Building and Evaluating a Learning Environment for Algorithm and Data Structures Courses

Eric Noel Fouh Mbindi

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Applications

Clifford A. Shaffer, Chair

Thomas L. Naps

Simin Hall

Stephen H. Edwards

Edward A. Fox

March 25, 2015

Blacksburg, Virginia

Keyword: Computer Science Education, eTextbook, Learning Technology, Automated Assessment, Algorithm Visualization, Student Evaluation, Data Analytics

Building and evaluating a learning environment for Algorithm and Data Structure Courses

Eric Noel Fouh Mbindi

ABSTRACT

Learning technologies in computer science education have been most closely associated with teaching of programming, including automatic assessment of programming exercises. However, when it comes to teaching computer science content and concepts, learning technologies have not been heavily used. Perhaps the best known application today is Algorithm Visualization (AV), of which there are hundreds of examples. AVs tend to focus on presenting the procedural aspects of how a given algorithm works, rather than more conceptual content. There are also new electronic textbooks (eTextbooks) that incorporate the ability to edit and execute program examples. For many traditional courses, a longstanding problem is lack of sufficient practice exercises with feedback to the student. Automated assessment provides a way to increase the number of exercises on which students can receive feedback. Interactive eTextbooks have the potential to make it easy for instructors to introduce both visualizations and practice exercises into their courses.

OpenDSA is an interactive eTextbook for data structures and algorithms (DSA) courses. It integrates tutorial content with AVs and automatically assessed interactive exercises. Since Spring 2013, OpenDSA has been regularly used to teach a fundamental data structures and algorithms course (CS2), and also a more advanced data structures, algorithms, and analysis course (CS3) at various institutions of higher education.

In this thesis, I report on findings from early adoption of the OpenDSA system. I describe how OpenDSA's design addresses obstacles in the use of AV systems. I identify a wide variety of use for OpenDSA in the classroom. I found that instructors used OpenDSA exercises as graded assignments in all the courses where it was used. Some instructors assigned an OpenDSA assignment before lectures and started spending more time teaching higher-level concepts. OpenDSA also supported implementing a "flipped classroom" by some instructors. I found that students are enthusiastic about OpenDSA and voluntarily used the AVs embedded within OpenDSA. Students found OpenDSA beneficial and expressed a preference for a class format that included using OpenDSA as part of the assigned graded work. The relationship between OpenDSA and students' performance was inconclusive, but I found that students with higher grades tend to complete more exercises.

This work is supported by the National Science Foundation, under grants DUE-1139861 and IIS-1258471.

Acknowledgments

First, I would like to thank my advisor, Dr. Clifford A. Shaffer, both for his guidance and support. I also want to thank him for giving the opportunity to work on the OpenDSA project since its inception. I thank Drs. Thomas L. Naps, Stephen H. Edwards, T. Simin Hall, and Edward A. Fox for serving on my thesis committee. Their recommendations and suggestions always brought fresh perspectives to my research.

Next, I would like to recognize Daniel Breakiron for his work on the OpenDSA content server, and the members of the Algorithm Visualization research group Sally Hamouda and Mohammed Farghally. I want to thank Dr. Ville Karavirta from the University of Turku in Finland for creating the JavaScript Algorithm Visualization (JSAV) library upon which many of the OpenDSA AVs and exercises are based.

I would also like to show my appreciate to the National Science Foundation, which partially supported my work through grants DUE-1139861 and IIS-1258571, and Virginia Tech for providing me a Graduate Teaching Assistantship.

Thanks to my friends and family for their love and support. Finally, I would like to extend my thanks to all the instructors who used OpenDSA in their classes, and to all the students who used OpenDSA.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Learning technologies in computer science education (CSEd) have been most closely associated with teaching of programming, including automatic assessment of programming exercises [4, 25]. A few recent examples include Problets [50] and Practice-it! [76] for tiny exercises, Web-CAT [28] for testing and automated grading for larger student projects, many teaching environments such as Alice [21] and GreenFoot [37], and even games such as LightBot [1]. Automated grading of programming alone has multiple survey articles [4, 25].

However, when it comes to teaching Computer Science content and concepts, learning technologies have not been heavily used. In earlier days there were some efforts to use programmed instruction, with perhaps the best known being the Little LISPer [32]. But use of the computer itself to teach Computer Science (as opposed to programming) has produced little. Perhaps the best known application is Algorithm Visualization [72, 31], of which there are hundreds of examples. There are also the beginnings of electronic textbooks that incorporate the ability to edit and execute program examples [57, 64]. But in terms of giving feedback to students or interactive exercises, we know of little. The only real example we are aware of is the TRAKLA2 system and its "algorithm simulations" or "proficiency exercises" [55].

Algorithm visualizations support the presentation of dynamic processes that pervade CS content, but which cannot be well represented or simulated using static physical media such as paper or static electronic formats such as PDF, ePUB, or MS-Word. Despite their educational usefulness, AV use in the classroom remains low. Surveys of instructors [53, 71, 47] indicate that one major impediment is the difficulty of fitting AVs into the instructors' current lectures mainly because of inconsistencies between the AVs and the other course materials (textbooks or course notes). Interactive eTextbooks make it easy for instructors to incorporate AVs into their courses as they provide complete units of instruction merging AVs and course content. In addition, eTexbooks contain interactive elements that along with AVs effectively convey dynamic processes, while engaging learners at a higher level of the Bloom taxonomy [13].

OpenDSA is an interactive eTextbook system for data structures and algorithms courses. OpenDSA makes use of Algorithm Visualizations to present the behavior of the algorithm that is being studied. Students are able to modify input data for the algorithm,

and simulate a step-by-step execution of the algorithm. In addition, OpenDSA includes a wide variety of automated assessment activities with immediate feedback. Students in DSA classes generally lack practice activities with immediate feedback due to the prohibitive cost of grading them. That can be solved by AVs and interactive exercises. All student interactions within OpenDSA are recorded, allowing for detailed analysis of how students use the system. Thus OpenDSA combines all of the key components required for effective eTextbooks together in one place: content, interactive visualization of dynamic process, interactive exercises with automated assessment, and interaction logging for system assessment.

The hypotheses guiding the development of OpenDSA include:

1. OpenDSA could result in an increased adoption by instructors since we believe (based on previous surveys) that instructors will be more willing to adopt a complete, ready to use set of materials rather than isolated standalone materials.
2. OpenDSA has the potential to improve student learning by providing them with dynamic presentation of dynamic processes through AVs, and by providing them with many more assessment activities than what is normally possible in a semester course.
3. Data collected within OpenDSA can help us determine if there is a correlation between a particular mode of use of the eTextbook and student performance, and can allow us to determine what features are most important in an eTextbook. The data can guide improvements in the eTextbook.

This thesis is organized as follows. Chapter 2 presents related work involving learning technologies for CSEd that makes use of visualization, automated assessment systems, data collection, and evaluation methods in education. We describe OpenDSA in Chapter 3. Chapter 4 presents our findings from examining the use of OpenDSA by instructors. The impact of OpenDSA on student's learning outcomes is investigated in Chapter 5. In Chapter 6 we give a summary of the contributions and findings of this dissertation and we lay out a blueprint for future work.

Chapter 2

Related Work

The idea of creating interactive class materials for CSEd has been present in the computing education community for many years. We can group efforts to create interactive eBooks for CS education into three categories: Paper textbook with assessment software, Moodle-type sites with assessment, and fully integrated eTextbooks. One important feature of fully integrated eTextbooks is automated assessment with feedback. In CSEd, there have been several interesting works, mainly in programming assessment, but also in algorithm simulation assessment using AVs, and programming visualization. We will describe some relevant automated assessment systems. Other capabilities of eTextbooks include adaptive content presentation and tutoring. Web technologies used in interactive eTextbooks give us more flexibility in the way we present content. We discuss approaches that have been used to display and personalize online educational content. Traditionally, evaluation of learning outcomes relies on observation of students and on measurement instruments such as tests, rating forms, interview schedules, and questionnaires. Online education expands the number and scope of instruments by allowing us to collect a large amount of usage data (interaction logs) in a non-invasive way. Log data analysis and evaluation are discussed in the final two sections of this chapter.

2.1 Paper textbook with assessment software

The first attempts to incorporate interactive elements into class materials consisted of supplementing a traditional hard copy textbook with software (or a website) for assessment and/or visualization.

One of the first attempts to provide CS students with many assessment activities is the Little LISPer written by Daniel Friedman. The Little LISPer content is a programmed text with questions on the left side of the page and answers on the right side. Readers will read the question, think about the question, write down their answer, and then compare their answer to the one on the right side. A collection of exercises without answers are included at the end of every chapter.

With computer labs becoming ubiquitous on college campus, CS educators moved more

toward programming assignment systems associated with a paper textbook. One example of this approach was implemented by Crescenzi and Nocentini. They published a (printed) textbook whose examples and illustrations are closely tied to the AlViE AV system [23]. The textbook was designed to be an extension of AlViE, with the textbook describing the algorithm and the data structures. All illustrations in the textbook were taken from the AlViE GUI. Readers are invited to visualize the execution of all algorithms presented in the book.

A similar project in this category is **Practice-It!** [76], an online learning environment built by Marty Stepp and Jessica Miller at the University of Washington. The majority of exercises in Practice-It come from a textbook written by Stuart Reges and Marty Stepp [66]. Thus, exercises are organized and grouped following the textbook structure. It was designed to help students practice programming in Java. Instructors can create lessons in the system and add problems to it. Most of the problems ask the student to write a short program, but short answer problems are also available. Coding exercises are assessed by compiling and executing students' code. The output is then compared to the output of a model solution (executed with the same input data). Short answer questions perform a string comparison between the student's answer and the (stored) solution. One of the main features of the system is the option for the students to do their assignments directly in Practice-It. To do so, the instructor needs to create a class in the system and to add problems to it, and then the students need to enroll in the class within Practice-It in order for the instructors to receive a notification when a solution is submitted.

These initiatives, while being a first step toward interactive CS class materials, suffered from several shortcomings. While practicing using a paper textbook coupled with an assessment software, learners and instructors need to go back and forth between the textbook and the software. It can be time consuming to use these systems in a classroom.

2.2 Moodle-type Sites with Assessment

Since most educational institutions use Learning Management Systems (LMS) like Moodle to deliver class content, CS educators have attempted to integrate assessment and interactive content into these systems. Rößling and Vellaramkalayil [70] presented a technology that integrates AVs into Moodle-based lessons. Their goal was to support visualization-based hypertextbooks in Moodle. Titterton et al. used a lab-centric mode of instruction for introductory CS courses at UC-Berkeley [78]. Their pedagogical approach was based on the hypothesis that *"Computer science students learn more by doing than by listening to lectures"*. Lab activities are delivered via Moodle and comprise a limited number of AVs along with many assessment activities that students must complete as they progress through a set of class materials. Assessment activities include quizzes (multiple choice questions), programming (modify provided code to comply with

exercise requirements, fill-in the blank coding activity, etc.), and debugging tasks (write test suite that reveals bugs in provided code, find arguments that reveal bugs in provided code, etc.). Quizzes are taken at the beginning of each lab and are graded by the LMS, while programming and debugging tasks are graded by a Teaching Assistant. Their materials are designed for introductory CS courses that aim mostly to develop students' programming skills.

Alharbi et al. [5] integrated digital teaching materials with Moodle. They used eXe (`http://www.exelearning.org/`), an open source authoring system for academic web content using XML and HTML. Their goal was to teach Operating Systems using visualizations. The AVs include a self-assessment feature similar to the one used in TRAKLA2, requiring students to simulate how the algorithm works on the given data set. Students perform a step by step execution of the algorithm on the data set, and receive instant feedback from the system. Students can view the model answer for the given exercise and can submit their solution to the LMS for further analysis by the instructor or teaching assistant.

Due to the nature of Moodle (being a content management system rather than an eTextBook authoring environment), all the projects in this category fell short in producing complete interactive eTextBooks. Moodle manages each class materials file as an individual resource, making it hard to perform cross referencing and seamless navigation between resources. In addition, searching for terms in all the resources is often problematic.

2.3 Fully Integrated eTextbooks

Many in the CS community have expressed the desire to create hypertextbooks, meaning online textbooks that integrate AVs, assessment exercises, and traditional text and images. Such technology is intended to address the shortcomings of systems previously described.

In 2008 Ross [68] described a Perl-based infrastructure for creating hypertextbooks. The infrastructure relies on the Dreamweaver toolkit. Three chapters of a theory of computing textbook were produced using this technology. The system requirements for these hypertextbooks hinder their wide adoption due to browser restrictions (Firefox was the only supported browser), use of Java applets, and specific screen resolutions.

Karavirta [41] integrated XAAL (eXtensible Algorithm Animation Language) AVs into hypertext documents. XAAL is a language designed to allow easy transformation of visualization between various formats. His solution is based on HTML and JavaScript, allowing the hypertextbook to be viewed in any browser without additional plugins. The learner can interact with the animation and draw annotations on it, but the current system does not store the annotation, nor does it support quizzes and tests. This

project can be considered as the forerunner to the JSAV (JavaScript Algorithm Visualization) [44] library used in OpenDSA.

An interesting approach to providing electronic resources is Connexions [9], a web-based system for authoring and sharing educational materials. Knowledge and concepts are organized within Connexions as small modules that can be easily exported and integrated into existing class materials, or can be combined to make new class materials. The available formats are PDF, ePub, and HTML, with only limited support for interactive exercises. Connexions aims to bring together a community of educational resources authors (faculty, industry professionals and students) in order for them to create new materials in a collaborative way. This online community also peer reviews the resources, thus enhancing the overall quality of the digital content in Connexions. Resources in Connexions are structured using an XML schema and are stored in a central repository. Metadata information (module title, authors, keywords or tags, and the relation with other modules) related to each module is stored in a database that enhances the search and retrieval of resources. Connexions' "Course Composer" allows instructors to browse the repository for modules, bundle them into chapters, and create a book for a specific course. Instructors can also add annotations in the eBook. Students need to install the "Module navigator/Roadmap", a web browser plugin (developed by the Connexions team) to visualize the relationships between the concepts in different modules or courses. Connexions documents can embed videos and sounds as well as LabView files (`http://www.ni.com/labview/`). Assessment capabilities are limited in Connexions to static multiple choice questions.

Miller and Ranum's interactive eTextbook for Python programming [57], and "CS Circles" [64] by Pritchard and Vasiga are Python courses for novice programmers. To the best of our knowledge, these are the closest projects to our idea of an interactive eTextBook. Miller and Ranum produced a complete book with features such as embedded video clips, active code blocks that can be edited and run right in the book by the learner, and a code visualizer that allows a student to step forward and backward through example code while observing the state of program variables. They used ReStructuredText and Sphinx (sphinx.pocoo.org) as the authoring system for their project. Sphinx is a document processing tool that makes it easy to output documentation to HTML, LATEX, and other formats. Miller and Ranum's book runs on the Google App Engine and includes assessment activities requiring students to write a small function to perform a single action. The grading system of Miller and Ranum's book is rudimentary and only provides students with simple pass/fail feedback upon completing an exercise. What is missing from this effort is a wider variety of automated assessment with immediate feedback. CS Circles is an exercise-centric online eTextbook for learning Python. They used the WordPress Content Management System as their authoring tool, and the CodeMirror plugin [36] to allow in-browser code editing and automated programming assessment. Both Miller and Ranum, and CS Circles eTextbooks use Guo's online Python tutor [34], an embeddable web program visualization for Python.

2.4 Automated Assessment Systems

Since automated assessment is an important component of our vision for an eTextbook, we have studied existing systems for this purpose.

Naps et al. [61] created a taxonomy to classify students' engagement with AV. The engagement taxonomy includes the following categories:

- *No viewing* indicates cases where no AV is used;
- *Viewing* involves both watching the AV execution and directing the pace of the AV;
- *Responding* involves answering questions about the visualization being viewed;
- *Changing* asks the learner to provide input data to the AV to direct its actions;
- When *Constructing*, students build their own visualization of the algorithm; and
- *Presenting* asks the learners to tell about the AV and gather feedback.

Myller et al. [58] extended the original taxonomy engagement to better capture the differences between student behaviors. They added four engagement levels, including *controlled viewing* and *entering input*, falling between the original levels of viewing and responding.

OpenDSA's interactive exercises are modeled after TRAKLA2 [55] exercises. TRAKLA2 has been one of the most widely used AV collections, routinely used throughout Finnish universities. It provides an outstanding example of operating at the responding engagement level of the engagement taxonomy defined by Naps et al [61].

TRAKLA2 allows the learner to control the visual representation of the data structures manipulated by the algorithm. Learners can "build" a data structure by dragging and dropping GUI elements. TRAKLA2 exercises ask learners to determine a series of operations that will change the state of the data structure to achieve some outcome. For example, students might build a tree data structure by repeatedly dragging new values to the correct locations in the tree. Alternatively, the learner can practice to gain understanding by examining a step-by-step execution of the algorithm (called the model solution). Many TRAKLA2 exercises include some tutorial text along with pseudocode to explain the algorithm, but their main purpose is to provide an interactive proficiency exercise.

Several experiments have shown the pedagogical effectiveness of TRAKLA2. Laakso et al. [52] reported on the use of TRAKLA2 exercises in data structures and algorithms courses at two universities in Finland. TRAKLA2 exercises were incorporated as classroom (closed lab) exercises, and supplemented lecture hours and classroom sessions.

TRAKLA2 exercises were also incorporated into the final exam (one out of five questions) and midterm (they replaced half of a question with a TRAKLA2 exercise in a total of five questions). A year after the introduction of TRAKLA2 in the curriculum, the activity of students increased in all aspects of the course, not only the part involving TRAKLA2 exercises. The average performance in classroom exercises rose from 54.5% to 60.3% (as percentage of exercises completed). Student opinions of TRAKLA2 were collected through an online survey. Eighty percent of the students reported having a good opinion of TRAKLA2. After a year of using TRAKLA2, the opinion of students on its suitability for instruction rose significantly. Ninety-four percent of students agreed that TRAKLA2 exercises helped in their learning process. Their opinion did not change regarding their preference on the medium used to access and perform the exercises: They preferred a mixed (online and traditional classroom) experience.

The usability of TRAKLA2 has been studied through a series of questionnaires and observations in a usability lab [52]. The observations did not find any critical usability issues, and revealed that 80% of the time was used to solve exercises while 14% of the time was used to get acquainted with the system interface. The students had a total of 15 minutes to complete all the exercises. This is an important finding since it is typical that any given AV is intended to be used from only a few minutes up to a couple of hours.

TRAKLA2 has been used to assess the impact of the extended engagement taxonomy (EET) on student performance. Laasko et al. [51] studied whether students using a visualization tool in pairs at the "changing" level of the EET outperformed pairs who used it at the "controlled viewing" level of the EET. First-year students were divided into two groups, both groups using the same textual materials. The treatment group (EET-level changing) used TRAKLA2 exercises while students in the control group (EET-level controlled viewing) viewed AVs containing equivalent information. All the students took an individual pretest and were asked to freely form pairs. They then had 45 minutes to study the learning materials and solve exercises together (using paper and pencil). All the students were also asked to take an individual post-test. The students were videotaped during the experiment. The first analysis of pre-test and post-test scores did not show a statistically significant difference between the treatment and the control group; but a second analysis of the video revealed that some students in the treatment group were not using TRAKLA2 as expected. They did not solve TRAKLA2 exercises, but instead just watched the model solution. Thus they were interacting at the EET-level "controlled viewing". After regrouping the students by creating a third group for the students in the treatment group interacting on the controlled viewing level, the analysis of the scores showed that the students in the treatment group who interacted on a changing level outperformed the controlled viewing groups.

Web-CAT [28] and **CodeWrite** [24] use testing as a mechanism to improve student programming skills. They both require students to write test cases and to submit them along with the actual program. Web-CAT assesses programs on three levels: *code*

correctness, *test completeness score with respect to the code* (how well the tests cover the submitted code) and *test completeness and validity score with respect to the problem* (how well the tests cover the behavior required in the assignment). Web-CAT allows multiple submissions and provides automatic feedback to the students who can then fix the errors in their code based on test results. CodeWrite requires students to write methods that will successfully pass all the test cases, and share correct code among all the student enrolled in the class for peer feedback and comparison.

Environment for Learning to Program (ELP) [79] is a web-based system consisting of a collection of programming exercises in the Java, C# and C languages. It has been used to support teaching introductory programming (Java and C#) courses at the Queensland University of Technology in Australia. ELP provides "filling-in-the-gaps" exercises consisting of a program with missing lines or blocks, and the students are asked to fill in the gaps with the correct lines of code [80]. ELP assesses the student solution statically by gathering some software engineering metrics and comparing them to the model solution. The metrics include the number of variables, statements and expressions, the Cyclomatic Complexity (the number of control paths in the student's solution), the number of unused parameters, etc. Dynamic assessment consists of running student's programs against a set of test data and comparing the output(s).

CodingBat [62] is a web site containing a large collection of programming exercises. Its goal is to give students the opportunity to practice, build their coding skill (in Java and Python) and deepen their understanding of programming concepts. CodingBat exercises provide immediate feedback and are intended to be used as lab exercises or as homework, or for self-study exercises. Exercises consist of short problem statements. Students are asked to write a single function to perform the action described in the problem statement. Upon completion, the student's submission is saved, compiled, and executed, and immediate feedback is provided to the student. One interesting feature of CodingBat is the "progress graphs" that display all student attempts at a problem. For each attempt (submission of the same exercise) the graph shows the submission time along with eventual compile errors (when the program does not run). If the code runs partially, a bar shows the proportion of the code working against the one failing. A working program is represented by a green bar.

JHAVE-POP [33] (`http://jhave.org/jhavepop/`) is a web environment for practicing elementary pointer and linked-list operations. Students can type code snippets in C++ or Java, and JHAVE-POP automatically generates a step-by-step visualization of the contents of memory as each program statement is executed.

2.5 Adaptive and Intelligent Web-based Educational Systems

Intelligent web-based educational systems or intelligent tutoring systems (ITS) have been used for two goals primarily: Curriculum sequencing and assessment support [15]. Curriculum sequencing aims to provide the student with a personalized sequence of learning materials while assessment support aims to help the student solve assessment activities. Adaptive web-based educational technologies use user models to adapt the content of online educational materials [15]. The majority of adaptive and or intelligent web-based educational systems use a coached practice approach [22]. This approach relies on the students being engaged in problem solving tasks, and the software makes decisions based on a student's input and performance.

Hsiao et al. developed **JavaGuide** an adaptive navigation software for **QuizJet** questions [38]. QuizJet is an assessment tool generating parameterized questions for Object-Oriented programing (in Java). Parameterized questions assign random values to parameters for every question, making them similar but significantly different. They allow authors to create more assessment activities with less effort. JavaGuide guides students throughout QuizJet questions based on student progress. It offers a different way for students to access QuizJet questions and implements a combination of "prerequisite-based" and "knowledge-based" approaches. The prerequisite-based approach recommends a topic once the student has completed all the prerequisites, while the knowledge-based approach recommends a topic based on the amount of knowledge already acquired by the student.

A similar effort is BITS (Bayesian Intelligent Tutoring System) [17]. It has been developed to help in teaching computer programing. It relies on a Bayesian network to make decisions regarding navigation support and prerequisite recommendation. The topics in BITS are annotated based on student knowledge. The system recommends a topic once all the prerequisites are met. BITS can recommend that the student read or reread a prerequisite when he/she provides an incorrect answer or indicates to the system that he/she does not understand the question or the concept. BITS can generate learning sequences. The student can specify a specific topic, and then the software will generate a learning sequence for that topic including prerequisites.

2.6 Data Collection and Analysis in Web Applications

Web-server logs are the primary way of collecting data related to web-based services. Web-server logs track four types of transactional data: **access logs**, **agent logs**, **error**

logs and **referrer logs** [10]. Access logs record data about the date and time of the transaction, the IP address of the client, the resource accessed, etc. Agent logs store information about a user's browser and operating system. Error logs store information about requests that are not satisfied by the server. Referrer logs contain information about other websites that link to a particular web server. Bertot et al. presented transactional log analysis as a three-step process comprising planning, data analysis and interpretation activities [10]. The **planning phase** consists of determining what type of information is needed. The **data analysis phase** requires the development or the use of programs that can parse, extract and present valuable information from the logged data, and the **interpretation** activities consist of analyzing the information generated by the data analysis program. Web-server logs are useful to answer questions about users (identified by IP address) and traffic through the server, software used by users, discrepancies in server behavior, and what sites point to the server. Server logs have been successfully used to cluster users based on the number of similar resources they accessed, and to implement a recommendation system [3] in AlgoViz (`http://algoviz.org`), an educational digital library for AVs.

While information provided by server logs is essential when designing or evaluating a web application, they do not capture user interactions with the system. Byrne et al. built a "taskonomy" (taxonomy) of web use [18]. Based on their study of users' interactions with websites, they defined six web tasks:

1. **Use information** refers to one or more activities in which a user tried to use a piece of information from the website. The activities include read, listen to, view, watch, duplicate, download, and display.

2. **Locate** refers to finding an item (word, links, image, etc.) on a page.

3. **Go to** tasks require the browser to change page. They can be initiated by the use of back/forward button, hyperlinks, or typing in a URL.

4. **Provide information** comprises all tasks requiring users to complete and submit a form.

5. **Configure** tasks change browser aspect (resize).

6. **React** (to environment) tasks require the user to respond to a demand from the browser/website like closing a dialog box, clicking on a particular button, etc.

Standard web-server logs do not capture data related to a user's interaction with website; these have to be done on the client side. One common way to uniquely identify and collect user information on the client-side is through the use of "cookies". The Internet Engineering Task Force RFC 2109 defines a cookies as a *"way to create a stateful session with HTTP requests and responses"* [49]. Typically, cookies contain a user name and/or

password, but the content can be modified to include any other relevant information. Scripting languages have also been used for data collection. An early client-side data collection tool is Listener [29]. **Listener** was developed using the AppleScript scripting language to collect information about a user's navigation through a website while using the Netscape browser. One advantage of Listener was the fact that it was able to run even in the absence of a web server. Modern website or Web 2.0 applications mostly use JavaScript-based scripting for their dynamic content and communication with the server. They provide a richer set of interactions between users and websites, generating a huge amount of data. Kiciman et al. developed **AjaxScope**, a tool for *"instrumenting and remotely monitoring the client-side execution of web applications within users' browsers"* [46]. AjaxScope is based on AJAX (Asynchronous JavaScript and XML) and collects data about JavaScript programs running on a webpage to detect performance issues, help with runtime analysis, and usability evaluation. Examples of AjaxScope data include errors in JavaScript code, infinite loop detection, inefficient string concatenation, mouse movements, and key strokes. AjaxScope has been designed with the goal of helping web developers debug and monitor web applications.

2.7 Educational Technology Evaluation Methods

The educational effectiveness of educational technology needs to be assessed in order to lower the barriers for technology adoption. Bullock et al. [16] present several models used when evaluating educational technologies.

- **Quasi-experimentation comparative** compares innovative technology to a traditional or different method of instruction. Bullock et al. note that when comparing technological innovation in education, comparative studies are most commonly used [16].
- **Objective-based evaluation** centers on having students reach pre-specified objectives. They usually use pre- and post-treatment measures like test or surveys.
- **Goal-free evaluation** aims to assess anticipated and unanticipated impacts of using a technology. This method is less formal than the first two, and uses open-ended measures like personal observation, focus groups, etc.
- **Illuminative evaluation** focuses on describing and interpreting the impact of the technology without measuring it.

When evaluating a new educational technology, evaluators can use the following instruments to collect qualitative and quantitative data:

- **Previous experience with computer or related/similar technology questionnaire:** These are used to gauge students' skills and attitude toward computer assisted education.
- **Task experience questionnaire:** They aim to assess students' current experience with the domain or the subject before starting the study. **Student confidence logs** are checklists by which students rate their ability to perform a specific task or their understanding of a principle. They are usually collected before exposing the students to a new technology.
- **Knowledge quizzes:** They are given to the students before and after the treatment, and are used to assess the impact of the technology on a specific learning objective.
- **Post-task questionnaires** are answered by students right after the intervention. They aim to collect students' attitude and feelings during the experiment, and specific information about the technology.
- **Focus groups or interviews** (with a sample of students) are used as a validator for the written answers (from post-task questionnaire) and as a way to collect information about aspects of the evaluation that were not included in the post-task questionnaires.
- **Learning resource questionnaires** find what learning resources (book, software, and website) the students use and information to evaluate their usefulness.
- **Post-course questionnaires** are asked at the end of the course and can be used when a learning resource questionnaire is not suitable.
- **Final exam performance** indicates students' performance on questions pertaining to the experiment during a final exam [26].

