Disseminating Learning Tools Interoperability Standards

Hamza Manzoor

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Clifford A. Shaffer, Chair
Stephen H. Edwards
Mohammed Seyam

May 13, 2019
Blacksburg, Virginia

Keywords: computer science education, learning tools interoperability, tutorials,
auto-grading, Jupyter notebooks

Disseminating Learning Tools Interoperability Standards

Hamza Manzoor

(ABSTRACT)

Until recently, most educational tools have worked in silos. If a teacher wanted her students to complete small programming exercises, record videos, and collaborate through discussion boards, three disconnected tools were probably needed. Learning Tools Interoperability (LTI) is a communication protocol that enables different learning tools to talk to each other and share scores with a Learning Management System (LMS). While most commercial LMS now support LTI, most educational software developed by small research efforts do not. This is often because of the lack of resources needed to understand the working of LTI and the process of using LTI in their applications. Our aim is to encourage the use of LTI within the CS Education community. We have developed tutorials that include example applications. We also provide a use case of how LTI is implemented in the OpenDSA eTextbook system. As another use case, we have enabled auto-grading of Jupyter Notebook assignments by providing immediate feedback to students and updating scores to the Canvas gradebook. We provide a Jupyter plugin to upload notebook files to the Web-CAT auto-grading system. We integrate Aalto University's ACOS content into OpenDSA as a third use case.

Disseminating Learning Tools Interoperability Standards

Hamza Manzoor

(GENERAL AUDIENCE ABSTRACT)

Until recently, most educational tools have worked in silos. If a teacher wanted her students to complete small programming exercises, record videos, and collaborate through discussion boards, three disconnected tools were probably needed. These disconnected tools did not integrate with the Learning Management Systems (LMS), such as Canvas and Moodle. Instructors had to manually manage these separate tools and enter scores into the LMS. There are standards such as Learning Tools Interoperability (LTI) that these learning tools can implement to enable them to talk to each other and to share scores with an LMS. However, most educational software developed by small research efforts do not support LTI. This is often because of the lack of resources needed to understand the working of LTI and the process of using LTI in their applications. We aim to encourage the use of LTI within the CS Education community. We have developed tutorials that include example applications. We also provide a use case of how LTI is implemented in OpenDSA, an eTextbook system developed at Virginia Tech. As another use case, we have enabled auto-grading of Jupyter Notebook (documents that run in a browser and can contain equations, visualizations, live code, and text) assignments by providing immediate feedback to students and updating scores to the Canvas gradebook. We provide a plugin to upload notebook files to the Web-CAT auto-grading system directly from the browser. We integrate Aalto University's ACOS content (Python and Java exercises) into OpenDSA as a third use case.

Dedication

In memory of my grandmother who passed away in January 2019.

Acknowledgments

First of all, I would like to thank Allah for his countless blessings on me.

I would also like to thank my advisor, Dr. Clifford A. Shaffer for his guidance and trust in me throughout my time at Virginia Tech. I would also like to thank my committee members, Dr. Stephen H. Edwards, and Dr. Mohammed Seyam.

All of my lab members have been of great help and I would like to thank them. Especially, I would like to thank Ayaan, Mostafa, Jieun, Thomas, Tyler and Jackson for their support, guidance, and help throughout my grad school life.

I would like to show my appreciation to Technology-enhanced Learning and Online Strategies (TLOS) for supporting the development of LTI Tutorials and auto-grading of Jupyter Notebooks. I would also like to thank National Science Foundation (NSF) for supporting my work through grants DLR-1740775, DLR-1740798, and DLR-1740765.

Finally, I would like to thank my parents, brothers and family members who have always believed in me and supported me through every thick and thin. I wouldn't be where I am today if it weren't for them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A common problem with educational software is that most of these tools do not talk to other tools. As an example, a teacher might want to add small programming exercises to her course. She might also have students record video submissions and collaborate through discussion boards. There could be a different tool for doing each of these tasks. While many Learning Management Systems (LMS) try to provide their own tools for some of these features, they are often inferior to tools built for the purpose, like dedicated forum system. Since these tools are not built within the instructor's LMS, they typically do not integrate with the LMS. And if they do, they still do not provide as much data to instructors in an integrated way as tools built within the LMS might. The answer certainly is not to wait for LMS developers to provide such features themselves. The answer is to let the various instructional tools cooperate with each other.

1.1 Motivation

When educational tools do not talk to each other, it is hard for instructors to make these tools work from one common place such as the LMS. This results in one of the following two scenarios:

1. Instructors opting not to use the tools, hindering the use of the best tools for improving

the learning process.

2. If the instructor decides to use the tools, she will then have to deal with the following problems:

    - Ask students to create separate accounts on each of the external tools she is using in her course.
    - Manage tasks such as assignment creation separately on each of the external tools.
    - Students will have to navigate between various tools for different tasks.
    - She must manually enter the scores of each student for each task into the LMS.
    - Click-stream data is not shared from the external tools to one central place, and hence that cannot be used to further improve the learning experience in a comprehensive way.

Domain-specific communication standards can potentially resolve these issues. If the service consumer (LMS) and service providers (educational tools) use shared standards, they will be able to communicate with each other.

The Learning Tools Interoperability (LTI) standard enables different learning tools to talk to each other and share data. LTI is a specification developed by IMS Global Learning Consortium to establish a standard way to integrate remotely hosted learning applications with the LMS. Most major LMS now support LTI. If an external tool has implemented LTI, then these LMS can consume scores and other data. This will help to resolve problems, for example:

1. Students will not have to create new accounts and navigate between different tools. They will be able to find all the course-related resources from within the LMS.

2. Teachers will not have to manage various tools separately.

3. Teachers will not have to manually keep track of the scores of each student on each task within each tool. The scores will be automatically entered in the gradebook of the LMS.

Though this standard has been around since 2010, many educational tools, especially the cutting edge tools created within small research groups, do not support this standard. It is often because of the lack of resources needed to understand the working of LTI and the process of using LTI in an application.

Similarly, Caliper [6] is another specification developed by IMS Global Learning Consortium. The purpose of Caliper is to define a standard to measure and share learning data. LTI enables external tools to connect to an LMS and report scores, but it doesn't allow sharing of click-stream data to one common place. If the learning applications implement Caliper as well, they will be able to share their collected learner analytics data to one common place, which can be used by instructors to improve the learning process.

1.2 Research Objectives

Our aim is to provide resources about LTI to educational tools developers. We do this both by providing tutorials and by developing useful integrations as use cases.

Our first step was to create a collection of tutorials about LTI for developers of learning applications, to help them understand and incorporate the standard. We developed tutorials that include example applications. We have also implemented several integrations. One use case shows how the OpenDSA eTextbook system [36] implements LTI. A second use case involves Jupyter Notebooks, which are becoming popular in academia for Python programming as-

signments and in the industry among data scientists [34]. Therefore, we wanted to enable the auto-grading of Jupyter Notebooks and submission of scores to an LMS for students. Our third use case involved adding the ACOS [2] exercises developed by Aalto University in OpenDSA, an e-Textbook system developed at Virginia Tech. This shows that different researchers can collaborate and use the resources developed by each other. OpenDSA does not have Python animations, and ACOS provides python exercises and animations. Similarly, we extended contents served by Mastery Grids [32], an open-source progress visualization environment which has open (social) learner model features designed at the University of Pittsburgh, by integrating CodeWorkout [20] small programming problems through LTI. This also demonstrates the ability to reuse materials developed by other researchers.

Chapter 2

Background

An LMS is a complex system that continuously evolves. Dagger et al. [22] discuss the evolution of eLearning platforms. They mention that the first generation of eLearning platforms were monolithic systems and used proprietary formats. Thus, they did not allow the sharing of content with other platforms. The first versions of Web-CT (1996) [25] and Blackboard (1997) [5] are examples of the initial eLearning platforms. The second generation of eLearning platforms was more flexible and allowed contents to be imported in LMS which followed certain standards. The initial version of SAKAI [15] launched in 2005 is an example of the second generation eLearning platform. SAKAI used web services to expose the limited functionality of the LMS [22]. The current generation of LMS became more modular and extendable. Dagger et al. describe the approaches used by various LMS to make their systems more extendable. Some LMS provide consumers the flexibility to extend the system by developing extensions or widgets. A number of standards have also emerged that support interoperability of learning tools with an LMS.

Some LMS were built with a component-oriented approach. The component-oriented approach allows implementing loosely coupled individual components into the system. Though this approach improves the scalability of the system, it limits the components to be developed with compatible technologies only. Open edX [12] is an example of an LMS which is built with a component-oriented architecture, called XBlocks [8]. Each XBlock provides a different feature, and Open edX is made up of various XBlocks. XBlocks limit the content

developers to implement using Python.

A few LMS were built with a Plug-in architecture, where each plug-in adds a new feature without modifications to the other features of the system. A plug-in based approach is more flexible than a component-oriented approach, as it allows third-party developers to create features that extend the system. However, the plug-in architecture still requires that the plug-ins be implemented using the same technologies as host systems and requires access to the database of the host system.

Google Gadgets [9] and OpenSocial [26] are examples of widget-based systems. A widget is defined as a portable application, typically packaged in a Zip archive, and implemented using HTML, JavaScript, and CSS. The JavaScript code of the widget is used to make calls to a set of standard APIs for making use of the services offered by the container [39]. Google and W3C provide specifications for widgets, and both specifications differ in meta-data and API specifications. The widget-based systems solve some of the problems faced by previous architectures, but they are still not very flexible. For example, in [39], Wilson et al. created widgets for collaboration features like chatting, voting, and a discussion forum in Moodle but the authors had to make modifications to W3C specifications. Also, there is no authorization specification defined. Therefore, the automatic login of students is not possible.

There exist standards to promote interoperability of learning tools. The Sharable Content Object Reference Model (SCORM) [19] standard by the Advanced Distributed Learning (ADL) [4] is a set of specifications defined for creating and sharing learning content. SCORM defines how the learning objects should be packaged to make them interoperable with a SCORM-compliant LMS. Though SCORM allows interoperability on learning tools, the last major update of SCORM was in 2004 and it is dying [16]. It also limits the content and requires eLearning tool developers to develop content mostly in Flash through SCORM authoring tools [33] because only the content designed in SCORM specifications can be

imported in the supported LMS [14]. Also, the licenses of the major authoring tools that publish content in the SCORM format such as Adobe Captivate [21] and Articulate Storyline [38] are expensive for small research groups.

Learning Tools Interoperability (LTI) [35] is an interoperability standard, developed by IMS Global Learning Consortium. The purpose of LTI is to establish a standard way to integrate remotely hosted learning applications within an LMS securely. LTI does not restrict external tool developers with a fixed way of developing tools. It allows them to build their tools however they like as long as those tools support the LTI API. LTI defines the connectivity standards and submissions of scores to the calling party, which in most cases is an LMS. The flexibility that LTI provides solves the interoperability issues that were present in the other standards and architectures. At the point of writing, most major LMS support LTI. Since our goal is to provide resources using the standards which are more commonly adopted, LTI seemed to be the best option. We have developed a tutorials website to provide resources for LTI. These tutorials are available at http://splice.cs.vt.edu/lti.

Chapter 3

LTI Tutorials

3.1 Background

Most learning institutes across the globe now use LMS such as Moodle [1], Canvas [11], or Blackboard [5] for content delivery and collaboration. These LMS typically come with a set of tools to support content delivery and collaboration. Apart from these built-in tools, many education providers have developed their own task-specific tools to improve the learning experience. These tools are called external tools because they are not part of the default installation of the LMS.

A common problem is that most of these external tools do not talk to any other tools. As an example, a teacher might want to add small programming exercises to her course. She might also want students to record video submissions, and to collaborate through discussion boards. Unless the LMS has its own tools for these specific tasks, the teacher must find external tools to support them. There do exist different tool for doing each of these tasks. However, since these tools are not built within the instructor's LMS, they typically do not integrate with the LMS.

3.2 Why do we need standards?

There are many task-specific third-party learning tools. For example, CodeWorkout [20] is a tool developed at Virginia Tech. It lets students answer and get credit for small programming exercises. As a third-party tool, CodeWorkout is hosted on Virginia Tech's servers. If computer science instructors want their students to solve programming exercises, they have two options:

1. Use CodeWorkout or an equivalent third-party tool.

2. Create their own tool similar to CodeWorkout.

Though using CodeWorkout seems like a reasonable option, if it does not integrate with local LMS then it is not straight forward for instructors and students to use. The first problem is that the instructors will first have to create a course and workouts within CodeWorkout. After that, they will have to ask students to create their accounts on CodeWorkout, and then students will have to enroll themselves to the course. Once students have done that, they will solve the programming exercises on the CodeWorkout website. Finally, the instructor or the teaching assistant will have to look at each of the scores, and they will need to enter those scores to the local LMS manually.

The second problem with this approach is that all the data (clickstreams, workout sessions, errors, etc.) stays within CodeWorkout, and the instructors do not have any picture of how students perform on a specific task except the final grades scored on the assessments. Instructors might want to know if all or some specific students are struggling on a particular topic so they can help those students understand that topic better.

To circumvent these issues, the other option is the typical "reinventing the wheel" scenario i.e. create a tool similar to CodeWorkout so that the instructors can have access to the

data. This seems absurd because CodeWorkout is just one tool, but if instructors want to use several tools in their courses, then how many tools can each instructor re-create?

If there are standards that allow connectivity of various tools seamlessly, both of the above-mentioned problems would be solved. For example, LTI enables various tools to connect with the LMS. So, in our example, LTI will address the first problem by allowing instructors to embed the CodeWorkout exercises within the LMS. Students will not have to create their new accounts on CodeWorkout website separately, and the instructors will not have to manually enter the scores back to the LMS, because LTI takes care of all of that.

To solve the second problem of not having access to the data, a standard called *Caliper Analytics* [6] can be used. Caliper Analytics allows institutions to collect the learning data from external tools through defined standards. If all tools implement this standard, they can share data with interested parties. In our example, if CodeWorkout implements this standard, instructors will be able to get the data from CodeWorkout and analyze where their students are struggling and help them accordingly.

The Experience API, also known as xAPI or Tin Can API is another specification that enables the collection of data about the learning experiences [31]. It allows the collection of data from various devices such as mobile and desktop computers. Unlike Caliper, it also allows the collection of data that a student has achieved offline on their mobile devices [31]. xAPI records the learning experiences in Learning Record Stores (LRS). These LRS can exist on their own or within an LMS [17]. There are notable similarities between xAPI and Caliper, and there is no clear distinction between the two [3]. However, a growing number of edtech companies are signing on to use Caliper [23]. If the learning applications implement Caliper or xAPI, they will be able to share data.

