HANDS-ON, WEB SERVICE BASED, SOFTWARE ARCHITECTURE LAB COMPONENT FOR SOFTWARE ENGINEERING COURSE

Crystal Cox and Michael A. Murphy
Computing Sciences Department
Coastal Carolina University
Conway, SC 29528
843-349-2144 crystal@coastal.edu, mmurphy2@coastal.edu

ABSTRACT

This paper describes a lab series component for learning multi-layer, web service based software architecture in an online, hybrid, or traditional format software engineering course. The labs complement the software engineering process and theory course topics with practical experience using current industry standard tools and technologies. No central infrastructure is required; students build the development/test environment from the ground up using free and widely used software, such as Apache Tomcat, Java, Jersey, and MySQL on a virtual machine. Survey results show that students see the labs as a valuable learning experience.

INTRODUCTION/BACKGROUND

Hands-on lab activities and practical experience with current tools and technologies is an important aspect of computer science and information systems education, as evidenced by the Accreditation Board for Engineering and Technology (ABET) suggested student learning outcomes. Outcome I from ABET's computing program criteria specifically addresses the "ability to use current techniques, skills, and tools necessary for computing practice" [1]. Addressing this desired outcome in advanced information systems and computer science courses with relevant hands-on activities in a hybrid or online delivery format can be especially challenging.

There are significant advantages to providing options for lab activities in a flexible format for traditional or online delivery, such as flexibility for students to work at their own pace and convenient times, and discussion forum participation from a broader range of students and a conversation history that other students can mine for tips. Additionally, this type of learning and online discussion is more similar to the way students will continue lifelong career learning. Other research has shown that lab activities can be as or more effective in an online format, but that hands-on learning is preferred regardless of format. Corter et al. found that remote and simulation labs, as opposed to traditional hands-on labs, resulted in equal or higher lab-related knowledge scores by undergraduate engineering students, but that students preferred the hands-on labs [3]. Interestingly, Breimer, Cotler, and Yoder compared video-based instruction to text-based instruction in an information systems computer lab course and found no difference in concept learning, task completion time, retention, and student impression [2].

Building from several semester-long iterations of experimenting with lab activities for hybrid and online courses, the authors have developed an effective, well-received series of text-based, hands-on labs for learning multi-layer web

service software architecture in a software engineering course, using freely available software. The labs are designed to be used in traditional in-person, hybrid, or online delivery courses. This paper describes the lab series and implementation experiences.

IMPLEMENTATION

The study setting is a four-year liberal arts university with approximately 10,000 students. The targeted software engineering course is shared by the computer science and information systems degree programs, which together have just over 300 students. Both degrees are offered only in a traditional, on-campus delivery format, but the authors have been allowed to experiment with hybrid (meets in-person one class per week) and fully online formats for individual courses within the program. These courses typically have 20-30 students each.

For software engineering and several other information systems courses, virtual machines have been employed to facilitate student learning by providing portable, customized execution environments with common versions of software available for student use. These environments were provided to the students in the form of downloadable appliances using the Open Virtualization Archive [4] format for use with the Oracle VirtualBox [6] desktop virtualization package. Since the students utilized a wide variety of personal laptop systems, with varying capabilities and varying operating systems, installation of a single virtualized environment was simpler than installation of multiple disparate pieces of software across different platforms.

A few challenges were experienced with the deployment of the virtual environments on student systems. Virtual machine performance was sometimes slow for the students using lower-end hardware, which sometimes lacked Hardware Virtual Machine [5] extensions to mitigate performance issues with sensitive instructions at the hardware level. The second, more minor, challenge was that updating the student computing environments post-deployment was difficult, resulting in environments that contained unresolved bugs for the duration of the semester. Work is underway at the university to provide students with remote access to virtual computing environments via a browser-based interface that is expected to work across multiple platforms, including mobile devices. These environments will be hosted on a virtualization cluster located on campus, with remote access provided to the environments via an interface incorporating the Guacamole remote display proxy [7]. Although there are some network latency and capacity challenges to be addressed, it is anticipated that access via this mechanism will resolve the performance issues experienced by students with less expensive computing hardware.

The Course

The software engineering course covers three broad topics: software engineering processes, software architectures, and advanced object-oriented programming. The prerequisite for the course is a CS2 course focusing on beginning data structures, searching and sorting, recursion, and object-oriented programming. It is important that this course transitions students from the primarily command-

line interface focus of their CS1 and CS2 courses to the n-tier architecture used throughout much of industry. The authors have chosen REST web services with Java for the practical application activities because of the increasing popularity and standardization of web services in the software industry, its simplicity (as compared to SOAP web services), the available support for programming REST web services in Java, and the fact that the students had taken CS1 and CS2 in Java.