Formative evaluation is commonly used when evaluating educational software. During the design phase, evaluators collect data (often preliminary user feedback) and work with developers to improve the software. Data collection methods are mostly qualitative (surveys, questionnaires and interviews) [82]. Field testing is inevitable because many problems can only be discovered when the software is used under realistic conditions [26]. In that case evaluators can use a combination of evaluation models.

Summative evaluation is less used because it is done after the tool has been introduced. The metric mostly used in summative evaluation is learning outcome but it is difficult to draw a conclusion on a technology based only on that metric because a wide number of factors (motivation, coercion, teaching style) can affect learning outcomes [26].

OpenDSA is a fully integrated eTextbook with automated assessment, thus allowing students to perform all book-related activities within the same environment. In addition, OpenDSA makes use of some of the successful assessment technologies previously described in this section, notably the TRAKLA2-style algorithm simulation exercises. In this dissertation, we used a combination of the data collection and research methods mentioned above for OpenDSA's evaluation.

Chapter 3

OpenDSA

3.1 Design goals

The OpenDSA project's goal is to build a complete set of materials and infrastructure to support courses with content on data structures and algorithms (DSA). The OpenDSA vision for an interactive eTextbook was described and presented in [74, 73]. Materials are provided under the MIT open source license. OpenDSA allows instructors to select from among the available materials those that they wish to integrate into their course. OpenDSA content is organized into modules. Modules are the basic unit within OpenDSA and correspond to sections in traditional paper books. They cover a specific topic or part of a topic.

OpenDSA's materials satisfy the following requirements:

- Modularity and reusability of book parts
- Platform and device independence
- Non-proprietary format
- Dynamic content
- Interactive simulations to promote active learning, problem solving, and critical thinking
- Interactive assessment activities with automated feedback
- Storage of student's exercise submissions or attempts and grades

OpenDSA's modules were designed with the following educational goals in mind:

1. *Leverage technology to present the material in a way that makes learning easier.*

   Several studies have measured the positive attitude towards AVs by educators and learners [61, 45]. Despite the fact that measuring pedagogical effectiveness of AVs is not an easy task, several AV systems have led to the improvement of learner's performance [40, 81].

2. *Include activities to engage and motivate students.*

AVs can engage students at different levels. The taxonomy defined by Naps et al. [61] defines a hierarchy for learner engagement. Higher engagement levels tend to provide a richer way for learners to interact (changing input data, responding to questions, etc.) with the technology.

3. *Include assessments to evaluate how well students learn.*

   Each module should include mechanisms for students to self-gauge how well they have understood the concepts presented. Self-assessment can increase learner's motivation, promote students' ability to guide their own learning and help them internalize factors used when judging performance [56, 6].

Each module should then integrate:

- Text and images
- Dynamic presentation (visualizations and interactive exercises)
- Rich, automated assessment exercises

3.2 The OpenDSA Authoring System

Since OpenDSA aims to integrate text, dynamic content, and automated assessment, we needed an authoring tool to glue all these elements together. We wanted a tool that would allow us to manipulate ASCII-format documents since most binary documents formats are proprietary and have some cross-software and device compatibility issues. Previous experiences with hypertextbook authoring have indicated that the authoring environment should be a standalone system not attached to a particular web development framework or environment [68]. We briefly attempted to create our own markup language and document processing engine. But we soon decided to use an existing open source system that was successfully used in [59] and [57]. We choose the reStructured-Text (ReST)[1] is a markup language originally designed to produce Python program documentation. Sphinx is the engine that converts ReST files to HTML, LaTeX or PDF documents according to specific "directives". A directive in Sphinx is equivalent to a macro in Microsoft Office documents or a markup command in LaTeX. A directive is implemented by a small Python program that controls how Sphinx should process the text included in the directive. Sphinx has many features of traditional document processing software such as cross referencing, generation of tables of contents, insertion of images, etc. One great advantage of Sphinx for our purpose is that you can include

[1]ReST (ReStructured Text) markup language and its compiler Sphinx (sphinx.pocoo.org) as our authoring tool for content.

ReST is not to be confused with the REST design model for web applications.

"raw HTML" code in your ReST document and it will be present in the output (HTML) file. This is a useful feature for creating dynamic webpages. We wanted our eTextbook to be platform independent so we opted for HTML5 output files and decided to write custom directives to fit our needs for dynamic webpages. We next describe a few directives that We wrote to support our content development efforts. While ReST supports the ability to pass through raw HTML to the output HTML pages, we do not actually want module authors to do so directly. Many of our custom directives allow convenient embedding of dynamic (JavaScript-based) content. Other directives augment support for creating book-length documents that Sphinx currently lacks.

- **Embedded dynamic content:** To increase the reusability of visualization components we decided to embed separate web pages containing dynamic content inside the book. The **avembed** directive embeds an HTML iframe containing an AV and/or interactive exercises inside a ReST document while the **inlineav** directive directly inserts an AV in the generated HTML page.
- **Code snippet support:** We designed OpenDSA to be programming language independent. We provide instructors with the feature of embedding samples of code located in separate files.
- **Numbered cross references:** This displays module numbers when citing other book elements as it is done in other authoring systems such as LaTeX.
- **"ToDo" text boxes:** During the development phase of the project we find it helpful to be able to specify items for module developers "to do" later. This directive provides necessary markup and generates a page listing all TODO elements.

While being an excellent tool for document authoring, ReST and Sphinx lack some functionality that are useful to book authors such as document objects (tables, figures, and equations) with numbering and displaying those numbers when citing objects. I created a **document processor** to fill that gap. The OpenDSA processor parses ReST files to generate a table of document objects and their respective numbers. The table is used by our custom *numbered cross references* directive to display numbers as hyperlinked text when referencing document objects. We also included support for displaying mathematical notation within OpenDSA webpages through Mathjax [20], a library that allows browsers to display math symbols defined using LaTeX markup.

3.3 AVs and Assessment

Assessment activities are the cornerstone for our model of an interactive eTextbook, as we believe that they have the greatest impact on student learning and performance.

Our assessment activities use visualization in an interactive way to help address the lack of practice activities found in most DSA courses. These activities are built using the **JavaScript Algorithm Visualization (JSAV) library** (`https://github.com/vkaravir/JSAV`) developed by our collaborators in Finland. The goal of JSAV is to provide a modern JavaScript framework for building interactive algorithm visualizations. The assessment part of each module should include one or many "proficiency exercises" and quizzes with automatic feedback.

Proficiency exercises are activities that require students to demonstrate how proficient they are with an algorithm by simulating its behavior. Proficiency exercises aim to verify that the learners understand how a given algorithm works. A student's proficiency is measured by one or a set of exercises using visualization that will ask the learner to:

- Predict the next step of the algorithm
- Provide input data that will make the algorithm behave in a specific way
- Simulate the execution of the algorithm or the behavior of a data structure (TRAKLA2-like exercises [55], pioneered by our collaborators)

Quizzes: Multiple choice questions and/or short answer questions are used to verify that the learner remembers definitions and understands concepts. They can be distributed throughout the module or at the end. We use the **Khan Academy** exercise framework (`https://github.com/Khan/khan-exercises`) to develop exercises using HTML and jQuery. The **Khan Academy** exercise framework also allows us to build more creative interactive exercise with or without AVs. The advantages of the framework include:

- Controlled randomization of questions
- Parameterization of exercises
- Automatic feedback
- Hints
- Built-in support for standard questions: Multiple choice, short answer

Programming exercises: OpenDSA includes in-browser code editing capability. Such exercises allow the student to perform programming assignments inside OpenDSA and get immediate feedback.

Content Dissemination

Our AVs and exercises are standalone artifacts that can be used independently of the eTextbook. I implemented an **oEmbed** (`http://oembed.com/`) web service for OpenDSA's AVs and exercises. oEmbed is an open format which is to allow embedding

content of a website on third party sites. The oEmbed endpoint will allow instructors to seamlessly embed our AVs and exercises into their own online course materials. An oEmbed endpoint has also been implemented by our collaborators to disseminate TRAKLA2 exercises rewritten using JSAV [42].

3.4 Infrastructure

This section briefly describes the current status of the OpenDSA infrastructure. OpenDSA implements a client-server model. The client consists of HTML documents created using Sphinx, and the backend is a web application that I built using the Django web framework with a MySQL database for data persistence.

3.4.1 Client-side technologies

HTML5 is the latest revision of the HTML standard and introduces elements and attributes that allow developers to create more dynamic and functional web pages. HTML5 is supported by all major browsers as well as most tablets and many smartphones. OpenDSA makes heavy use of HTML5's **localStorage** feature. This is a set of functions and procedures that allow web developers to store data in a web browser. We use localStorage to store all user interactions with the eTextBook and all data received from the server. This allows the frontend to keep all transactions in a coherent state, and allows interactions with the server to be cached and transferred in batches to reduce network traffic.

We used JavaScript as our scripting language and Asynchronous JavaScript and XML (**AJAX**) for communication with the web server. JavaScript is natively supported by all modern browsers and when coupled with AJAX, allows us to send (asynchronous) event-based requests to the backend through HTTP methods (HEAD, GET, POST,OPTIONS, DELETE). We encode all the data exchanged between the front end and the backend in JSON. JSON is a subset of JavaScript that simplifies the parsing and handling of objects within the browser. It is also supported by most modern web browsers. Daniel Breakiron was the primary developer for the client-side ("frontend" implementation).

3.4.2 Data collection server technologies

We built OpenDSA's data collection server using the Django web framework. Django is an open-source Python Web development framework that implements the model/-view/controller architectural pattern.

We wanted our data collection infrastructure to be easily integrated into course management systems (CMS). We have integrated the OpenDSA data collection server into A+, a CMS created and used by our collaborators in Finland [43]. We use A+ to manage the relationship between courses, eTextbooks, instructors and students. A book is attached to a course instance that is taught by an instructor. When using a particular book instance, students are automatically enrolled in the associated course. There are also OpenDSA eTextbook instances that are created for testing purposes.

OpenDSA uses the **REpresentational State Transfer** (REST) design model for web applications [30]. REST's goal is to simplify use, development, and deployment of web applications. REST requires applications to comply with the following constraints.

- **Use of Client-Server architecture :** OpenDSA is composed of a content server (set of HTML documents created using Sphinx) that delivers HTML pages that are viewed on the user's browser. The HTML pages are clearly distinct from the actions of the data collection server (Django application). Neither the user's browser nor the content server are involved in any data persistence operations required by the data collection server. This constraint allows the components to work independently.
- **Stateless:** Each message between the user's browser and the data collection server contains all information necessary (in the form of JSON objects) for the service to perform its function, with no client context being stored on the server between requests.
- **Cacheable:** OpenDSA's webpages uses the localStorage object to cache information from the server's responses.
- **Layered system:** The data collection server and user's browser communicate via an API. The API, based on the Model/View/Controller architecture pattern implemented by Django, makes the system scalable.
- **Uniform Interface:** Within OpenDSA, all communication between the browser and the data collection server is mediated through HTTP/S GET and POST methods.

We used django-tastypie (`http://tastypieapi.org/`) to build OpenDSA's data collection API. An overview of the API is shown in Figure 3.1. Cross-site HTTP requests are authorized only to sites explicitly listed in the data collection server configuration settings. This allows us to control which sites can serve OpenDSA content "textbooks" to a given instance of the data collection system. Likewise, it also allows third parties to independently set up just an OpenDSA content server (and find a willing partner with a data collection server that will accept their input) or both a content server and data collection server.

The API exposes the following services.

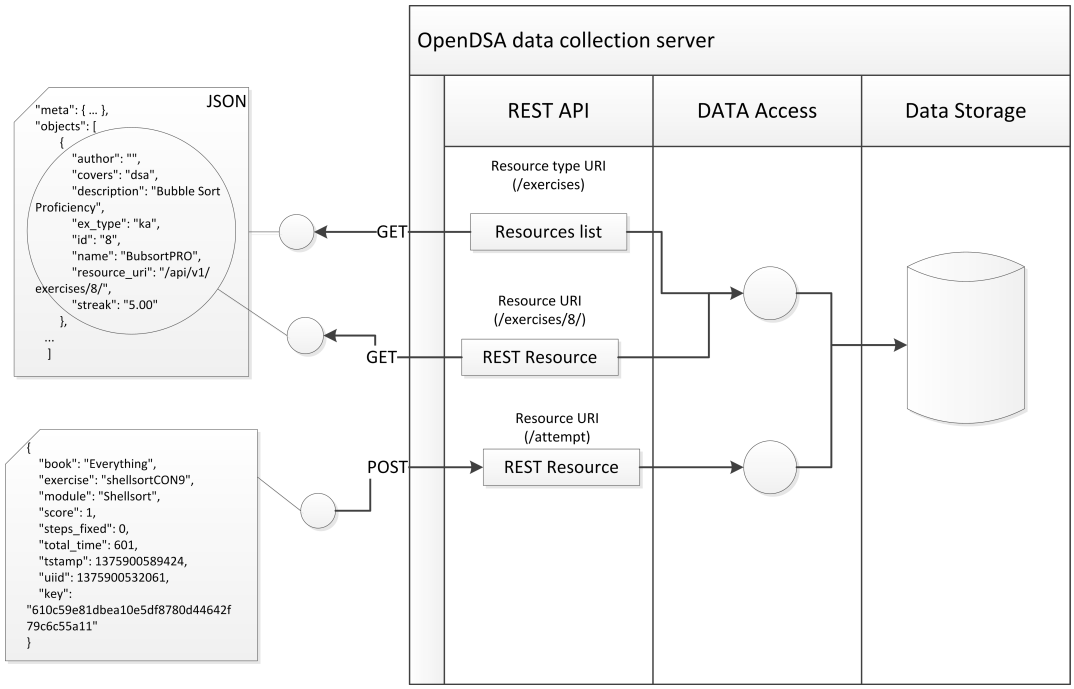

Fig. 3.1: Overview of OpenDSA data collection server API.

1. **Authentication** allows users to login and/or logout. OpenDSA uses a key authentication mechanism. In that model, a new key is generated every time the user connects to the server. After the initial login only the key is used to authenticate the user as shown in Figure 3.2. This mechanism allows us to forbid connection from a different host at the same time by the same user.

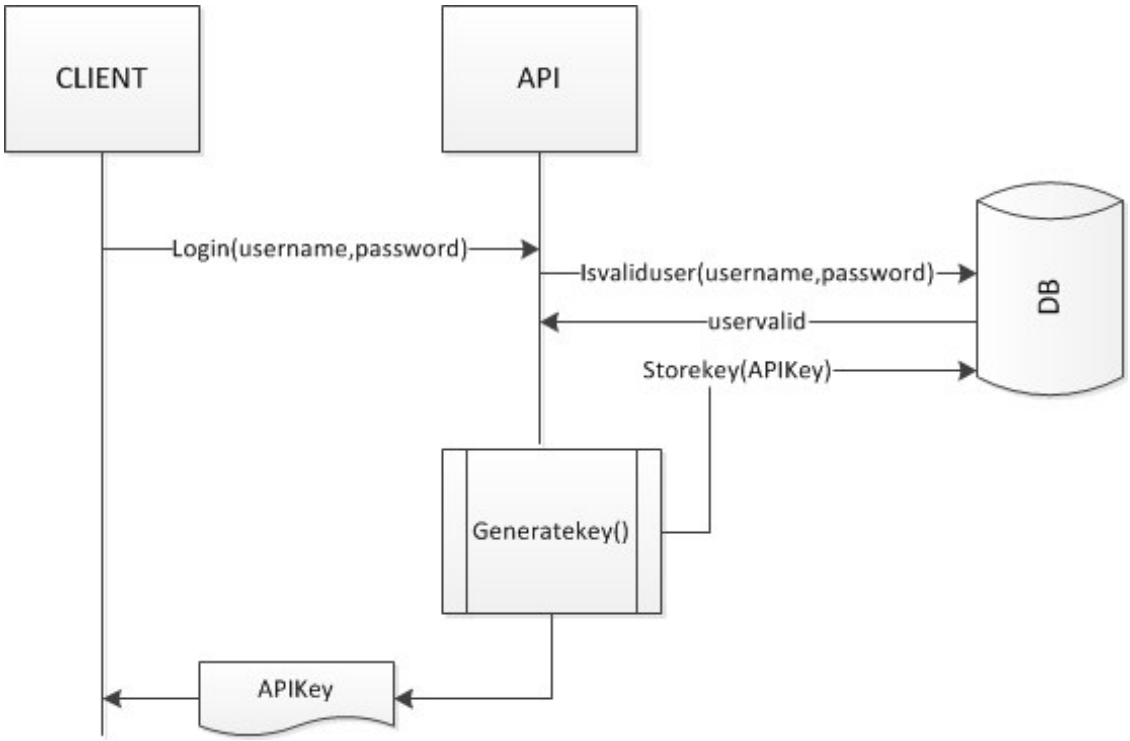

Fig. 3.2: Sequence diagram for the authentication service

2. **Book components management**. In OpenDSA, book components such as module information (name, author, topic, etc.), and exercise information are stored into the database through the API. The book components management service is responsible for that task. The sequence diagram in Figure 3.3 describes how it works.

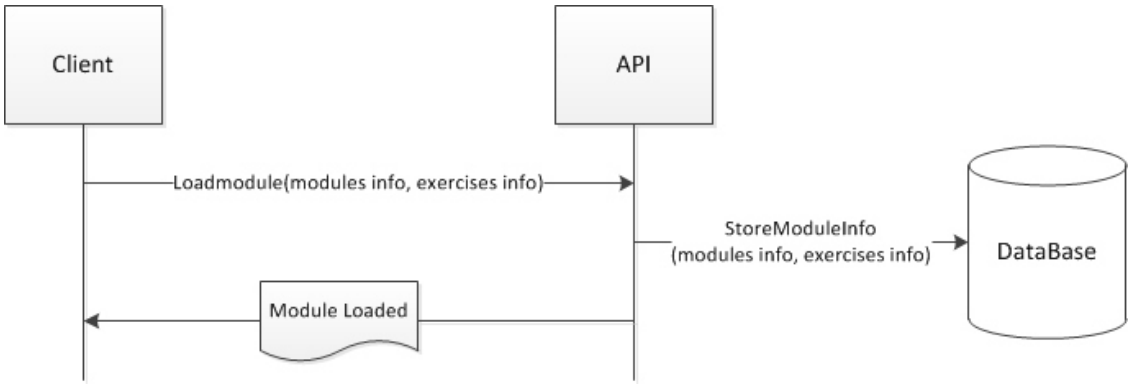

Fig. 3.3: Sequence diagram for the book components management service

3. **Exercises attempt management** is responsible of storing all user-exercise in-

teraction data such as time of the attempt, exercise name, attempt correctness, etc., as shown in Figure 3.4.

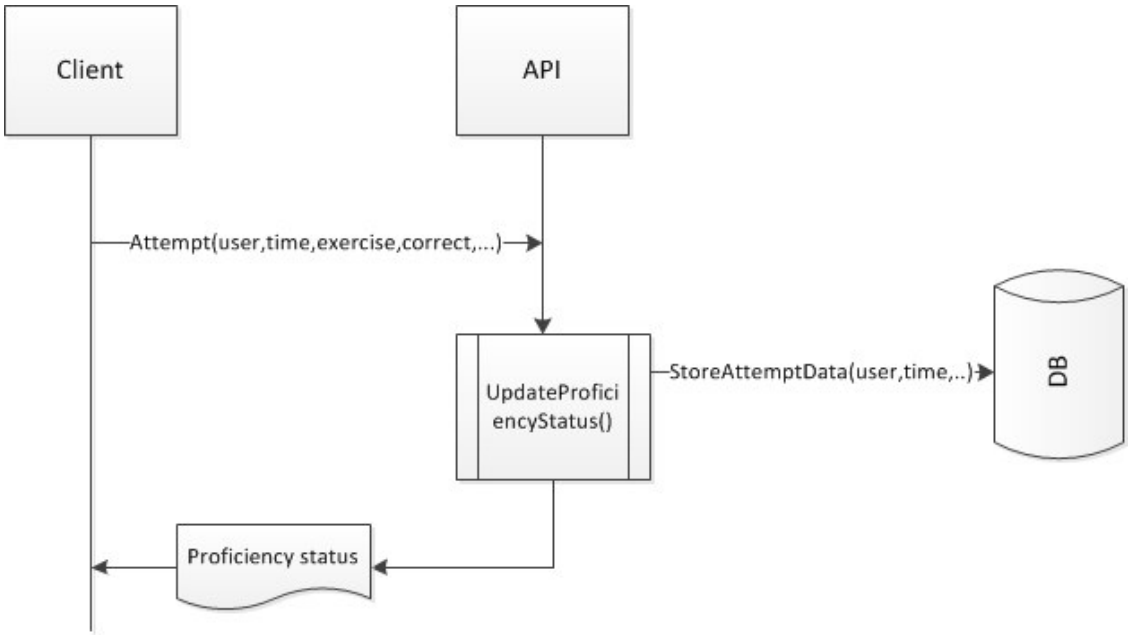

Fig. 3.4: Sequence diagram for the exercises attempt service

4. **Frontend events management** allows us to record every action that occurred on the frontend, like button clicks made by the students, windows and HTML iframe openings and closings, etc.
5. **Student progress management** provides the frontend with the student progress on a specific exercise, such as how close the student is to being proficient.
6. **Student proficiency management** returns the student's proficiency status on an exercise or a module.

3.5 Usability evaluation

User-reported incidents is a technique used during remote (post-deployment) usability evaluation. It requires that users report bugs and usability problems as they use the software. Castillo et al. [19] showed that users can effectively recognize, report and rank critical incidents. In addition to user-reported bugs, we also assessed user frustration when performing post-deployment usability evaluation of OpenDSA.

1. **User frustration**

We used surveys to collect users' overall experience with OpenDSA. Student reactions were highly positive, however they reported some frustration with the visual elements (tracking their work) sometimes being out of sync. It is important for students to see their progress in real time and to have the assurance that their work is being recorded by the data collection server.

- **OpenDSA feedback infrastructure**

  Several visual feedback items are displayed every time an interactive activity is completed.

  Proficiency for mini-slideshows is indicated by the appearance of a green check mark on the right or left side of the slideshow container as shown in Figure 3.5. When a mini-slideshow is completed, the information about its completion are sent to the data collection server. If the status of the submission is SUBMITTED (to the data collection server), a "Saving..." message will appear beneath the check mark but will be hidden once the status changes to STORED (in the data collection server). If the status is set to ERROR (communication issue between the client and the data collection server), a warning indicator will appear (to draw the user's attention to the exercise) and the saving message will be replaced by an error message and a "Resubmit" link which allows the user to resend their score data without redoing the exercise.

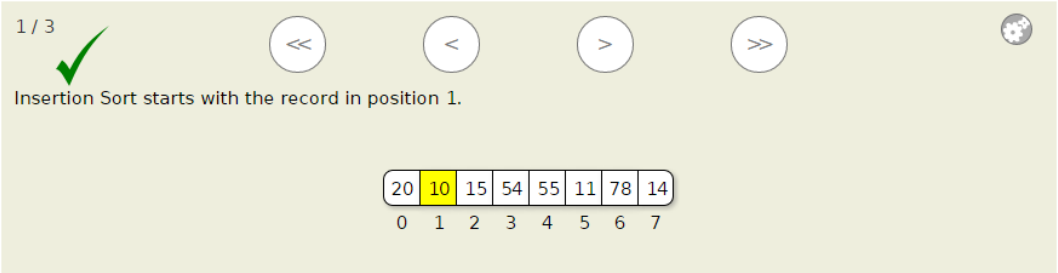

Fig. 3.5: Insertion sort mini-slideshow with green check mark

Proficiency for embedded exercises is indicated by the color of the button used to show or hide the exercise. Red indicates the user is not proficient as shown in Figure 3.6. Yellow indicates the user's score has been submitted to the data collection server or an error occurred during the submission. Green indicates that the user is proficient and their proficiency has been verified and stored by the data collection server (see Figure 3.7).

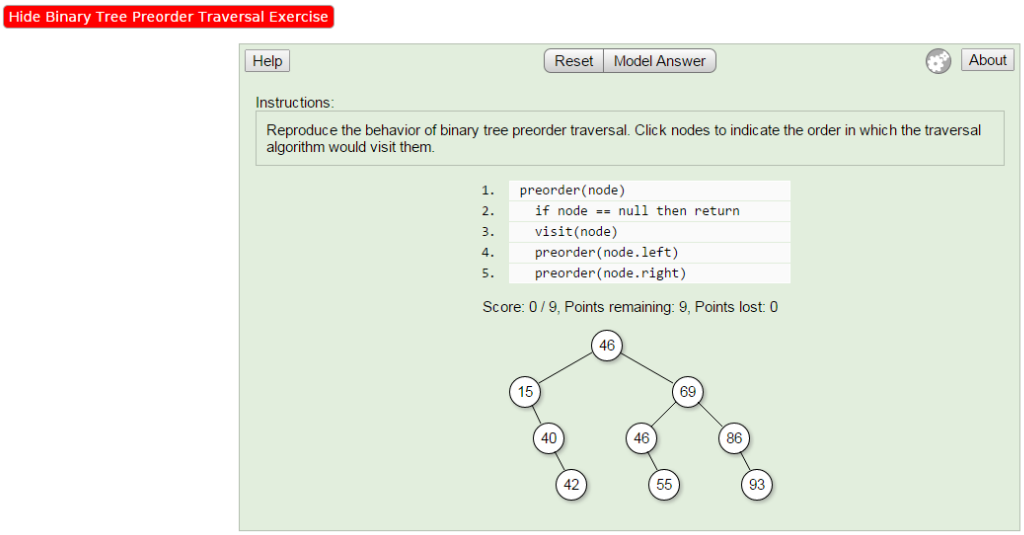

Fig. 3.6: Tree preorder traversal exercise with red show/hide button

Proficiency for modules occurs when a user obtains proficiency for all the required exercises in a module. It is indicated by the words "Module Complete" that appears in green at the top of the module page (see Figure 3.7). If "Module Complete" appears in yellow, the user has obtained proficiency with all the required exercises but one or more of them have not yet been successfully submitted and verified by the data collection server. In general, to obtain module completion a user must complete all exercises marked as "required" in the book instance configuration file. If a module does not contain any required exercises, module completion cannot be obtained unless the configuration file sets the variable "dispModComp" to "true" for the given module. Inversely, if "dispModComp" is set to "false", module completion will not be awarded even if the user completes all the required exercises.

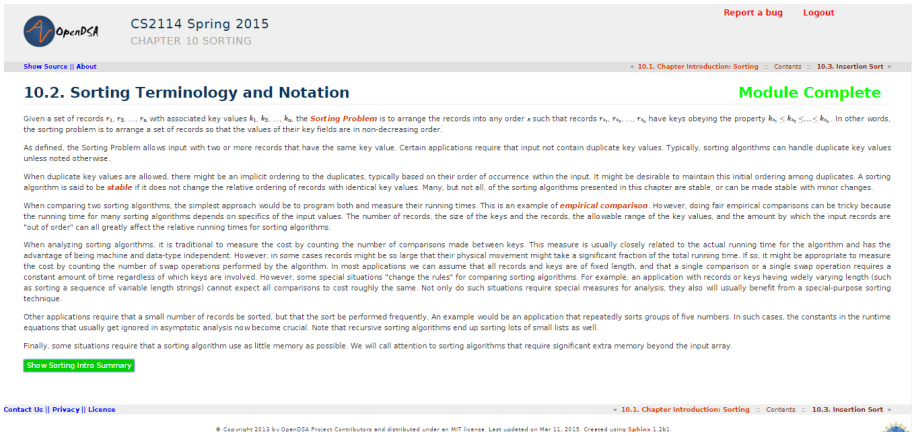

Fig. 3.7: Module page with proficiency indicator for module and exercise

Table of contents proficiency is indicated by a small green check mark next to a module showing that the module has been completed as shown in Figure 3.8.

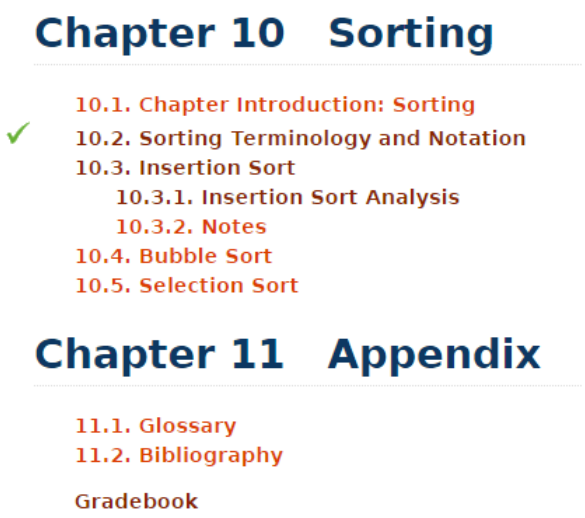

Fig. 3.8: Section of table of content with module proficiency indicator

The gradebook page shows the score for exercises and modules with which the user is proficient; they are highlighted in green. Figure 3.9 shows a gradebook page with completed exercises highlighted in green.

