily able to determine the cause of issues and correct them because the scope of a problem can be narrowed down to a specific software or hardware component. Minimizing course downtime ensures that students are not severely impacted when a system failure occurs. Furthermore, when students are unable to conduct robot testing on a course region, the modular system ensures that the other regions are available for testing.

Over the past two years, the program has utilized the course scoring software to log data on thousands of robot testing runs. The data were filtered to eliminate runs that were either less than five seconds or had zero points. The processed data start in late March of each year when students are consistently testing with the QR code on their robot, since taking data earlier would not guarantee student runs since some teams would not be using QR codes yet.

Figure 3 shows the number of runs conducted daily on each course region in 2017 and 2018. After filtering out runs based on the criteria above, 15,626 tests were conducted from March 21st to April 5th in 2017 and 12,596 tests were conducted from March 25th to April 4th in 2018.

It was previously stated that the microservices-style architecture provides a stable testing environment for students. Figure 4 shows instances where tests were being conducted while one or more regions were down. The graphs show the number of tests conducted on a region during each hour block. Places where points are significantly lower than the rest indicate that testing on the corresponding region was not occurring or was occurring at a significantly reduced rate. This demonstrates how the modular system of components minimizes the impact that software and hardware failures have on the students because they can test on other regions while one is not functioning properly. TA teams can then work on the hardware and software failures to fix them while students are still testing robots on other courses, leading to parallelization of work.

The reliability provided by this modular course computer system benefits students in several ways. Because the software provides a consistent and instant mechanism for them to receive feedback, they can focus on enhancing the design of their robot without having to worry about inconsistencies on how they are scored. The system also provides the robots with additional ways to interact and make decisions on the course because of the information transmitted to the programmable controller from the Robot Positioning System. Finally, when the students compete in the robot competition, the system ensures that all students are scored in the same manner and without discrepancy.

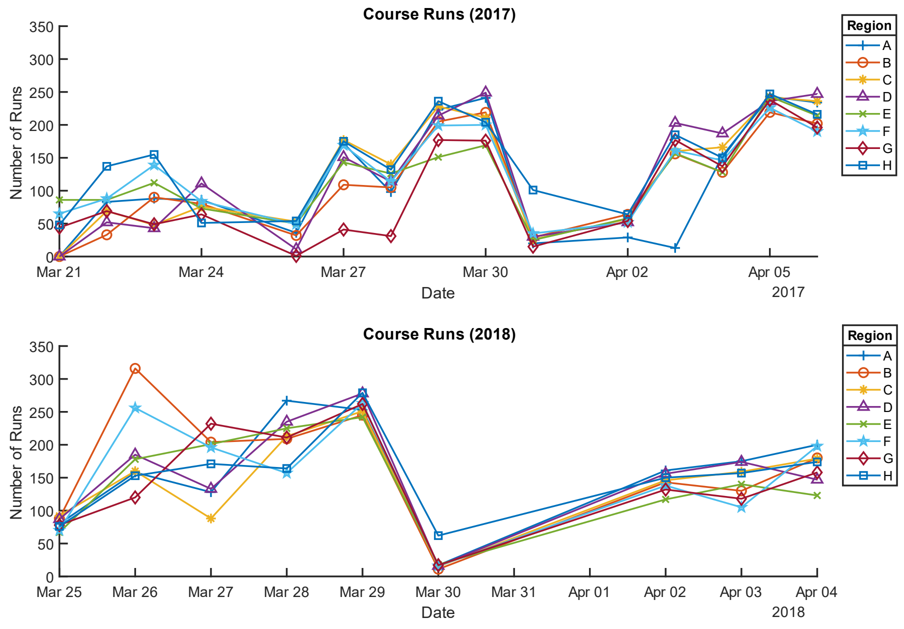

Figure 3: Total runs conducted on each course region by day from two successive years.