The above examples provide motivation for using standards. Fortunately, IMS Global Learn-

ing Consortium has already developed these standards. However, the problem is that these standards have been around for a while now, but most smaller learning tool developers have not implemented them in their tools. Therefore, we have focused our efforts on how to help such developers to adopt these standards. For learning tool developers, we have created sample applications and provided tutorials to illustrate the use of LTI. As an example, we show how OpenDSA has adopted LTI. For educators, we explain how there are many learning tools out there that they can use to improve their courses and add to their LMS.

In the next two sections, we explain what these two standards are.

3.3 What is LTI?

Learning Tools Interoperability (LTI) is a specification developed by IMS Global Learning Consortium. The purpose of LTI is to establish a standard way to securely integrate remotely hosted learning applications with platforms like LMS and similar educational environments.

Learning applications are called *Tools*, and they are delivered by *Tool Providers*. These are things like interactive exercises that a student might do for a grade.

The LMS (systems like Canvas, Moodle, etc) is called a *Tool Consumer*. At the moment, Canvas is the LMS that is most advanced in its support for LTI, since Instructure is a major participant in the IMS consortium. Other LMS such as Moodle also support LTI, and instructors can use the techniques described in our tutorials to integrate tools with other LMS.

The LTI specification enhances the functionality provided by a Tool Consumer and provides instructors with various applications that they can embed into their course. For example:

1. CodeWorkout is a tool developed at Virginia Tech. It lets students answer and get

credit for small programming exercises. As a third-party tool, CodeWorkout is hosted on Virginia Tech's servers. Canvas (or another LTI-compliant LMS acting as a Tool Consumer) integrates CodeWorkout seamlessly as a Canvas Assignment embedded in a Canvas page. This enables students to work coding exercises from within Canvas in the same way that they might do any Canvas-native assignment, and receive a score in the Canvas gradebook for doing that assignment.

2. The Wikipedia LTI application lets instructors search through English Wikipedia articles and link to or embed these articles into course material.

3. The Piazza LTI application adds a link to Canvas course navigation for Piazza discussions and auto-logs the user into the piazza discussion area from the LMS.

3.3.1 How does it work?

LTI enables the tool consumer to send a user to the tool provider in a trusted way. The trust assertion allows the user to be automatically signed in to the tool provider and directed to the specific content that is being provided by the tool provider.

Launch: The LMS (consumer) launches the tool (provider) in an iframe, so it feels like a part of the LMS.

Security: LTI Authentication uses the OAuth standard, which is a secure protocol over HTTPS for communication between different systems. The consumer and a provider share a consumer key and shared secret via an HTTP POST request. They are used to sign any messages passed between the two systems. The signature is done through OAuth so that either party can verify the signatures.

3.4 Caliper

Learning data or "learning analytics" is generated when students interact with online learning tools. These data can be combined with student records and other data points to support student success. These data can be used by faculty for proactive intervention with a student, or to infer how people use the tool and thus can help to improve the learning process. Instructors can also see which behaviors and content produce the desired learning outcomes.

Caliper is a specification developed by IMS Global Learning Consortium. The purpose of Caliper is to define a standard to measure and share learning data. Historically, separate data has been captured by each learning application. One result is that each organization needs to reinvent the analytics wheel. A separate problem is that these different data sources cannot be merged and therefore, none of the parties have a complete picture. Thus, a standard way of capturing and sharing learning data enables more efficient development of learning analytics features in learning environments. Caliper attempts to overcome these problems by defining a common language for labeling learning data and provides a standard way of measuring learning activities [10].

Though we did focus our efforts on LTI, we also provided a basic introduction for Caliper on our Tutorials website.

3.5 What we did

Previous sections explain that standards are essential for educational tools to interoperate and share data with each other, and that LTI and Caliper are good standards which solve the problems of interoperability and data sharing. If all learning software developers use these standards, educators can use tools easily in their courses, and these tools can share

their data with the interested parties.

These standards have been available for several years, but most of the smaller learning software developers either do not know about these standards or they do not have enough resources to adopt these standards. This could be in part because there are no complete and easy to follow guides or tutorials about how to implement these standards in existing or new learning tools. Documentation by IMS Global Learning Consortium is not easy to follow. We decided to address the need and provide tutorials and sample applications to developers for them to understand LTI. Figure 3.1 shows how the tutorial website looks. In the following sections, we briefly explain the contents of the tutorial website.

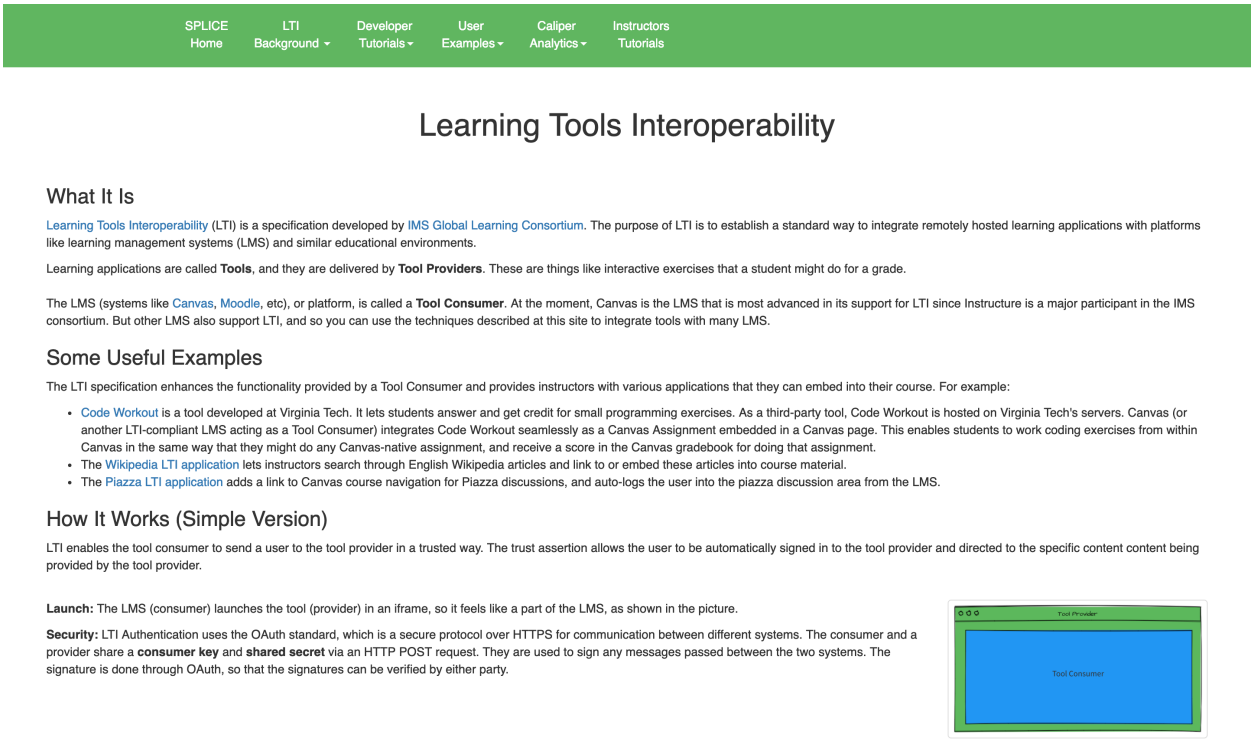


Figure 3.1: SPLICE LTI tutorials homepage.

3.5.1 LTI Introduction

We provide an introduction to LTI, which includes what it is and associated terminologies, such as *Tool Provider*, *Tool Consumer*, and *LMS*. We also provide useful examples of tools to reiterate the importance of LTI. Finally, we provide a simplified version of how LTI works. The LTI introduction in Section 3.3 is from the LTI introduction section of the website.

3.5.2 Developer Tutorials

The developer tutorials section is where we provide resources for developers of learning applications on how to make their tools LTI compatible. Following are the resources we provide under developer tutorials.

3.5.2.1 Building an LTI Tool Provider

This section explains the necessary steps required to build an LTI Tool Provider. Any tool can be converted to an LTI tool provider by following these steps. Here we are briefly describing those steps.

1. **Setup a Launch URL:** The first step is setting up an endpoint that will accept an HTTP POST message from a Tool Consumer. This endpoint has to verify that the received request is an LTI request or not.

2. **Authenticate the request:** The next step is to verify the authenticity of the tool consumer. All LTI requests are signed through OAuth to ensure that the data is not altered en-route to the tool provider and the request is not a duplicate request. To verify the authenticity, the request should have all the required OAuth parameters

3. **Launch:** After the verification of the request, the next step is to launch the tool. If the user session is required, then a user session can be created using a *user_id* field received in the POST request. After that, the tool should be configured to launch.

4. **Provide Configuration for Tool Consumer:** Tool provider has to provide the configuration so that it can be added within a Tool Consumer. Therefore, an XML configuration which includes the *launch_url* and other parameters should be provided.

5. **Tool Consumer Registration:** Tool provider needs to provide a procedure for instructors to register and get their key, secret and a URL to the XML configuration. Therefore, the next step is setting up a method for registration which will provide the key and a secret.

6. **Get SSL Certificate:** Tool Provider needs to have a secure connection for it to open in an iframe. Therefore, getting an SSL certificate is the next step.

We explain these steps in detail on our tutorials website. If any tool implements these steps, it will be ready to be used by any tool consumer.

3.5.2.2 Rails LTI Tool Provider

This section provides the steps of creating an LTI tool provider in Ruby on Rails. We created a sample stand-alone calculator Rails application and provided the code for that on GitHub. We then walk through the steps on converting that into an LTI Tool Provider. Since it is only a calculator and does not need to send scores back, we only show the basic steps mentioned in the above section. This 13 step tutorial describes using the Rails LTI gem, which helps to perform most of the LTI tasks such as validating and authenticating LTI requests. At the end of the tutorial, the user will have a working LTI-compliant calculator

application running in an LMS. We also provide the complete source code of the LTI version of the same calculator application.

3.5.2.3 Rails LTI Tool Provider - Send Scores back

This section provides a more advanced tutorial for creating an LTI tool provider in Ruby on Rails, which sends scores back to the tool consumer. We created a sample stand-alone Quiz application in Ruby on Rails and provided the code on GitHub. This 17 step tutorial shows how to use the Rails LTI gem to perform most of the LTI tasks, such as validating and authenticating LTI requests. It shows the steps to convert our Quiz application into an LTI Tool Provider, which also sends scores back to the tool consumer. Following these steps, the user can take the stand-alone Quiz application or their own Rails application and convert that into an LTI Tool Provider. At the end of this tutorial, the user will have a working LTI-compliant Quiz application running in an LMS which, after completing the questions, sends the scores back to the tool consumer. We provide the complete source code for the application.

3.5.2.4 Generating an SSL Certificate

The LTI tools have to be running over HTTPS for them to be served by the tool consumers. This section of the tutorial explains the steps required to generate an SSL certificate on a local machine, allowing a developer to test their locally running LTI tool. The tutorial has 6 steps to set up an SSL certificate on local machines.

3.5.2.5 LTI in OpenDSA - Stand-alone Exercises

The OpenDSA eTextbook system is an LTI-compliant content provider, allowing it to expose materials to any LMS that acts as an LTI consumer, such as Canvas or Moodle. To further illustrate the use of LTI, we explain how OpenDSA implements LTI. OpenDSA implements LTI for three different scenarios.

1. Present individual exercises and visualizations within an LMS.
2. Present exercises and visualizations from an integrated book instance.
3. Allow instructors to add exercises and visualizations to their courses.

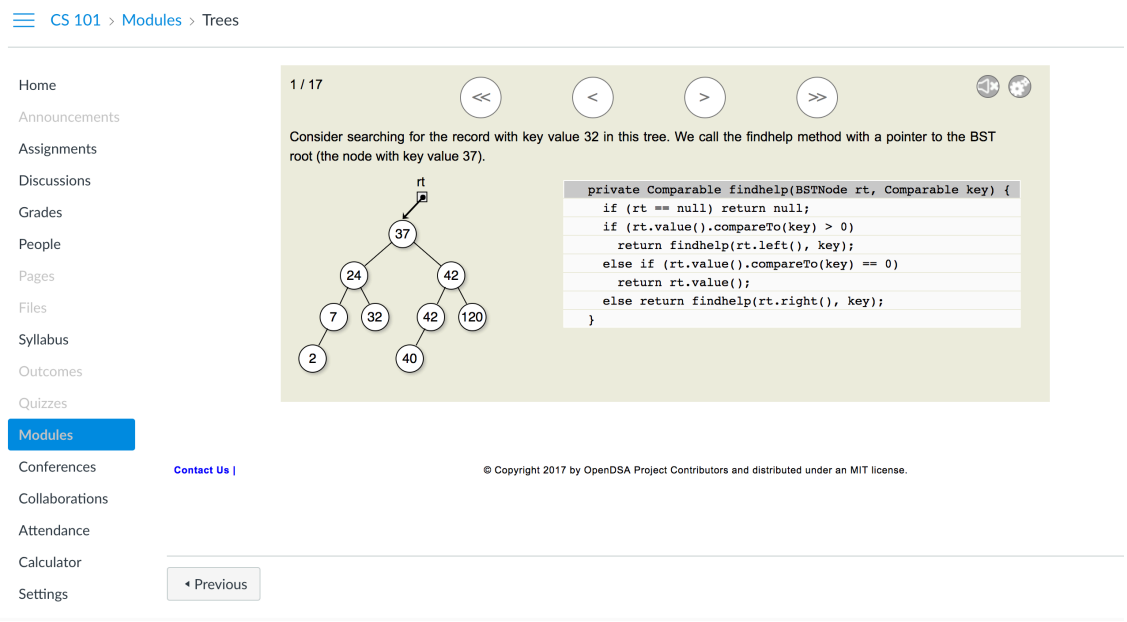


Figure 3.2: An OpenDSA exercise within Canvas.