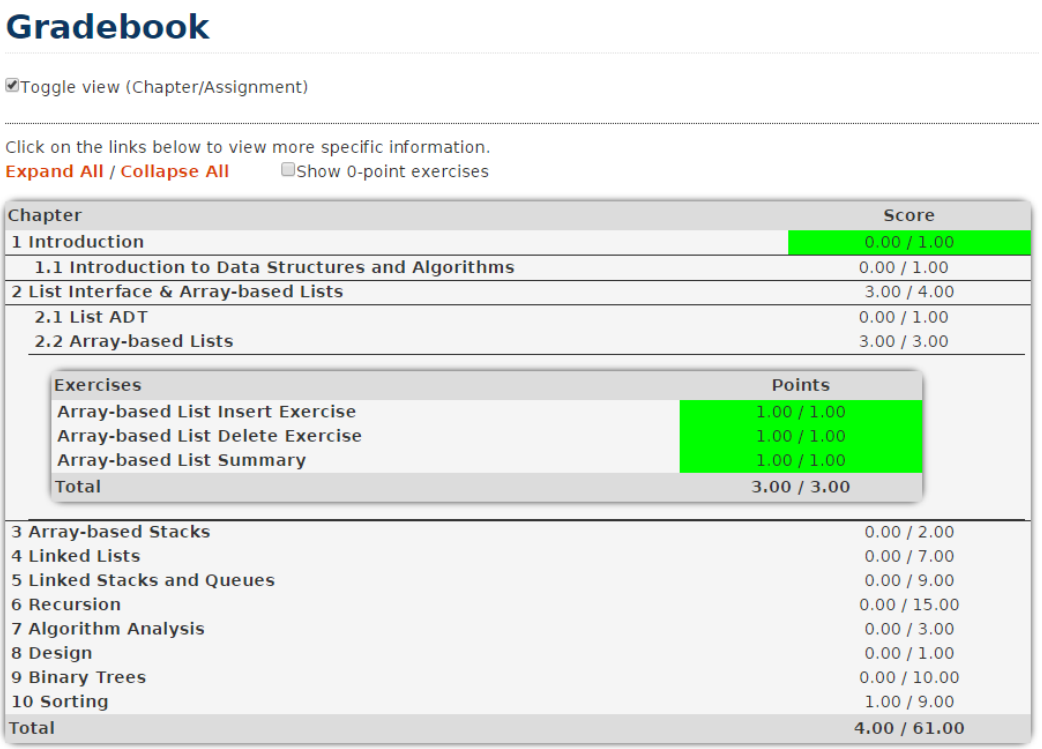

Fig. 3.9: Gradebook page

- **Local proficiency cache issues**

  The primary purpose of the local proficiency cache is to allow anonymous (guest) users to maintain their progress and to allow OpenDSA to function without the data collection server, but a secondary purpose is to make pages more responsive for logged in users. Locally caching user proficiency status (using browser's localstorage) allows the page to immediately display the proper proficiency indicators rather than waiting for a response from the server. All proficiency related data stored inside browser's local storage are identified by the book instance. We noticed several cases of local storage data out of synchronization with the data collection server. The scenarios causing the corruption are when students use different book instances in the same browser and there is a communication error or delay (with the data collection server) for one or all the book(s), and when several users use the same browser. Symptoms of local cache corruption include proficiency indicators not showing upon completion of an interactive item or proficiency indicators disappearing after reloading a module's page. Clearing the browser's localstorage typically fix the issue.

- **Student reaction to proficiency indicators**

  Students are particularly attentive to proficiency indicators. We asked the students in CS3114-F14 if the proficiency indicators changed their behavior,

and how. We received 33 answers. Eighth students reported that proficiency indicators did not affect them while working, while 25 acknowledged the importance of proficiency indicators. Students reported that proficiency indicators allow them to check where they stand in regards to graded assignments. Some students also mentioned that it motivated them to do more exercises because of the instant "gratification/validation". Based on students responses, it was no surprise that any problem with proficiency indicators would be immediately noticed and reported by students. Students expressed their frustration whenever there was an issue affecting the (timely) display with the proficiency indicators.

2. **User reporting of bugs**

   We allow users to automatically report bugs or incidents while working on OpenDSA content (see Figure 3.10). We used Wilson's usability scale [83] to rank the reported bugs by their gravity level, defined as follows:

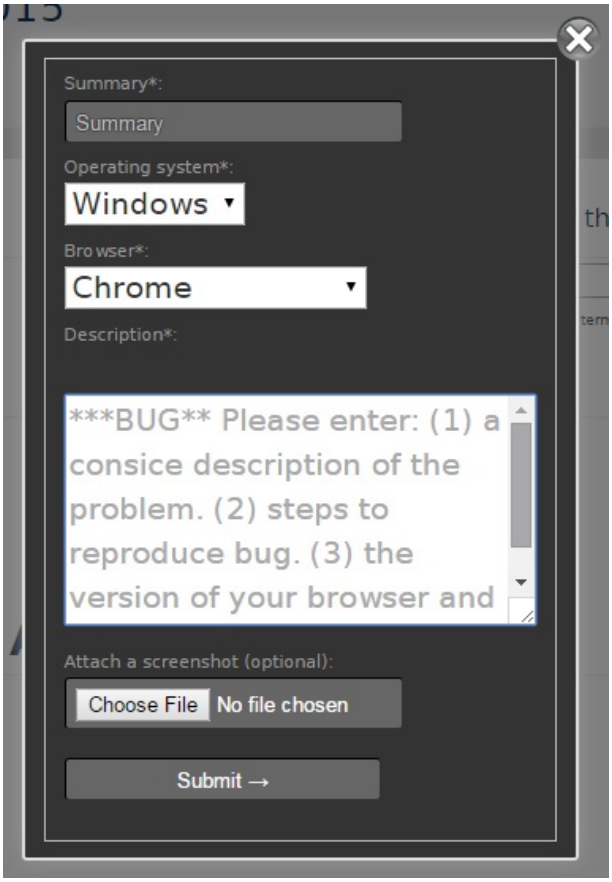

Fig. 3.10: Bug report form

Level 1: Catastrophic error causing irrevocable loss of data or damage to the hardware or software. The problem could result in large-scale failures that prevent many people from doing their work. Performance is so bad that the system cannot accomplish business goals.

Level 2: Severe problem, causing possible loss of data. User has no workaround to the problem. Performance is so poor that the system is universally regarded as unusable.

Level 3: Moderate problem causing no permanent loss of data, but wasted time. There is a workaround to the problem. Internal inconsistencies result in increased learning or error rates. An important function or feature does not work as expected.

Level 4: Minor but irritating problem. Generally, it causes no loss of data, but the problem slows users down slightly. There are minimal violations of guidelines that affect appearance or perception, or mistakes that are recoverable.

Level 5: Minimal error. The problem is rare and causes no data loss or major loss of time. Minor cosmetic or consistency issue.

43 bugs have been reported since Summer 2014. More than 700 students used OpenDSA during the same time frame. See Table 3.1 for information about bugs distribution across gravity levels.

Table 3.1: Bugs reported grouped by gravity level

Gravity level	# of bugs
Catastrophic error	0
Severe problem	2
Moderate problem	10
Minor problem	14
Minimal error	8

The majority of incidents reported by users stemmed from errors with local proficiency caching issues resulting in proficiency indicators not updating. It resulted in some frustration for the users. Students reported bugs about not receiving credit for exercises, completion check not working, etc. Clearing the browser's local storage and/or cache typically solved the problem.

The most severe problem that we encountered were caused by a broken SSL certificate signature chain which resulted in the lack of communication between the client and the data collection server. The problem symptoms were as follows: students could not log in, clearing the cache did not solve the problem and the problem occurred on most browsers. The JavaScript console showed that all requests were

made through a secure URL (https), but the requests failed with the following message "Failed to load response data network error, ERR_INSECURE_RESPONSE". We estimate that about a third of all users (during Fall 2014) encountered the problems. Reinstalling the certificate fixed the problem.

3. **Findings and Recommendations**

   Usability testing showed us that users are attentive to the visual proficiency indicators and that they are concerned when they are out-of-sync. To minimize that risk, we are working on improving our caching mechanism. In addition, we added an entry in the FAQ page containing instructions about clearing browser's localStorage. Most user-reported incidents have been fixed and most of the bugs we are receiving now are related to inconsistencies in the text or AVs (that is, problems with specific content, not with the system infrastructure).

Chapter 4

OpenDSA in the classroom

4.1 Driving Hypotheses

Through the years, successive surveys have shown positive views on AVs and other learning technologies by CS educators [61, 71]. But that positive view has not resulted in widespread integration and use of AVs in the classroom. Some reasons for the low use of AVs in the past have included technical limitations such as limited platform independence, limited control by teachers over the AV (when they are not AVs developers), difficulty to find good AVs, or inconsistencies between the AVs and the other course materials (textbooks or course notes) [53, 71, 47]. We built OpenDSA with the goal of overcoming these obstacles.

Assessment systems with automated feedback provide more practice for students and less grading time for instructors. In addition, students can self-teach and evaluate their knowledge of the mechanical aspects of a topic through the assessment system included in the eTextbook. We think instructors interested in experimenting with new pedagogical approaches (blended learning, "flipped" classroom, etc.) can take advantage of the affordances provided by eTextbooks to implement such approaches.

We hypothesize that:

- ***"The model of an electronic textbook as a complete unit of instruction with AVs and interactive exercises will address issues that hindered AV use under prior models."***
- ***"The new pedagogical features supplied by OpenDSA will support pedagogical changes in the way courses that use OpenDSA are conducted."***

To test our hypothesis we will answer the following research questions:

1. How well does OpenDSA's design address obstacles to the use of AV systems in the classroom?
2. What pedagogical changes occur when OpenDSA is used in the classroom?

In this chapter we take a look at how OpenDSA fares in regard to previously identified AVs adoption roadblocks [53, 71, 47]. We also study how OpenDSA had been used for teaching so far, and investigate early trends in OpenDSA use.

4.2 OpenDSA vs. AV adoption obstacles

Knobelsdorf et al. [47] divided obstacles to AV systems' adoption into four categories:

1. Learning and assessment media difference: AV systems have traditionally been developed to support learning without integrated assessment. This situation had forced students and instructors to switch back and forth between tools. Knobelsdorf et al. [47] claimed that the disconnect between AV systems and assessment systems is one of the reason why teachers do not observe visualization tools to be educationally effective.
2. AV tools run on limited platforms: Many AVs systems run only on specific systems, thus forcing instructors to use systems they are not familiar with. Instructors willing to use a visualization need to make sure that it is compatible with their computing systems. They sometimes have to install additional software to be able to use the AV. This situation can be one reason why instructors reported trouble with integrating AV material into computer science courses [71].
3. AV creation is a "divided" activity: AV systems development has mostly been limited to small groups of AV developers with few crowd sourcing attempts. The main consequence of this situation is a tight coupling between the AV developer, the AV, and how the AV is used. This situation makes AVs too rigid to be adapted to different teaching situations [47, 60].
4. The educational effectiveness of visualizations: Some instructors still debate the educational effectiveness of AV systems [75, 35].

4.2.1 Addressing learning and assessment media difference

OpenDSA integrates AVs and interactive exercises with tutorial content. The benefits of integrating AVs with assessment activities include:

1. **Learning and assessment platforms uniformity**

   OpenDSA module pages integrate visualizations and a wide variety of assessment activities. Students do not need to leave the tutorial system to practice and gauge their understanding of the topic they are studying. This is a big improvement from Moodle-type sites with assessment, or paper textbook with assessment software.

2. **Ubiquitous automated grading**

   OpenDSA supports client-side grading and storage of data (HTML5 local storage) for student self-monitoring. Only small programming exercises need the data collection server for grading. Unlike most courseware that operates behind a login wall, OpenDSA textbook instances, AVs, and exercises can be used by anyone at any time to get automatic feedback. Signing in allows the system to preserve and relate students' progress, and report that progress to the instructor. But is not a requirement for practicing.

4.2.2 Addressing limited platform support

The use of HTML5 and JavaScript to develop OpenDSA modules ensures that the content can be displayed on all major web browsers. To guarantee forward compatibility, the major browsers (Chrome, Firefox, and Safari) are supported . As of October 2014, 96% of OpenDSA's traffic originated from supported browsers. OpenDSA generally does work on Internet Explorer, so its limited use by students is not caused by of our lack of support.

All AVs and interactive exercises can be embedded in third-party educational sites

Instructors can use OpenDSA's oEmbed endpoint to embed AVs and interactive exercises into their existing online web tutorials. oEmbed is an open format that allows embedding content of a website on third party sites. The oEmbed endpoint allows instructors to seamlessly embed our AVs and exercises into their own online course materials. When used in this way, instructors can decide if they want to use the OpenDSA backend to track their students' progress.

The OpenDSA data collection server stores students' attempts and progress data, and allows instructors to export students' grades into several file formats such as PDF, CSV, and Excel spreadsheet. The data exported from OpenDSA can in turn be imported into commonly used online learning tools like Sakai, Blackboard, or Moodle.

4.2.3 Addressing AV systems creation divisiveness and its consequences

OpenDSA is an open source project. OpenDSA code is publicly available on GitHub (`https://github.com/opendsa`). The developer base of OpenDSA is not huge (it is not really crowd sourced) but it does involve people from multiple institutions in multiple countries (so far developers are from six universities in 3 countries) with differing needs and goals. CS educators are constantly adding new content that suits their need, so the

OpenDSA system has to accommodate a fairly wide range of use cases, and be flexible. In addition to using open source technology, OpenDSA includes features that make it easy to create content that is less coupled with its author. OpenDSA design goals include giving instructors control over the content they use in their course. OpenDSA provides the following services to help instructors customize their course content.

1. **OpenDSA infrastructure supports the creation and the use of customized books**

   Instructors have the ability to create their own content. They can configure the book to fit their class setting. Configuration options include the following: picking the modules and exercises to be included in the book, selecting exercises to be completed by the students in each module, and setting the points awarded to students upon completion of an exercise.

2. **OpenDSA can be integrated with other systems**

   OpenDSA's backend exposes an API that allows other systems to log their data in OpenDSA's database. Existing assessment systems can connect to our API and use HTTP requests to send and retrieve their data. Security is maintained by granting explicit permission to specific client sites through **HTTP access control**. However, third parties can also install a copy of our server-side system in order to support their own assessment process.

3. **Localization**

   OpenDSA supports a sophisticated internationalization framework that attempts to make it as easy as possible to support compiling textbook instances in various (natural) languages. The configuration system allows a book compiler to specify the language of choice, and the system will take module versions in the target language whenever available (the default language is English). When building a new book instance, the "lang" variable in the configuration file is used to set up the book language. OpenDSA makes it easy for module authors to add support for a new language by storing the translations in a JSON file.

   The OpenDSA framework makes it easy to be able to compile book instances with code snippets from the instructor's desired programming language(s). The most important principle for managing code snippets is that they should be taken from working programs that can properly support testing of the code that you included into your modules. All such sourcecode should be uploaded in the SourceCode directory within OpenDSA, with each coding language having its own subdirectory. A given AV can have an associated JSON file that defines the configuration for alternate coding languages (including such things as the sourcecode filename).

4. **Privacy**

When deciding to adopt a learning environment system, educators have to take privacy into consideration if they do not want to host their own instance of the learning environment. Instructors have to make sure that information like students' name, email address, and even grades are not accessible to entities outside of their academic institution. OpenDSA can generate a set of anonymous account names, with a prefix set by the instructor. The list of accounts created is then emailed to the instructor. This feature ensures that outside entities will not have access to students' personal information since only anonymized usernames are stored into OpenDSA database.

4.2.4 Addressing educational usefulness of AV systems

In their meta-study of AV effectiveness [40], Hundhausen et al. found AV educational effectiveness inconclusive. Instead they concluded that investigating the educational effectiveness of visualization should focus on students' engagement, because the ways AVs are used have a major impact on students' learning success. We investigate the impact of OpenDSA on student learning performance in Section 5.4.

4.2.5 OpenDSA adoption in practice: Linköping University (Sweden) case study

Linköping University provided us the opportunity to assess OpenDSA adoption in practice. It allows us to observe how obstacles to AV systems adoption came into play when a particular university not already involved with the project used OpenDSA. As a result we improved the system to make it easier to install. The main instructor of a C++ and Data Structures course at Linköping University wanted to use OpenDSA in his class. Due to privacy constraints, they decided to run their own instance of the OpenDSA server. In addition, they wanted to customize their book instance to convert the code snippets and examples from Java/Processing to C++. Installation and configuration took place during Summer 2014. OpenDSA was used to teach a junior level C++ and Data Structures course during Fall 2014.

We interviewed the instructor after the installation to collect information about his experience with installing and customizing OpenDSA. We related the issues he reported to AV systems adoption obstacles. The following issues arose during the adoption process.

1. **Issues stemming from AV systems creation divisiveness**

    - **Incomplete documentation:** The lack of sufficient documentation became a problem that when installing OpenDSA at Linköping University. We

worked with the instructor to update OpenDSA documentation based on the questions and difficulties they encountered.

- **Undetected bugs:** Being the first to install a system from scratch, the people at Linköping University uncovered several bugs. The instructor mentioned that the OpenDSA team fixed most of the bugs and that the system ran smoothly and was quite robust during the Fall semester.
- **Customization:** Unlike other instructors who used OpenDSA with Java, C++ is used to teach Data Structures at Linköping University. The instructor had to customize his book instance to convert code snippets and AV, to display C++ instead of Java code. The instructor reported that the configuration process worked as needed in regard to language configuration. Some work was needed in order to have the visualizations highlight the corresponding lines of code in C++.

2. **Other sources of obstacles: Institutional obstacles** have been identified as a major impediment to technology integration and adoption [11]. The instructor reported that the main difficulty was to install OpenDSA on Linköping University computing systems, with little timely support from the university system administrator. Installation would have been smoother if it had been done on a machine that the instructor had total control over.

4.3 OpenDSA use by Instructors

In order to assess the impact of OpenDSA on classroom instruction, we followed a cohort of instructors using OpenDSA. We used a qualitative research approach since we were concerned with exploring behavioral changes occurring in natural settings [54]. Among all the qualitative research strategies, the case study approach seems the most suitable for our investigation. Yin [85] defined a case study research strategy as *"an attempt to examine a contemporary phenomenon in its real-life context, especially when the boundaries between phenomenon and context are not clearly evident"*. That definition fully captured our intentions.

Population: We collected our data from instructors who used OpenDSA between Fall 2012 and Fall 2014, inclusive. We treated each teaching offering as a separate case study. We categorized instructors' experiences in order to extract trends for the whole group. In order to get an accurate picture of the relationship between instructors and OpenDSA, we classify the instructors based on their relationship with the project. The categories are as follows:

- Grant support: for instructors contributing to the OpenDSA project and with research financially supported by a grant related to the OpenDSA project.

- Research interest: for instructors contributing to the OpenDSA project and not financially supported by a grant related to the OpenDSA project, but with an interest in using their involvement in OpenDSA as part of their own research.
- Instructional involvement: for instructors using OpenDSA with no other involvement (research or grant support). They used OpenDSA materials strictly for their instructional value.

Data Collection: We used a combination of online surveys (see Appendix A) and interviews (in-person or via Skype) to collect data. Our research was approved by the Institutional Review Board of Virginia Tech (see Appendix C). The questionnaires were administered at the end of the school term (Spring, Fall, Summer semesters) and examined instructor's opinion of OpenDSA and how they used it. We were interested in knowing how OpenDSA was used in the classroom, and how the instructor used the interactive exercises and if the exercises contributed to students' final grades.

The following sub-sections summarize the situations and observations (directly quoted) of instructors involved in the main case studies. We also labeled any pedagogical change that occurred.

4.3.1 Case study 1: CS3114-F12

Treatment section

Institution	Virginia Tech
Term	Fall 2012
Instructor's ID	I_1
Relationship with the OpenDSA project	Grant support
Course level	CS3
How used	Students spent their class time working through equivalent content in the form of online material. During some class periods, the treatment group received lecture or group discussion rather than solely working on the modules.
Assignments	One assignment due at the end of the 3 week period
In class use	Tutorial pages used in lieu of PowerPoint slides
Contribution to final grade	5%

The instructor is the principal investigator of the OpenDSA project.

Pedagogical change: More active classroom. The instructor experimented with a flipped classroom for one week.

Control section

Institution	Virginia Tech
Term	Fall 2012
Instructor's ID	I_2
Relationship with the OpenDSA project	No
Course level	CS3
How used	The control group received standard lecture and textbook for three weeks on the topics of sorting and hashing, similar to what has been typically done in this course for many years.
Assignments	No
In class use	No
Contribution to final grade	N/A

4.3.2 Case study 2: T1061220

Institution	Aalto University
Term	Spring 2013
Instructor's ID	I_3
Relationship with the OpenDSA project	Grant support
Course level	CS3
How used	Supplemental material listed on course website. JSAV proficiency exercises were used in another online tutorial.
Assignments	No
In class use	No
Contribution to final grade	N/A

4.3.3 Case study 3: CS3114-SP13-A

Institution	Virginia Tech
Term	Spring 2013
Instructor's ID	I_4
Relationship with the OpenDSA project	No
Course level	CS3
How used	OpenDSA was used as the main resource to teach Sorting and Hashing topics.
Assignments	One mandatory homework covering sorting and hashing
In class use	Traditionally used PowerPoint slides plus the board to show examples; this time used OpenDSA visualizations as examples. Instructor had previous experience with an AV system; he had used the TRAKLA2 Heap tutorial.
Contribution to final grade	5%

Instructor comments: I am comfortable with all the students' getting an A (mastery approach of OpenDSA) because homework are used to force students to study. If I use OpenDSA exercises for all homework assignments it will be worth about 15% of the class grade. OpenDSA provides the students with more opportunities to practice. Students will spend a lot of time on OpenDSA and benefit from it. Students sometimes cannot remember teacher's explanation from their notes. OpenDSA visualizations will help them in that matter.

4.3.4 Case study 4: CS3114-SP13-B

Institution	Virginia Tech
Term	Spring 2013
Instructor's ID	I_5
Relationship with the OpenDSA project	No
Course level	CS3
How used	OpenDSA exercises were used as homework mandatory assignments covering sorting and hashing
Assignments	OpenDSA exercises replaced (previous) hashing and sorting mandatory homework assignments
In class use	No
Contribution to final grade	8%

Instructor comments: OpenDSA was useful to the students. It gave them different prospective on the concepts and real practice on the topics. OpenDSA augmented the classical textbook, reading + lecture with "self-paced" online assignments. I think people respond better when presented a concept through different ways.

4.3.5 Case study 5: CS223

Institution	Alexandria University
Term	Spring 2013
Instructor's ID	I_6
Relationship with the OpenDSA project	No
Course level	CS3
How used	OpenDSA was used as the main resource to teach Hashing.
Assignments	One mandatory assignment on hashing
In class use	Replaced PowerPoint slides with OpenDSA pages.
Contribution to final grade	3%

Instructor comments: The most beneficial part was the exercises with immediate feedback in contrast to traditional paper textbooks. In addition, students can easily

understand the concepts from the materials (if they missed the class or they did not understand in class). I like the animation, and I used them to show examples in class. I could not use PowerPoint slides anymore with OpenDSA.

4.3.6 Case study 6: CS3114-F13

Institution	Virginia Tech
Term	Fall 2013
Instructor's ID	I_1
Relationship with the OpenDSA project	Grant support
Course level	CS3
How used	OpenDSA was the textbook for the course
Assignments	Mandatory assignments due before the end of the semester
In class use	OpenDSA pages used during lectures
Contribution to final grade	20%

The instructor is the principal investigator of the OpenDSA project.

4.3.7 Case study 7: CS3114-SP14

Institution	Virginia Tech
Term	Spring 2014
Instructor's ID	I_7
Relationship with the OpenDSA project	No
Course level	CS3
How used	Assigned weekly readings and exercises as homeworks for supplementing in-class lecture material. In class, we covered a majority of the chapters, but in a slightly different order to accommodate programming project needs.
Assignments	Mandatory homework assignments were assigned and due on a weekly basis. Entire chapters were assigned. But due to various problems that were resolved by the OpenDSA team during the course of the semester, students were given until the end of the semester to complete the homework assignments without penalty.
In class use	No
Contribution to final grade	20% of final grade.

Instructor comments: Positive, it greatly simplified the management of homeworks. Interactive exercises added a lot to students' understanding of the material. Online materials worked well with integrating with other online course materials. Made it feel more relevant to the class than a paper textbook. Convenient access. I referred to it frequently to refresh my own understanding of topics and to link students to relevant material when answering questions online. Something to think about is how to better integrate OpenDSA into the classroom/lecture, perhaps by using the interactive exercises in class, workshops, etc.

4.3.8 Case study 8: CS2114-SP14

Section 1

Institution	Virginia Tech
Term	Spring 2014
Instructor's ID	I_5
Relationship with the OpenDSA project	Research interest
Course level	CS2
How used	Homework assignments
Assignments	Assigned weekly readings and exercises as homeworks for supplementing in-class lecture material and Moodle homework quizzes.
In class use	
Contribution to final grade	2%

Section 2

Institution	Virginia Tech
Term	Spring 2014
Instructor's ID	I_8
Relationship with the OpenDSA project	Grant support
Course level	CS2
How used	Homework assignments
Assignments	Assigned weekly readings and exercises as homeworks for supplementing in-class lecture material and Moodle homework quizzes.
In class use	No
Contribution to final grade	2%

4.3.9 Case study 9: CSCI204

Institution	Beloit College
Term	Spring 2014
Instructor's ID	I_9
Relationship with the OpenDSA project	No
Course level	CS2
How used	OpenDSA was the textbook for the course. The assessment activities were mandatory. The assessment activities were due before the class for the reading they included so there were many due dates. The penalty was 70% for not finishing an assessment by defined time. The goal of the penalty was to ensure that each lecture starts at a known point for everybody.
Assignments	Weekly assignments
In class use	I have been shifting toward doing less mechanics and more ideas in class. In using OpenDSA I made this shift more complete. I looked at the results from assessment activities and asked the students at the start of class if they understood the material, esp. how the data structure/algorithm worked. I then went over any needed areas at this point but moved on to the concepts as soon as I could (or right away if no issues). Note I also made other changes not directly related to OpenDSA. For example, I started using clickers.
Contribution to final grade	8% of the grade in the course but I allowed them to drop the two lowest scores.

Pedagogical change: Assignments due before lecture, more active classroom.

Instructor comments: I found requests for changes to be very responsive. The best I have ever seen. The textbook uses ideas I have tried or wanted in that the questions aren't just fill in the blank but visually relate to the data structure/algorithm. Overall, I was happy with the textbook.

4.3.10 Case study 10: CSCI102

Institution	New York University
Term	Spring 2014
Instructor's ID	I_{10}
Relationship with the OpenDSA project	No
Course level	CS2
How used	OpenDSA was a supplemental resource. The primary textbook is selected by the department for all sections of Data Structures course.
Assignments	One mandatory assessment. The others were optional practice exercises. The main reason for that was that the data structures presented by OpenDSA and the primary textbook differed slightly.
In class use	"I sometimes used the visualizations provided in OpenDSA instead of creating my own or drawing things on the board".
Contribution to final grade	-

Instructor comments: I think that OpenDSA provides a great resource for students (and me). The visualizations (especially the ones for sorting) were the biggest benefit. Some sections/chapters do not have visualizations yet - I think that would greatly contribute to the improvement of OpenDSA.

4.3.11 Case study 11: CS2114-S14-1

Institution	Virginia Tech
Term	Summer 2014
Instructor's ID	I_5
Relationship with the OpenDSA project	Research interest
Course level	CS2
How used	Homework assignments
Assignments	Assigned weekly readings and exercises as homeworks for supplementing in-class lecture material and Moodle homework quizzes.
In class use	No
Contribution to final grade	2%

4.3.12 Case study 12: CS2114-S14-2

Summer II 2014 VT

Institution	Virginia Tech
Term	Summer 2014
Instructor's ID	I_8
Relationship with the OpenDSA project	Grant support
Course level	CS2
How used	Homework assignments
Assignments	Assigned weekly readings and exercises as homeworks for supplementing in-class lecture material and Moodle homework quizzes.
In class use	No
Contribution to final grade	3%

4.3.13 Case study 13: CS3114-F14

Institution	Virginia Tech
Term	Fall 2014
Instructor's ID	I_1
Relationship with the OpenDSA project	Grant support
Course level	CS3
How used	OpenDSA was the textbook for the course
Assignments	Mandatory assignments due before the end of the semester
In class use	OpenDSA pages used during lectures
Contribution to final grade	20%

4.3.14 Case study 14: COP3534

Institution	Florida International University
Term	Fall 2014
Instructor's ID	I_{11}
Relationship with the OpenDSA project	No
Course level	CS3
How used	Supplementary material
Assignments	Students had assigned weekly readings and exercises from OpenDSA. They were required to complete them prior to the associated lectures.
In class use	Because OpenDSA was available, I was able to engage my students more in class with active learning activities. I could make the readings and activities required work to be completed prior to class (and easily check that it was completed), so that I could focus on quick refreshers during lecture time with more hands-on activities to tackle more complex topics. I would not have been able to do this without OpenDSA.
Contribution to final grade	15%

Pedagogical change: Assignments due before lecture, more active classroom.