The authors have taught the course six times in the past three years, in traditional, hybrid, and online formats. The same content has been covered each time, but the lab activities have evolved from in-class free-form activities to the current set of well-defined, self-contained labs as a result of student feedback, trial-and-error, and now-predictable troubleshooting issues.

The Labs

This lab series aims to provide hands-on activities for software architecture and advanced object-oriented programming. Over the course of the semester, with 14 labs and two pre-labs, students build a proof-of-concept recipe ingredient management system using RESTful web services in Java and a MySQL database. These labs are done as a practical application component of the course to complement the software engineering process and theory topics. The lab topics progress from setting up the virtual machine and development server environment, to implementing the various layers and components of the web application, to refactoring the application using design patterns. Students are encouraged and sometimes required to discuss the labs in online forums each week.

LAB SERIES PROGRESSION

Pre-Labs: Ingredient and Recipe POJOs (Plain Old Java Objects)

The first pre-lab requires students to recall their object-oriented programming skills from the previous semester to create Ingredient and Recipe classes. This exercise covers public/private modifiers, attributes and methods, interfaces and inheritance, the concepts of composition, coupling, and cohesion, and common scaffolding methods in Java classes. The second of these two pre-labs introduces/refreshes data structures from the Java Collection hierarchy, including lists, sets, and maps. For this pre-lab, they enhance the Recipe class by implementing a list of ingredients, map of ingredients and amounts, and set of tags along with methods for traversing the structures in order to calculate items such as total calories or total cost. These two pre-labs are not required for the remaining labs; a simpler set of classes is created later in order to keep the focus on other topics, but later labs or exams may require incorporating these more complete classes into the final web application.

Topics: object-oriented design, composition, interfaces, data structures

Labs 1-3: Virtual Machines and MySQL Server

The first two labs in the series set up the virtual environment, using Oracle's VirtualBox software. Students learn about virtual machines, server software and ports, and command line operation. The third lab installs MySQL Workbench, a graphical interface for the MySQL server. Students learn basic SQL for creating

tables, inserting data, and selecting data. During this lab, they create a basic Ingredient database table and populate it with test data to be used in the later labs.
Topics: virtual machines, servers and ports, database server, SQL

Labs 4-5: Java from the Command Line and JDBC

Since the virtual machines are configured with very basic levels of memory, IDEs like eclipse may be too resource-intensive to run efficiently. This opportunity is used to teach students the basics of compiling and running Java programs from the command line, which requires that they learn about system environment variables PATH and CLASSPATH. Lab 5 requires students to write and test a Java program to retrieve data from the MySQL database. Since it requires an extra JAR file for the MySQL driver, this is where they learn about third-party library JAR files.
Topics: command interface, system variables, coding without IDE, ISO files, JDBC, third-party JARs, programmatic database access

Labs 6-7: Setting up a Web Server and A First Web Service

Lab 6 installs the Apache Tomcat web server. Students learn basic web server administration, such as configuring ports, adding administrative users by modifying xml files, and starting/stopping the server program from the command line. They also set up a basic static website and learn about the server directory structure. After learning enough system administration for web application development, Lab 7 returns the focus to code, showing the new annotation syntax for turning a Java method into a web service. Students learn the basic structure of a Java web application and necessary components. The goal of this lab is just to get all the pieces in place to get a simple web service up and running. The next lab goes into more detail about RESTful web services and HTTP in general.
Topics: XML files, server administration, web services, web application structure

Labs 8-10: Calling Web Services and Passing Parameters from Javascript

Lab 8 covers basic characteristics of HTTP and RESTful web services. Students start working with specific method types and configuring the web services. They write small jQuery functions to build an HTTP request, using AJAX to receive the response and update the page using the Document Object Model (DOM). This lab also covers passing path and query parameters. Labs 9 and 10 bring together the JDBC experience from Lab 5 and the web service creation from Labs 7 and 8. The web service methods, which up to this point just returned mock data, are modified to build SQL statements and retrieve actual ingredient data from the database. The data is returned as just a concatenation of strings representing ingredients at this point, but later will be sent as an object. Students learn the security, performance, and maintenance benefits of PreparedStatement objects. Since mistakes during this lab are often server errors, students also learn how to use the server logs for troubleshooting. By the end of these labs, students see a web page with three buttons, each of which makes a call to a different web service with appropriate parameters, and displays the returned data from the database on the page.
Topics: HTTP method types, REST best practices, jQuery/AJAX, DOM, JDBC, PreparedStatement, server logs