This tutorial details how OpenDSA supports a student who sees an individual exercise or visualization from within the LMS. Figure 3.2 shows what a student might see from within an LMS. The *launch* endpoint is called when a student accesses an exercise from within

an LMS. In the case of individual exercises, the *launch_ex* endpoint is called from within *launch*. This tutorial explains the *launch_ex* endpoint. Within this endpoint, OpenDSA first finds the course offering to which this assignment belongs. After that, OpenDSA calls the *lti_authorize!* method to validate the received request. This method returns true if the received request is a valid LTI request. Then, OpenDSA calls *ensure_user* to check if a user with the same email exists or not and creates a new user if that email is not found. Finally, OpenDSA signs in the user to start her session. Starting the user session is followed by a call to *lti_enroll* with the course offering object, which enrolls the user in this particular course. Once OpenDSA validates the request and enrolls the user, it finds the current exercise using the short name and displays that exercise to the user. When a student completes an exercise, OpenDSA sends a new request to the *assessment* endpoint from JavaScript. The assessment method would send a score back to the Tool Consumer if this exercise was launched as an assessment.

3.5.2.6 LTI in OpenDSA - Book Instance Exercises

Along with adding the individual exercises, OpenDSA also allows instructors to create books on its website from pre-built modules. The modules used in those books are automatically created as Canvas modules and assignments using the Canvas API. Those modules are linked to a book instance in OpenDSA, and therefore they are handled slightly differently from the individual exercises. This tutorial explains how OpenDSA handles the LTI requests for book instance exercises.

When the content of OpenDSA is added into a course and a student clicks on an OpenDSA exercise, the Tool Consumer sends an LTI request to OpenDSA's *launch* endpoint. This endpoint validates if *custom_inst_book_id* is available or not. If this custom parameter is available, it means that the OpenDSA book instance was created on OpenDSA's website and

modules were created in Canvas using the Canvas API. In that case, this exercise belongs to an OpenDSA book instance. OpenDSA finds the book instance from the custom book instance ID received in the request and fetches the course offering of this book instance. In OpenDSA, a course offering uniquely identifies a course. After that, OpenDSA calls the *lti_authorize!* method to validate the received request. This method returns true if the received request is a valid LTI request. Then, OpenDSA calls *ensure_user* to check if a user with the same email exists or not and creates a new user if that email is not found. Finally, it signs in that user to start her session. Starting the user session is followed by a call to *lti_enroll* with the course offering object, which enrolls the user in this particular course. Once the request has been validated, and a user has been enrolled, OpenDSA reads the HTML content of the exercise, which is shown to the user in an iframe within the tool consumer. When a student completes an exercise, OpenDSA sends a new request to the *assessment* endpoint, which sends a score back to the Tool Consumer if this exercise was launched as an assessment.

3.5.2.7 LTI in OpenDSA - Instructors

This tutorial explains how OpenDSA makes use of LTI to support instructors who wish to add individual OpenDSA visualizations or exercises to their courses. When an instructor wants to add OpenDSA content, he sees a list of exercises that he can add, as shown in Figure 3.3.

OpenDSA handles this process, starting from showing the list of exercises to returning the link of the selected exercise back to the tool consumer within the *resource* endpoint. OpenDSA checks that this is a known LMS instance by calling the *ensure_lms_instance* method. For example, Virginia Tech's Canvas instance (canvas.vt.edu) is different from Instructure's public Canvas instance (canvas.instructure.com). This information is used to

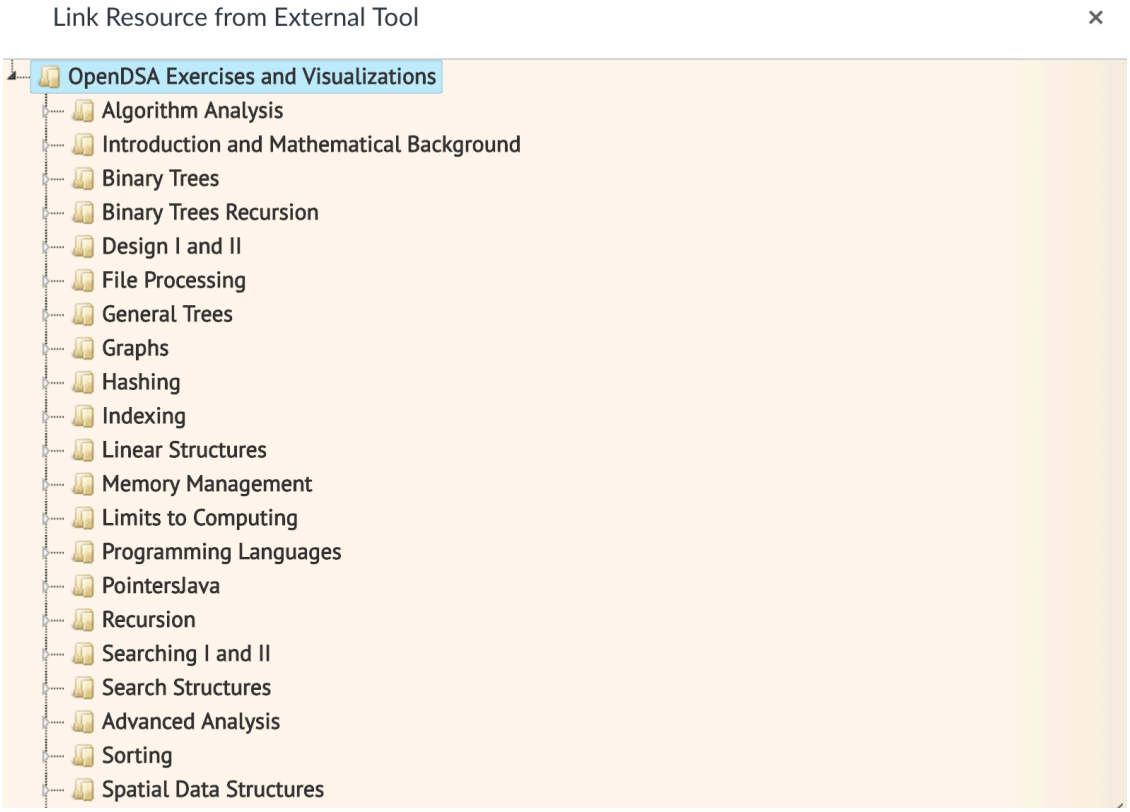


Figure 3.3: The list of OpenDSA exercises that can be added to a course within the LMS

create the correct URL that will be embedded in the exercises at the end. Once the LMS instance is fetched, OpenDSA fetches the course offering using the LMS instance ID and course number received in the request. Then OpenDSA creates a launch URL and validates if the request received from the Tool Consumer is an authentic LTI request or not by calling the *lti_authorize!* method. After validating the request, OpenDSA returns to the *resource* endpoint and signs in the user in the next step. Finally, OpenDSA fetches the names of all exercises and shows them to the instructor in a tree view.

3.5.3 User Examples

The tutorial website includes user examples to help instructors improve their courses and teach more effectively. Following are the resources we provide under user examples.

3.5.3.1 Add Tool Providers to Canvas

This section shows where instructors can find a list of LTI-compatible tools. Instructors can take one or more tools from this list to use them in their Canvas courses by following this tutorial. As an example, we use ChemVantage [7], a tool that provides free Chemistry quizzes, homework assignments, and practice exams that keep students engaged by allowing multiple submissions until students succeed. By the end of this tutorial, instructors will have their choice of tool working within their Canvas course.

3.5.3.2 Add Tool Providers to Moodle

We also provide tools that can be used with Moodle. Instructors can use one or more of these tools in their Moodle courses by following this tutorial. This tutorial again uses the example of ChemVantage. By the end of this tutorial, instructors will have their choice of tool working within their Moodle course.

3.5.3.3 Add OpenDSA exercises to Canvas

In this section, we walk through how instructors can add OpenDSA exercises and visualization in their Canvas courses. A course instructor who is using the Canvas LMS can add individual OpenDSA materials directly into their Canvas courses by following our tutorial. The tutorial is divided into three parts. The first part is an eight-step tutorial which explains

how instructors can add OpenDSA as an external tool in Canvas. After adding OpenDSA as an external tool, instructors can add OpenDSA content as either graded or ungraded content. The second part is a five-step tutorial which goes through the process of adding ungraded content into a Canvas course. Finally, a seven-step tutorial explains the process of adding graded content into a Canvas course. Once the content is added, students will be able to read the prose, interact with the visualizations, and complete the exercises from within the Canvas. The gradable exercises will report scores to the Canvas' gradebook.

3.5.3.4 Add OpenDSA exercises to Moodle

This section is similar to the previous section but for Moodle. A course instructor who is using the Moodle LMS can add individual OpenDSA materials directly into their courses by following our seventeen step tutorial. Unlike Canvas, adding graded or ungraded content is very similar in Moodle. Once the instructors select the exercise they want to add, they also have to choose if they would like this OpenDSA exercise or visualization as a graded or ungraded content by going to the *Grade* section. Then they can select the type of exercise from the *Type* drop-down. The three available types are *None*, *Scale*, and *Points*. Once the content is added, students will be able to read the prose, interact with the visualizations, and complete the exercises from within the Moodle. The gradable exercises will report scores to Moodle's gradebook.

3.5.3.5 Configure an OpenDSA eTextbook

OpenDSA allows instructors to configure their own textbooks from its list of topics. After configuring their textbook, a course instructor who is using Canvas LMS can directly import the OpenDSA materials to Canvas. This is accomplished through the Canvas API. In this

tutorial, we explain all the settings and configurations along with the steps to add and remove chapters from the book. At the end of this tutorial, the instructors will have a custom book configured which they can later use to populate their Canvas course.

3.5.3.6 Add an OpenDSA eTextbook to Canvas

In this section, we walk through the process of using OpenDSA books to generate a Canvas course. Instructors can either use a pre-built book or configure their own book by following the instructions as mentioned in the above section. Once they have a book configured, the instructors can follow this twenty-eight step tutorial which walks them through the creation of their course, course offering, and LMS instance on the OpenDSA website. In the end, OpenDSA automatically creates assignments and modules within the Canvas course.

The tutorial resources described above are available at http://splice.cs.vt.edu.

3.6 Feedback

We talked to various researchers from the CS Education community at the 4th SPLICE workshop at SIGCSE 2019. Most of them appreciated the work done in the development of these tutorials. A few of those researchers have also requested LTI tutorials for other programming languages. JavaScript and Java are the most commonly requested languages. For example, Dr. Horstmann, a professor at San Jose State University, specifically asked that he needs tutorials in Java language because his tools are in Java, and most of his developers are undergraduate students.

Though we cannot say with any certainty if educational tool developers are using our tutorials or not, we know that the University of Pittsburgh has implemented LTI in their learning

platform, Mastery Grids, for it to consume materials from other educators using our tutorials. They are now consuming CodeWorkout exercises for small programming assignments.

Chapter 4

Auto-Grading Jupyter Notebooks

4.1 Introduction

Many courses in computer science would benefit if students were to not only write code, but also visualize the data, type equations, and write explanations. Jupyter Notebooks provide a platform where all these different tasks can be combined in one place. Instructors can provide self-contained notebooks with all the instructions and starter code in one place [27]. Jupyter notebook is gaining popularity [29] and is a common choice for many Data Scientists and for Python programming [34].

Instructors can create notebooks and distribute them to students through an LMS. The students download the assignment, write their solutions, and submit their files back to the LMS. The instructor or teaching assistants download the submissions and manually grade them. They also have to manually enter the points scored by each student in the LMS. Generally, feedback is provided to students only long after the assignment is complete.

Our aim is to auto-grade Jupyter notebook assignments and provide helpful feedback to students on their submissions. In the traditional setting, students get limited feedback as they progress through assignments because their assignment is only graded once they have submitted the final version [18]. Auto-grading Jupyter notebook assignments helps students get feedback as they progress through the assignment. Auto-grading notebooks can also

reduce the workload for instructors and TAs.

4.2 nbgrader vs. Web-CAT for Jupyter Notebooks

There exist auto-graders for programming assignments, including Web-CAT [24], BOSS [30], CourseMaker [28] and Bottlenose [37]. Previously, none of these supported auto-grading of Jupyter Notebook assignments. nbgrader [27] was developed by the creators of Jupyter Notebooks to auto-grade Jupyter Notebook assignments. In nbgrader, instructors can use the formgrader extension or the command line to create an assignment. During assignment creation, instructors can define the auto-graded and the manually graded cells. They have to provide the solution code and unit tests for the auto-graded cells. Finally, they can generate the student version of the assignment that does not include solution code and hidden unit tests. When students complete their assignments, they upload their solution files to their LMS. Instructor or teaching assistants download those files and move them to the "submissions" directory to auto-grade those assignments through the formgrader extension or nbgrader commands. However, nbgrader does not submit scores to an LMS, nor does it provide automated feedback to students. It does generate feedback for students, but that has to be manually delivered to each student, and therefore, it only happens after the final submission.

Web-CAT [24] is an auto-grading system with support for assignments in various programming languages. For Web-CAT, instructors have to create assignments on Canvas and Web-CAT separately. Adding the Canvas assignment URL to the Web-CAT assignment links these two together. Finally, when students are done with their assignment, they can use a plugin in their programming environment (such as Eclipse) to directly upload the assignment to Web-CAT. It shows them their feedback right away and submits the scores directly to the

LMS.

Our initial plan was to support LTI in nbgrader, which would have helped in sending scores to the LMS after auto-grading. But adding LTI support does not solve all the shortcomings of nbgrader. Teachers would still have to collect the assignments, move them to the *submissions* directory, and manually share feedback with students. Whereas Web-CAT already has the ability to send scores to Canvas and provides immediate feedback, it also offers other features such as giving extra credit for early submissions and defining rules for late submissions. Therefore, supporting Jupyter notebook auto-grading in Web-CAT would provide more features. Since Web-CAT provides an API for submission of assignments, we were able to create Jupyter Notebook extensions to upload notebooks directly to Web-CAT.

4.3 Auto-grading Jupyter Notebooks with Web-CAT

As discussed in the last section, we decided to use Web-CAT for auto-grading and providing immediate feedback to students. However, Web-CAT initially had no support for Jupyter notebooks. So, we implemented functionality within Web-CAT to handle "ipynb" files, which is a Jupyter notebook file format. During the assignment creation process, nbgrader is also used because nbgrader provides a toolbar through which instructors can select the type of each cell within the notebook. The nbgrader toolbar provides the functionality to mark the cells with different options such as *Autograded Answer* and *Autograded Tests*. We used this toolbar to mark the cells for Web-CAT to identify the auto-graded answer and the auto-graded tests cells. The toolbar also allows associating points to the *Autograded Tests* cells.