Instructor comments: Overall, I find OpenDSA to be an exceptional resource. I strongly believe that my students would not have learned as much or done as well in my classes if I had not used OpenDSA. I consider it a critical part of my course. At this point, I would have to completely redesign my course if I (and my students) did not have access to OpenDSA. Both the comprehensiveness and quality of the materials and exercises are exceptional. In terms of improvements, my primary recommendation involves improving the usability and user experience of OpenDSA. Cross-browser compatibility was also an issue at times for some students, although I did not experience any issues myself. Finally, it would be nice if there were related in-class activity suggestions for the various topics and chapters in OpenDSA.

4.3.15 Case study 15: CS2401

Institution	University of Texas El Paso
Term	Fall 2014
Instructor's ID	I_{12}
Relationship with the OpenDSA project	No
Course level	CS2
How used	Supplementary reading material for the class
Assignments	Students had to complete all exercises
In class use	No
Contribution to final grade	5%

Instructor comments: Many students reported problems. Sometimes, students showed me that OpenDSA was working for some exercises and for others they were receiving errors. Many times, the problems were so frequent that I did not report them to the OpenDSA team. Other than the reported problems, I think, it was good. More error-free exercises will improve the experience. Some students personally told me that OpenDSA was very helpful.

4.3.16 Case study 16: CS208

Institution	Massachusetts Bay Community College
Term	Fall 2014
Instructor's ID	I_{13}
Relationship with the OpenDSA project	No
Course level	CS2
How used	Supplementary material
Assignments	At first, the exercises were mandatory (with grade). But, due to issues with exercises and the grades (I guess we were using it during a reconstruction), I made them optional.
In class use	
Contribution to final grade	0

Instructor comments: Unfortunately, due to issues mentioned before, it did not work well for my class last semester. Another issue was the tightly coupled content, which removed the ability to design my own chapters and flow. I am going to hopefully create my own version this summer, but if the content are more modularized and decoupled, it will make it a lot easier to create a book that matches my students' level. It would be nice if the exercises are more streamlined, so they are not either very basic and simple or high level (with missing steps) without much instructions.

4.3.17 Case study 17: TDDD86

Institution	Linköping University (Sweden)
Term	Fall 2014
Instructor's ID	I_{14}
Relationship with the OpenDSA project	Research interest
Course level	CS3
How used	OpenDSA was the course main book
Assignments	All exercises to be completed before the final.
In class use	The class is based on lectures and lab. Used ODSA both in lectures, show slideshows and AVs. students were assigned modules exercises The students did not have to do the exercises before the lectures.
Contribution to final grade	11 credits for the class, with 2 (pass/fail) credits for OpenDSA.

Instructor comments: Wanted to try a flipped classroom, OpenDSA is definitely helping making the transition. Before I was still using AVs, so the lecture did not change much now all is in one place, and the presentation is more uniform. No need for the students to navigate several links.

4.4 Summary of pedagogical changes and uses across case studies

One goal of this research is to uncover and understand teachers' views and experiences when introducing new technology into their classrooms. We hoped to find what factors influenced instructors' decision to use OpenDSA, how OpenDSA was used in the classroom, and if any changes occurred because of the introduction of OpenDSA. Before discussing pedagogical changes, we will describe the uses of OpenDSA in the classroom.

4.4.1 Wide variety of use and reuse

When using OpenDSA, instructors have to decide if they want to use OpenDSA as the main course material or as secondary/supplemental material. Table 4.1 shows this aspect of OpenDSA status in the classroom.

Table 4.1: Instructors use of OpenDSA

Main textbook	**Secondary textbook (mandatory)**	**Supplemental textbook (non-mandatory)**
I_1	I_4	I_3
I_6	I_5	
I_9	I_7	
I_{14}	I_8	
	I_{10}	
	I_{11}	
	I_{12}	
	I_{13}	

OpenDSA as main textbook and instructor's involvement in the project: Three instructors (I_1, I_9, I_{14}) used a complete version of OpenDSA as the main textbook in their class. I_6 only used a week's worth of material (covering Hashing). I_6 used OpenDSA when the project was still at an early stage when materials covered only two topics (sorting and hashing). One instructor (I_1) is the main investigator of the OpenDSA project, while the two others (I_9, I_{14}) only used OpenDSA for teaching. I_{14} wanted to run his own instance of the OpenDSA server and also wanted to change the programming language of his book instance from Java to C++. To make all these changes happen, he started contributing to the project. I_9 started contributing to the project because he wanted a book instance tailored for his CS2 course. I_{13} expressed his dissatisfaction with OpenDSA content and indicated that they will contribute to the project in order to create an OpenDSA book instance more relevant to his class. Based on experiences of I_{14}, I_{13}, and I_9 we conjecture that **independent contribution to OpenDSA stems from using OpenDSA as main course material**, thus requiring instructors to have a greater control of the book instance's content.

Use/Reuse of OpenDSA over time: We wanted to see if instructors used OpenDSA more than one time to teach the same course. Unfortunately, the history is short considering that instructors only began using the materials in 2013, and most instructors only teach a relevant course once per year. To answer our question, we used data from Fall 2012 to Spring 2015. We collected teaching information from instructors interviews, but also from instructors' personal websites and university courses listing web pages. Table 4.2 summarizes our findings.

Eight instructors have not used OpenDSA since their initial use of the system. Among them, five (I_4, I_7, I_{11}, I_{12}, and I_{13}) have not taught that same course since. I_6 and I_{10} listed OpenDSA as a supplemental resource during their subsequent teachings. Only one instructor (I_5) did not use it again to teach CS3114 in Spring-2015 because he is

teaching the course with another instructor who does not use OpenDSA. However, this instructor continues to use OpenDSA in his CS2 course. Five instructors (I_1, I_5, I_8, I_9, I_{14}) used OpenDSA in each course offering they taught during the time frame of our investigation. If we remove the 5 instructors who have not taught a DSA course since, we have an overall reuse rate of 62% for OpenDSA (60% for instructors not on an OpenDSA research grant). That rate jumps to 87% if we include instructors who listed OpenDSA as a supplemental (optional) resource in their subsequent teachings. Based on our data we can say that **educators tend to reuse OpenDSA in their courses**.

Use of OpenDSA assessment activities: All instructors indicated that assessment activities are the most valuable feature of OpenDSA. They reported that it saved them time creating and grading assignments. With the exception of I_{10}, all **instructors used OpenDSA assessment activities as mandatory graded assignments.** I_{10} mentioned that the main textbook of the course was selected by the department for all sections of Data Structures courses and that the data structures presented by OpenDSA and the primary textbook differed slightly. However, the students in her class had to complete one OpenDSA assignment.

The contribution of OpenDSA exercises to students' final grade is an indicator of the value put on OpenDSA exercises. The contribution ranged from 2% up to 20%, with CS3 courses intructors awarding more weight to OpenDSA exercises. The reason is that OpenDSA was initially created for traditional CS3-level courses, and they tend to have more content that is covered by OpenDSA. CS3 instructors who used a complete OpenDSA book instance (I_1, I_7, and I_{11}) assigned an average of 18.3% of their course final grade to OpenDSA exercises, while CS2 instructors (I_9, I_5, I_8, I_{12}) assigned 3.4% of their course grade to OpenDSA exercises. This is in line with the amount of the course grade that these instructors typically assign to data structures homework.

4.4.2 Trends emerging from early adoption

More practice for students: One of the main changes mentioned by instructors is the availability of more assessment opportunities for students. I_7 reported that previously he was assigning monthly homework. With OpenDSA he switched to weekly assignments. I_5 and I_8 added OpenDSA exercises to existing Moodle quizzes, thus greatly increasing the number of homework assignments. Eight instructors reported assigning weekly homework.

Assignments due before lecture: I_9 and I_{11} assigned homework before the corresponding lecture. Instructor I_9 in particular used the instructor's view of OpenDSA to track students' progress on assignments to determine if he needed to cover the mechanics of the data structure/algorithm. Students were penalized for not doing the exercise before lecture time. OpenDSA allowed him to automatically flag late assignments and apply the penalty. It was important for him to have the students completing the as-

Table 4.2: Instructors use of OpenDSA over time

Instructor	**Initial use term**	**Subsequent use term(s)**	**Comments**
I_1	Fall-2012	Fall-2013, Fall-2014	OpenDSA used as course textbook in all offerings
I_4	Spring-2013	-	Has not taught the course since
I_5/CS3114	Spring-2013	-	Teaching one section of CS3114 during Spring-2015 but decided not to use OpenDSA due to coordination with another instructor.
I_6	Spring-2013	-	Taught the course during Spring-2014, OpenDSA was listed on the course website as a supplemental resource.
I_7	Spring-2014	-	Has not taught the course since
I_5/CS2114	Spring-2014	Summer-2014-1, Fall-2014, Spring-2015	OpenDSA used for mandatory graded assignment in all offerings
I_8	Spring-2014	Summer-2014-2, Spring-2015	OpenDSA used for mandatory graded assignment in all offerings
I_9	Spring-2014	Spring-2015	OpenDSA used as course textbook in all offerings
I_{10}	Spring-2014	-	Taught the course during Fall-2014 and Spring-2015. OpenDSA listed on the courses website as a supplemental resource. Visualizations sometimes used during lectures
I_{11}	Fall-2014	-	Has not taught the course since
I_{12}	Fall-2014	-	Has not taught the course since
I_{13}	Fall-2014	-	Has not taught the course since
I_{14}	Fall-2014	Spring-2015	OpenDSA used as course textbook in all offerings

signments before lecture because he experimented with a "flipped classroom". I_{11} used OpenDSA activities to verify that the students were acquainted with the mechanics of the data structure/algorithm before lecture time.

More active classroom: Active learning is an instructional approach requiring students to do meaningful learning interactive activities in the classroom [63]. The type of activity used in an active classroom setting can engage students at different levels of the revised Bloom's taxonomy [48]. The revised taxonomy contains the following categories: remember, understand, apply, analyze, evaluate, and create (see Figure 4.1). In a recitation-style lecture, it is hard to engage students at levels above "understand". We think that technology and tools like OpenDSA can help, especially for activities related to the "remember" and "understand" levels, thus allowing instructors to focus more on higher level activities. An active classroom can be implemented in many ways. In traditional CS courses, a programming lab is the most common instance of an active classroom. Lately, flipped or inverted classrooms have received a lot of attention by CS educators. Bishop et al. [12] defined a flipped classroom as "*an educational technique that consists of two parts: interactive group learning activities inside the classroom, and direct computer-based individual instruction outside the classroom*". Several studies have shown that students have a positive attitude toward flipped classrooms [12, 7].

I_9 and I_{11} leveraged OpenDSA assessment activities to implement a more active classroom. I_9 spent more time working on students' misconceptions and on higher-level concepts. He also introduced clickers (a classroom response system) in his class to make his class more active. I_{11} spent lecture time working on students' misconceptions and more complex topics with hands-on activities. I_1 experimented with a flipped classroom during the pilot study. He had the students work on OpenDSA modules during lecture instead of the traditional Powerpoint/slides type lecture. However, he reverted to more traditional lectures due to feedback from students. Nevertheless, he reported spending less time explaining procedural aspects of algorithm when lecturing, leaving more time to cover other topics.

Other instructors (I_7, I_{14}) have expressed their interest in moving toward a more active class in the future. I_7 mentioned being interested in using interactive exercises in class, or OpenDSA-based workshops during lecture. Like I_{11} reported in her comments, "in-class activity suggestions for the various topics and chapters in OpenDSA" will greatly help instructors transition toward a more active classroom. Based on the above observations, we find that **instructors interested in moving to a more active classroom are finding themselves enabled or at least encouraged by OpenDSA**.

1.0 Remember – Retrieving relevant knowledge from long-term memory.
- ***1.1 Recognizing***
- ***1.2 Recalling***

2.0 Understand – Determining the meaning of instructional messages, including oral, written, and graphic communication.
- ***2.1 Interpreting***
- ***2.2 Exemplifying***
- ***2.3 Classifying***
- ***2.4 Summarizing***
- ***2.5 Inferring***
- ***2.6 Comparing***
- ***2.7 Explaining***

3.0 Apply – Carrying out or using a procedure in a given situation.
- ***3.1 Executing***
- ***3.2 Implementing***

4.0 Analyze – Breaking material into its constituent parts and detecting how the parts relate to one another and to an overall structure or purpose.
- ***4.1 Differentiating***
- ***4.2 Organizing***
- ***4.3 Attributing***

5.0 Evaluate – Making judgments based on criteria and standards.
- ***5.1 Checking***
- ***5.2 Critiquing***

6.0 Create – Putting elements together to form a novel, coherent whole or make an original product.
- ***6.1 Generating***
- ***6.2 Planning***
- ***6.3 Producing***

Fig. 4.1: Revised Bloom's Taxonomy structure [48]

4.4.3 OpenDSA use incentives

Out of the 12 instructors[1] who used OpenDSA between Fall 2012 and Fall 2014, two (I_1 and I_8) were financially supported by research grants related to OpenDSA. Seven instructors were associated (during that time or previously) with Virginia Tech and learned about OpenDSA from grant-supported instructors. I_6 and I_{12} are former Virginia Tech students, and I_{14} was introduced to OpenDSA by our collaborators in Finland. Four instructors (I_9, I_{11}, I_{10}, and I_{13}) found OpenDSA while looking for resources for their courses. They did not have previous contact with the grant-supported instructors. Figure 4.2 shows the distribution of instructors over time. Spring 2015 data were included to show the evolution of the number of independent instructors.

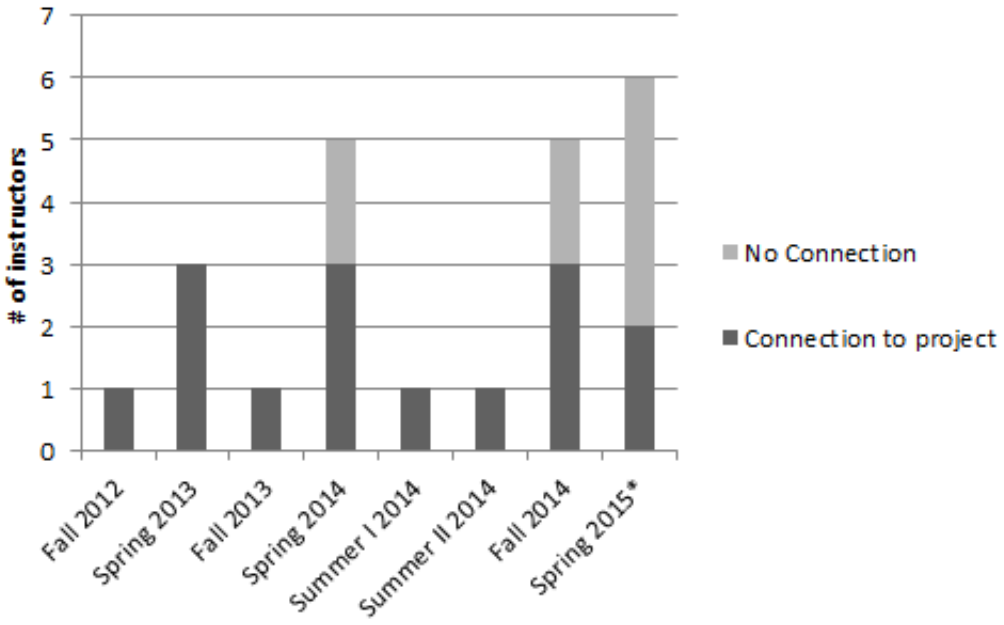

Fig. 4.2: OpenDSA users by term grouped by previous connection to the OpenDSA project.

For obvious reasons, grant-supported instructors had financial motivations to use OpenDSA in their class. However, instructors not affiliated with the OpenDSA project have mainly pedagogical reasons to use OpenDSA. They found OpenDSA while looking for interactive materials. The fact that all non-Virginia Tech affiliated instructors (except for I_{14}) first sent inquiry emails to the address listed on the project web page is a confirmation of their pedagogical interest in OpenDSA, rather than personal contacts. Based on the above findings we can say that **there is an interest in OpenDSA from CS educators not affiliated with the project**.

[1]We excluded I_3 from the count since he used JSAV exercises in his own tutorial and I_2 because he did not actually use OpenDSA.

4.4.4 Implications

The trends identified in this study have many implications for the future of OpenDSA and for similar projects.

Building easily customizable systems: We saw that some instructors wanted a custom-made version of OpenDSA and that they sometimes needed to implement some of the changes themselves. We need to design systems that use accessible technology and will allow instructors to easily carry out modifications. In that regard, I_7 and I_{13} expressed their interest in modifying their book instance's content in the future. OpenDSA was designed to be easily customizable, and use accessible, open source software.

Cloning will help returning users: For returning users it can be frustrating to do the same thing twice. Educational systems should provide users with the possibility of cloning old courses/activities/assignment. Cloning assignments is not yet available in OpenDSA.

Adding in-class activities: eTextbooks should support in-class activities. I_9 reported that he looked at the results from assessment activities (due before class) and then went over any needed areas or to the concepts as soon as he could (or right away if no issues). The implication is that we should build better monitoring (visualization) tools to automatically inform him about students' progress. We should provide a better integration between systems like OpenDSA and lecture materials, and through crowd sourcing, we should collect best in-class hands-on activities and integrate them into OpenDSA.

4.5 Conclusion

In this chapter we investigated if OpenDSA would be easier to integrate in the classroom and what impact it will have on pedagogy. Specifically, we answered the following questions:

1. How well does OpenDSA's design address obstacles in the use of AV systems in the classroom?

   By integrating AVs and interactive exercises, OpenDSA successfully addresses learning and assessment media difference issue. To circumvent limited platform obstacle, OpenDSA content was designed to can be displayed on all major browsers. In addition, all AVs and interactive exercises can be in third-party educational sites. OpenDSA makes it easier for AV developers and educators to collaborate when creating educational content. OpenDSA allows the creation of customized AVs, interactive exercises, and books.

2. What pedagogical changes occur when OpenDSA is used in the classroom?

   Instructors used OpenDSA in various ways. Some used it as the main textbook in their course, others used it as secondary textbook. All instructors used OpenDSA exercises as mandatory assignments. The use of OpenDSA resulted in more practice for students. OpenDSA's automated grading capabilities allows instructors to request homework to be completed before lecture and support instructors implementing a flipped classroom.

Chapter 5

Impact of OpenDSA on students

The use and effectiveness of technology depends in part on users' attitudes. We assessed student's attitudes towards OpenDSA and also their perception of OpenDSA's impact on learning and ability to learn course materials. We hypothesize that the interactive nature of OpenDSA, coupled with the fact that students receive immediate feedback upon completion of an exercise, will lead students to have a positive attitude toward OpenDSA. Furthermore, the variety and the quantity of assessment activities within OpenDSA will encourage students to use OpenDSA voluntarily (for example to study for examinations). We were also interested in the impact of OpenDSA on student performance.

5.1 Hypothesis

"OpenDSA will positively impact student performance and motivation."

Our hypothesis is about measuring the impact of OpenDSA on students' learning outcomes. In particular we want to see how the way that OpenDSA is used by instructors inside the classroom will affect students' attitude towards OpenDSA, students' use of OpenDSA outside the classroom (for homework and/or review), and students' performance in Data Structures and Algorithms courses.

To test our hypothesis we will answer the following research questions:

1. What is students' experience with OpenDSA?
2. How do students prefer to use OpenDSA in the classroom?
3. Does the way students use OpenDSA correlate to students' learning performance?

We used the statistical software R [65] to analyze the data collected during our investigations.

5.2 Student experience

In this section we present findings on learners' attitude and motivation towards OpenDSA.

5.2.1 Student opinions

Participants

We used questionnaire responses to collect data about students' experiences with OpenDSA. Participants were students enrolled in CS3 courses at Virginia Tech (n=57)and Alexandria University in Egypt (CS223, n=18). Students at Virginia Tech were divided into two sections CS3114-SP13-A (n=21) and CS3114-SP13-B (n=36). Each section was taught by a different instructor, both having no relationship with the OpenDSA project.

Materials and procedure

Students in CS3114-SP13-A used OpenDSA as the main resource for studying sorting and hashing topics. The instructor used OpenDSA as lecture aide and the students had to complete all OpenDSA exercises on sorting and hashing. Students in CS3114-SP13-B only had to complete OpenDSA exercises as homework mandatory assignments covering sorting and hashing. In CS223, OpenDSA was used as the main resource to teach hashing and OpenDSA exercises were used as mandatory homework. Participants in all groups completed a questionnaire about their experience using OpenDSA. The questionnaires can be found in the Appendix B. We use a qualitative method of content analysis when examining student responses.

Results

We used codes to label ideas expressed in each response. With the codes we wanted to capture both students' opinion of OpenDSA, and also elements that influenced their opinion. We derived the codes from students' responses, and grouped responses with the same code together. Table 5.1 shows a description of each code.

Out of 75 responses collected, 61 (or 81%) expressed an overall "positive" opinion toward OpenDSA. We received 2 "negative" and 3 "neutral" opinions. The negative answers referenced bugs encountered while using OpenDSA. For example one student wrote "Buggy and many wrong answers in the exercises. It would be great if more accurate". Twenty eight responses mentioned interactive elements (AVs and exercises) as main

Table 5.1: Coding scheme description

Label	**Description**
Positive	Positive experience with no details
Positive-interactivity	Positive experience - OpenDSA interactive elements (AVs or exercises)
Positive-frustration	Positive experience - OpenDSA limitations or defects
Neutral	Neither good or bad experience
Neutral-frustration	Neither good or bad experience - OpenDSA limitations or defects
Negative-frustration	Negative experience - OpenDSA limitations or defects

contributors to their positive experience. One student wrote "The interactive features allowed me to learn better and faster". Another wrote "It was an excellent way of learning new material. The visual exercises were the most helpful".

Frustration was expressed in 13 responses (out of 75); bugs and confusing questions were the main sources of frustration reported by the respondents. Examples of responses expressing frustration include: "Great experience. Frustrating at times because the interaction could use a bit of tweaking" and "Aside from the initial set up confusion it was enjoyable. Had some trouble with stalling/slowness at times". A positive experience was the most frequent opinion expressed by students. Table 5.2 summarizes survey responses analysis data.

Table 5.2: Student responses analysis summary

Response	**Frequency**	**Percentage**	
Positive	33	44%	92% Positive
Positive-interactivity	28	37%	
Positive-frustration	8	11%	
Neutral	1	1%	5% Neutral
Neutral-frustration	3	4%	
Negative-frustration	2	3%	3% Negative
Total	75	100%	

We wanted to see if the way OpenDSA was used in the classroom had an impact on student opinions. OpenDSA was used during lectures in CS3114-SP13-A but it was not in CS3114-SP13-B. The proportions of responses in the two groups are shown in Table 5.3. The difference in the distribution of responses between CS3114-SP13-A and CS3114-SP13-B students was not statistically significant (Chi-squared 2-sample test for equality of proportion, $\alpha = 0.05$) indicating that the use of OpenDSA as lecture aid did not have an impact on students' opinion of OpenDSA.

Table 5.3: CS3114-SP13-A and CS3114-SP13-B responses percentages

Response	CS3114-SP13-A	CS3114-SP13-B
Positive	40%	39%
Positive-interactivity	44%	39%
Positive-frustration	12%	9%
Neutral	-	-
Neutral-frustration	-	4%
Negative-frustration	4%	9%

We created a word cloud to visualize students' responses. We concatenated all student answers (N=75) and used Python's NLTK[1] (Natural Language Toolkit) library to count bigram (every sequence of two adjacent words in the text) frequency in students responses. We then use Wordle[2] to build bigram word cloud images of student responses, see Figure 5.1.

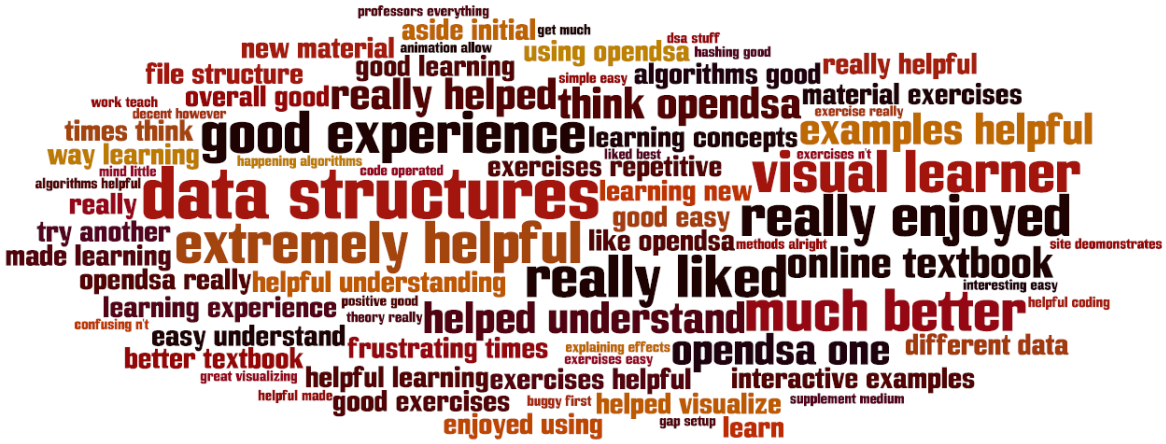

Fig. 5.1: Student opinion words cloud (Spring 2013).

The word cloud shows that students used positive terms to describe their experience with OpenDSA. "Extremely helpful", "good experience", and "really helpful" were among the terms heavily used by students in all three sections.

5.2.2 Perceived learning benefits

We investigated how students ranked OpenDSA in terms of perceived learning benefits.

[1] `http://www.nltk.org/` (accessed: 04/09/2015)
[2] `http://www.wordle.net/` (accessed: 04/09/2015)

Participants

Participants were sophomore students enrolled in CS2114-SP14 at Virginia Tech (n=54). CS2114-SP14 is a programming intensive class with 2 hours of programming labs each week. OpenDSA exercises were used as weekly homework mandatory assignments. At the end of the semester students answered survey questions to report about their experience using OpenDSA.

Materials and procedure

The questionnaire used in this study can be found in Appendix B. Students were asked: "This class included these elements: (1) lectures, (2) OpenDSA Exercises, (3) textbook (paper), (4) programming labs, (5) reading quizzes, and (6) interacting with instructor. Please rank these in order from the one that gave you the most learning gains to the one that gave you the least". Thus they ranked class items from 1-most useful to 6-least useful in regards to their learning gains in the course.

Results

Students enrolled in CS2114-SP14 on average ranked OpenDSA exercises second, with an average score of 2.48. In CS2114-SP14, OpenDSA was used as a supplemental (mandatory) resource for homework assignments (reading and exercises). The main course textbook is a traditional paper textbook. Programming labs were ranked first with an average score of 1.42, the course (paper) textbook was ranked third (average score = 3.77). Lectures, interacting with instructor, and reading quizzes were respectively ranked fourth, fifth and sixth with average scores 4.13, 4.19, and 4.57, respectively.

A Friedman test was run to identify if the difference between perceived leaning benefits was statistically significant. Because the Friedman test is only applicable to complete block designs, we discarded incomplete blocks (data from students who did not rank all class elements). In total 11 incomplete evaluations were discarded resulting in 43 evaluations used in our analysis. Figure 5.2 shows rankings of class elements and boxplots of score differences ($\alpha = 0.05$) for each pair of class items. The statistically significant differences boxplots are colored in green.

A Friedman test revealed a significant difference between the ranking of class items ($X^2(5) = 101.5, p < 0.01$). A post-hoc test using the Wilcoxon test with Bonferroni correction identified 3 clusters of items as shown in Table 5.5. The p-values of score differences are displayed in Table 5.4. Students attributed most of the learning gains in the course to programming labs first, then to OpenDSA exercises. The other class items were perceived to have a significantly lower impact in learning benefits.