Lab 11: Encapsulation, Marshaling, and JSON

In this lab, web service methods are modified to build Ingredient objects and then return their JSON representations. Students learn about marshaling objects for transport with the JSON format. They use Google's Java libraries for converting objects to JSON and Javascript libraries for converting the JSON string back into an object to be processed on the page.
Topics: JSON, marshaling, POJOs

Labs 12-13: Submitting Form Data and Receiving Response Objects

Previous labs have focused on retrieving data; Lab 12 introduces the submission of web form data for creating resources on the server. Students again study the REST architecture and the use of HTTP methods GET, PUT, POST, and DELETE for the common operations of retrieve, update, create, and delete (CRUD operations). They create a web form and build an HTTP request object in Javascript. In this lab, more web service annotations are introduced, such as @Consumes for specifying acceptable input data to the service. Lab 13 modifies the web service methods to return actual standardized HTTP response objects with appropriate status codes and the data objects embedded in the message body.
Topics: HTTP request and response objects and method types, more web service annotations, using form data, jQuery/AJAX, HTTP status codes, REST best practices

Lab 14: Design Patterns and Refactoring

Lab 14 is all about refactoring the current codebase to achieve better separation of concerns and lower coupling. Students learn about design patterns in general, and then practice using the Singleton and Façade patterns. The Singleton pattern is used to encapsulate the database connectivity, while the Façade is used to gather the database interaction code and simplify the process of retrieving and creating ingredients from the perspective of the web service.
Topics: refactoring, MVC, design patterns, singleton, façade, code quality

RESULTS

Students in the Fall 2014 software engineering course and students in the optional follow-up software engineering II course were surveyed with the following questions using a 5-point Likert scale.

- Q1: The labs helped me understand multi-layered software application architecture.
- Q2: The labs gave me practical experience in developing advanced web applications.
- Q3: The labs helped prepare me for job interviews and possible my next job.
- Q4: The labs helped me improve my independent learning skills.
- Q5: The labs provided a valuable learning experience.

The results from both classes showed that over 90% of students agreed or strongly agreed with questions 1, 2, 3, and 5, while the rest were neutral. On question 4, approximately half responded agree or strongly agree, and half neutral.

CONCLUSIONS

Overall, this lab component has been successful at helping students gain experience with current tools and technologies for multi-layer web service based software applications. Although they could be used in any delivery format, students appreciate the hands-on learning experience in the online, self-paced environment, especially when they have an option to come in for in-person help if/when necessary.

One very interesting and positive side effect of the online setting was noticed in the discussion forums. Students were encouraged to use the online discussion forums to ask for help troubleshooting errors they encountered while doing the labs. They were also encouraged to offer advice when they could, based on their own experience with the lab. At the beginning of the semester, students made posts of the nature "My service doesn't work—what am I doing wrong?", which obviously is not enough information to get useful help. By the end of the semester, the same students were posting much more detailed descriptions of their problems, including specific errors, code samples, log files, and details of steps they had already taken. This improvement directly supports ABET's student learning outcome of the recognition for and ability to engage in lifelong learning, as it shows that students are better prepared to engage with other programmers in online forums to collectively troubleshoot and solve problems. This level of collective problem solving and personal contributions did not occur when the class was taught in a traditional setting.

REFERENCES

[1] ABET Criteria for Accrediting Computing Programs 2014-2015, http://www.abet.org/cac-criteria-2014-2015/, retrieved November 28, 2014.

[2] Breimer, E., Cotler, J., Yoder, R. Video vs. text for lab instruction and concept, *CCSC Northeastern Conference,* 2012.

[3] Corter, J., Nickerson, J. Esche, S., Chassapis, C., Seongah, I., & Jing, M. Constructing reality: A study of remote, hands-on, and simulated laboratories. *ACM Transactions on Computer-Human Interaction, 4(2), article 7*, 2007.

[4] Crosby, S., Doyle, R., Gering, M., Gionfriddo, M., Hand, S., Hapner, M., & Wilson, J. Open virtualization format specification. *Standards and Technology, no. DSP0243 in DMTF Specifications, Distributed Management Task Force*, 2009.

[5] Kivity, A., Kamay, Y., Laor, D., Lublin, U., & Liguori, A. kvm: the Linux virtual machine monitor. In *Proceedings of the Linux Symposium*, vol. 1, pp. 225-230, 2007.

[6] Li, P. Selecting and using virtualization solutions: our experiences with VMware and VirtualBox. *Journal of Computing Sciences in Colleges*, *25*(3), 11-17, 2010.

[7] Mulfari, D., Celesti, A., Villari, M., & Puliafito, A. Using Virtualization and Guacamole/VNC to Provide Adaptive User Interfaces to Disabled People in Cloud Computing. In *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)* (pp. 72-79). IEEE, Dec. 2013.