4.3.1 Preparing Jupyter Notebook Assignments

The first phase of this project involved adding the functionality in Web-CAT to support auto-grading of Jupyter notebooks. The instructor creates a Notebook assignment which includes solution code and unit tests marked by the nbgrader toolbar. Web-CAT generates a student version of the assignment by removing the solution code and unit tests from the notebook. The student version of the notebook file can be downloaded from Web-CAT and distributed to students through the LMS. The following steps are performed in Web-CAT once the instructor uploads the Jupyter Notebook file.

1. Insert a brand new cell to the front with the assignment description info for the extension toolbar. This is described in detail in the next sections.
2. Strip out any auto-grading test cells
3. Strip out any solution blocks and replaces them with *# Insert your work here* comments.
4. Strip out any output content from code cells, which might be from the instructor solution.

Figure 4.1 shows an auto-graded cell generated from the nbgrader toolbar. Web-CAT removes the code written between *### BEGIN SOLUTION* and *### END SOLUTION* and replaces these comments with *# Insert your work here* comments. Figure 4.2 shows the auto-graded test cell that is removed from the student's version. These tests are used for validating student submissions.

Once students complete their assignments or projects and submit them to Web-CAT, Web-CAT runs the tests which were stripped from the student's version and calculates points. The assertion failure message is displayed to students when they fail a particular unit test.

Figure 4.1: nbgrader cell indicating auto-graded solution cell.

Figure 4.2: nbgrader cell indicating auto-graded tests.

4.3.2 Jupyter Notebook Extension

Once students complete their assignments or projects, they have to upload their submissions to Web-CAT. The usual way to upload assignments to Web-CAT is to log in at the Web-CAT's website, navigate to the assignment page, and upload the submission files manually. After uploading, students will receive their scores and feedbacks. They have to repeat this process for each submission. To make this process easier, we created an extension which allows students to submit their assignments and projects from within their Jupyter notebook.

Web-CAT provides an API to submit assignment files to Web-CAT. The API requires the following three parameters along with the submission files:

1. **course:** Unique identifier for a course in Web-CAT.
2. **a:** Name of the assignment.
3. **d:** Name of the institution that this course belongs to.

We created two Jupyter extensions, a front-end extension, and a server extension. The front-end extension helps in modifying the default Jupyter notebooks, which was required to add a submission button on the notebooks. But since the notebooks run in the browser, we do not have access to the file system, and hence, we cannot access the assignment file and post

it to Web-CAT through its API. Therefore, we had to use a server extension to get access to the file.

4.3.2.1 Front-end Extension:

We developed the front-end extension, which adds a button labeled "*Submit to Web-CAT*" at the top of a Jupyter Notebook. The assignment is submitted to Web-CAT when students click on this button. Figure 4.3 shows the submit button.

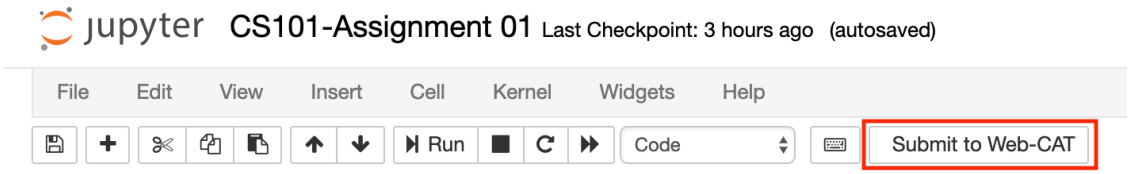


Figure 4.3: Submit to Web-CAT button

Since the Web-CAT API requires three parameters (course, a, d), these parameters are added as comments in the first cell as shown in Figure 4.4.

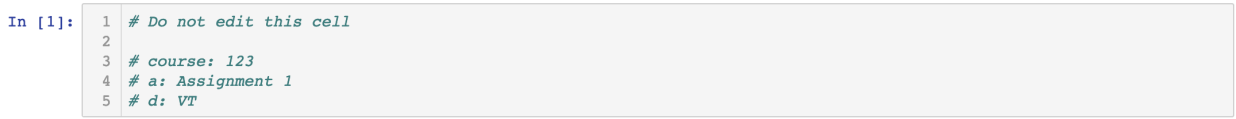


Figure 4.4: Web-CAT assignment parameters in the first cell of a notebook

The plugin reads this first cell and parses these three parameters. If these parameters are not present, the plugin throws an error message as shown in Figure 4.5.

The plugin also fetches the file path from the browser URL. Then it sends this file path with Web-CAT parameters to the server extension.

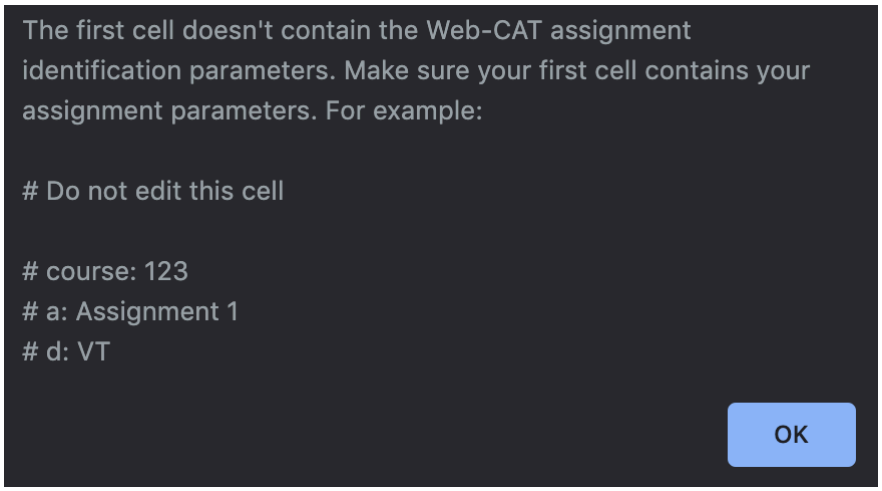


Figure 4.5: Error message shown when Web-CAT assignment parameters are not present in the first cell

4.3.2.2 Server Extension:

The server extension creates a URL and listens for requests sent to this URL. The front-end extension sends the file path and Web-CAT parameters to this URL. When the server extension receives the request, it reads the parameters from the received request and fetches the file from the file system. Finally, it sends the request to the Web-CAT API with the notebook file, along with Web-CAT assignment identification parameters. Web-CAT sends a response back with a URL on which the scores and feedback can be viewed. The server extension sends the response containing this URL back to the front-end extension. In the end, the front-end extension opens this URL in a pop-up. Figure 4.6 shows the output shown to the students. It is a pop-up showing the scores received on the assignment.

4.3.3 Putting it all together

Initially, instructors have to create their assignment, including the solution code and unit tests. They have to mark the cells containing the solution code and unit tests using the

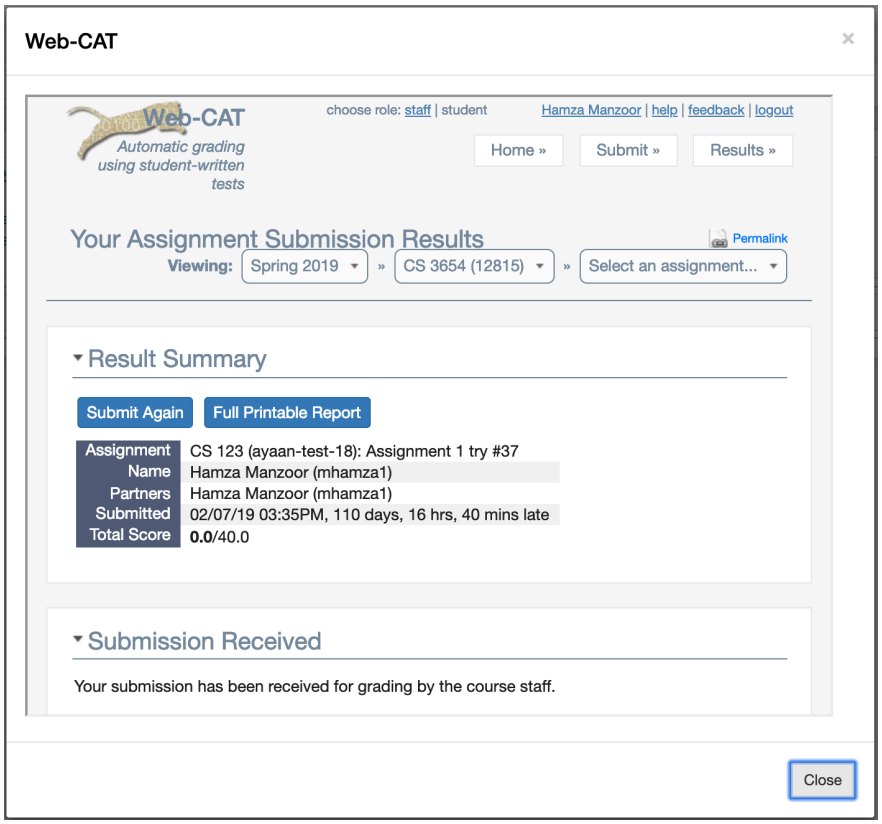


Figure 4.6: Pop-up showing the scores received on the assignment.

nbgrader toolbar's cell types *Autograded Answer* and *Autograded Tests* respectively. Then, they have to create an assignment in Canvas. After that, they have to create a course at Web-CAT and then an assignment within that course. They have to paste the Canvas assignment URL in the Web-CAT assignment to link the Canvas assignment with this Web-CAT assignment. While creating the assignment, they have to select *JupyterPlugin* in order for Web-CAT to know that it is a Jupyter Assignment. Finally, Web-CAT will generate a student version, which instructors can download from *Browse Raw Files* section. The student version will not have any solutions or reference unit tests.

Instructors can distribute the assignment to students through the LMS. When students complete their assignment, they can upload their submissions using the Jupyter Notebook extension, which will provide them with their scores and feedback right away. The received

scores are submitted directly to the gradebook of the LMS. They can improve their code and resubmit as many times as they like by working on the feedback, as long as they are within the deadline of the assignment.

4.4 Results and Feedback

Two sections of the course CS-3654: Intro Data Analytics & Visualization at Virginia Tech used Web-CAT for auto-grading Jupyter Notebooks in Spring 2019. We designed two surveys to analyze the effectiveness of using Web-CAT for auto-grading Jupyter Notebooks. One survey was designed for students and the other for instructors and teaching assistants.

4.4.1 Student's survey

This 15 question survey was designed for students. Students were asked questions such as, if they preferred continuous feedback through Web-CAT and if they believe their performance improved with continuous feedback. The instructor of one section asked students to fill the survey in class, and the other instructor asked students through a Canvas announcement. We got 99 responses, but 5 responses had incomplete data, and therefore, we removed those entries from our analysis. The questions asked in this survey can be found in Appendix A.1.

Figure 4.7 shows the responses in percentage when students were asked if getting immediate feedback from Web-CAT improved their performance? From the survey responses, we can say that 80% of students believe that getting immediate feedback from Web-CAT improved their performance.

This also aligns with our analysis from the Web-CAT data. Students on average had 11.4 submissions on each homework. Only 3% of cases have a single submission. 14% students had

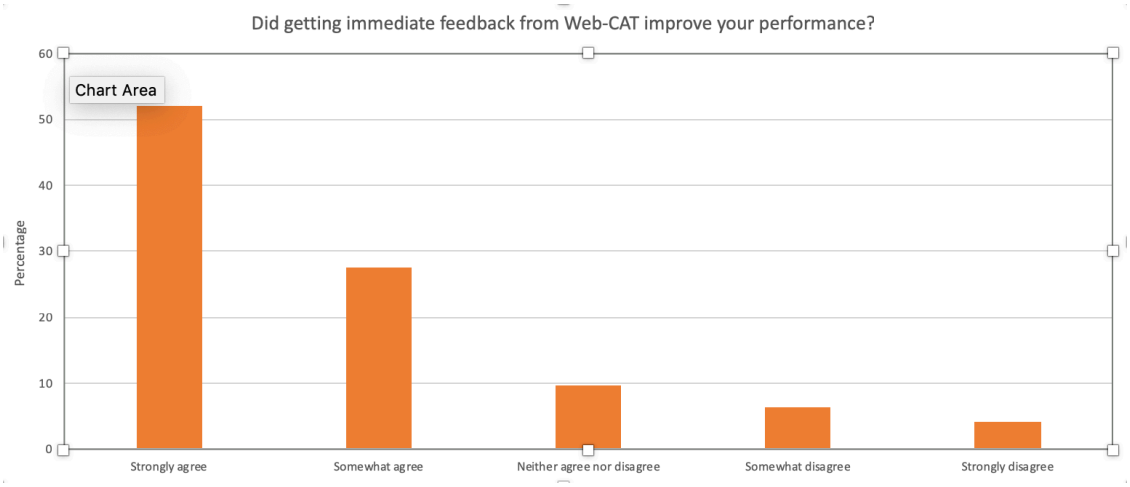


Figure 4.7: Survey question to students asking if getting immediate feedback improved their performance.

over 20 submissions to improve their scores. For example, one student had 69 submissions on Homework 3 and finally managed to score 100.

Similarly, 75% of students say that they will recommend future instructors to use Web-CAT for auto-grading and immediate feedback in their courses. Figure 4.8 shows the percentage of responses on a 10 point scale.

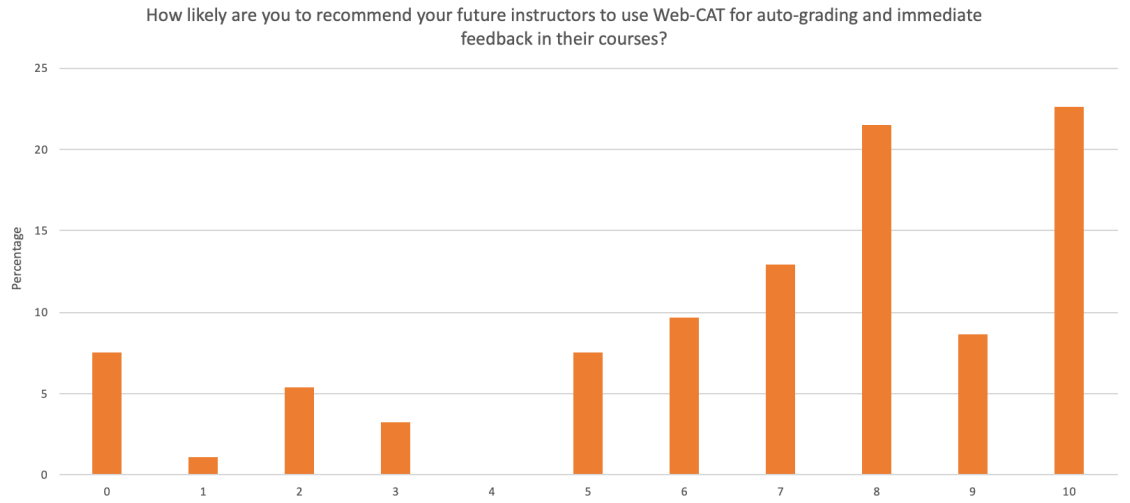


Figure 4.8: Responses to survey question asking students how likely are they to recommend using Web-CAT for Jupyter Notebooks to other instructors.