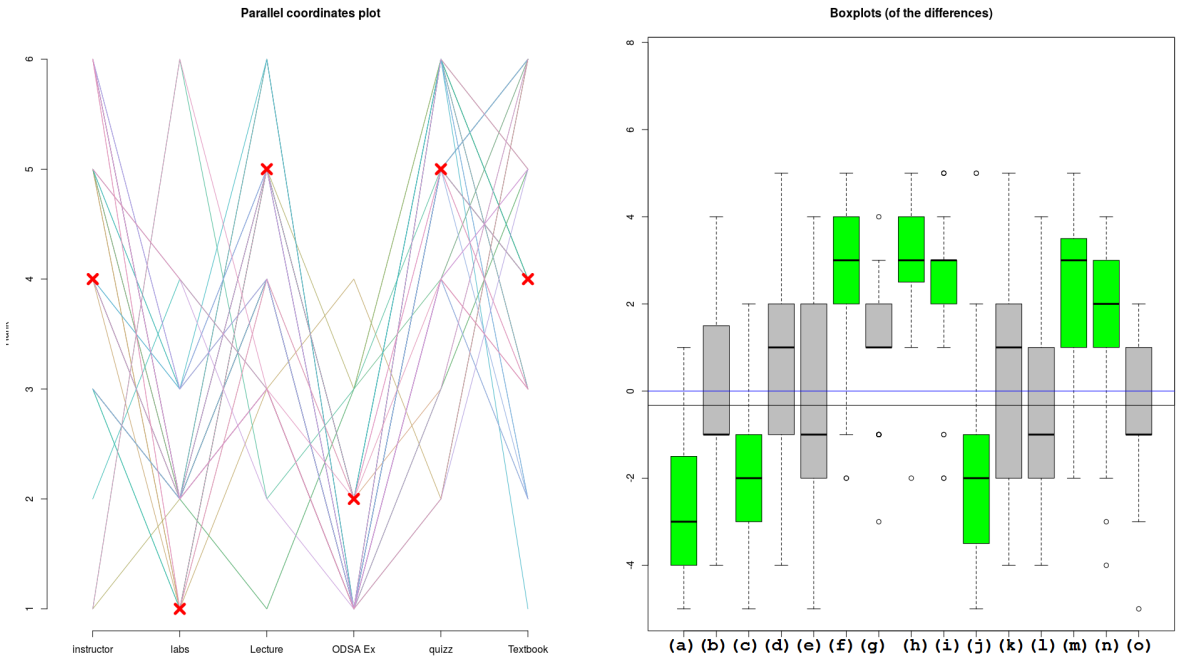

Fig. 5.2: Class items coordinates plot and boxplots of class items score difference (statistically significant differences in green).
(a): labs-instructor, (b): lecture-instructor, (c): ODSA Ex-instructor, (d): quizz-instructor, (e): textbook-instructor, (f): lecture-labs, (g): ODSA Ex-labs, (h): quizz-labs, (i): textbook-lab, (j): ODSA Ex-lecture, (k): quizz-lecture, (l): textbook-lecture, (m): quizz-ODSA Ex, (n): textbook-ODSA Ex, (o): textbook-quizz

5.2.3 Student use of AVs

OpenDSA's animations can be grouped in two categories that we will call: "simple slideshow" and "Complete AV". Simple slideshows typically present a static example to illustrate a particular point or part of an algorithm. Complete AVs illustrate the full execution of an algorithm, and allow the student to provide their own input or specify that a random input should be generated. Regardless of the type of visualization, the student is always in control of the pace. In his master's thesis [14], Breakiron investigated whether students will complete AVs voluntarily or only when required to. To address that question, he compared the behavior of students in CS3114-SP13-A against those in CS3114-SP13-B. Students in CS3114-SP13-A were not required to complete AVs (mini slideshows), and they did not receive credit for doing so. However, students in CS3114-SP13-B were required to complete and received credit for completing AVs. The analysis of the total number of AVs started and the total number of AVs completed revealed that 100% of CS3114-SP13-B students started all 44 AVs in the book instance with 89.23% completing all 44 AVs. In CS3114-SP13-A, 100% of students completed at least one AV but only 5.88% of the students completed all 44 AVs.

In his ongoing work, Mohammed Farghally is investigating whether students read algorithm analysis content. Such content is mainly prose with no interactive elements

Table 5.4: Wilcoxon test with Bonferroni correction p-values (statistically significant differences in green)

	Labs	**Lecture**	**OpenDSA Ex**	**Quizz**	**Textbook**	**Instructor**
Labs		$1.3e-11$	$6.1e-05$	$1.5e-13$	$6.6e-12$	$3.7e-12$
Lecture			$4.0e-07$	1.00	1.00	1.00
OpenDSA Ex				$4.4e-09$	$8.1e-07$	$5.4e-07$
Quizz					0.21	1.00
Textbook						1.00
Instructor						

Table 5.5: Clusters of class items ranked by perceived learning efficacy

Cluster	**Class element(s)**	**Average Ranking**
1	Programming labs	1.42
2	OpenDSA Exercises	2.48
3	Textbook (paper)	4.03
	Lectures	
	Interacting with instructor	
	Reading quizzes	

embedded in it. He analyzed the time spent by students on each page and concluded that students do not read algorithm analysis content.

In this section, we investigated students use of AVs when working on OpenDSA exercises and found that students go through AVs when working on assignments. That coupled with the above results might suggest that students prefer content presented using visualizations rather than just text.

Participants

Participants were sophomore students enrolled in CSCI102 course at New York University (n=79). In CSCI102, OpenDSA was used as supplemental resource. However, students had to complete one OpenDSA exercise.

Materials and procedure

Students in CSCI102 had to complete only one mandatory OpenDSA assignment. The assignment consisted of a small programming exercise about using a List ADT as shown on Figure 5.3. Out of the 79 students enrolled in the course, 72 completed the assignment. Most students (65) completed it between February 7-14, 2014 (the due date was February 13).

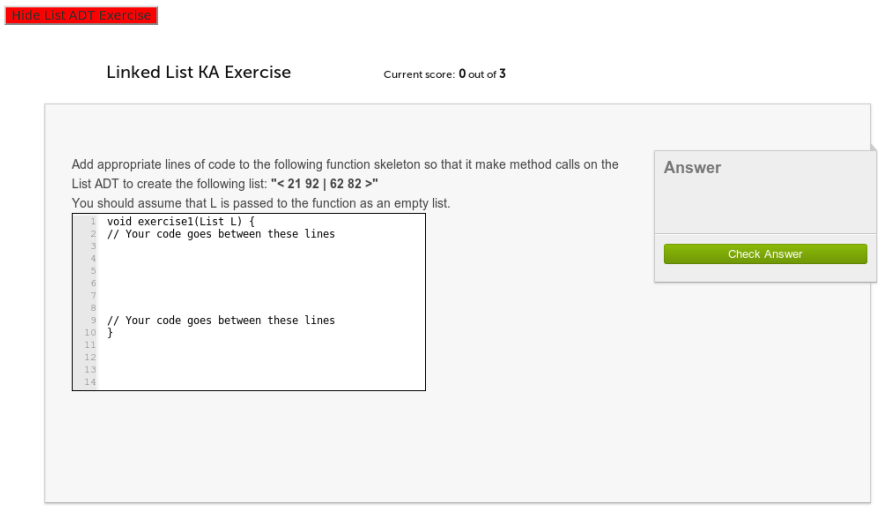

Fig. 5.3: Linked List programming exercise

On the eTextbook page with the programming exercise were two visualizations. We will refer to them as AV1 and AV2. The AVs describe the behavior of linked lists, and do not contain any lines of code as shown in Figure 5.4. Thus, students could not use the visualizations to find a direct answer to the problem, but instead might use them to learn about the mechanics of lists, which would prepare them to solve the exercise. We used student interaction data recorded by the OpenDSA data collection server to investigate CSCI102 students use of AVs.

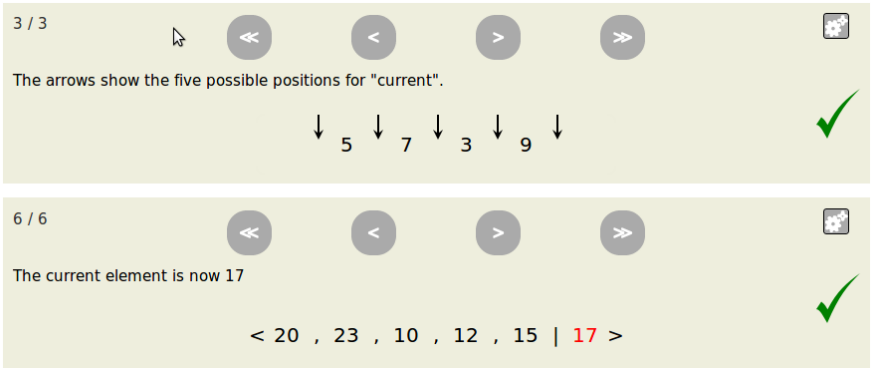

Fig. 5.4: Linked List visualizations

Results

For each day of the assignment, we plotted the number of exercises completed along with the number of AV completed. Figure 5.5 shows an almost one-to-one correlation

between the number of exercises completed and the number of AVs visited, indicating that almost all of the students used the AVs when solving the exercise. Each day during the active period, the number of students viewing the AVs closely matched the number of students working the exercise. In all, out of 72 students, 63 students viewed AV1 and 61 students viewed AV2.

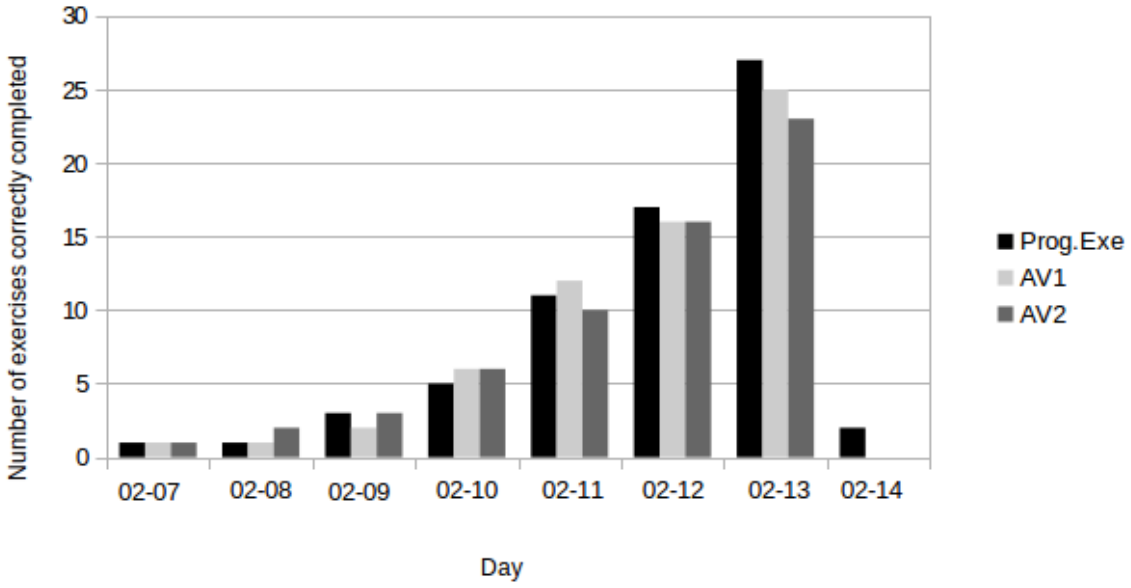

Fig. 5.5: Mandatory exercise and AVs completion distribution (CSCI102)

The programming exercise was "hidden" behind an HTML button, thus providing us with a way to track when the student displayed (and we assume read) the exercise instructions. We wanted to know how the AVs were incorporated into student's study workflows. We identified the following scenarios based on the order in which students completed the AVs and exercises.

1. The student goes through the AVs, then displays and completes the programming exercise
2. The student displays the exercise (reads the instructions), goes through the AVs, then completes the exercise
3. The student displays and completes the exercise, then goes through the AVs
4. The student displays and completes the exercise, and never goes through the AVs

We found that 38 students followed Scenario 1, 27 students followed Scenario 2, and 4 students followed Scenario 4. No students were found for Scenario 3. Among the students who followed scenario 2, 4 unsuccessfully attempted the programming exercise before deciding to go through the module's content. The majority of the students in that category did not make any attempt to work on the exercise prior to interacting with the AVs.

The fact that the majority of students who looked at the exercise decided to go through the AVs before attempting the exercise is an indication that they found the AVs useful. The next most used interactive item from the OpenDSA material used by this class was viewed by only 12 students (see Table 5.6). Recall that these students did not have required exercises related to the remaining content.

Table 5.6: Interactive items ranked by student attempts

Exercise	Count
List programming exercise	72
List AV 1	63
List AV 2	61
Array-Based Queues AV	12
Array-Based List AV	10

5.2.4 Student voluntary use of OpenDSA

We sought to find out if students would only use OpenDSA when required, or if they would use it voluntarily such as to review for exams.

Participants

Participants included in this study were students enrolled in CS3114-SP13-A, CS2114-SP14, and CSCI102 courses.

Procedure

We used interaction data collected by the data collection server to analyze students use of OpenDSA. We plotted timelines of interactions within OpenDSA and added class events (homework due date, exams date) to the graph.

Results

Logs analysis showed that students used the book to review for exams, thus indicating the usefulness of OpenDSA as a study aid. Figures 5.6, 5.7 and 5.8 present timelines showing exercise completions. Clusters of interactions appear not only at homework due dates as expected, but also immediately prior to the midterm.

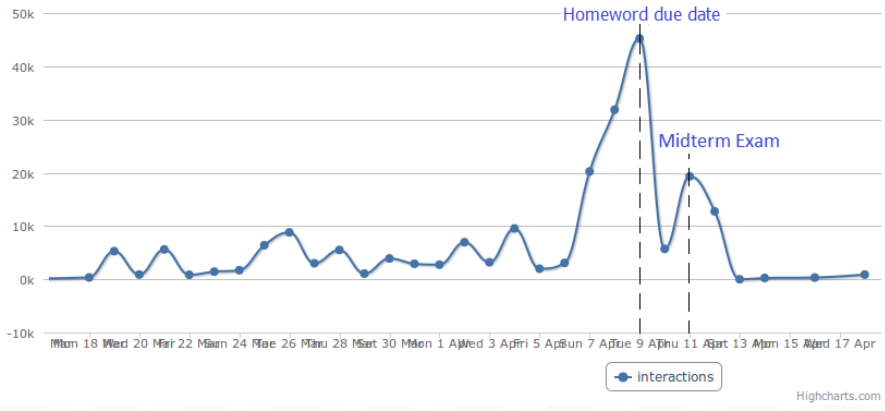

Fig. 5.6: Timeline of interactions: CS3114-SP13-A (sorting and hashing chapters).

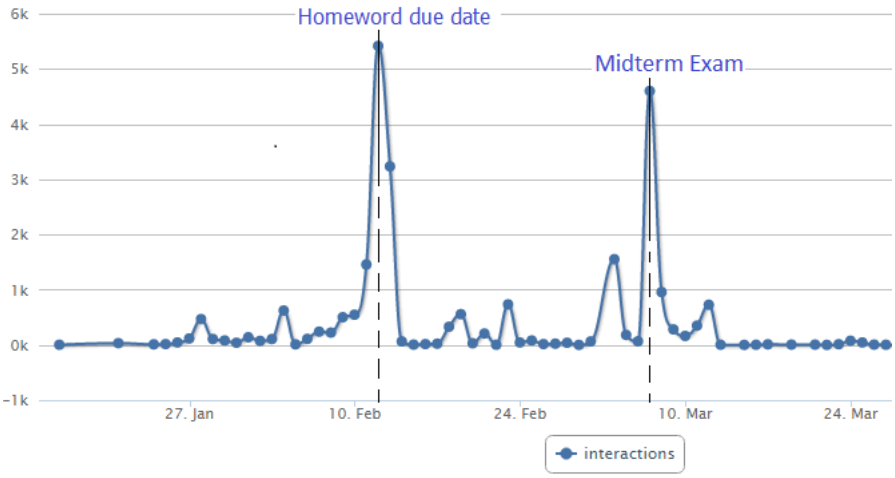

Fig. 5.7: Timeline of interactions: CSCI102.

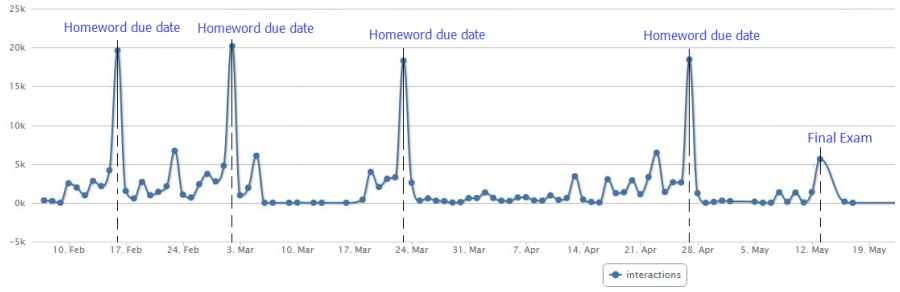

Fig. 5.8: Timeline of interactions: CS2114-SP14.

We further analyzed data from CS2114-SP14 to find out when students actually study. In that course, all the assignments were due Sunday at 11:59PM, while the lecture sessions happened on Mondays and Wednesdays. The students usually had a week to complete a given assignment associated with the prior week's lectures. We analyzed our log data to see when the students usually work on OpenDSA exercises.

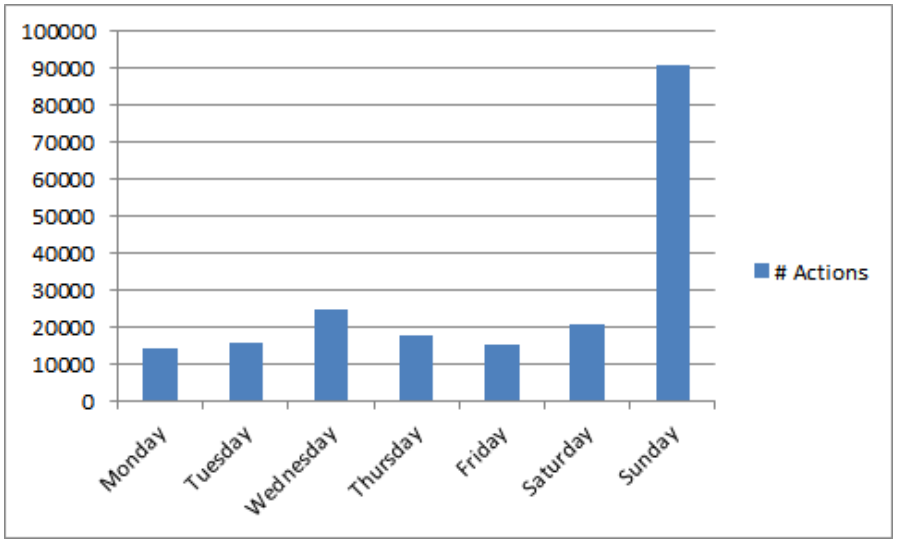

Fig. 5.9: Daily distribution of all interactions (CS2114-SP14).

Figures 5.8 and 5.9 show that most exercises were done on the due date (Sundays). Furthermore, on Sundays, most of the work is done between 2pm and midnight as shown in Figure 5.10. The above results showed that most students waited until the last day to do their homework. The relative surge of activity on Tuesday, May 13 corresponds to the course final examination. This activity represents use of the materials as a study aid, rather than for completing graded assignments.

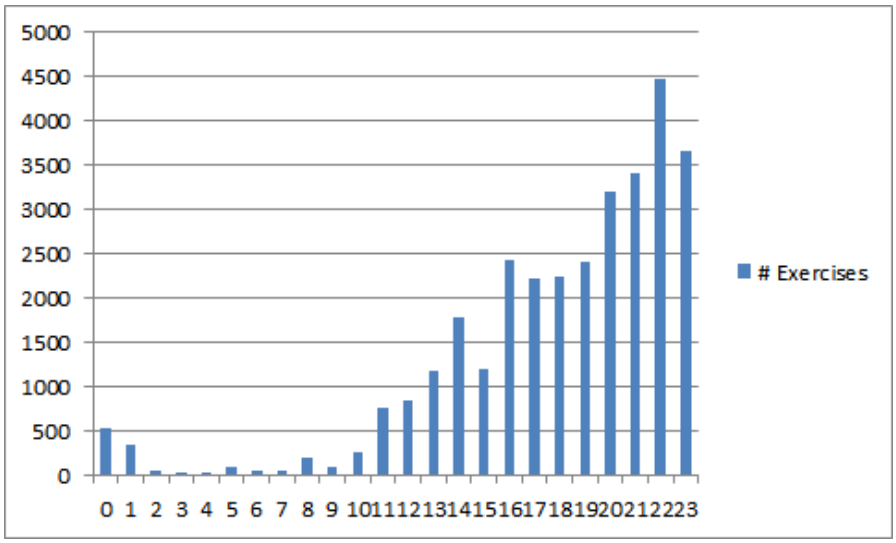

Fig. 5.10: CS2114-SP14 Hourly distribution of exercises performed on Sundays

The analysis of interactions logs indicated that students use OpenDSA voluntarily to review for exams, regardless of the way OpenDSA was used in the classroom. The found similar behavior in all courses that used OpenDSA.

5.3 OpenDSA use preference

5.3.1 OpenDSA use and students' ability to learn course material

In order to assess how students would prefer to use OpenDSA in the classroom, we asked the students to rate their ability to succeed in the course under various scenarios of use for OpenDSA.

We hypothesize that the variety and the quantity of assessment activities within OpenDSA will encourage students to use OpenDSA voluntarily (for example to study for examinations). In addition, OpenDSA's assessment activities provide the learner with many mastery-based experiences, thus improving students' ability to learn course material. The ability to learn course material is loosely related to self-efficacy. Bandura [8] defined self-efficacy as "*the expectation of personal efficacy*", and argued that it is a good predictor of the amount and duration of effort one will put into completing a task. OpenDSA provides students with multiple mastery-based experiences (many exercises with automated feedback).

Self efficacy has been used to evaluate the impact of teaching approaches in CS courses. Hundhausen et al. used self-efficacy as one of the measures to evaluate the effectiveness of "studio-based" instruction in a CS1 course. In a quasi-experimental comparison, they investigated the difference in self-efficacy between students enrolled in the studio-based course and students enrolled in the traditional course. They measured self-efficacy in both groups at the beginning and at the end of the experiment. They found that self-efficacy decreased in both groups. However, student's self-efficacy decreased more in the traditional course [39]. Zingaro [86] reports that peer instruction improved students' self-efficacy in a CS1 course.

Participants

Participants were students enrolled in CS2114-SP14 (CS2) and CS3114-F14 (CS3) at Virginia Tech. In both groups, OpenDSA exercises were used as mandatory assignments. OpenDSA was the class textbook in CS3114-F14.

Materials and procedure

At the end of the semester students answered survey questions asking them to rate their ability to learn course material under several (hypothetical) scenarios. We asked students in CS2114-SP14 to use a likert scale of 1 (low confidence) to 5 (high confidence) to rate their confidence in their ability to learn the CS2114-SP14 course materials under the following conditions:

- Scenario 1: I could succeed in learning the CS2114 course material if the class was only taught using a standard textbook with Moodle quizzes.
- Scenario 2: I could succeed in learning the CS2114 course material using OpenDSA on my own if I was required to do the OpenDSA exercises for a grade.
- Scenario 3: I could succeed in learning the CS2114 course material using OpenDSA on my own with no graded exercises.
- Scenario 4: I could succeed in learning the CS2114 course material using OpenDSA if the instructor goes over OpenDSA's content during lecture but I am NOT required to do the exercises for a grade.
- Scenario 5: I could succeed in learning the CS2114 course material using OpenDSA if the instructor goes over OpenDSA's content during lecture and I AM required to do the OpenDSA exercises for a grade.

Students enrolled in CS3114-F14 were asked to answer the same question but scenario 1 was removed since OpenDSA was the textbook of the course and the course did not include Moodle quizzes. The questionnaire used in this study can be found in Appendix B.

Results

The analysis of student responses showed that students ranked scenarios involving using OpenDSA during lecture and for graded assignment higher. In CS2114-SP14, out of the 49 responses we collected, 38% gave scenario 5 the highest score of 5, and 69% of the responders ranked their confidence in succeeding in the class under scenario 5 with a score of 4 or 5. Scenarios involving mandatory OpenDSA exercises (for grading) received higher ability to learn course content scores, thus showing that students have a preference for for that type of use of OpenDSA. The above results also corroborate our hypothesis that the mastery-based experience provided by OpenDSA exercises will help student ability to learn course materials. Figure 5.16 shows score distribution for each scenario for CS2114-SP14 students. We found similar results from CS3114-F14 responses. Tables 5.7 and 5.8 shows the average score for each scenario.

Table 5.7: OpenDSA use scenarios and ability to learn (CS2114-SP14)

Scenario	**Ability to learn score** (n=49)
Scenario 1	2.64 (SD=1.22)
Scenario 2	3.6 (SD=1.01)
Scenario 3	3.02 (SD=1.09)
Scenario 4	3.34 (SD=1.04)
Scenario 5	4.2 (SD=0.82)

CS3114-F14 students answers are summarized in Table 5.8

Table 5.8: OpenDSA use scenarios and ability to learn (CS3114-F14)

Scenario	**Ability to learn** (n=34)
Scenario 2	3.9 (SD=0.96)
Scenario 3	2.9 (SD=1.08)
Scenario 4	3.1 (SD=0.91)
Scenario 5	4.5 (SD=0.61)

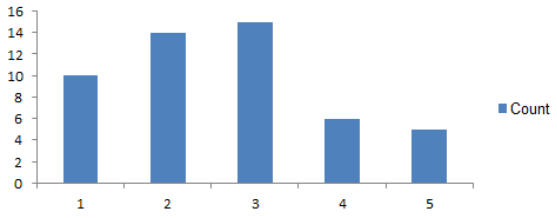

Fig. 5.11: Scenario 1 rank distribution

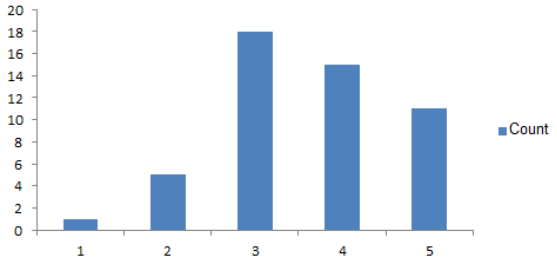

Fig. 5.12: Scenario 2 rank distribution

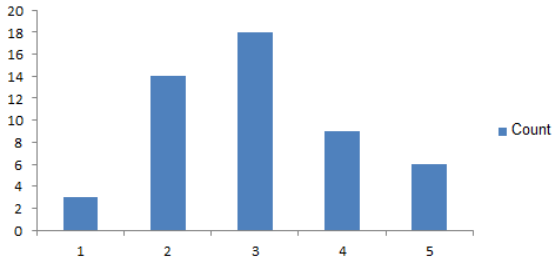

Fig. 5.13: Scenario 3 rank distribution

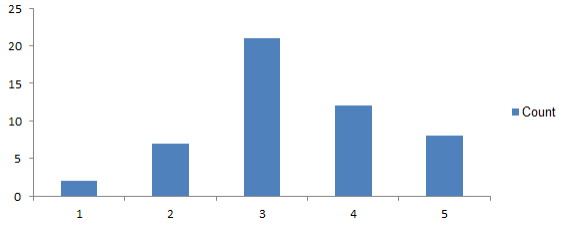

Fig. 5.14: Scenario 4 rank distribution

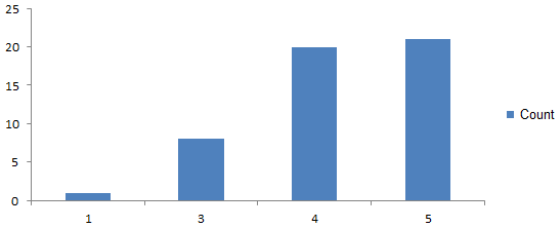

Fig. 5.15: Scenario 5 rank distribution

Fig. 5.16: Scenarios rank distribution: CS2114-SP14

A Kruskal Wallis test revealed a significant effect for scenarios of use on ability to learn course materials for CS2114-SP14 ($X^2(4) = 51.9, p < 0.01$) and CS3114-F14 ($X^2(3) = 48.5, p < 0.01$). A post-hoc test using Wilcoxon tests with Bonferroni correction identified two clusters for CS2114-SP14 data as shown in Table 5.9.

Table 5.9: Clusters of OpenDSA use scenarios ranked by ability to learn course material (CS2114-SP14)

Cluster	**Scenario(s)**
1	Scenario 5
2	Scenario 2
	Scenario 4
	Scenario 3
	Scenario 1

A post-hoc test using Wilcoxon tests with Bonferroni correction identified three clusters for CS3114-F14 data as shown in Table 5.10.