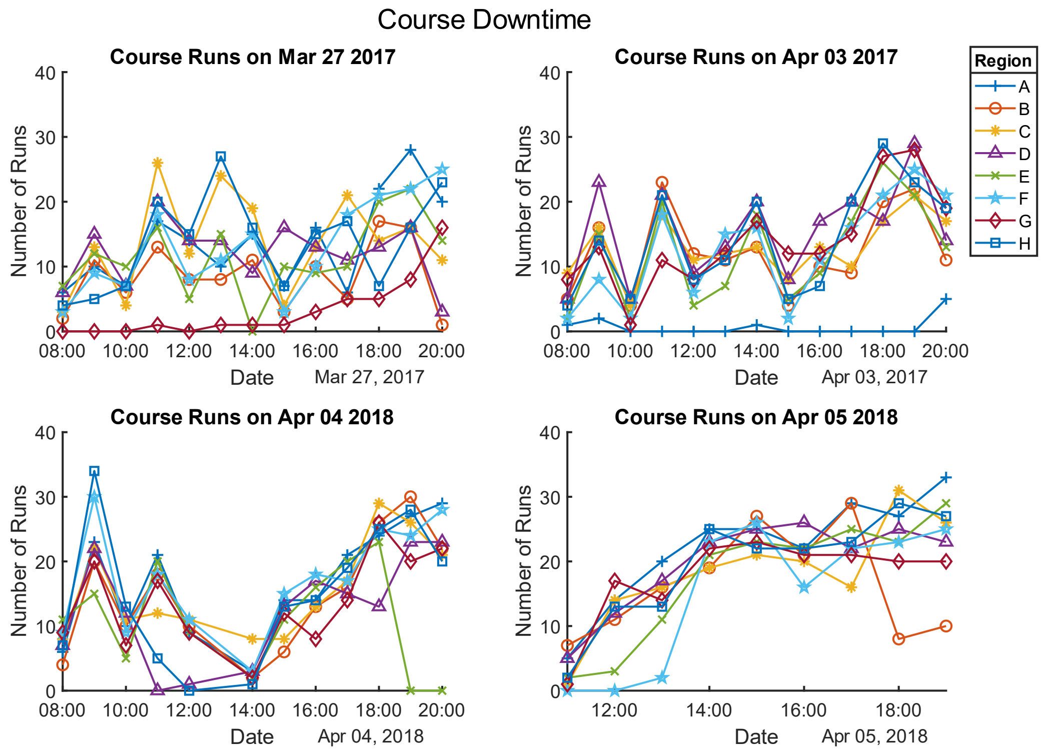

Figure 4: Selected dates of instances of course region failures and lack of impact on other regions.

Conclusion and Future Work

This paper has described the various computer hardware and software components utilized to run a complex robotics design project electronic course. Implementing these components as microservices in both the physical structure of the computer system and in the TA development team structure has shown great advantages in terms of modularity, specialization, and knowledge transfer. In addition, data were presented showing the limited impact of software or hardware failures on student experience with the robot courses due to the separation of concerns implementation. Students are able to continue working on their robots even while certain course regions or components are being repaired by experienced teaching assistants. This model may have implications for other first-year engineering programs which have large cornerstone design projects with many computer system components. In the future, more data from the students can be obtained to determine if the students notice the smoothing effects of the modular system.

References

1. Morin, B., Kecskemety, K., Harper, K., & Clingan, P. (2013), The Inverted Classroom in a First-Year Engineering Course. *2013 ASEE Annual Conference & Exposition Proceedings*. https://peer.asee.org/22605
2. Frank, D., Kolotka, K., Phillips, A., Schulz, M., Rigney, C., Drown, A., . . . Freuler, R. (2017). Developing and Improving a Multi-Element First-Year Engineering Cornerstone Autonomous Robotics Design Project. *2017 ASEE Annual Conference & Exposition Proceedings*. doi:10.18260/1-2--28143
3. Schulz, M., Danish, E., Leonhardt, T., Jackson, J., Frank, D., & Freuler, R. (2017). Modular System of Networked Embedded Components for a First-Year Engineering Cornerstone Design Project. *2017 ASEE Annual Conference & Exposition Proceedings*. doi:10.18260/1-2--28683
4. Frank, D., Witt, K., Hartle, C., Enders, J., Beiring, V., & Freuler, R. (2016). A Low-Cost Robot Positioning System for a First-Year Engineering Cornerstone Design Project. *2016 ASEE Annual Conference & Exposition Proceedings*. doi:10.18260/p.26355
5. Vernier, M., Wensing, P., Morin, C., Phillips, A., Rice, B., Wegman, K., . . . Freuler, R. (2014). Design of a Full-Featured Robot Controller for Use in a First-Year Robotics Design Project, *2014 ASEE Annual Conference & Exposition Proceedings*. https://peer.asee.org/20260
6. Harper, K., Zierden, H., Wegman, K., Kajfez, R., & Kecskemety, K. (2015). Teaching Assistant Professional Development Through Design: Why They Participate and How They Benefit. *2015 ASEE Annual Conference and Exposition Proceedings*. doi:10.18260/p.24806