We also received some valuable feedback from free response questions. Most students say that getting feedback on homework is useful and tremendously improved their scores on assignments, while others say that feedback is generic and should be more specific.

4.4.2 Instructor's survey

A 16 question survey was designed for instructors and Teaching Assistants. It asked questions such as whether the number of students asking for feedback has significantly reduced or not. We got 5 responses from instructors and TAs. The questions asked in this survey can be found in Appendix A.2.

When asked if they believe that their load has been reduced using Web-CAT for auto-grading Jupyter Notebooks, 4 of 5 agreed and 1 had a neutral response. Similarly, 3 responses tell us that the number of students asking for feedback was significantly reduced.

3 instructors/TA also believe that Web-CAT with Jupyter plugin is better than nbgrader. When asked if they will recommend using Web-CAT for auto-grading Jupyter notebooks to other instructors, all 5 responses suggested "Yes".

We also received feedback from instructors in free response questions. A couple of them indicated that the process of creating assignments on Web-CAT is cumbersome and the UI should be improved. One instructor provided detailed feedback explaining why Web-CAT is better than nbgrader, but he also suggested that the feedback provided by nbgrader has the advantage that it embeds the feedback directly into the student's notebook format as an HTML page.

Chapter 5

Adding ACOS Content to OpenDSA

5.1 Introduction

The OpenDSA eTextbook system provides materials to support courses in a wide variety of Computer Science-related topics and allows users to choose from available modules to put together a custom textbook for a course. That book can be exported to Canvas as individual modules and assignments. Most OpenDSA exercises are developed by researchers and students at Virginia Tech. Similarly, many other researchers have worked on similar projects and have their own exercises. For example, ACOS, which is a smart learning content server developed as a joint project of Aalto University and the University of Pittsburgh, enhances the re-usability of online learning activities by decoupling the content and the existing interoperability protocols including LTI. ACOS is capable of serving multiple smart contents including Java and Python animations and exercises. OpenDSA does not have Python exercises. One solution would be to develop these exercises at Virginia Tech, which would require months of work. Better is to integrate existing exercises written by other researchers into OpenDSA, such that they seem like a part of OpenDSA. In this project, we made use of the Tool-Consumer capabilities of OpenDSA to integrate animations and exercises from ACOS.

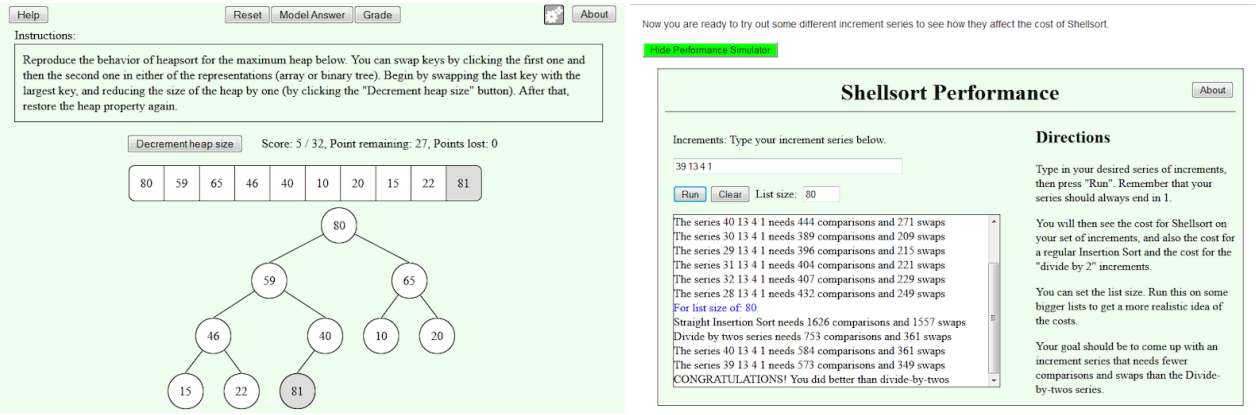
5.2 What are OpenDSA and ACOS?

5.2.1 OpenDSA

OpenDSA is an eTextbook project developed at Virginia Tech. OpenDSA materials include many visualizations and interactive exercises that support courses in a wide variety of Computer Science-related topics such as Data Structures and Algorithms (DSA), Formal Languages, and Programming Languages [13]. OpenDSA has hundreds of visualizations and exercises. Interactive algorithm visualizations illustrate most algorithms and data structures in the OpenDSA collection. OpenDSA combines textbook-quality prose with interactive visualizations and exercises, and allows students to practice as many times as they like. It also allows students to control the pacing of visualization and to enter their own test cases to see how the algorithm or data structure works on their input. It provides immediate feedback to students on every step of an exercise.

Figure 5.1a shows an example of an exercise in OpenDSA. Here, students reproduce the behavior of the algorithm to create a maximum heap by swapping records in an array. Similarly, Figure 5.1b shows a simulation of the performance of shellsort. Students can analyze the cost of shellsort on their own custom series of diminishing increments.

OpenDSA implements LTI and works as a tool provider to an LMS. A course instructor who is using the Canvas LMS can add individual OpenDSA materials directly into their Canvas course. Instructors can use pre-built books, or they can create their own book from the available modules and exercises. Once they create a book, they can easily export that to Canvas, and students directly interact with OpenDSA within Canvas. Instructors can set the number of points for each assessment. Once a student finishes the assignment, his score is reported back to Canvas and is displayed in the Canvas gradebook. Figure 5.2 shows an



(a) Exercise to reproduce the behavior of max-heap

(b) Simulator to analyze the performance of shellsort

Figure 5.1: OpenDSA visualizations and exercises

OpenDSA module within Canvas. Due to LTI, students cannot tell any difference whether the module or assignment is a Canvas-native module, or is served through an external tool.

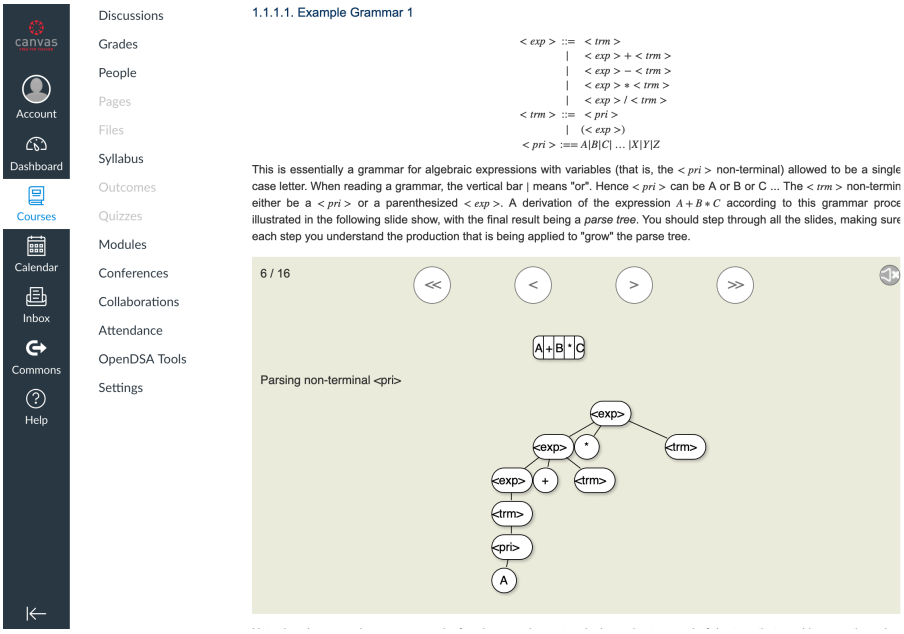


Figure 5.2: An OpenDSA module delivered in Canvas.

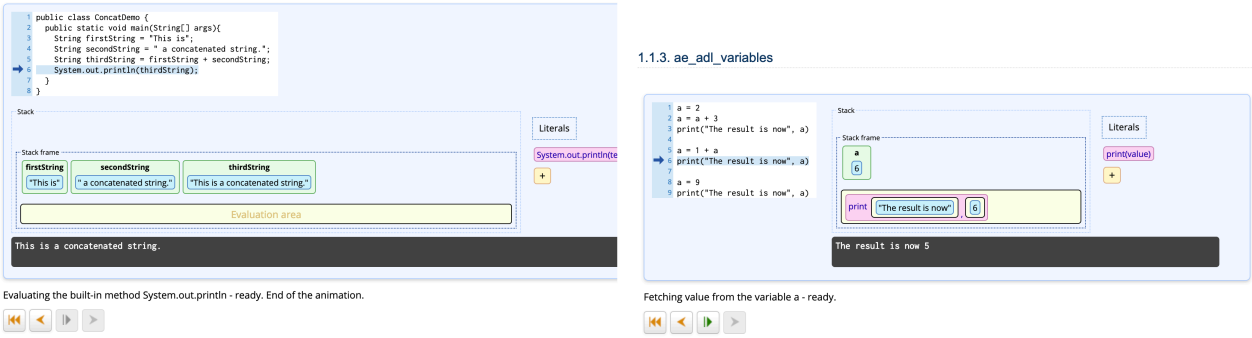
5.2.2 ACOS: Advanced Content Server

ACOS server is a smart learning content server developed as a joint project between Aalto University and the University of Pittsburgh [2]. It enhances the re-usability of online learning activities by decoupling the content type from existing interoperability protocols including LTI. ACOS is capable of serving multiple smart content types including Java and Python animations and exercises.

The key concept of ACOS is that the content and protocol developers do not have to worry about supporting different protocols. They can focus their efforts on content and their protocols independently. Instructors can use various content materials running on different protocols, and learners do not need to know if the content they are viewing is served through external tools. ACOS currently supports simple HTML content, and content served through LTI, A+ (a protocol similar to LTI developed at Aalto University for sharing external content), and Pitt protocol (a similar protocol developed by researchers at the University of Pittsburgh).

ACOS includes a collection of Python exercises and animations. The Python exercises are in the form of Parsons problems. Similarly, ACOS provides Java animations. These Java and Python animations are for beginner-level programmers, starting from *Hello World* to more advanced topics like recursion and classes. Figure 5.3a shows a Java animation for String concatenation. The animation shows all the steps involved in this program, including the values printed to the console and stack updates on every step. Similarly, Figure 5.3b shows an example Python animation for variable manipulation, i.e., how the value of a variable is updated in a stack when we change its value or add something to it.

ACOS server also provides Parsons problems in Python. These are a type of code completion problem in which a student has to place the mixed up code blocks into the correct order.



(a) Java animation of strings concatenation (b) Python animation of variable manipulation

Figure 5.3: ACOS Java and Python Animations

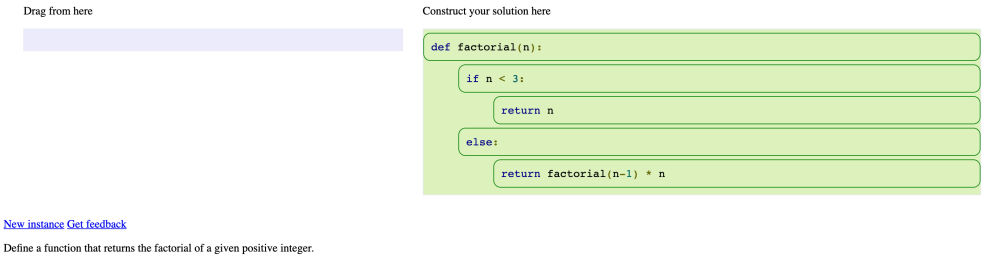


Figure 5.4: ACOS Parsons problem for Factorial program.

Figure 5.4 shows an example factorial program, which is solved by dragging the code blocks from left to right in the correct order.

5.3 Why do we need to integrate ACOS with OpenDSA?

OpenDSA currently has a wide variety of Computer Science-related topics for CS-2 and post CS-2 level courses such as Data Structures and Algorithms (DSA), Formal Languages, and Programming Languages. All the prose, exercises and visualization in OpenDSA are a result of years of efforts by various researchers and students. Others have developed content and exercises to support other topics, such as offered in CS-1 level courses. To add these to OpenDSA has traditionally meant rewriting or porting them individually. A significant

amount of time will be required to re-develop the animations and exercises. Therefore, if someone else has already spent their time creating these visualizations and exercises, it is more effective for OpenDSA to add them seamlessly within the textbook. By integrating with ACOS, OpenDSA will be able to support the topics for CS-1 level courses for both Python and Java.

5.4 OpenDSA as a Tool-Consumer

Historically, OpenDSA started only as an LTI tool provider and Canvas could add OpenDSA modules through the Canvas API. Then, instructors wanted to add support for coding exercises in OpenDSA modules. The first phase involved using CodeWorkout as another tool provider in Canvas. To accomplish this, OpenDSA content creators would add CodeWorkout exercises as a part of the OpenDSA module. Then instructors would create their courses on OpenDSA and OpenDSA would populate the Canvas course through the Canvas API. The Canvas API calls would not only create OpenDSA modules in Canvas, but also it would create CodeWorkout assignments separately. Now, this goes against the model of *OpenDSA as a textbook* because OpenDSA as a textbook should be able to add all the materials in itself, but in this case, a separate Canvas assignment required students to navigate away from the OpenDSA module and open the assignment in a separate window. Also, OpenDSA did not have a complete picture of students' performance because CodeWorkout and Canvas directly communicated with each other. Finally, OpenDSA could not group multiple assignments because each CodeWorkout assessment meant a separate assignment in Canvas.