Table 5.10: Clusters of OpenDSA use scenarios ranked by ability to learn course material (CS3114-F14)

Cluster	**Scenario(s)**
1	Scenario 5
2	Scenario 2
3	Scenario 3
	Scenario 4

The findings from the two groups seems to indicate that students might have a preference for a class format in which OpenDSA exercises are used as mandatory assignments.

Effect of OpenDSA use on student's ability to learn course material

We further investigated what aspect of OpenDSA use had the most impact on student's perceived ability to learn course content. We used data from CS3114-F14 to assess the impact of OpenDSA use in lecture and mandatory OpenDSA exercises on student ability to learn course content. We built a model with "ability score (ability)" as outcome and with "OpenDSA used in class lectures (ODSAIC)" and "OpenDSA exercises used as graded homework (ODSAFG)" as explanatory variables. The levels of the explanatory variables and the relationship between explanatory variables and scenarios of use are shown in Table 5.11.

Table 5.11: OpenDSA use scenarios

Scenario	ODSAIC	ODSAFG
S2	0	1
S3	0	0
S4	1	0
S5	1	1

ODSAIC: OpenDSA used during lectures
ODSAFG: OpenDSA exercises used for graded assignments

Table 5.12 shows average ability score for each combination of explanatory variables levels. Figure 5.17 shows ability score evolution based on combinations of explanatory variables levels.

Table 5.12: Ability to learn mean score by OpenDSA use scenarios (CS3114-F14)

		ODSAFG	
		0	1
ODSAIC	0	2.91	3.91
	1	3.12	4.50

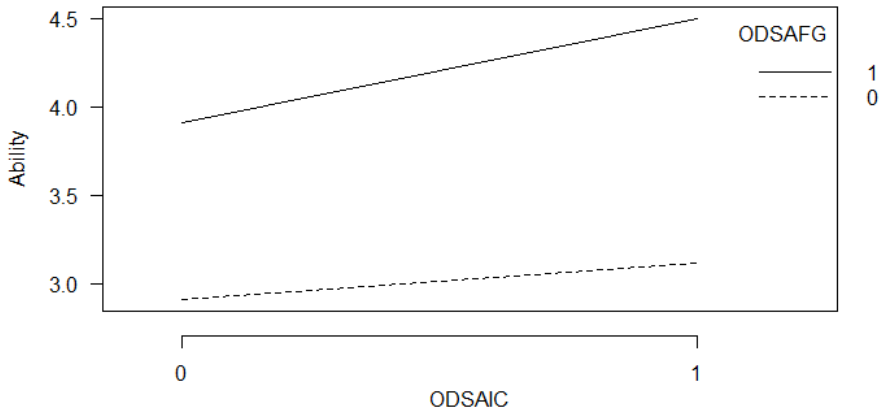

Fig. 5.17: Effect of OpenDSA use on student ability to learn course material, CS3114-F14

We investigated if the effect of explanatory variable interactions was significant in the evolution of ability score. The result of the ANOVA test for the statistical model including ODSAIC and ODSAFG interaction is shown in Table 5.13. ODSAIC and ODSAFG effects were statistically significant.

Table 5.13: ANOVA for CS3114-F14 data

Source	Df	Sum Sq	Mean Sq	F value	Pr(>F)
ODSAIC	1	5.36	5.36	6.46	0.012*
ODSAFG	1	48.24	48.24	58.156	$4.24e-12$**
ODSAIC*ODSAFG	1	1.24	1.24	1.498	0.2232
Residuals	132	109.50	0.83		

Signif. codes: ('**':0.001), ('*:0.05)

The effect of the using OpenDSA during lecture and using OpenDSA exercises for graded assignments simultaneously is not significant on student ability to learn course content.

Using OpenDSA as lecture aide or using OpenDSA exercises for graded assignments significantly affected students perception of their ability to learn course content. Using OpenDSA exercises as graded assignments seems to have the most effect as shown in Table 5.14.

Table 5.14: Parameter estimates:CS3114-F14

Term	Estimate	Std. Error	t value	Pr(>|t|)
Intercept	2.8162	0.1355	20.780	$< 2e-16^{**}$
ODSAIC1	0.3971	0.1565	2.537	0.0123^{*}
ODSAFG1	1.1912	0.1565	7.612	$4.44e-12^{**}$

Signif. codes: ('**':0.001), ('*:0.05)

5.3.2 Student use of mobile devices

As handheld wireless mobile computers become more ubiquitous, their potential use and benefit in education is being examined. Learning technology developers and users are excited about the opportunities provided by mobile devices [67]. In their meta-analysis of mobile learning studies, Wu et al. [84] concluded that most research in mobile learning studies resulted in positive outcomes. Also, they found that mobile learning was most frequently used by students in the professional and applied sciences field. Computer Science was the second most investigated discipline (after Languages and Linguistics) in regard to mobile learning. Other studies show that college students want to use mobile devices more for learning and hope that their instructors and institution will create an environment that promotes the use of mobile devices [27].

OpenDSA gives us an opportunity to study the use of mobile devices for learning in CS courses. We looked at log data to get a sense of the proportion of interactions originating from mobile devices (smartphone, small size tablet, full size tablet). Since more students own tablets (handheld devices with screen size above 7"), we are interested in measuring the use of mobile devices to access online tutorials.

During Spring 2014, in addition to analyzing the logged data, we collected data from students in CS2114-SP14 to uncover the factors that encouraged or prevented the students from using their mobile device. In CS2114-SP14, students had to complete weekly OpenDSA assignments, and OpenDSA homework accounted for 2% of the total course grade. Most students enrolled in courses using OpenDSA own a mobile device in addition to their Laptop/PC. We tracked the origin of user interactions between Fall 2013 and Spring 2014, and found that less than 1.5% of the traffic originated from a mobile device. We surveyed students enrolled in CS2114-SP14 to gather their experiences and opinion about mobile devices and OpenDSA. We got a total of 51 responses. They revealed that:

- 68% of respondents own a laptop computer with no touch screen
- 49% own a touch screen laptop
- 72% own a smartphone
- 17% own a small tablet (approx. 7")
- 15% own a full-size tablet (approx. 10")

The mobile devices ownership distribution in CS2114-SP14 is consistent with the general US population statistic[3] , and can be generalized at least to the other VT courses that we studied. From the log data, we found that out of 176 students enrolled in CS2114-SP14, only 5 accessed OpenDSA from a mobile device at least one time. The ratio of mobile users was equally low in other VT courses using OpenDSA. We did not expect smartphone owners to access the materials via their devices, because several factors make it difficult to effectively use them to study (screen size, battery life, and usability concerns). But even small- and full-sized tablet owners rarely tried to access the tutorials from their devices. This situation seemed to contradict previous research showing students positive attitude towards using full size tablets to study [69]. We tried to determine if the low use of mobile devices was due to some serious usability bug within OpenDSA. When asked to rank course resources by positive learning impact, students ranked OpenDSA as the second most important (after programming labs), thus proving that the students did not have a negative opinion of the OpenDSA system. The low number of mobile users seems to indicate that the reason not to use mobile devices was made independently of OpenDSA, since the vast majority (97%) of students never accessed OpenDSA from a mobile device.

We then sought to see if the nature of the course made it harder for students to use mobile devices. We asked students if they ever used their mobile devices to study in other courses. We got 15 positive answers out of 51 responses. But there were only 2 positive answers when asked if they used mobile devices to access OpenDSA content. Further investigation will be necessary to understand the difference. The most common reason mentioned by students to justify their preference for non-mobile devices was the "convenience". Most students said that it is more convenient to use a laptop or a desktop computer when working on OpenDSA assignments. They mentioned the bigger screen and the full keyboard of laptops/desktops as key elements, confirming the findings in [69]. As we mentioned earlier, many OpenDSA exercises are proficiency exercises, suitable for touch interaction, and do not require typing text. Even the KA exercises do not require the user to type in significant text. We saw earlier that most students wait until late to do their homework. Several studies have shown that students experienced greater stress toward the end of an assignment period, when they are rushing to complete

[3] `http://www.pewinternet.org/factsheets/mobile-technology-fact-sheet/` (accessed: 04/09/2015)

the assignment on time [77]. For students who study close to the deadline, it makes no sense to include a mobile device into the study workflow. They will go for the device that will give them more capabilities (bigger screen, possibility to open multiple tabs, full keyboard, etc.) to complete the homework in a small amount of time. Under those circumstances, mobile devices cannot compete with laptops/desktops.

5.4 OpenDSA and student performance

In our opinion, the single most important pedagogical feature of OpenDSA is the interactive exercises. We investigated the relationship between OpenDSA exercises and student performance on tests. We only focused on students' declarative knowledge, hence only students' performance on written tests are included in our analysis.

OpenDSA includes three main categories of exercises: **simple questions**, **proficiency exercises**, and **programming exercises**. Simple questions include traditional T/F, multiple choice, or fill-in-the-blank questions. We will label those "KA exercises". Proficiency exercises are visual algorithm simulation activities and comprise two types of exercises. Mini proficiency exercises (labeled "KAV exercises") are exercises requiring the student to perform only few steps of an algorithm and full proficiency exercises (labeled "PE exercises") asked the student to perform all the steps of an algorithm.

We collected the following data for each student:

- Total number exercises attempted (Total.KA for simple questions, Total.KAV for mini proficiency exercises, and Total.PE for full proficiency exercises).
- Total number of completed exercises (Correct.KA for simple questions, Correct.KAV for KA mini proficiency exercises, and Correct.PE for full proficiency exercises)

5.4.1 Students' scores quartile analysis

We collected and analyzed the data from CS3114-F13. The course's OpenDSA instance had a total of 95 required exercises, of which 36 were KA simple questions sets, 30 were KA mini proficiency exercises, 26 were full proficiency exercises, and 3 were other interactive activities. We divided the data into two groups: M1 for all activity until the first midterm examination, and M2 for all activity between the first and second midterms. The final exam was on sections with little or no interactive elements in OpenDSA, so it is not included in our analysis.

Before Midterm 1, students were required to solve a total of 15 KA simple question sets, 12 KA proficiency exercises, and 6 JSAV proficiency exercises. Between Midterm

1 and Midterm 2, they had to complete 19 KA simple question sets, 18 KA mini proficiency exercises and 20 JSAV full proficiency exercises. Figures 5.18 and 5.19 show the performance distribution. The $x - axis$ of the Correct.KA, Correct.KAV, and Correct.PE histograms represent the number of exercises correctly completed. In the case of KA simple questions and KA proficiency exercises it is the number of times a student reached the proficiency score. For JSAV proficiency exercises, it is the number of times a student reached the proficiency threshold (correctly simulated 90% of an algorithm's steps).

We see from the histograms that most students correctly completed most exercises only one time regardless of the type of question. The exception was the JSAV proficiency assignments before the second midterm, where the majority of students correctly completed exercises more than once. We can explain the exception by the fact that Midterm 2 covered the Sorting and Hashing chapters. Both included concepts that were fairly new to students. Midterm 1 covered topics where the students had had previous exposure (linear data structures, algorithm analysis and binary trees).

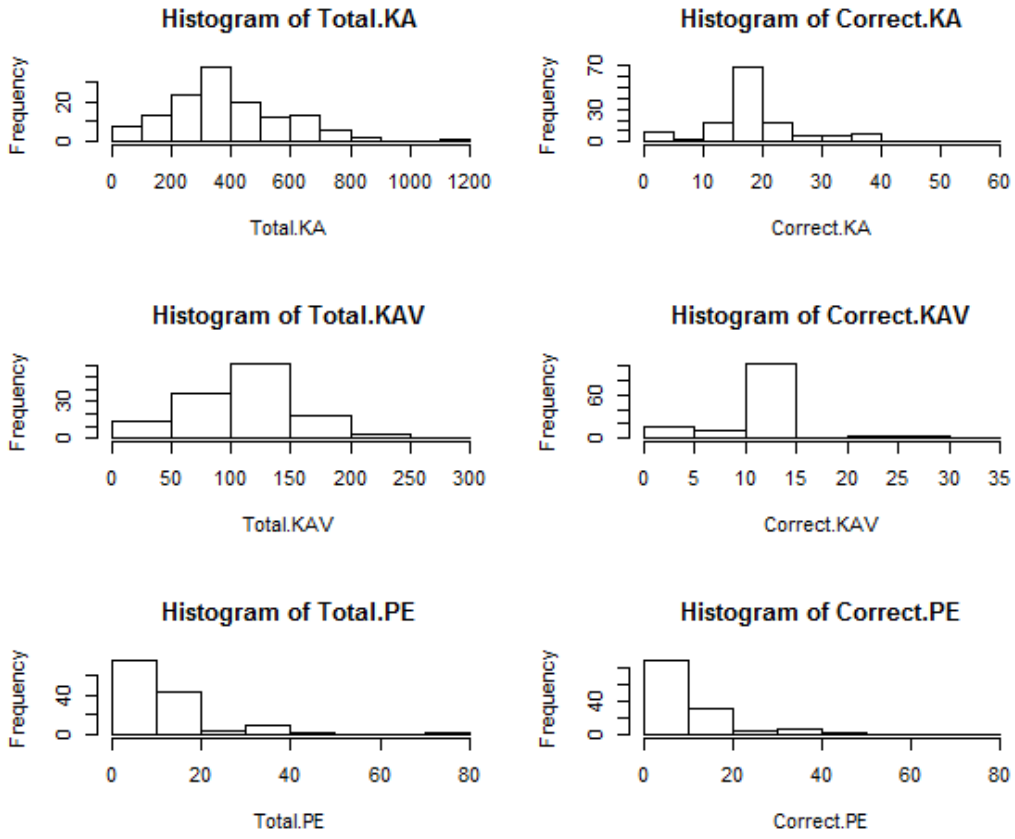

Fig. 5.18: CS3114-F13 - Midterm 1 distribution

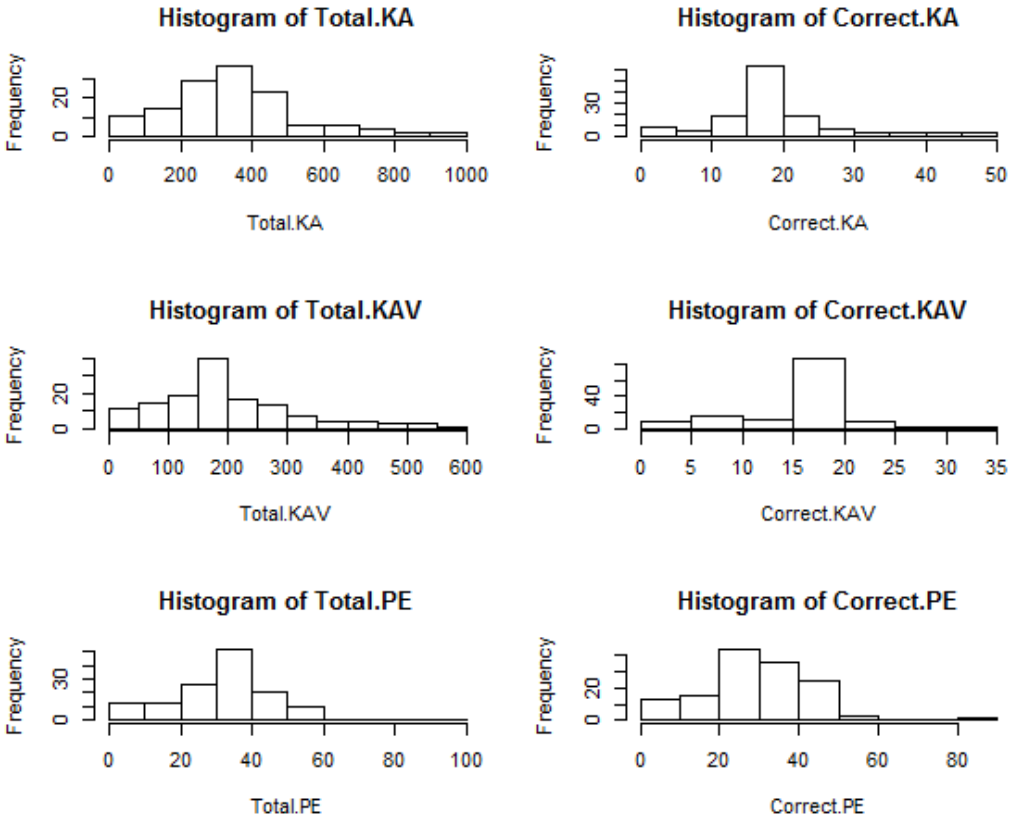

Fig. 5.19: CS3114-F13 - Midterm 2 distribution

Since there was no penalty associated with attempting an exercise several times, we expected to see a difference between the students who correctly completed the exercises "post-proficiency". We found a low correlation between the amount of correct exercises (regardless of the type) completed by the students and exam scores as shown in Table 5.15. The negative correlation between the total number of KA simple questions attempted suggests that some students were just trying out all the provided answers until they find the right one, without deeply thinking about the exercise's answer.

We wanted to know if analysis of student midterm scores will show different patterns of OpenDSA exercise use. We computed the average number of correct exercises performed by students grouped by score quartile as follows: $q0$ represents students who performed below the 25^{th} percentile; $q1$ represents students with score between the 25^{th} and the 50^{th} percentile; $q2$ represents students with score between the 50^{th} and the 75^{th} percentile; and $q3$ represents students with score above the 75^{th} percentile. Tables 5.16 and 5.17 show the statistics of the different groups. As shown in Figure 5.20, **students with higher scores tend to correctly complete more exercises than those with**

Table 5.15: CS3114-F13 - Correlation table

	Score	
Variables	Midterm 1	Midterm 2
Total.KA	−0.11	0
Correct.KA	0.32	0.29
Total.KAV	0.09	0.01
Correct.KAV	0.29	0.25
Total.PE	0.15	0.20
Correct.PE	0.19	0.20

lower score.

Table 5.16: CS3114-F13 - Average correct exercises completed, grouped by Midterm 1 score

Groups	Correct.KA	Correct.KAV	Correct.PE
$q0$ (N=35)	15.14 (SD=8.04)	9.17 (SD=4.85)	9.37 (SD=7.36)
$q1$ (N=37)	19.05 (SD=9.82)	11.43 (SD=4.11)	11.75 (SD=8.36)
$q2$ (N=32)	19.43 (SD=5.89)	11.53 (SD=3.91)	13.93 (SD=14.43)
$q3$ (N=31)	23.64 (SD=9.15)	12.70 (SD=5.71)	13.87 (SD=12.18)

Table 5.17: CS3114-F13 - Average correct exercises completed, table grouped by Midterm 2 score

Groups	Correct.KA	Correct.KAV	Correct.PE
$q0$ (N=38)	15.42 (SD=5.91)	14.55 (SD=6)	26.89 (SD=12.76)
$q1$ (N=31)	16.83 (SD=10.11)	14.22 (SD=6.57)	26.45 (SD=13.08)
$q2$ (N=33)	18.09 (SD=7.27)	16.33 (SD=4.55)	31.24 (SD=14.93)
$q3$ (N=33)	24.60 (SD=12.09)	18.78 (SD=5.75)	34.03 (SD=11.85)

The difference in the average number of correct exercises between student groups was significant at the $p < 0.05$ level for KA simple questions and KA mini proficiency exercises for Midterm 1 and Midterm 2. However, the difference was not statistically significant for full proficiency exercises. The difference was still not significant when merging the number of correct mini proficiency exercises with the number of correct full proficiency exercises.

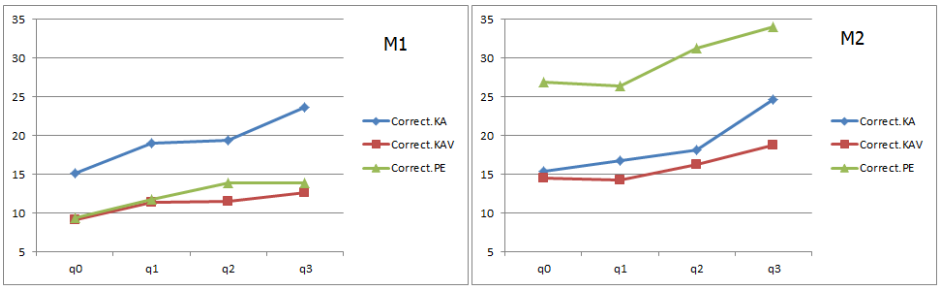

Fig. 5.20: CS3114-F13 - Midterm 1 and Midterm 2 exercises means per quartile

We can say that higher scores on written tests are associated with a high number of correct exercises (beyond the minimum level required to receive full homework credit). These findings can be used to detect struggling students, since they are less likely to redo an exercise. In addition, students with the lowest grades have the lowest "number of correct exercises to number of exercises attempted" ratio.

5.4.2 Extra work in OpenDSA and student grades

We investigated if "extra work" in OpenDSA correlates to higher scores at exams. To investigate the relationship, we used the data from the CS3114-F13 course. To quantify "extra work", we grouped students into three clusters based on the number of exercises completed. We named the clusters based on the level of completion: "Below" for students' who did not or barely completed the amount of work required by the instructor in the course; "Average" for students' who completed about what was required in the course; and "Above" for students who did more work in OpenDSA than what was required. The number of required exercises (by the instructor) is in the "Average" cluster. Figure 5.21 displays the clusters distribution.

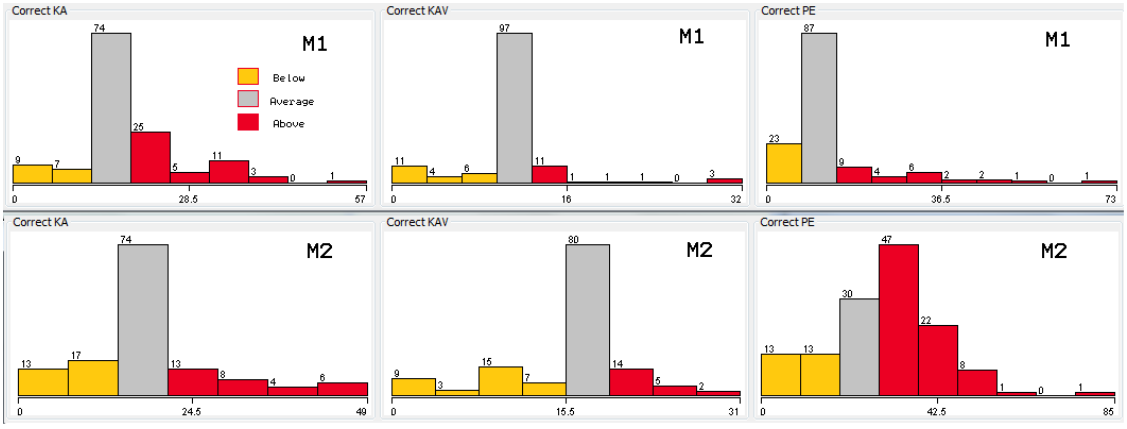

Fig. 5.21: CS3114-F13 - OpenDSA exercises completion distribution

The bounds for each level are shown in Table 5.18 (M1 and M2 represent Midterm 1 and Midterm 2, respectively).

Table 5.18: CS3114-F13 - Exercises completion categories thresholds

	KA		KAV		PE	
Level	M1	M2	M1	M2	M1	M2
Below	$[0, 12]$	$[0, 14]$	$[0, 9]$	$[0, 15]$	$[0, 7]$	$[0, 18]$
Average	$[13, 19]$	$[15, 21]$	$[10, 12]$	$[16, 19]$	$[8, 14]$	$[19, 28]$
Above	$[20, 57]$	$[22, 49]$	$[13, 32]$	$[20, 31]$	$[15, 73]$	$[29, 85]$

For each category, we computed the average scores grouped by type of exercises. Tables 5.19, 5.20, 5.21, 5.22, 5.23, and 5.21 summarize our results for Midterm 1 and Midterm 2. The mean midterm score increased with the number of exercises completed.

Table 5.19: Average Midterm 1 scores grouped by correct KA exercises completion category

Group	Score
Below (N=16)	63.62 (SD=16.26)
Average (N=74)	74 (SD=11.53)
Above (N=45)	76.9 (SD=13.77)

Table 5.20: Average Midterm 1 scores grouped by correct KAV exercises completion category

Group	Score
Below (N=21)	68.23 (SD=16.42)
Average (N=97)	73.9 (SD=12.41)
Above (N=17)	79.58 (SD=12.99)

Table 5.21: Average Midterm 1 scores grouped by correct PE exercises completion category

Group	Score
Below (N=23)	67.13 (SD=15.23)
Average (N=87)	74.9 (SD=12.67)
Above (N=25)	75.76 (SD=12.87)

Table 5.22: Average Midterm 2 scores grouped by correct KA exercises completion category

Group	Score
Below (N=30)	69.16 (SD=13.29)
Average (N=74)	71.79 (SD=16.46)
Above (N=31)	80.77 (SD=11.69)

Table 5.23: Average Midterm 2 scores grouped by correct KAV exercises completion category

Group	Score
Below (N=34)	67.11 (SD=17.08)
Average (N=80)	74.61 (SD=14.26)
Above (N=21)	78.14 (SD=13.74)

Table 5.24: Average Midterm 2 scores grouped by correct PE exercises completion category

Group	Score
Below (N=26)	68.26 (SD=12.50)
Average (N=30)	69.36 (SD=22.08)
Above (N=79)	76.40 (SD=12.13)

Except for Midterm 2 PE exercises, the difference in the average score between student groups was significant at the $p < 0.05$ level for all three types of exercises and for both midterms. Post hoc t-tests ($\alpha = 0.05$) identified the following clusters (mean score difference statistically significant) for each midterm and for each type of exercise:

- Midterm 1
  - KA: [Below], [Average, Above]
  - KAV: [Below], [Above]
  - PE: [Below], [Above]
- Midterm 2
  - KA: [Below, Average], [Above]
  - KAV: [Below], [Above]
  - PE: N/A

KA exercises: For Midterm 1 we had two clusters with average and above levels clustered together. Correctly completing exercises at level below (what was required in the course) was associated with lower midterm scores compared to students who worked at levels average and above (Midterm 1). We obtained two clusters for Midterm 2 data. Below and average levels grades were not significantly different. Doing work at level average or below was associated with lowest midterm scores compared to students who performed at level above. We can conclude that for KA exercises, performing "above" what was required by the course was associated to higher scores at midterms.

KAV exercises: For Midterm 1 and Midterm 2 we obtained two clusters. The difference in mean scores was statistically significant between levels above and below. Since mean score of performers at level average could not be differentiated from those performing at levels below and above, we could not make any conclusion regarding a potential relationship between level of work on KAV exercises and scores at midterms.

PE exercises: For Midterm 1, PE exercises data behave like KAV exercises data. We obtained two clusters, with no statistically significant difference between levels average and below, and between level average and above. No clusters were identified for Midterm 2. It was not possible to conclude any potential relationship between level of work on PE exercises and grades at midterms.

5.4.3 Grade fluctuations and OpenDSA exercises

In the absence of a controlled experiment, we used the epidemiology technique "dose-response relationship" to investigate whether work in OpenDSA impacts students' performance. The idea here is to try to verify two conditions:

- Students who completed a lot of exercises and had high scores (at midterms), obtained lower scores when performing fewer OpenDSA exercises.
- Students who completed few OpenDSA exercises and had lower scores, saw an improvement in their scores when performing more OpenDSA exercises.

We identified all students with score difference between Midterm 1 and Midterm 2 of at least one standard deviation (of Midterm 2 scores) in both directions (drop or increase), and see if the difference in score can be explained by their use of OpenDSA. For each exercise type (KA, KAV, PE), we computed the mean number of exercises completed by the subjects before Midterm 1 and before Midterm 2. Out of 135 students, we found 42 with a positive score fluctuation and 10 with a negative score fluctuation.

1. **Positive fluctuation analysis**

   Here we are looking at students who experienced an increase of their scores percentile rank of at least one standard deviation between Midterm 1 and Midterm 2. The results for subjects in this category are shown in Table 5.25.

   Table 5.25: CS3114-F13 - Positive fluctuation summary (normalized data)

	KA		KAV		PE	
(n=42)	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Midterm 1	0.3	0.16	0.31	0.15	0.14	0.1
Midterm 2	0.41	0.23	0.52	0.22	0.37	0.15

   We saw that the number of exercises completed increased for all types. The difference between the mean number of exercises completed between Midterm 1 and Midterm 2 was statistically significant for KA, KAV, and PE exercises ($\alpha = 0.01$).

   All types of exercises seem to have a significant positive effect in score increase under the protocol we followed, we can conclude that more exercises completion seems to have a positive effect on student score.

2. **Negative fluctuation analysis**

   Here we are looking at students who experienced a drop of at least 1 standard deviation in their scores rank between Midterm 1 and Midterm 2. The results for subjects in this category are shown in Tables 5.26.

   The average number of exercises completed decreased only for KA and KAV exercises. We did not observe a change in the number of PE exercises completed between Midterm 1 and Midterm 2. The difference in mean number of exercises completed was significant only for KA exercises. The difference was not statistically significant for PE and KAV exercises.

Table 5.26: CS3114-F13 - Negative fluctuation summary (normalized data)

	KA		KAV		PE	
(n=10)	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
Midterm 1	0.39	0.15	0.49	0.27	0.24	0.18
Midterm 2	0.26	0.12	0.39	0.25	0.24	0.15

Only KA exercises seem to have a significant effect in negative score fluctuation. We cannot conclude that change in OpenDSA behavior impacted scores differences.

The above results do not allow us to affirm causality between OpenDSA and students' performance. However, we observed that KA exercises seems to have an effect on student performance on written tests.

5.5 Discussion

We performed many user studies when exploring OpenDSA impact on students and found the following.

- **Positive opinion:** Students rated their experience with OpenDSA positively and used it to review for exams. They found OpenDSA helpful and indicated that OpenDSA was among the most useful resources. Students in CS2114-SP14 ranked OpenDSA as the second most useful class element. CS2114-SP14 is a programming-intensive course, and given that OpenDSA exercises accounted for only 2% of students final grade, which makes the ranking of OpenDSA right behind programming labs more meaningful.
- **AVs use and importance:** Students used AVs when solving problems. We have evidence showing that students will use AVs voluntarily when studying. We also have early results showing that students don't read text attentively. AVs represent an alternative (when possible) way of presenting content that will increase students' interaction with class materials.
- **Use in the classroom:** Most students recognized the value of OpenDSA exercises and think they would perform better when they are used as mandatory assignments. That result was statistically significant. Also they think that using OpenDSA during lectures will positively affect their performance.
- **Pedagogical impact of OpenDSA exercises:** We found that students with higher exam scores tend to complete more exercises. However, we could not reach

a definitive conclusion when investigating causality relationship between OpenDSA and students exam scores. We investigated if completing a lot of OpenDSA exercises translate to better scores. There seems to be a threshold effect (between OpenDSA exercises and grades) that our clustering strategy did not capture for KAV and PE exercises. Further analysis will be needed to verify our assumption. We found a relationship between KA exercises and student score on written exams. This is likely due to the fact that exams in CS3114-F13 disproportionately cover algorithm analysis topics. Since KAV and PE exercises cover the behavior of the algorithm or the data structure, the only activities allowing students to practice algorithm analysis topics within OpenDSA are KA exercises. It is then understandable that KA exercises have a greater impact student scores.

- **Low use of mobile devices:** The low use of mobile device might be explained by a combination of structural and behavioral factors. Structural factors include device characteristics and some usability issues, especially for small screen size devices. There was a lot of procrastination, thus leading to limited amount of time to complete homework. It is possible that this had an influence on the device selected.

5.6 Conclusion

In this chapter we investigated the impact of OpenDSA on students. The goal of our investigation was to answer the following questions:

1. What is students' experience with OpenDSA?

   Opinion surveys revealed that students have a positive opinion of OpenDSA. Students described described their experience using OpenDSA positively. They also expressed some frustration due to bugs or inconsistencies in OpenDSA. The way OpenDSA was used in the classroom did not affect students' responses. Students found OpenDSA to be one of the most useful resources in the CS2 offering at Virginia Tech. Only programming labs were ranked higher in terms of perceived impact on learning gains. We have evidence that students used AVs when solving exercises within OpenDSA and that they used OpenDSA voluntarily to review for exams

2. How do students prefer to use OpenDSA in the classroom?

   When asked to rate how the use of OpenDSA could affect their ability to learn course materials, students enrolled in the CS2 and CS3 offering at Virginia Tech expressed their preference to scenarios involving using OpenDSA during lectures and for graded assignments. Further analysis of students' responses showed a statistically significant impact of using OpenDSA during lectures or using OpenDSA

exercises as graded assignment on students ability to learn course materials. The effect of using OpenDSA during lectures and use OpenDSA exercises as graded assignment simultaneously was not statistically significant. We also found that students do not use mobile devices to access OpenDSA content

3. Does the way students use OpenDSA correlate to students' learning performance?

   A causality relationship between OpenDSA and students' learning performances could not be established. We found that students with higher scores tend to correctly complete more exercises than those with lower grades. We were unable to explain students' score (at written exams) fluctuations by their use of OpenDSA. However, completion of KA exercises seems to explain students' score fluctuation. Likely due to the fact that the topic covered by the exams (algorithm analysis) could only be practiced through KA exercises. Current effort lead by Mohammed Farghally to create AVs and interactive exercises specifically for algorithm analysis topic will allow us to have a clearer picture on the impact of OpenDSA exercises impact on student exam scores when the exam disproportionately covers such topics (like it was in CS3114-F13).

Chapter 6

Conclusions and Future Work

There is growing interest toward use of interactive online course materials at all levels of education. MOOCs (Massive Open Online Courses) and the popularity of practice systems like Khan Academy and Code Academy are part of that trend. MOOCs and online courses especially drive a need for scalable methods of assessment that do not require direct involvement of limited human resources. Automated assessment of exercises provides an excellent way to scale up assessment when it is possible. For many traditional computer science courses, a long-standing problem is lack of sufficient practice exercises with feedback to the student. Another problem in computer science is the low use of technology for teaching concepts. Algorithms visualizations are an attempt to solve that problem, but despite their perceived educational benefits, the adoption of AVs has been slow. Interactive eTextbooks have the potential to make it easy for instructors to introduce both visualizations and practice exercises into their courses. We built OpenDSA with the goal of addressing roadblocks identified in previous studies. Specifically we addressed the following objectives:

1. **AVs system adoption obstacles**

   Several obstacles to AV system use and adoption have been identified in previous studies. Because it is harder to squeeze a new technology into existing materials, we think instructors will be more willing to adopt complete units of instruction rather than standalone AV systems. As we described in Chapter 4, we designed and implemented OpenDSA with the goal of addressing AV adoption roadblocks. Specifically, we wanted OpenDSA to provide pedagogically useful interactive elements, integration of AVs and exercises, portability and interoperability, customizable materials, instructor collaboration support, and privacy support. Trends from early adoption show that most instructors are satisfied with the system and think that it is beneficial to their students. In addition, OpenDSA can help transition toward a more active classroom. The automated assessments included in OpenDSA allow students to self-teach and evaluate their declarative knowledge of the topic, and make it possible for instructors to focus on higher level concepts. We found that instructors who were already interested in moving to a more active classroom found themselves enabled or at least encouraged by OpenDSA.

2. **Present information in a way that will positively impact students ex-**

periences

Students' perception and affect toward a learning technology are important factors in students' experience and impact students' performance. In Chapter 5, we assessed students' attitude toward OpenDSA and found that students rated their experience positively. Furthermore, students found OpenDSA pedagogically beneficial and used it voluntarily. We found that students used the AVs when studying and potentially might prefer content presented using visualization over prose.

3. **Provide students with more practice activities**

   The limited amount of practice for tasks other than programming is a problem for many CS courses. OpenDSA includes a wide variety of assessment activities. As shown in Chapters 4 and 5, instructors and students found OpenDSA exercises useful. All instructors used OpenDSA assessment in their class and it contributed to students final grade in almost all classes. Students expressed that using OpenDSA in class and being required to do the exercises might increase their chances of succeeding in the course.

Future work

OpenDSA had been used to teach several CS2 and CS3 courses in the past two and a half years, and we can now infer some benefits from using it. We also can now assess what additions should be made to the system. Planned future work in OpenDSA includes the following:

1. **Integration of analytics tools:**

   We are collecting large amounts of data within OpenDSA. So far the data has only been used for research purposes. We think that the data collected might be useful to instructors. We should build or integrate data visualization tools into the instructor's view and use data analytics tools to help instructors interpret the collected data. We can for example add features that will help instructors to monitor students' progress, identify hard topics for students, and help identify struggling students.

2. **Migration towards a more scalable system:**

   There is a need for OpenDSA to accommodate a growing number of users. Since most instructors will rely on projects members to host their OpenDSA instance, we need to develop a more robust system. Also, with the popularity of MOOCS, we should think about offering OpenDSA on a MOOC platform. There is an ongoing effort lead by Hossameldine Shahin to move OpenDSA to the OpenEdX platform [2].

3. **Addition of in-class activities:**

   While many instructors have an interest in adopting "flipped" classrooms or other active learning approaches, for most instructors making major changes in teaching approach is a slow evolution. New technologies can speed up that process by lowering the frictions or barriers to change. To better support instructors we should incorporate in OpenDSA in-class hands-on activities. We should collect such activities from CS educators, evaluate them, and incorporate them into OpenDSA.

4. **Support for course notes/slides:**

   Several instructors expressed the desire to be able to create course notes using OpenDSA(HTML slides + AVs). We should implement a solution that use current OpenDSA architecture (ReStructuredText and Sphinx) to fully integrate the feature with the rest of the OpenDSA system.

5. **Better Support for presenting analytical content:**

   Algorithm analysis topics pose challenges in more advanced DSA courses that follow after CS2. Interactive exercises and visualizations for analytical content are challenging to implement, since there is no dynamic content to be presented and the analysis material is more abstract than procedural algorithm dynamics. We should create interactive content (AVs and exercises) for algorithm analysis topics.

6. **Expanded content:**

   There are ongoing efforts to create OpenDSA content for advanced algorithm topics such as NP-Completeness, programming languages topics, and formal languages topics.

Bibliography

[1] Light-Bot. `http://lightbot.com/`. Accessed: 04/09/2015.

[2] Open edx. http://code.edx.org. Accessed: 04/09/2015.

[3] Monika Akbar, Clifford A. Shaffer, and Edward A. Fox. Deduced social networks for an educational digital library. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '12, pages 43–46, Washington, DC, USA, 2012. ACM.

[4] Kirsti M Ala-Mutka. A Survey of Automated Assessment Approaches for Programming Assignments. *Computer Science Education*, 15(2):83–102, 2005.

[5] A. Alharbi, F. Henskens, and M. Hannaford. Integrated Standard Environment for the Teaching and Learning of Operating Systems Algorithms Using Visualizations. In *Fifth International Multi-Conference on Computing in the Global Information Technology (ICCGI), 2010*, pages 205 –208, Sept. 2010.

[6] Heidi Andrade and Anna Valtcheva. Promoting Learning and Achievement Through Self-Assessment. *Theory Into Practice*, 48(1):12–19, 2009.

[7] Paul Baepler, J.D. Walker, and Michelle Driessen. It's not about seat time: Blending, flipping, and efficiency in active learning classrooms. *Computers & Education*, 78(0):227 – 236, 2014.

[8] Albert Bandura. Self-efficacy: Toward a unifying theory of behavioral change. *Psychological review*, 84(2):191, 1977.

[9] R. G. Baraniuk, C. S. Burrus, B. M. Hendricks, G. L. Henry, A. O. Hero, D. H. Johnson, D. L. Jones, J. Kusuma, R. D. Nowak, J. E. Odegard, L. C. Potter, K. Ramchandran, R. J. Reedstrom, P. Schniter, I. W. Selesnick, D. B. Williams, and W. L. Wilson. Connexions: DSP education for a networked world. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2002*, volume 4, pages IV–4144 –IV–4147, May 2002.

[10] John Carlo Bertot, Charles R. McClure, William E. Moen, and Jeffrey Rubin. Web usage statistics: Measurement Issues and Analytical Techniques. *Government Information Quarterly*, 14(4):373 – 395, 1997.

[11] Dawn Birch and Bruce Burnett. Bringing academics on board: Encouraging institution-wide diffusion of e-learning environments. *Australasian Journal of Educational Technology*, 25(1), 2009.

[12] Jacob Lowell Bishop and Matthew A Verleger. The flipped classroom: A survey of the research. In *Proceedings of ASEE Annual Conference Proceedings*, Atlanta, GA, June 2013.

[13] Benjamin Samuel Bloom. *Taxonomy of educational objectives : the classification of educational goals. Handbook I, Cognitive domain / by a committee of college and university examiners.* Benjamin S. Bloom, Editor. David McKay, New York :, 1956.

[14] Daniel Aubrey Breakiron. Evaluating the Integration of Online, Interactive Tutorials into a Data Structures and Algorithms Course. Master's thesis, Virginia Polytechnic Institute and State University, 2013.

[15] P. Brusilovsky et al. Adaptive and intelligent technologies for web-based eduction. *Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching*, 13(4):19–25, 1999.

[16] Cheryl Bullock and John Ory. Evaluating Instructional Technology Implementation in a Higher Education Environment. *American Journal of Evaluation*, 21(3):315–328, 2000.

[17] C.J. Butz, S. Hua, and R.B. Maguire. A Web-Based Intelligent Tutoring System for Computer Programming. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, 2004. WI 2004.*, pages 159 – 165, sept. 2004.

[18] Michael D. Byrne, Bonnie E. John, Neil S. Wehrle, and David C. Crow. The tangled Web we wove: a taskonomy of WWW use. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '99, pages 544–551, Pittsburgh, Pennsylvania, USA, 1999. ACM.

[19] José C. Castillo, H. Rex Hartson, and Deborah Hix. Remote usability evaluation: Can users report their own critical incidents? In *CHI 98 Cconference Summary on Human Factors in Computing Systems*, CHI '98, pages 253–254, Los Angeles, California, USA, 1998. ACM.

[20] Davide Cervone. Mathjax: a platform for mathematics on the web. *Notices of the AMS*, 59(2):312–316, 2012.

[21] Stephen Cooper, Wanda Dann, and Randy Pausch. Alice: a 3-D tool for introductory programming concepts. In *Proceedings of the fifth annual CCSC northeastern conference on The journal of computing in small colleges*, CCSC '00, pages 107–116, USA, 2000. Consortium for Computing Sciences in Colleges.

[22] A.T. Corbett, K.R. Koedinger, and J.R. Anderson. Intelligent tutoring systems. *Handbook of human computer interaction*, pages 849–874, 1997.

[23] Pierluigi Crescenzi, Giorgio Gambosi, Roberto Grossi, C. Nocentini, and W. Verdese. AlViE [Computer software]. `http://www.algoritmica.org/software/`, 2007. Accessed: 04/09/2015.

[24] Paul Denny, Andrew Luxton-Reilly, Ewan Tempero, and Jacob Hendrickx. CodeWrite: supporting student-driven practice of Java. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, SIGCSE '11, pages 471–476, Dallas, TX, USA, 2011. ACM.

[25] Christopher Douce, David Livingstone, and James Orwell. Automatic test-based assessment of programming: A review. *Journal on Educational Resources in Computing (JERIC)*, 5(3), September 2005.

[26] Stephen W. Draper, Margaret I. Brown, Fiona P. Henderson, and Erica McAteer. Integrative evaluation: an emerging role for classroom studies of CAL. *Computers & Education*, 26(13):17 – 32, 1996. Computer Assisted Learning Selected Contributions from the CAL 95 Symposium.

[27] Dahlstrom E., Walker J. D., and Dziuban C. ECAR Study of Undergraduate Students and Information Technology. Technical report, EDUCAUSE Center for Analysis and Research, 2013.

[28] Stephen H. Edwards. Using software testing to move students from trial-and-error to reflection-in-action. In *Proceedings of the 35th technical symposium on Computer science education*, SIGCSE '04, pages 26–30, Norfolk, Virginia, USA, 2004. ACM.

[29] Darin R. Ellis, Thomas B. Jankowski, Jarrod E. Jasper, and Balaji S. Tharuvai. Listener: A tool for client-side investigation of hypermedia navigation behavior. *Behavior Research Methods, Instruments, & Computers*, 30:573–582, 1998.

[30] Roy T. Fielding and Richard N. Taylor. Principled design of the modern Web architecture. *Transactions on Internet Technology (TOIT)*, 2(2):115–150, May 2002.

[31] Eric Fouh, Monika Akbar, and Clifford A Shaffer. The Role of Visualization in Computer Science Education. *Computers in the Schools*, 29(1-2):95–117, 2012.

[32] D.P. Friedman and M. Felleisen. *Little LISPer*. Scientific Research Associates, 1974.

[33] David Furcy. JHAVEPOP: Visualizing linked-list operations in C++ and Java. *Journal of Computer Science in Colleges*, 25(1):32–41, October 2009.

[34] Philip J. Guo. Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. In *Proceedings of the 44th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '13, pages 579–584, Denver, Colorado, USA, 2013. ACM.

[35] J.S. Gurka and W. Citrin. Testing effectiveness of algorithm animation. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 182–189, Sep 1996.

[36] M. Haverbeke. CodeMirror (Version 2.x). `http://codemirror.net/`, 2011. Accessed: 04/09/2015.

[37] Poul Henriksen and Michael Kölling. Greenfoot: combining object visualization with interaction. In *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*, OOPSLA '04, pages 73–82, Vancouver, BC, CANADA, 2004. ACM.

[38] I-Han Hsiao, Sergey Sosnovsky, and Peter Brusilovsky. Adaptive Navigation Support for Parameterized Questions in Object-Oriented Programming. In Ulrike Cress, Vania Dimitrova, and Marcus Specht, editors, *Learning in the Synergy of Multiple Disciplines*, volume 5794 of *Lecture Notes in Computer Science*, pages 88–98. Springer Berlin Heidelberg, 2009.

[39] Christopher Hundhausen, Anukrati Agrawal, Dana Fairbrother, and Michael Trevisan. Does Studio-based Instruction Work in CS 1?: An Empirical Comparison with a Traditional Approach. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, pages 500–504, Milwaukee, Wisconsin, USA, 2010. ACM.

[40] Christopher D. Hundhausen, Sarah A. Douglas, and John T. Stasko. A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages & Computing*, 13(3):259 – 290, 2002.

[41] Ville Karavirta. Seamless Merging of Hypertext and Algorithm Animation. *Transactions on Computing Education (TOCE)*, 9(2):10:1–10:18, June 2009.

[42] Ville Karavirta and Petri Ihantola. Initial Set of Services for Algorithm Visualization . In *Proceedings of the Sixth Program Visualization Workshop*, pages 67–71, Darmstadt, Germany, 2011.

[43] Ville Karavirta, Petri Ihantola, and Teemu Koskinen. Service-oriented approach to improve interoperability of e-learning systems. In *13th IEEE International Conference on Advanced Learning Technologies*, 2013.

[44] Ville Karavirta and Clifford A Shaffer. Jsav: the javascript algorithm visualization library. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 159–164. ACM, 2013.

[45] Colleen Kehoe, John Stasko, and Ashley Taylor. Rethinking the evaluation of algorithm animations as learning aids: an observational study. *International Journal of Human-Computer Studies*, 54(2):265 – 284, 2001.

[46] E. Kıcıman and B. Livshits. AjaxScope: A platform for remotely monitoring the client-side behavior of Web 2.0 applications. In *Proceedings of the 21st Symposium on Operating Systems Principles*. ACM, 2007.

[47] Maria Knobelsdorf, Essi Isohanni, and Josh Tenenberg. The reasons might be different: Why students and teachers do not use visualization tools. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, Koli Calling '12, pages 1–10, Koli, Finland, 2012. ACM.

[48] David R Krathwohl. A revision of Bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218, 2002.

[49] D. Kristol and L. Montulli. HTTP State Management Mechanism. RFC 2109, RFC Editor, February 1997.

[50] Amruth Kumar. Dynamically generating problems on static scope. In *Proceedings of the 5th annual conference on Innovation and Technology in Computer Science Education*, ITiCSE '00, pages 9–12, Helsinki, Finland, 2000. ACM.

[51] Mikko-Jussi Laakso, Niko Myller, and Ari Korhonen. Comparing Learning Performance of Students Using Algorithm Visualizations Collaboratively on Different Engagement Levels. *Educational Technology & Society*, 12(2):267–282, 2009.

[52] Mikko-Jussi Laakso, Tapio Salakoski, Linda Grandell, Xuemei Qiu, Ari Korhonen, and Lauri Malmi. Multi-perspective study of novice learners adopting the visual algorithm simulation exercise system trakla2. *Informatics in Education*, 4(1):49–68, May 2005.

[53] Ronit Ben-Bassat Levy and Mordechai Ben-Ari. We work so hard and they don't use it: Acceptance of software tools by teachers. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, ITiCSE '07, pages 246–250, Dundee, Scotland, 2007. ACM.

[54] Yvonna S. Lincoln and Egon G. Guba. *Naturalistic inquiry*, volume 75. Sage, 1985.

[55] L. Malmi, V. Karavirta, A. Korhonen, J. Nikander, O. Seppälä, and P. Silvasti. Visual Algorithm Simulation Exercise System with Automatic Assessment: TRAKLA2. *Informatics in Education*, 3:267–288, 2004.

[56] James H. McMillan and Jessica Hearn. Student self-assessment: The key to stronger student motivation and higher achievement. *Educational Horizons*, 87(1):40–49, 2008.

[57] Bradley N. Miller and David L. Ranum. Beyond PDF and ePub: toward an interactive textbook. In *Proceedings of the 17th ACM annual conference on Innovation and Technology in Computer Science Education*, ITiCSE '12, pages 150–155, Haifa, Israel, 2012. ACM.

[58] Niko Myller, Roman Bednarik, Erkki Sutinen, and Mordechai Ben-Ari. Extending the engagement taxonomy: Software visualization and collaborative learning. *Transactions on Computing Education (TOCE)*, 9(1):7:1–7:27, March 2009.

[59] T.R. Naidu, M. Verma, V. Choppella, and G. Chalapakay. Synthesizing customizable learning environments. In *International Conference on Technology for Education (T4E), 2010*, pages 122 –129, july 2010.

[60] T Naps, G Rössling, J Anderson, S Cooper, W Dann, R Fleischer, B Koldehofe, A Korhonen, M Kuittinen, L Malmi, et al. Evaluating the educational impact of algorithm visualization. *ACM SIGCSE Bulletin*, 2003.

[61] Thomas L. Naps, Guido Rössling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, and J. Ángel Velázquez-Iturbide. Exploring the role of visualization and engagement in computer science education. In *Working group reports from ITiCSE on Innovation and technology in computer science education*, ITiCSE-WGR '02, pages 131–152, Aarhus, Denmark, 2002. ACM.

[62] Nick Parlante. CodingBat. `http://codingbat.com/`, 2006. Accessed: 04/09/2015.

[63] Michael Prince. Does Active Learning Work? A Review of the Research. *Journal of Engineering Education*, 93(3):223–231, 2004.

[64] David Pritchard and Troy Vasiga. CS Circles: An In-Browser Python Course for Beginners. In *Proceedings of the 44th Technical Symposium on Computer Science Education*, SIGCSE '13, pages 591–596, Denver, Colorado, USA, 2013. ACM.

[65] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

[66] Stuart Reges and Marty Stepp. *Building Java Programs: A Back to Basics Approach, 2nd edition*. Addison Wesley, 2010.

[67] Jeremy Roschelle. Keynote paper: Unlocking the learning value of wireless mobile devices. *Journal of Computer Assisted Learning*, 19(3):260–272, 2003.

[68] Rockford J. Ross. Hypertextbooks and a Hypertextbook Authoring Environment. In *Proceedings of the 13th annual conference on Innovation and Technology In Computer Science Education*, ITiCSE '08, pages 133–137, Madrid, Spain, 2008. ACM.

[69] Jonathan P Rossing, Willie M Miller, Amanda K Cecil, and Suzan E Stamper. ilearning: The future of higher education? student perceptions on learning with mobile tablets. *Journal of the Scholarship of Teaching and Learning*, 12(2):1–26, 2012.

[70] Guido Rößling and Teena Vellaramkalayil. A Visualization-Based Computer Science Hypertextbook Prototype. *Transactions on Computing Education (TOCE)*, 9(2):11:1–11:13, June 2009.

[71] Clifford A. Shaffer, Monika Akbar, Alexander Joel D. Alon, Michael Stewart, and Stephen H. Edwards. Getting algorithm visualizations into the classroom. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, SIGCSE '11, pages 129–134, Dallas, TX, USA, 2011. ACM.

[72] Clifford A. Shaffer, Matthew Cooper, and Stephen H. Edwards. Algorithm Visualization: A Report on the State of the Field. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, SIGCSE '07, pages 150–154, Covington, Kentucky, USA, 2007. ACM.

[73] 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*, Koli Calling '11, pages 112–117, Koli, Finland, 2011. ACM.

[74] Clifford A. Shaffer, Thomas L. Naps, and Eric Fouh. Truly interactive textbooks for computer science education. In *Proceedings of the 6th Program Visualization Workshop*, PVW '11, pages 97–103, 2011.

[75] John Stasko, Albert Badre, and Clayton Lewis. Do algorithm animations assist learning?: An empirical study and analysis. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 61–66, Amsterdam, The Netherlands, 1993. ACM.

[76] Marty Stepp and Jessica Miller. Practice-It! `http://practiceit.cs.washington.edu/`, 2011. Accessed: 04/09/2015.

[77] Dianne M Tice and Roy F Baumeister. Longitudinal study of procrastination, performance, stress, and health: The costs and benefits of dawdling. *Psychological science*, pages 454–458, 1997.

[78] Nathaniel Titterton, Colleen M. Lewis, and Michael J. Clancy. Experiences with lab-centric instruction. *Computer Science Education*, 20(2):79–102, 2010.

[79] Nghi Truong, Peter Bancroft, and Paul Roe. A web based environment for learning to program. In *Proceedings of the 26th Australasian computer science conference - Volume 16*, ACSC '03, pages 255–264, Darlinghurst, Australia, 2003. Australian Computer Society, Inc.

[80] Nghi Truong, Peter Bancroft, and Paul Roe. Learning to program through the web. In *Proceedings of the 10th annual conference on Innovation and technology in computer science education*, ITiCSE '05, pages 9–13, Caparica, Portugal, 2005. ACM.

[81] Jaime Urquiza-Fuentes and J. Ángel Velázquez-Iturbide. A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems. *Transactions on Computing Education (TOCE)*, 9:9:1–9:21, June 2009.

[82] Tim Weston. Formative Evaluation for Implementation: Evaluating Educational Technology Applications and Lessons. *The American Journal of Evaluation*, 25(1):51 – 64, 2004.

[83] Chauncey Wilson and Kara Pernice Coyne. The whiteboard: Tracking usability issues: to bug or not to bug? *interactions*, 8(3):15–19, 2001.

[84] Wen-Hsiung Wu, Yen-Chun Jim Wu, Chun-Yu Chen, Hao-Yun Kao, Che-Hung Lin, and Sih-Han Huang. Review of trends from mobile learning studies: A meta-analysis. *Computers & Education*, 59(2):817–827, 2012.

[85] Robert K Yin. The case study crisis: Some answers. *Administrative science quarterly*, pages 58–65, 1981.

[86] Daniel Zingaro. Peer instruction contributes to self-efficacy in cs1. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 373–378, Atlanta, Georgia, USA, 2014. ACM.

Appendix A

Instructors Questionnaires

A.1 Fall 2012 and Spring 2013

1. How did you use OpenDSA with your class?
   (a) Did students use OpenDSA in classroom?
   (b) Did you use OpenDSA as a lecture aid? How much?
2. How did you use OpenDSA assessment activities?
   (a) Was a grade associated with doing OpenDSA exercises? If so, what fraction of course grade?
3. Did you change the way you conducted your class because of OpenDSA?
   (a) If yes how? [Homework, lecture]
   (b) Do you think OpenDSA conflicted with your pedagogy style or was consistent with it?
   (c) Do you think OpenDSA is useful for in class teaching?
   (d) what other factors influenced your decision?
4. What OpenDSA modules did you use?

5. Do you think your students enjoyed working with OpenDSA? What feedback did they give you?

6. Do you think OpenDSA was useful to your students? If so, how?

7. If OpenDSA had more modules (relevant to your course), do you think that you would use them?

8. What items/features do you want improved/changed in OpenDSA?

A.2 Spring 2014

Default Question Block

1. How did you use OpenDSA with your class?

2. How did you use OpenDSA assessment activities?
[Mandatory or not? Did the students get credit from doing OpenDSA exercises? What was the contribution of OpenDSA exercises in student final grade? One due date or several due dates?]

3. Did you changed the way you conducted your class because of OpenDSA? Please explain why.

4. Did you use all OpenDSA content available that was relevant to your course? Why if you did not.

5. Do you think OpenDSA was useful to your students?

6. How do you rate OpenDSA instructors' view?
[usability, reliability, interoperability with your institution LMS]

7. Please describe your overall opinion of your experience with the OpenDSA etextbook materials and exercises. And give us your suggestions on how OpenDSA can be improved.

8. Are you planning on using OpenDSA in the future?

Appendix B

Students Questionnaires

B.1 Pre-treatment survey: CS3114-SP13-A, CS223

Data Structures and Algorithms Survey
Opinions on Electronic Textbooks

We are conducting a study of your experiences and opinions regarding the potential use of electronic textbooks and online materials in a data structures and algorithms course. The principal investigators for this study are: Dr. Cliff Shaffer and Dr. T.S. Hall. There is no risk for you to participate in this study. The survey is conducted anonymously, and all data reporting will be in aggregate form.