To overcome these problems, the OpenDSA team came up with a new architecture, which allows OpenDSA to act as both a tool provider and a tool consumer. This new architecture enables OpenDSA to integrate with other external tools. In this particular example,

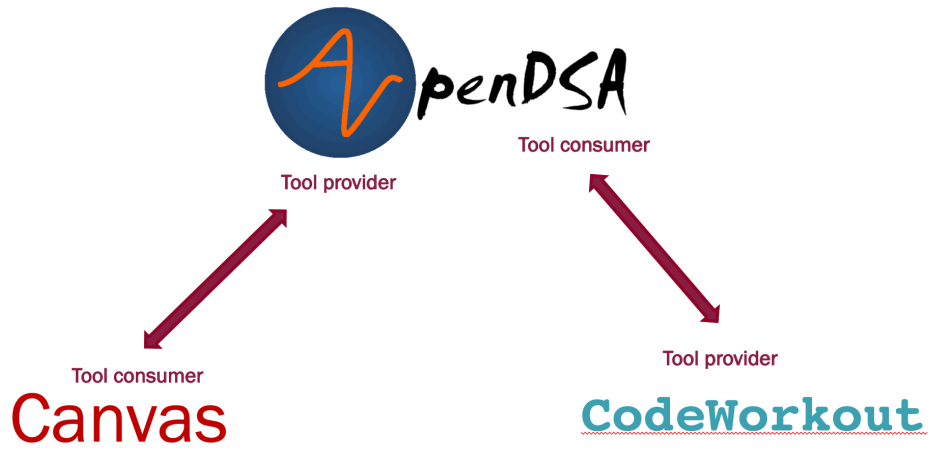


Figure 5.5: OpenDSA New Architecture.

OpenDSA adds CodeWorkout as an external tool and adds the CodeWorkout assignment within its modules, as shown in Figure 5.5. Now, Canvas adds OpenDSA as an external tool and has no information about CodeWorkout. Canvas sends a request to OpenDSA, and then OpenDSA sends a request to CodeWorkout to add coding exercises in the textbook. CodeWorkout sends a response to OpenDSA and OpenDSA then embeds the CodeWorkout exercise and sends a complete module to Canvas. This allows students to solve programming exercise from within the same page where they read prose and visualization in Canvas. This also allows OpenDSA to have a complete picture of how a student is performing on other exercises. Figure 5.6 shows an OpenDSA exercise which is added inside an OpenDSA module that is shown within Canvas as its module.

5.5 Integration Details

The integration of OpenDSA and ACOS is possible because OpenDSA works as both a tool consumer and a tool provider, as explained in the above section.

Figure 5.6: A CodeWorkout exercise embedded within an OpenDSA module, displayed through Canvas.

5.5.1 Adding a new tool to OpenDSA

OpenDSA acts as a tool provider to Canvas to serve textbooks, and within those books, OpenDSA calls other tools such as CodeWorkout to serve various coding exercises. We added ACOS as a new tool within OpenDSA to serve the Python exercises and animations of Java and Python. To do so, first we added three new tools *mastery-grid-jsparsons-python*, *mastery-grid-java-animations*, and *mastery-grid-python-animations* in the OpenDSA external tools script. This script is called during the book compilation process if external tools are used. Adding these tools allows instructors or content developers to add various exercises through external tools in their books. Though these exercises and animations are all served by ACOS, the reason why we used three different tools instead of one is that ACOS serves these different exercise types through three different URLs. Since each tool can have only one unique URL, we had to consider these as three different tools. We then added these into the OpenDSA database as LTI tools. When one or more of these tools are used in an

OpenDSA module, OpenDSA fetches the URL from the database by looking through the tool name and sends an LTI request to the respective URL of ACOS. ACOS delivers the exercise or animation in an iframe, which is delivered as a part of the OpenDSA module.

5.5.2 Integrating Python Exercises

ACOS provides Parsons problems in Python. These problems allow students to arrange the mixed-up code blocks in the correct order to make the program run. These problems are added as an external tool called *mastery-grid-jsparsons-python*. 34 Parsons problems can be added to OpenDSA books. These problems start with the basics of Python (Hello World) to nested loops. The complete list of Python exercises can be found in Table B.1 in Appendix B.1.

These are graded exercises, so when OpenDSA sends a request to ACOS, it expects a score back from it once a student finishes the exercise. These scores are then passed on to Canvas and are shown in the Canvas gradebook. Figure 5.7 shows an example of a Parsons problem served through Canvas via the OpenDSA eTextbook.

5.5.3 Integrating Python Animations into OpenDSA

ACOS also provides Python animations. These animations are Python program visualizations about various topics covered in CS1 courses. These animations take the reader through each process step by step, such as showing changes in the stack when a particular line of code is executed, and how the output is calculated and finally printed to the console.

These animations are added as an external tool called *mastery-grid-python-animations*. 56 Python animations can be added to OpenDSA books. These animations start with the basics

Figure 5.7: A Parsons problem within an OpenDSA eTextbook delivered via Canvas.

of Python and move onto more complicated topics, such as loops, objects, classes, and file handling. The complete list of Python animations can be found in Table B.2 in Appendix B.1. Figure 5.9 shows an example of Python function animation served through Canvas via the OpenDSA eTextbook. This animation explains how parameter passing and return values work in Python functions.

5.5.4 Integrating Java Animations into OpenDSA

Apart from Python exercises and animations, ACOS also provides Java animations. These animations are Java program visualizations about various topics covered in CS1 courses. These animations are similar to the Python animations, but for Java.

These animations are added as an external tool called *mastery-grid-java-animations*. 53 Java animations can be added to OpenDSA books. These animations cover the basics of Java

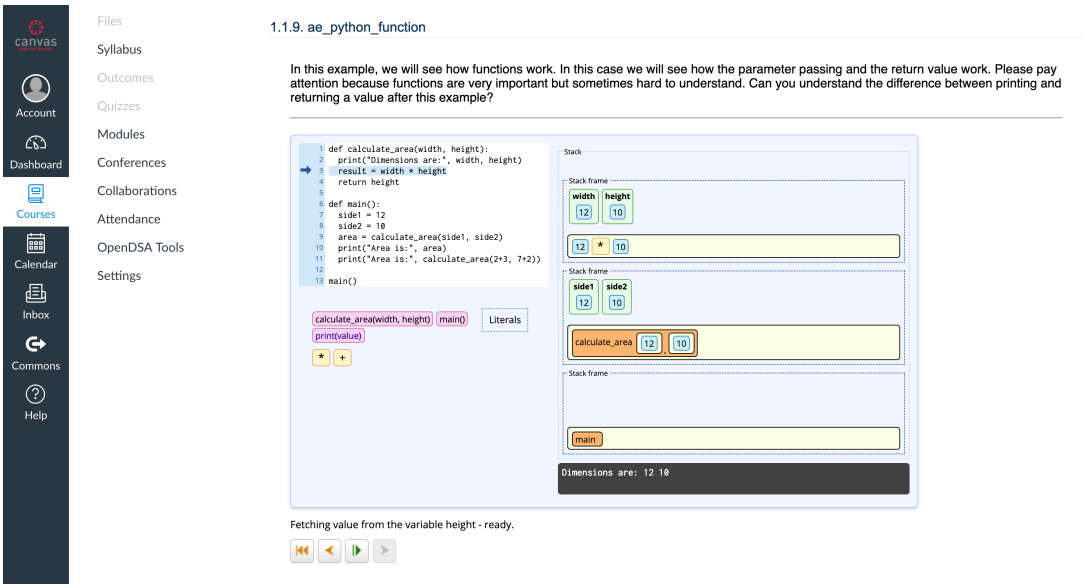


Figure 5.8: Python function animation served through Canvas via OpenDSA eTextbook.

programming from printing *Hello World* to the console to advanced topics, such as multi-dimensional arrays, inheritance, and polymorphism. The complete list of Java animations can be found in Table B.3 in Appendix B.1. Figure 5.9 shows an example of an animation of the switch statement. It is served through Canvas via the OpenDSA eTextbook.

5.6 Integrating CodeWorkout Exercises into Mastery Grids

Similar to adding ACOS content in OpenDSA, we also extended the content supported by Mastery Grids [32]. Mastery Grids is an open-source progress visualization environment which has open learner model features designed at the University of Pittsburgh. We integrated CodeWorkout [20] programming problems into Mastery Grids.

Authoring smart learning content to fulfill the requirements of programming courses at different levels (e.g. beginner, advanced) requires a lot of time. Fortunately, integrating

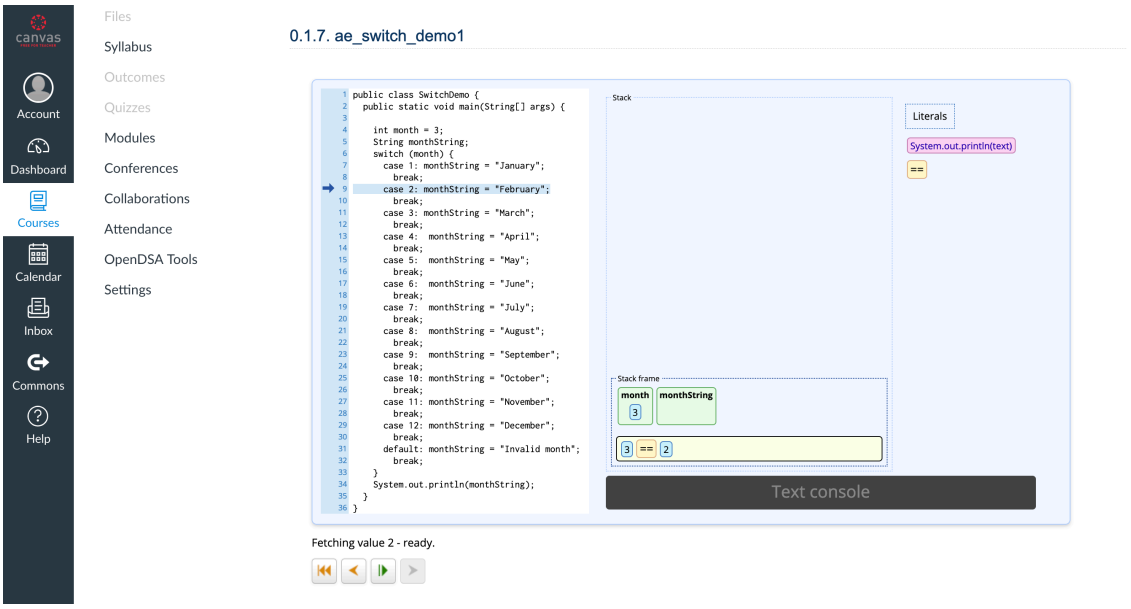


Figure 5.9: Java animation of switch statement served through Canvas via OpenDSA eTextbook.

already existing tools and learning content through standardized protocols such as LTI is the solution to this problem. To implement this solution, we have selected CodeWorkout to demonstrate the importance and applicability of smart content integration. CodeWorkout is an LTI-provider, and has many Java programming exercises in advanced topics.

Collaborating with the University of Pittsburgh, we have created a practice environment for Java learners with multiple topics such as Recursion, Trees, etc. Figure 5.10 shows an instance of a CodeWorkout exercise accessed from the Mastery Grids interface. A student with access to Mastery Grids can access CodeWorkout exercises without being required to log in to the CodeWorkout system. After a student completes and submits her answer successfully, the student modeling service which is already a part of Mastery Grids is updated automatically, and the student can view her progress status via colored squares as shown in Figure 5.10. This integration is another use case to demonstrate the importance of reusing the materials developed by other researchers.

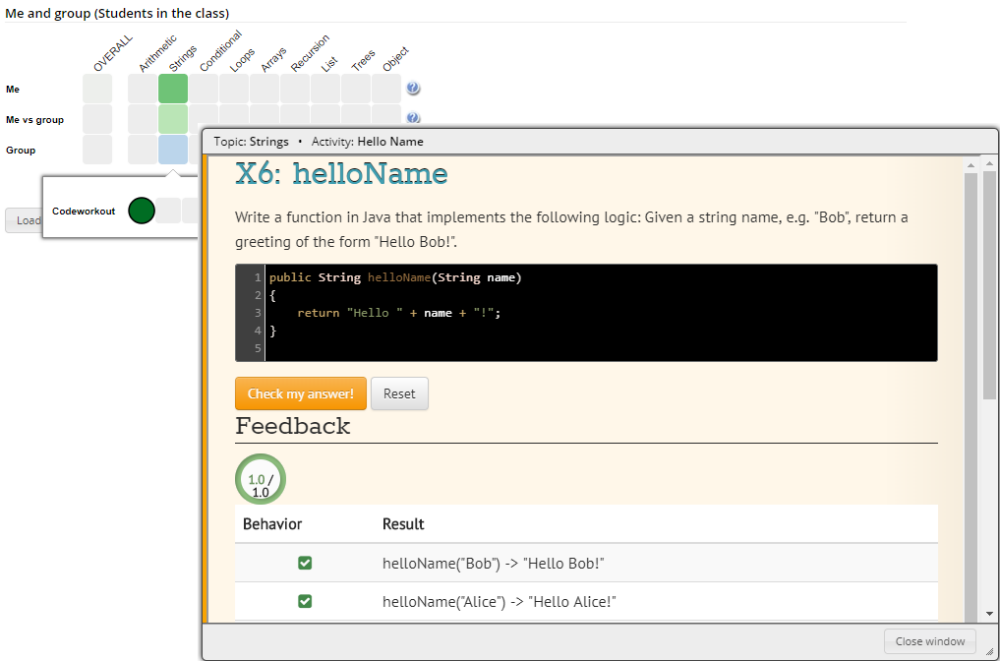


Figure 5.10: An instance of a CodeWorkout exercise accessed from Mastery Grids interface.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The theme of this thesis is to provide resources to educational content developers who seek to use the LTI interoperability standard. Our first task was to provide tutorials and resources on LTI to developers of learning applications. Our second task was to use LTI for auto-grading of Jupyter Notebooks and submission of scores to the LMS. We want students to get immediate feedback so they can know where they went wrong and improve their programs. This auto-grading of notebooks also makes it easier for instructors to handle a larger number of students in CS courses. Our third task was to add Computer Science exercises developed by the University of Pittsburg (ACOS) into the OpenDSA e-textbook system developed at Virginia Tech. This integration shows that different institutions can collaborate and use the resources developed by each other.

Many educational tools exist that would be helpful to use if they were interoperable with existing LMS. We started by reviewing the various standards for interoperability. We took as a goal to provide awareness about the one which is supported by the major LMS. We found that most current LMS support LTI. In our attempt to support LTI in the applications developed at Virginia Tech, we found that a lack of resources available to many developers made it hard to implement this standard properly. Therefore, we have developed tutorials for LTI for developers of learning applications, to help them understand and implement this

standard in their applications. The tutorials also include example applications and insights into how OpenDSA, one popular LTI tool, implements LTI.

We noticed that the increase in the number of students enrolling into the computer science courses has made it extremely difficult for instructors to grade all the assignments and manually enter all the scores in the LMS. Therefore, we implemented auto-grading of Jupyter Notebooks and submission of scores to the LMS. We have used Web-CAT to auto-grade Jupyter assignments and projects. We have also implemented a Jupyter extension which enables students to upload their submissions directly to Web-CAT from their Jupyter Notebooks running their browsers. Auto-grading notebooks has also enabled students to get instant feedback on their submissions.

From our survey results, we can say that the performance of students has improved due to continuous feedback. Students typically make multiple submissions to score more on their homework. The number of students asking instructional stuff for feedback has also reduced. Both instructors and students are in favor of using Web-CAT for auto-grading Jupyter Notebooks, and they would recommend other instructors to do the same.

We have used LTI to extend OpenDSA to support Python programming exercises and animations. We wanted to demonstrate the benefits of using the standards in learning applications. OpenDSA had Java programming exercises and animations for various computer science related topics, and it took years of effort from contributors to develop them. ACOS provides Python exercises and animations. Making use of the fact that both ACOS and OpenDSA support LTI, we have added ACOS materials into OpenDSA. Now OpenDSA textbooks can take advantage of these Python exercises and animations.

Since the launch of our LTI tutorial website, and through various SPLICE workshops, we have seen more awareness about LTI among the CS Education community. For example,

the University of Pittsburgh has implemented LTI in their learning platform, Mastery Grids. This allows it to consume materials provided by other educational software developers. Mastery Grids can now use CodeWorkout exercises for small programming assignments.

6.2 Future Work

LTI has enabled learning tools to talk to each other and share data. We have provided LTI tutorials to be used by developers of learning applications. Apart from the generic tutorials, the sample applications and tutorials we offer are in Ruby on Rails only. We have received feedback from various researchers at 4th SPLICE workshop at SIGCSE 2019 that they want tutorials in other languages. JavaScript is the most commonly requested programming language. Future work could be to provide more tutorials for other languages, starting with JavaScript.

LTI solves some interoperability problems, and allows tools a limited ability to talk to each other. But learning data (clickstreams, workout sessions, errors, etc.) gathered by various learning tools stays with those tools, and is not shared through LTI. Caliper Analytics allows institutions to collect the learning data from external tools through defined standards. Caliper is not yet popular among the CS education community. In the future, we should provide tutorials about Caliper, along with sample applications.

For our use case involving Jupyter Notebooks, we used Web-CAT for auto-grading. Web-CAT provides feedback separate from the original notebook file. Students get their feedback on Web-CAT's website, or in a pop-up if they use the extension to upload notebook files to Web-CAT. The feedback that we received from instructors who used both Web-CAT and nbgrader for auto-grading in their courses is that nbgrader provides better feedback because it embeds feedback directly in the original notebook files. We can try to follow the same

approach of providing feedback within the notebook file, but it is not as simple in the case of Web-CAT. nbgrader provides feedback at the very end when students have submitted their final version of the assignment. Embedding feedback within the notebook file makes sense in the end. Whereas in the case of Web-CAT, students get feedback right away, and they might want to improve their submissions after getting feedback. Therefore, changing the original notebook file through the plugin might not be the best approach. We will probably have to create a new notebook file or generate an HTML file of the notebook with feedback embedded in it as nbgrader does.

Another feedback we received from the instructors is that in the case of Data Science courses, they are not concerned with individual units and therefore, unit tests do not solve all the problems. For example, a student might be asked to generate a table. There is no assert statement to compare the tables, so instructors have to write extra code to do this. In the future, we can also provide code samples to illustrate commonly used output formats.

We also need a better evaluation for Tutorials and Jupyter Notebooks. We need a way to identify if users are using our tutorials or not. One method is to track how much time users are spending on tutorials pages and if they are scrolling through the entire tutorial or not. For Jupyter Notebooks, we asked students and instructors to answer survey questions after the end of the course. Only one of the assignments used nbgrader in that course. To compare if instructors prefer nbgrader or Web-CAT, we need to have a better comparison. Secondly, we do not have a comparison to show if students performed better with continuous feedback through Web-CAT. We need comparison with a course that did not use auto-grading to compare how students performed on the same assignments when there was no constant feedback.

Finally, OpenDSA, for the most part, provides proper documentation. But when we were trying to add ACOS content into OpenDSA, we realized that there is no documentation for

adding external tools. We should document the process for adding more external content into OpenDSA.

Bibliography

[1] About Moodle. URL https://docs.moodle.org/30/en/About_Moodle.

[2] Acos-Server. URL https://github.com/acos-server/acos-server.

[3] ADL Experience API (xAPI) and IMS Caliper Discovery Review - ADL Initiative, . URL https://adlnet.gov/news/adl-experience-api-and-ims-caliper-discovery-review.

[4] The Advanced Distributed Learning Initiative, . URL https://adlnet.gov.

[5] Blackboard LMS. URL https://www.blackboard.com/learning-management-system/.

[6] Caliper Analytics. URL https://www.imsglobal.org/activity/caliper.

[7] ChemVantage. URL https://www.chemvantage.org/About.

[8] edx/XBlock. URL https://github.com/edx/XBlock.

[9] Google Gadgets. URL https://developers.google.com/gadgets/.

[10] IMS Global | Glossary. URL https://www.imsglobal.org/glossary.

[11] Canvas by Instructure. URL https://www.canvaslms.com/.

[12] Open edX, . URL https://open.edx.org/.

[13] OpenDSA eTextbook Systems, . URL https://opendsa-server.cs.vt.edu.

[14] SCORM vs. LTI: What's the Difference? URL https://rusticisoftware.com/blog/scorm-vs-lti/.

[15] Sakai Learning Management System | Higher Education. URL https://www.sakailms.org/.

[16] Why SCORM 2004 Failed and What That Means for Tin Can, May 2016. URL https://www.efrontlearning.com/blog/2013/04/why-scorm-2004-failed-what-that-means-for-tin-can.html.

[17] The Difference Between AICC, SCORM xAPI, Jun 2018. URL https://www.webanywhere.com/2018/06/24/the-difference-between-aicc-scorm-xapi/.

[18] Maha Aziz, Heng Chi, Anant Tibrewal, Max Grossman, and Vivek Sarkar. Auto-grading for Parallel Programs. In *Proceedings of the Workshop on Education for High-Performance Computing*, page 3. ACM, 2015.

[19] Oliver Bohl, Jörg Scheuhase, Ruth Sengler, and Udo Winand. The Sharable Content Object Reference Model (SCORM) - A Critical Review. In *International Conference on Computers in Education, 2002. Proceedings.*, pages 950–951. IEEE, 2002.

[20] Kevin Buffardi and Stephen H Edwards. Introducing CodeWorkout: An Adaptive and Social Learning Environment. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pages 724–724. ACM, 2014.

[21] Adobe Captivate. Adobe Captivate, 2010.

[22] Declan Dagger, Alexander O'Connor, Seamus Lawless, Eddie Walsh, and Vincent P Wade. Service-oriented e-Learning Platforms: From Monolithic Systems to Flexible Services. *IEEE Internet Computing*, 11(3):28–35, 2007.

[23] Ron Drabkin. Meet Caliper, the Data Standard That May Help Us (Finally) Measure Edtech Efficacy - EdSurge News, Dec 2018. URL https:

//www.edsurge.com/news/2017-05-23-meet-caliper-the-data-standard-may-help-us-finally-measure-edtech-efficacy.

[24] Stephen H Edwards and Manuel A Perez-Quinones. Web-CAT: Automatically Grading Programming Assignments. In *ACM SIGCSE Bulletin*, volume 40, pages 328–328. ACM, 2008.

[25] Murray W Goldberg, Sasan Salari, and Paul Swoboda. World Wide Web-Course Tool: An Environment for Building WWW-based Courses. *Computer Networks and ISDN Systems*, 28(7-11):1219–1231, 1996.

[26] Lynne Grewe. *OpenSocial Network Programming*. John Wiley & Sons, 2009.

[27] Jessica B Hamrick. Creating and Grading IPython/Jupyter Notebook Assignments with NbGrader. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 242–242. ACM, 2016.

[28] Colin A Higgins, Geoffrey Gray, Pavlos Symeonidis, and Athanasios Tsintsifas. Automated Assessment and Experiences of Teaching Programming. *Journal on Educational Resources in Computing (JERIC)*, 5(3):5, 2005.

[29] Tony Hirst. The Growing Popularity of Jupyter Notebooks, Sep 2018. URL https://blog.ouseful.info/2018/09/10/the-growing-popularity-of-jupyter-notebooks/.

[30] Mike Joy, Nathan Griffiths, and Russell Boyatt. The BOSS Online Submission and Assessment System. *Journal on Educational Resources in Computing (JERIC)*, 5(3):2, 2005.

[31] Kin Chew Lim. Case Studies of xAPI Applications to E-Learning. In *The Twelfth International Conference on eLearning for Knowledge-Based Society*, pages 3–1, 2015.

[32] Tomasz D Loboda, Julio Guerra, Roya Hosseini, and Peter Brusilovsky. Mastery Grids: An Open Source Social Educational Progress Visualization. In *European Conference on Technology Enhanced Learning*, pages 235–248. Springer, 2014.

[33] Brendan Noud. To SCORM or Not to SCORM, That is The Question - LearnUpon Blog. URL https://www.learnupon.com/blog/to-scorm-or-not-to-scorm-that-is-the-question/.

[34] Jeffrey M. Perkel. Why Jupyter is Data Scientists' Computational Notebook of Choice, Oct 2018. URL https://www.nature.com/articles/d41586-018-07196-1.

[35] Charles Severance, Ted Hanss, and Josepth Hardin. IMS Learning Tools Interoperability: Enabling a Mash-up Approach to Teaching and Learning Tools. *Technology, Instruction, Cognition and Learning*, 7(3-4):245–262, 2010.

[36] Clifford A Shaffer, Ville Karavirta, Ari Korhonen, and Thomas L Naps. OpenDSA: Beginning a Community Active-ebook Project. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, pages 112–117. ACM, 2011.

[37] Mark Sherman, Sarita Bassil, Derrell Lipman, Nat Tuck, and Fred Martin. Impact of Auto-grading on an Introductory Computing Course. *Journal of Computing Sciences in Colleges*, 28(6):69–75, 2013.

[38] Articulate Storylineis. Articulate Storyline. 2017.

[39] Scott Wilson, Paul Sharples, and Dai Griffiths. Distributing Education Services to Personal and Institutional Systems Using Widgets. In *Proceedings of Mash-Up Personal Learning Environments-1st Workshop MUPPLE*, volume 8, pages 25–33, 2008.

Appendices

Appendix A

Jupyter Notebooks Survey Questions

A.1 Student's Survey

1. How familiar were you with Jupyter Notebooks before this course?
    - Used it for several assignments or projects
    - Used it for one or two assignments or projects before
    - Knew about it but never used it
    - Did not know much about it before
2. Do you think Jupyter Notebooks are better than traditional IDEs or text editors for Python programming?
    - Jupyter Notebooks are better
    - Traditional IDEs (or a plain text editor) are better
    - I like both
3. If you have a choice, are you likely to choose Jupyter Notebooks for your future Python assignments and projects?
    - Yes
    - Maybe

- No

4. What are your thoughts on Jupyter Notebooks?

5. How familiar were you with Web-CAT before this course?

   - Regularly used it before for many assignments
   - Used it for a small number of assignments
   - Knew about it but never used it
   - Never heard about it before

6. The first two assignments did not use Web-CAT for auto-grading, so you have some experience with having no auto-grading available. Did getting immediate feedback from Web-CAT improve your performance on these assignments?

   - Strongly agree
   - Somewhat agree
   - Neither agree nor disagree
   - Somewhat disagree
   - Strongly disagree

7. Do you think that we should continue to use Web-CAT for auto-grading and automatic feedback for future assignments and projects?

   - Yes, it is very helpful
   - Yes, it is somewhat helpful
   - It makes no difference
   - No, it is not of much help

- No, it is completely useless

8. Do you have anything else to tell us about the feedback provided by Web-CAT? Is there anything else you would like Web-CAT to provide?

9. Did you use the Web-CAT plugin for Jupyter Notebooks for your assignments submissions?

   - Yes
   - Sometimes
   - No
   - I do not know

10. Do you believe that it is more convenient to use the Web-CAT plugin for Jupyter Notebooks than it is to submit directly on Web-CAT's website?

    - The plugin is more convenient
    - Submitting directly on Web-CAT's website is more convenient
    - Both are equally convenient
    - Neither are convenient and I'd prefer something else

11. Given the choice, would you use the Web-CAT plugin for Jupyter Notebooks for your future assignments and projects in this course and other courses which will use Web-CAT for auto-grading?

    - Definitely yes
    - Probably yes
    - Might or might not

- Probably not
- Definitely not

12. Did you face any issues with the plugin? How would you like to see the plugin improved?

13. How likely are you to recommend your future instructors to use Web-CAT for auto-grading and immediate feedback in their courses?

    - 0
    - 1
    - 2
    - 3
    - 4
    - 5
    - 6
    - 7
    - 8
    - 9
    - 10

14. How convenient do you find the entire process of using Web-CAT for auto-grading and immediate feedback for Jupyter assignments and the Web-CAT plugin for uploading assignments? How would you like to improve this process?

15. Is there anything else you would like to mention?

A.2 Instructor's Survey

1. How many times have you been a Teaching Assistant or Instructor before this course?
    - 1 time
    - 2 times
    - >= 3 times
2. How familiar were you with Jupyter Notebooks before this course?
    - Used it for several assignments or projects
    - Used it for one or two assignments or projects before
    - Knew about it but never used it
    - Did not know much about it before
3. How experienced were you with grading programming assignments or projects before this course?
    - Have graded several assignments or projects before
    - Have graded one or two assignments or projects before
    - Never graded programming assignments before
4. How experienced were you with grading programming Jupyter notebooks assignments or projects before this course?
    - Have graded several assignments or projects that used Jupyter notebooks before
    - Have graded one or two assignments or projects that used Jupyter notebooks before

- Never graded an assignment that used Jupyter notebooks before

5. How familiar were you with Web-CAT for auto-grading before this course?

   - Regularly used it before for many assignments
   - Used it for a small number of assignments
   - Knew about it but never used it
   - Never heard about it before

6. The first two assignments did not use Web-CAT for auto-grading, so you have some experience with a manual grading. Did getting immediate feedback from Web-CAT reduced your workload and made your life easier?

   - Strongly agree
   - Somewhat agree
   - Neither agree nor disagree
   - Somewhat disagree
   - Strongly disagree

7. Now students get immediate feedback on their submissions. Is the number of students coming to you for feedback reduced when using Web-CAT for auto-grading or is it about the same?

   - The number of students asking for feedback has significantly reduced when using Web-CAT for auto-grading
   - The number of students asking for feedback has slightly reduced when using Web-CAT for auto-grading
   - It is about the same.

- The number of students asking for feedback has slightly increased when using Web-CAT for auto-grading
- The number of students asking for feedback has significantly increased when using Web-CAT for auto-grading

8. Do you have anything else to tell us about the feedback provided by Web-CAT to the students? What were the issues faced by students with Web-CAT? Is there anything else you would like Web-CAT to provide?

9. Did you see students facing issues with the Web-CAT plugin for Jupyter Notebooks?

   - Yes, many students faced issues
   - Yes, but only a few students faced issues
   - No, plugin is easy to use for student and no one faced any issues
   - I do not know

10. Given the choice, would you recommend the Web-CAT plugin for Jupyter Notebooks to students in this course and other courses which will use Web-CAT for auto-grading assignments with Jupyter Notebooks?

    - Definitely yes
    - Probably yes
    - Might or might not
    - Probably not
    - Definitely not

11. What were the common issues faced by students with the plugin? How would you like to see the plugin improved?

12. Have you used nbgrader for auto-grading Jupyter Notebooks before?

    - Used it for several assignments or projects
    - Used it for one or two assignments or projects before
    - Knew about it but never used it
    - Did not know much about it before

13. If you have used nbgrader before, then which do you prefer? Web-CAT with Jupyter Plugin, or nbgrader?

    - Web-CAT with Jupyter Plugin is better
    - nbgrader is better
    - They are about the same
    - I do not know both

14. How likely are you to recommend your future instructors to use Web-CAT for auto-grading and immediate feedback in their courses?

    - 0
    - 1
    - 2
    - 3
    - 4
    - 5
    - 6
    - 7

- 8
- 9
- 10

15. How convenient do you find the entire process of using Web-CAT for auto-grading and immediate feedback for Jupyter assignments, and the Web-CAT plugin for uploading assignments? How would you like to improve this process?

16. Is there anything else you would like to mention?

Appendix B

ACOS exercises added to OpenDSA

B.1 Python Parsons Problems

Table B.1: List of Python Parsons problems added in OpenDSA from ACOS

Name	**Description**
ps_hello	Construct a program that prints out hello and world on separate lines.
ps_simple_function	Create a function that prints out Hello functions.
ps_simple_params	Create a function that adds 2 to the input parameter
ps_return_bigger_or_none	Construct a function that returns the bigger value of the given arguments. If a and b are equal, it should return None.
ps_python_addition	Construct a program that adds numbers prints the value 8.
ps_python_iteration_addition	Construct a program that adds 2 in a loop until the value becomes 8 and prints the value 8.
ps_python_iteration_multiplication	Construct a program that multiples number with 2 in a loop and prints values 1,2,4,8 and finally prints ”The end!”.

Continued on next page

Table B.1 – *Continued from previous page*

Name	**Description**
ps_python_calculate_function	Define a function that returns the second value multiplied by two and added by the first value. The program should print 23 at the end.
ps_python_nested_calls	Define a calculate function and a double function which will return the double of the input value. Then construct a program that first prints out 20 and then 36.
ps_python_recursive_factorial	Define a recursive function that returns the factorial of a given positive integer.
ps_python_class_person	Write a program that defines a class "Person" and prints out "Safiira. Nice to meet you!"
ps_python_modulo_is_even	Construct a function that will return True if a given number is even, otherwise false.
ps_python_list_iteration_zoo	Construct a program that prints out all the animals in the zoo-variable by looping through the list.
ps_python_nested_lists_indexing	Construct a program that first prints out [[1, 2, 3], [4, 5, 6]], then [4, 5, 6], and finally 6 from the nested list.
ps_python_string_indexing	Construct a program that first prints out strings "Py", "th", "o", and "n" from the "Python" string.
ps_python_dict_keys	"storage" dictionary object stores the amount of certain items in storage. Construct a program that prints out a list of all the items that are more than one currently stored.

Continued on next page

Table B.1 – *Continued from previous page*

Name	**Description**
ps_python_dict_values	Construct a program that loop through the dictionary and prints out how many items in total are in storage.
ps_python_list_to_dict	Construct a program that casts list to dictionary and prints out moo and oink, in that order.
ps_python_dict_filter	Construct a program that prints out all the circles that are larger than 5.
ps_python_bigger_than	Assume that num1 and num2 have been initialized to numbers, so that number1 is bigger. Construct a program that correctly prints out that number1 is indeed bigger
ps_python_conditionals_temperature	Construct a program that outputs "Cold", when the temperature is 15 degrees Celsius or below. It prints "Moderate" when the temperature is over 15 degrees but no more than 25 degrees. If the temperature is over 25 degrees, it prints "Hot".
ps_python_printing_file_contents	Construct a program that opens up a file and prints out each line of the file.
ps_python_try_except	Construct a program that prints out various Celsius temperatures in Fahrenheit and throws an exception if the temperature is below absolute zero.

Continued on next page

Table B.1 – *Continued from previous page*

Name	**Description**
ps_python_nested_ifs	Construct a program using nested if statements that prints out a different sentence depending on the time of day. Night is considered to be before 7 a.m., morning is from 7 a.m. until noon, afternoon is until 5 p.m. and rest is considered evening.
ps_python_comparisons	Construct a program that prints out whether variable a is bigger than b.
ps_python_add_to_list	Construct a function that adds a given amount to all items in a list.
ps_python_swap	Construct a program that swaps the values of x and y variables.
ps_python_xor	Construct a program that mimics an XOR gate (exclusive or). When input_a and input_b are the same, it should print out 0 and in other cases print out 1.
ps_python_for_odd_or_even	Construct a program that goes through a list of numbers and prints out whether they are odd or even.
ps_python_string_join	Construct a program that prints out a sentence from a given list of words by joining words in a list.
ps_python_try_adding	Construct a function that adds two numbers together and handles non-numeric input through exception handling.

Continued on next page

Table B.1 – *Continued from previous page*

Name	**Description**
ps_python_class_point	Construct a class Point which has a method to calculate distance from another instance of Point.
ps_python_2d_list	Construct a program that prints out [[0, 1, 2], [3, 4, 5], [6, 7, 99]] from the given equation.
ps_python_nested_loops	Construct a program that first prints out 15, then 14, then 12, then 9 and finally 5 on consecutive lines using nested loops.

B.2 Python Animations

Table B.2: List of Python Animations added in OpenDSA from ACOS

Name	**Description**
ae_adl_hello	Animation for printing Hello World to console
ae_adl_variables	Animation showing addition of numbers and variables
ae_adl_simple_arithmetics	Animation showing simple addition and multiplication
ae_adl_arithmetics2	Animation showing complex addition and multiplication
ae_adl_swap	Animation displaying the swapping of two variables
ae_adl_comparison	Animation on comparing variables
ae_adl_logic	Explanation of how the logic operator behaves
ae_adl_greet	Evaluating the input function
ae_adl_input1	Handling of addition and multiplication of two input variables

Continued on next page

Table B.2 – *Continued from previous page*

Name	**Description**
ae_adl_input2	Calculate area of circle given the radius as input
ae_adl_hiscore1	Explanation of simple if statement using game scores
ae_adl_hiscore2	Another explanation of simple if statement using game scores
ae_adl_ifelse	Explanation of if else statements to check if a number is even or odd
ae_adl_ifelifelse	Explanation of else if statements
ae_adl_nested_if	Explanation of nested if statements
ae_adl_while	Demonstration of while loop
ae_adl_for	Demonstration of for loop
ae_adl_strings	Explanation of string concatenation and use of length function
ae_adl_lists1	Explanation of the working of Python lists
ae_adl_lists2	Demonstration of looping through the list
ae_adl_lists3	Multiplying each element in list by a constant
ae_adl_format1	Demonstration of strings formatting using format function
ae_adl_format2	Demonstration of formatting floating point numbers using format function
ae_adl_findmax	Finding the maximum element in the list
ae_adl_functions1	Use of basic functions.
ae_adl_returnvalue	Use of function which returns value.

Continued on next page

Table B.2 – *Continued from previous page*

Name	**Description**
ae_adl_functions2	Use of function which takes multiple input parameters and returns a value
ae_adl_functions3	Use of multiple functions where return value of one function is passed as an input parameter to the second function
ae_adl_recursion	Explanation of recursion using the factorial program
ae_adl_dict1	Explanation of the working of Dictionaries in Python
ae_adl_dict2	Changing values in Python Dictionaries
ae_adl_dict3	Demonstration of looping through Dictionaries to find the maximum value
ae_adl_objects1	Explanation of basic Python Objects
ae_adl_objects2	Explanation of Python Objects by defining a complex class
ae_adl_objects3	Use of multiple objects of a class
ae_python_intro	Add two values and assign the result to variable dollars.
ae_python_assignment	Explanation of how an assignment operator works
ae_python_input	Use of two important built-in functions. The function input is used to read data from the user and print is used to print out the given values.
ae_python_float	Explanation of how type conversion works in Python.
ae_python_if	Explanation of how boolean values True and False can be used together with an if statement.

Continued on next page

Table B.2 – *Continued from previous page*

Name	**Description**
ae_python_while	Another explanation of while loop with if condition
ae_python_for	Another explanation of the for loop
ae_python_function	Working of functions. How the parameter passing and the return value work.
ae_python_list	Using lists with functions
ae_python_split	Explanation of how strings can be converted into a list by using the split method that splits a string with the given separator and returns a corresponding list.
ae_python_dict	Another animation of using a dictionary in Python
ae_python_file	Explanation of how we can read the contents of a file.
ae_python_class1	Create two Bus instances and call methods to change and query the state of the object.
ae_python_class2	Explanation of how the class is implemented and how the methods work.
ae_adl_vals_refs1	Explanation of how variables are not passed by reference to functions
ae_adl_vals_refs2	Explanation of how lists are passed by reference to functions
ae_adl_vals_refs3	Explanation of cloned list objects point to the same objects.
ae_adl_tryexcept1	Demonstration of basic try, except statements
ae_adl_tryexcept2	Demonstration of try, except statements using functions

Continued on next page

Table B.2 – *Continued from previous page*

Name	**Description**
ae_adl_file1	Reading integers from a file
ae_adl_file2	Reading strings from a file

B.3 Java Animations

Table B.3: List of Java Animations added in OpenDSA from ACOS

Name	**Description**
ae_BJ4_ch01_s05_HelloPrinter	Displays a greeting in the console
ae_BJ4_ch01_s05_PrintTester	Prints various strings and numbers to the console
ae_BJ4_ch02_s03_VariableDemo	Explanation of variable creation and manipulation
ae_BJ4_ch02_s05_MethodDemo	Use of various string functions such as length and replace
ae_BJ4_ch02_s06_ConstructorDemo	Use of constructors to initialize objects
ae_BJ4_ch02_s07_AccessorMutatorDemo	Use of accessors and mutators of an object
ae_BJ4_ch02_s09_MoveTester	Create a rectangle object and move it to the right
ae_BJ4_ch02_s10_CopyDemo	Declare two object variables and copy the first into the second. Mutating one object changes the other as well because both refer to the same object.
ae_BJ4_ch03_s01_CounterDemo	Use counter class to increment the counter in order to demonstrate classes

Continued on next page

Table B.3 – *Continued from previous page*

Name	**Description**
ae_BJ4_ch03_s06_BankAccountTester	Another demonstration of the use of classes
ae_BJ4_ch03_s06_CashRegisterTester	More complicated example of classes
ae_JavaTutorial_4_1_3	Demonstrate the use of integer and long
ae_arithmetic_v2	Demonstration of various arithmetic operations
ae_StringExample_v2	Example of strings concatenation
ae_JavaTutorial_4_2_7	Demonstration of nested if else statements
ae_decisions1_v2	Demonstration of else if statement
ae_relational_operators_v2	Explanation of the use of relational operators
ae_comparison_operators_v2	Explanation of comparison operators
ae_while_v2	Demonstration of while loop
ae_do_while_v2	Demonstration of Do While loop
ae_JavaTutorial_4_6_6	Demonstration of for loop with multiple variables
ae_nested_loops_v2	Animation of the use of nested for loops
ae_use_array_v2	Example of using arrays in Java
ae_JavaTutorial_4_7_5	Example of initializing and looping through the arrays
ae_JavaTutorial_4_7_8	Example of looping through an array and the use of break statement
ae_EnhancedForLoopDemo	Examples of more complicated for loops
ae_use_2darray_v2	Explanation of two dimensional arrays
ae_arraylist2_v2	Demonstration of the use of lists in Java
ae_JavaTutorial_4_7_4	Example of looping through a Java list

Continued on next page

Table B.3 – *Continued from previous page*

Name	**Description**
ae_simple_inheritance_1	Basic example of inheritance in Java
ae_inheritance_polymorphism_1	Example of polymorphism in Java
ae_jeg_interface1_v2	Demonstration of the use of Interfaces in Java
ae_Measure	More complicated example of the use of interface with multiple classes
ae_wrapper_class_v2	Example of the use of wrapper classes in Java
ae_exception_v2	Demonstration of exception handling in Java
ae_inheritance_polymorphism_2	More complicated animation of Polymorphism in Java
ae_constant_math_demo	Example of using Math constants like Pi
ae_constant_demo	Multiply float with a constant and cast to an integer
ae_cash_register_constant_demo	Example of final variables
ae_primitive_datatype_demo	The casting of primitive datatypes such as integer and short to double
ae_do_while_demo	More complicated example of the Do While loop
ae_nested_for_demo	Example of nested for loops
ae_while_demo	Example of while loop to sum all the digits of a number
ae_concat_demo	Example of strings concatenation in Java
ae_conditional_demo1	Examples of conditional OR and AND
ae_switch_demo1	Demonstration of the Switch statement

Continued on next page

Table B.3 – *Continued from previous page*

Name	**Description**
ae_switch_demo2	Another demonstration of the Switch statement
ae_unary_demo	Demonstration of the use of unary operators
ae_tostring_demo	Convert double to string and find the number of digits before and after decimal point
ae_for_demo	Basic for loop demo
ae_multidimarray_demo	Use of multidimensional strings array
ae_array_demo	Another example of a basic integer array