You may ask questions about this research by contacting Dr. Shaffer at **(540) 231-4354** or **shaffer@cs.vt.edu**.

For questions about your rights as a research participant, contact Virginia Tech's Office of Research Administration at **(540) 231-6866**.

Your participation is voluntary. If you do not wish to participate, simply do not fill out the survey. You must be 18 or older to take part in this research.

Thank you for your participation.

I. Prior to this semester, have you ever taken a class where the textbook or most of the course content was online?

a) Yes
b) No

If <u>yes</u>, please answer the following questions about your favorite class that contained online materials or activities:

1- The course was (check one)

a) Entirely online: online instruction only, no class meetings
b) Partly online: combination of online and in-class instruction

2- The course materials included (check all that apply)

a) Static material: like PDF files, HTML pages, or something like you would get on an eBook reader
b) Static material with multimedia: like recorded videos and images
c) Interactive activities: like navigation buttons, drag and drop utilities, etc.
d) Assessment exercises with answer feedback
e) Tools to develop and post your own content

3- You interacted with the course instructor via (check all that apply)

a) Forum
b) Weblog
c) E-mail
d) Listserv
e) Chat
f) Video conferencing
g) Teleconferencing
h) Face-to-face meetings
i) Other _____

4- You interacted with your classmates via (check all that apply)

a) Forum
b) Weblog
c) E-mail
d) Listserv
e) Chat
f) Video conferencing
g) Teleconferencing
h) Face-to-face meetings
i) Other _____

5- Compared to a typical in-person course, the online course was (check one)

a) Much easier
b) A little bit easier
c) About the same level
d) A little more difficult
e) Much more difficult

II. Prior to this semester, have you ever taken a class where there was a significant amount of questions or exercises that were graded automatically?

a) Yes
b) No

If <u>yes</u>, please answer the following questions about the online class where automatically graded questions played the most important role:

1- The automatically graded materials were (check all that apply)

a) Online tests and quizzes
b) Programming assignments
c) Other (please specify) _____
_____

2- How did your course grade compare to your usual grades for in-person courses?

a) Much higher
b) A little higher
c) About the same
d) A little lower
e) Much lower

3- Tell us your opinion about this type of course: did you like the experience?

_____
_____
_____
_____

4- The course textbook "Data Structures and Algorithms" by Shaffer is available for free as a PDF online. Alternatively, a paper copy could be printed or purchased in bound form. Did you get a paper copy of the textbook?

a) Yes, and when I read the book, I mostly used the paper copy.
b) Yes, but I mostly used the electronic copy of the book.
c) No, I only used the electronic copy of the book.
d) No, I did not use this textbook.

III. The following questions ask you about the concept of an online textbook for your data structures and algorithms course. This textbook would be of a particular type, far different from just a PDF of a paper book. This form of electronic textbook would include algorithm visualizations, questions and exercises that you would work through, and immediate feedback to help you determine if you understand the material.

1- Would you like to take a course built around such an electronic textbook?

a) 1 No, not at all
b) 2
c) 3
d) 4
e) 5 Yes, very much

Please explain why or why not?

_____
_____
_____

2- What is your preference between a traditional "lecture based" course, versus a course that is "lab based" in the sense of spending much of the time working through an online textbook to learn the content?

a) I strongly prefer a traditional “lecture based” course
b) I slightly prefer a traditional “lecture based” course
c) I am neutral
d) I slightly prefer a lab-based course
e) I strongly prefer a lab-based course

3- If you were taking a class focused on an online textbook with visualizations and exercises as described above, would you prefer to do all of your work at home (perhaps with weekly deadlines), or would you prefer to take the course as a "lab course" where you come to class on a regular schedule, the instructor is available to interact with you, but the main focus is still to work through the online material?

a) I definitely would prefer doing it at home
b) I slightly would prefer doing it at home
c) I am neutral
d) I slightly would prefer the lab-based class
e) I definitely would prefer the lab-based class

4- Compared to a traditional lecture-based data structures and algorithms course, what do you think your grade would be if you took a lab-based course that relies on online content as described above?

a) A much worse grade for the lab-based course
b) A slightly worse grade for the lab-based course
c) About the same grade for either course
d) A slightly better grade for the lab-based course
e) A much better grade for the lab-based course

IV. Please answer the following questions about your Learning Preferences:

1- If I don't fully understand the material presented in class,

a) I am turned off, I spend minimal time working on the course material
b) I am slightly discouraged, and spend a little less time working on course material
c) I am not affected, and spend the same amount of time working on course material
d) I am slightly challenged, and spend a little more time working on course material
e) I am highly challenged, and spend more time reading and working on course material

2- In a typical 3-credit hour class, over the course of a semester, please estimate the fraction of your time spent on each of the following.

a) Reading the textbook or other course material _____ % of time
b) Working on examples and practice problems _____ % of time
c) Working on homework assignment _____ % of time
d) Working on class projects _____ % of time
e) Other, please specify_____ _____ % of time

3- What type of assessment do you think is better for you?

a) Small incremental assessments (e.g. short quizzes at least once a week)
b) A few major tests (e.g. two midterms and a final)

4- Please rate your confidence about working through an online textbook with visualizations, interactive exercises, and questions with feedback (on your own).

a) No confidence
b) Little confidence
c) Some confidence
d) A fair amount of confidence
e) A great deal of confidence

Thank You

B.2 Post-treatment survey: CS3114-F12, CS3114-SP13-A, CS223

Data Structures and Algorithms Post-treatment Survey
Opinions on Your Experience with OpenDSA

We are conducting a study of your experiences and opinions regarding use of the OpenDSA interactive tutorials. The principal investigators for this study are: Dr. Cliff Shaffer and Dr. T. S. Hall. There is no risk for you to participate in this study. The survey is conducted anonymously, and all data reporting will be in aggregate form.

You may ask questions about this research by contacting Dr. Shaffer at **(540) 231-4354** or **shaffer@cs.vt.edu**.

For questions about your rights as a research participant, contact Virginia Tech's Office of Research Administration at **(540) 231-6866**.

Your participation is voluntary. If you do not wish to participate, simply do not fill out the survey. You must be 18 or older to take part in this research.

Thank you for your participation.

1. Over the past few weeks, you have used the OpenDSA materials. Please estimate the total number of hours that you have spent during this time for your class. Please include:
   - hours spent in class,
   - time spent working the OpenDSA tutorials and exercises,
   - time spent looking at other material, and
   - time spent preparing for and taking any associated test on the material.

   Please DO NOT include time spent working on any programming projects for this course.

2. Did you spend any significant amount of time looking at related materials other than the OpenDSA tutorials? If so, how much time? What did you look at?

3. How much time did you spend studying the topics covered by OpenDSA beyond the time that you spent going through the OpenDSA materials in order to complete the exercises? That is, how much time did you spend specifically studying for your exam, as opposed to completing the exercises? How does this compare to the time that you would have spent studying for the exam if this content were taught with lecture and textbook?

4. Please describe your overall opinion of your experience with the OpenDSA online textbook materials and exercises.

5. Considering your experience with the OpenDSA materials as compared to the alternative of standard lecture and textbook, please rate your preference for how the material should be presented.
   A. I definitely preferred using OpenDSA to lecture and textbook
   B. I somewhat preferred OpenDSA to lecture and textbook
   C. I am neutral, I think either would be about the same
   D. I somewhat would have preferred doing these topics with lecture and textbook
   E. I definitely would have preferred doing these topics with lecture and textbook

6. How do you think that your learning gains using OpenDSA compare with what you would have learned using regular lecture and textbook?
   A. I believe that I learned a lot more with OpenDSA
   B. I believe that I learned somewhat more with OpenDSA
   C. I believe that I would have learned about the same amount either way
   D. I believe that I would have learned somewhat more with lecture and textbook.
   E. I believe that I would have learned a lot more with lecture and textbook

7. Please explain why you gave your answer for Question (6)

8. Now consider the amount of time that you spent using OpenDSA, and compare that with how much time you think that you would have spent in class, reading the textbook, doing a homework set, and studying for the test. How do you think that the amount of time you spent using OpenDSA compares to the time you would have spent if the class had been conducted using lecture and textbook, with a regular, paper homework set?

   - I think I spent a lot more time using OpenDSA than I would have with lecture, textbook, and paper homework.
   - I think I spent somewhat more time using OpenDSA than I would have with lecture, textbook, and paper homework.
   - I think I spent about the same amount of time as I would have with lecture, textbook, and homework.
   - I think that I spent somewhat less time because we used OpenDSA.
   - I think that I spent much less time because we used OpenDSA.

9. Do you think that the amount of time that you spent using OpenDSA was reasonable for the amount of material covered and the amount of learning achieved?

10. When you are working through an OpenDSA module, do you read the text material and slideshows in the order they appear, and then do the exercises? Or do you skip directly to the exercises, and read the text as necessary to get the exercises done?

11. Did you encounter any significant bugs in the OpenDSA system that kept you from using it as intended for a significant period of time?

12. OpenDSA uses text and visualizations to explain the algorithms. The algorithms themselves are often explained using slideshows that mix explanatory text with the visualization for the algorithm. An alternate design to the slideshows would be a video with audio narration replacing the current text, synchronized with a screen display of the algorithm visualization. Comparing text-based slideshows as currently implemented in OpenDSA and an audio-narrated visualization as described replacing the slideshows, which do you think you would prefer?

    A. I definitely would prefer the existing text-based slideshows to audio narration
    B. I somewhat would prefer the existing text-based slideshows to audio narration
    C. I am neutral, I think either would be about the same
    D. I somewhat would prefer audio narration to the existing text-based slideshows
    E. I definitely would prefer audio narration to the existing text-based slideshows

13. There are many ways that OpenDSA materials might be used in class. Two extremes might be (i) to spend class time working the interactive OpenDSA materials in a "lab" setting with the teacher available to answer individual questions, or (i) to have the teacher give lectures on the materials, but tied directly to using OpenDSA as the "textbook".

    A. I prefer using class time to work on the OpenDSA materials.
    B. I prefer using class time to get an overview or lecture from the teacher about the topic, and working the OpenDSA exercises at home.
    C. I have no preference.

14. Given OpenDSA materials available for the full semester's content, what fraction of class time used to cover content should be given to working the tutorials in class (with the remaining time spent by the instructor lecturing)? For example, 0% doing modules in class vs. 100% lecturing, 50% doing module vs. 50% lecturing, etc.

Now, assume a class design where most days the instructor lectures on the course content, covering one or two OpenDSA modules. Associated with each class day is a "mini-homework assignment" to do the modules covered in that class. Each such assignment might be worth 5-10 points (collectively for about 10-20% of the total course grade). The idea is that this forces you to keep current with the material.

15. Would you like this approach to structuring the class?

    A. Yes, I very much like this idea
    B. Yes, I somewhat like this idea
    C. I am neutral
    D. No, I am somewhat against this idea
    E. No, I am very much against this idea

16. If the class were structured this way, would you prefer that the modules must be done prior to class (to prepare you for the lecture), or would you prefer that they must be done within a day or two after class (to consolidate the lecture and keep you on track)?

    A. Very much prefer prior to class
    B. Slightly prefer prior to class
    C. Neutral on these alternatives
    D. Slightly prefer after class
    E. Very much prefer after class

17. This class included these elements: (1) lectures, (2) OpenDSA, (4) textbook, (5) written homework, (7) tests, (8) coursenotes, and (9) interacting with instructor. Please rank these in order from the one that gave you the **most** learning gains to the one that gave you the **least**.

18. Now, please give us your suggestions on how OpenDSA should best be used, such as having class with lots of time to work the modules, versus having a class where the content is overviewed and mostly you work the exercises out of class. Or maybe you have other ideas on how OpenDSA should be used.

19. Do you have any other suggestions or comments about OpenDSA or your experience?

Here are some questions related to your basic communications infrastructure as it relates to your classes.

20. Do you have some email account that you read at least once per day?

21. Does your official university email account get forwarded to your most-frequently-monitored email account? That is, do you see your university email pretty much every day?

Thank You

B.3 Post-treatment survey: CS3114-SP13-B

Data Structures and Algorithms Post-treatment Survey
Opinions on Your Experience with OpenDSA

We are conducting a study of your experiences and opinions regarding use of the OpenDSA interactive tutorials. The principal investigators for this study are: Dr. Cliff Shaffer and Dr. T. S. Hall. There is no risk for you to participate in this study. The survey is conducted anonymously, and all data reporting will be in aggregate form.

You may ask questions about this research by contacting Dr. Shaffer at **(540) 231-4354** or **shaffer@cs.vt.edu**.

For questions about your rights as a research participant, contact Virginia Tech's Office of Research Administration at **(540) 231-6866**.

Your participation is voluntary. If you do not wish to participate, simply do not fill out the survey. You must be 18 or older to take part in this research.

Thank you for your participation.

1. Please describe your overall opinion of your experience with the OpenDSA online textbook materials and exercises.

2. Considering your experience with the OpenDSA materials as compared to the alternative of standard lecture and textbook, please rate your preference for how the material should be presented.
   A. I definitely preferred using OpenDSA to lecture and textbook
   B. I somewhat preferred OpenDSA to lecture and textbook
   C. I am neutral, I think either would be about the same
   D. I somewhat would have preferred doing these topics with lecture and textbook
   E. I definitely would have preferred doing these topics with lecture and textbook

3. How do you think that your learning gains using OpenDSA compare with what you would have learned using regular lecture and textbook?
   A. I believe that I learned a lot more with OpenDSA
   B. I believe that I learned somewhat more with OpenDSA
   C. I believe that I would have learned about the same amount either way
   D. I believe that I would have learned somewhat more with lecture and textbook.
   E. I believe that I would have learned a lot more with lecture and textbook

4. Please explain why you gave your answer for Question (3)

5. Would you have liked to have completed the OpenDSA tutorials on sorting prior to your midterm on sorting? Do you think that doing the tutorials before your midterm would have affected your midterm grade?

6. Now consider the amount of time that you spent using OpenDSA, and compare that with how much time you think that you would have spent in class, reading the textbook, doing a homework set, and studying for the test. How do you think that the amount of time you spent using OpenDSA compares to the time you would have spent if the class had been conducted using lecture and textbook, with a regular, paper homework set?

    - I think I spent a lot more time using OpenDSA than I would have with lecture, textbook, and paper homework.
    - I think I spent somewhat more time using OpenDSA than I would have with lecture, textbook, and paper homework.
    - I think I spent about the same amount of time as I would have with lecture, textbook, and homework.
    - I think that I spent somewhat less time because we used OpenDSA.
    - I think that I spent much less time because we used OpenDSA.

7. Do you think that the amount of time that you spent using OpenDSA was reasonable for the amount of material covered and the amount of learning achieved?

8. When you are working through an OpenDSA module, do you read the text material and slideshows in the order they appear, and then do the exercises? Or do you skip directly to the exercises, and read the text as necessary to get the exercises done?

9. Did you encounter any significant bugs in the OpenDSA system that kept you from using it as intended for a significant period of time?

10. OpenDSA uses text and visualizations to explain the algorithms. The algorithms themselves are often explained using slideshows that mix explanatory text with the visualization for the algorithm. An alternate design to the slideshows would be a video with audio narration replacing the current text, synchronized with a screen display of the algorithm visualization. Comparing text-based slideshows as currently implemented in OpenDSA and an audio-narrated visualization as described replacing the slideshows, which do you think you would prefer?

    A. I definitely would prefer the existing text-based slideshows to audio narration
    B. I somewhat would prefer the existing text-based slideshows to audio narration
    C. I am neutral, I think either would be about the same
    D. I somewhat would prefer audio narration to the existing text-based slideshows
    E. I definitely would prefer audio narration to the existing text-based slideshows

11. There are many ways that OpenDSA materials might be used in class. Two extremes might be (i) to spend class time working the interactive OpenDSA materials in a “lab” setting with the teacher available to answer individual questions, or (i) to have the teacher give lectures on the materials, but tied directly to using OpenDSA as the “textbook”.

    A. I prefer using class time to work on the OpenDSA materials.
    B. I prefer using class time to get an overview or lecture from the teacher about the topic, and working the OpenDSA exercises at home.
    C. I have no preference.

12. Given OpenDSA materials available for the full semester's content, what fraction of class time used to cover content should be given to working the tutorials in class (with the remaining time spent by the instructor lecturing)? For example, 0% doing modules in class vs. 100% lecturing, 50% doing module vs. 50% lecturing, etc.

Now, assume a class design where most days the instructor lectures on the course content, covering one or two OpenDSA modules. Associated with each class day is a "mini-homework assignment" to do the modules covered in that class. Each such assignment might be worth 5-10 points (collectively for about 10-20% of the total course grade). The idea is that this forces you to keep current with the material.

13. Would you like this approach to structuring the class?

    A. Yes, I very much like this idea
    B. Yes, I somewhat like this idea
    C. I am neutral
    D. No, I am somewhat against this idea
    E. No, I am very much against this idea

14. If the class were structured this way, would you prefer that the modules must be done prior to class (to prepare you for the lecture), or would you prefer that they must be done within a day or two after class (to consolidate the lecture and keep you on track)?

    A. Very much prefer prior to class
    B. Slightly prefer prior to class
    C. Neutral on these alternatives
    D. Slightly prefer after class
    E. Very much prefer after class

15. This class included these elements: (1) lectures, (2) OpenDSA, (3) textbook, (4) written homework, (5) tests, (6) coursenotes, and (7) interacting with instructor. Please rank these in order from the one that gave you the **most** learning gains to the one that gave you the **least**.

16. Now, please give us your suggestions on how OpenDSA should best be used, such as having class with lots of time to work the modules, versus having a class where the content is overviewed and mostly you work the exercises out of class. Or maybe you have other ideas on how OpenDSA should be used.

17. Do you have any other suggestions or comments about OpenDSA or your experience?

Here are some questions related to your basic communications infrastructure as it relates to your classes.

18. Do you have some email account that you read at least once per day?

19. Does your official university email account get forwarded to your most-frequently-monitored email account? That is, would you see your official university email pretty much whenever you check your email? Or do you only see it when you explicitly go to check it?

Thank You

B.4 Post-treatment survey: CS2114-SP14

Software Design and Data Structures Post-treatment Survey

Opinions on Your Experience with OpenDSA

We are conducting a study of your experiences and opinions regarding use of the OpenDSA interactive tutorials. The principal investigator for this study is Dr. Cliff Shaffer. There is no risk for you to participate in this study. The survey is conducted anonymously, and all data reporting will be in aggregate form.

You may ask questions about this research by contacting Dr. Shaffer at (540) 231-4354 or shaffer@cs.vt.edu.

For questions about your rights as a research participant, contact Virginia Tech's Office of Research Administration at (540) 231-6866.

Your participation is voluntary. If you do not wish to participate, simply do not fill out the survey. You must be 18 or older to take part in this research.

Thank you for your participation.

Recall that you used online textbook materials and exercises on the topics of lists, stacks, queues, binary trees and sorting.

To help us improve OpenDSA for future students, please answer the following questions

1. Please describe your overall opinion of your experience with the OpenDSA online textbook materials and exercises.

2. Considering your experience with the OpenDSA materials as compared to the alternative of standard textbook with paper homework, please rate your preference for how the material should be presented.

A. I definitely preferred using OpenDSA to lecture and textbook

B. I somewhat preferred OpenDSA to lecture and textbook

C. I am neutral, I think either would be about the same

D. I somewhat would have preferred doing these topics with textbook and paper homework

E. I definitely would have preferred doing these topics with textbook and paper homework

3. How do you think that your learning gains using OpenDSA compare with regular textbook reading assignments and reading quizzes in Moodle?

A. I believe that I learned a lot more with OpenDSA

B. I believe that I learned somewhat more with OpenDSA

C. I believe that I learned about the same amount either way

D. I believe that I learned somewhat more with textbook and Moodle quizzes

E. I believe that I learned a lot more with textbook and Moodle quizzes

4. Please explain why you gave your answer for Question (3)

5. When you are working through an OpenDSA module, do you read the text material and slideshows in the order they appear, and then do the exercises? Or do you skip directly to the exercises, and read the text as necessary to get the exercises done?

6. Did you encounter any significant bugs in the OpenDSA system that kept you from using it as intended for a significant period of time?

7. There are many ways that OpenDSA materials might be used in class. Two extremes might be (i) to spend class time working the interactive OpenDSA materials in a "lab" setting with the teacher available to answer individual questions, or (ii) to have the teacher give lectures on the materials, but tied directly to using OpenDSA as the "textbook".

A. I would prefer using class time to work on the OpenDSA materials.

B. I would prefer using class time to get an overview or lecture from the teacher about the topic, and working the OpenDSA exercises at home.

C. I have no preference.

8. Now, please give us your suggestions on how OpenDSA should best be used, such as having class with lots of time to work the modules, versus having a class where the content is overviewed and mostly you work the exercises out of class. Or maybe you have other ideas on how OpenDSA should be used.

9. This class included these elements: (1) lectures, (2) OpenDSA exercises, (3) textbook, (4) programming labs, (5) reading quizzes, and (6) interacting with the instructor.

Please rank these in order from the one that gave you the most learning gains to the one that gave you the least.

The following questions ask you about your confidence in your ability to learn the CS2114 course materials under various conditions. Answer each question on a scale of 1 to 5, where 1 means that you are not at all confident and 5 means that you are very confident.

10. I could succeed in learning the CS2114 course material if the class was only taught using a standard textbook with Moodle quizzes.

11. I could succeed in learning the CS2114 course material using OpenDSA on my own if I was required to do the OpenDSA exercises for a grade.

12. I could succeed in learning the CS2114 course material using OpenDSA on my own with no graded exercises.

13. I could succeed in learning the CS2114 course material using OpenDSA if the instructor goes over OpenDSA's content during lecture but I am NOT required to do the exercises for a grade.

14. I could succeed in learning the CS2114 course material using OpenDSA if the instructor goes over OpenDSA's content during lecture and I AM required to do the OpenDSA exercises for a grade.

15. Not counting time spent in class or lab, how many hours have you spent this semester on the topic of recursion? Include time that you spent reading the textbook, course notes, or online materials, and time spent working on homework problems involving recursion.

16. How many hours did you spent on solving the Coding Bat exercises on recursion?

17. On a scale of 1-5, rate your confidence level about your mastery of recursion. (1 being least confident to 5 being most confident)

The following questions ask you about your use of mobile devices for studying and interacting with OpenDSA content.

18. What type of electronic devices do you possess? Check All That Apply.

A. Desktop [Computer]

A. Laptop [Computer] (no touch screen)

B. Touch screen laptop [Laptop computer with touch screen]

C. Smartphone

D. Small tablet (approx 7")

E. Full-size tablet (approx 10")

Call the following items mobile devices: Smartphones, small tablet, and full-size tablet.

19. Do you use your mobile device(s) to study for any course?

A. Yes

B. No

20. Did you use your mobile device(s) to access (read) OpenDSA content?

A. Yes

B. No

21. Did you use your mobile device(s) to do OpenDSA exercises?

A. Yes

B. No

22. If you answered Yes to either Question (22) or (23), please describe your motivation and experience accessing OpenDSA from your mobile device(s).

23. If you answered No to either Question (22) or (23), and you own a mobile device, please explain why you did not access OpenDSA from your mobile device.

24. Break down (by listing percentages) your use of electronics devices for studying CS2114 course materials. The total should be 100.

A. Desktop B. Laptop (no touch screen) C. Touch screen laptop

C. Smartphone D. Small tablet (approx 7")

E. Full-size tablet (approx 10")

Appendix C

IRB approval letters

VirginiaTech

Office of Research Compliance
Institutational Review Board
North End Center, Suite 4120, Virginia Tech
300 Turner Street NW
Blacksburg, Virginia 24061
540/231-4606 Fax 540/231-0959
email irb@vt.edu
website http://www.irb.vt.edu

MEMORANDUM

DATE: December 4, 2013

TO: Cliff Shaffer, Tahereh Hall

FROM: Virginia Tech Institutional Review Board (FWA00000572, expires April 25, 2018)

PROTOCOL TITLE: Interactive e-TextBook

IRB NUMBER: **11-953**

Effective December 4, 2013, the Virginia Tech Institution Review Board (IRB) Chair, David M Moore, approved the Amendment request for the above-mentioned research protocol.

This approval provides permission to begin the human subject activities outlined in the IRB-approved protocol and supporting documents.

Plans to deviate from the approved protocol and/or supporting documents must be submitted to the IRB as an amendment request and approved by the IRB prior to the implementation of any changes, regardless of how minor, except where necessary to eliminate apparent immediate hazards to the subjects. Report within 5 business days to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.

All investigators (listed above) are required to comply with the researcher requirements outlined at:

http://www.irb.vt.edu/pages/responsibilities.htm

(Please review responsibilities before the commencement of your research.)

PROTOCOL INFORMATION:

Approved As:	**Exempt, under 45 CFR 46.110 category(ies) 2**
Protocol Approval Date:	**November 8, 2011**
Protocol Expiration Date:	**N/A**
Continuing Review Due Date*:	**N/A**

*Date a Continuing Review application is due to the IRB office if human subject activities covered under this protocol, including data analysis, are to continue beyond the Protocol Expiration Date.

FEDERALLY FUNDED RESEARCH REQUIREMENTS:

Per federal regulations, 45 CFR 46.103(f), the IRB is required to compare all federally funded grant proposals/work statements to the IRB protocol(s) which cover the human research activities included in the proposal / work statement before funds are released. Note that this requirement does not apply to Exempt and Interim IRB protocols, or grants for which VT is not the primary awardee.

The table on the following page indicates whether grant proposals are related to this IRB protocol, and which of the listed proposals, if any, have been compared to this IRB protocol, if required.

Date*	OSP Number	Sponsor	Grant Comparison Conducted?
05/31/2012	11247105	National Science Foundation	Not required (Exempt approval)

* Date this proposal number was compared, assessed as not requiring comparison, or comparison information was revised.

If this IRB protocol is to cover any other grant proposals, please contact the IRB office (irbadmin@vt.edu) immediately.

VirginiaTech

Office of Research Compliance
Institutational Review Board
North End Center, Suite 4120, Virginia Tech
300 Turner Street NW
Blacksburg, Virginia 24061
540/231-4606 Fax 540/231-0959
email irb@vt.edu
website http://www.irb.vt.edu

MEMORANDUM

DATE: January 31, 2014

TO: Cliff Shaffer, Eric Noel Fouh Mbindi, Sally Hamouda, N. Dwight Barnette

FROM: Virginia Tech Institutional Review Board (FWA00000572, expires April 25, 2018)

PROTOCOL TITLE: Tutorials for Teaching Recursion and Pointers

IRB NUMBER: **14-036**

Effective January 31, 2014, the Virginia Tech Institution Review Board (IRB) Chair, David M Moore, approved the New Application request for the above-mentioned research protocol.

This approval provides permission to begin the human subject activities outlined in the IRB-approved protocol and supporting documents.

Plans to deviate from the approved protocol and/or supporting documents must be submitted to the IRB as an amendment request and approved by the IRB prior to the implementation of any changes, regardless of how minor, except where necessary to eliminate apparent immediate hazards to the subjects. Report within 5 business days to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.

All investigators (listed above) are required to comply with the researcher requirements outlined at:

http://www.irb.vt.edu/pages/responsibilities.htm

(Please review responsibilities before the commencement of your research.)

PROTOCOL INFORMATION:

Approved As:	**Expedited, under 45 CFR 46.110 category(ies) 5,7**
Protocol Approval Date:	**January 31, 2014**
Protocol Expiration Date:	**January 30, 2015**
Continuing Review Due Date*:	**January 16, 2015**

*Date a Continuing Review application is due to the IRB office if human subject activities covered under this protocol, including data analysis, are to continue beyond the Protocol Expiration Date.

FEDERALLY FUNDED RESEARCH REQUIREMENTS:

Per federal regulations, 45 CFR 46.103(f), the IRB is required to compare all federally funded grant proposals/work statements to the IRB protocol(s) which cover the human research activities included in the proposal / work statement before funds are released. Note that this requirement does not apply to Exempt and Interim IRB protocols, or grants for which VT is not the primary awardee.

The table on the following page indicates whether grant proposals are related to this IRB protocol, and which of the listed proposals, if any, have been compared to this IRB protocol, if required.

Date*	OSP Number	Sponsor	Grant Comparison Conducted?
01/23/2014	13024208	University of Wisconsin Madison	Compared on 01/31/2014
01/23/2014	11247105	National Science Foundation	Compared on 01/31/2014

* Date this proposal number was compared, assessed as not requiring comparison, or comparison information was revised.

If this IRB protocol is to cover any other grant proposals, please contact the IRB office (irbadmin@vt.edu) immediately.

VirginiaTech

Office of Research Compliance
Institutational Review Board
North End Center, Suite 4120, Virginia Tech
300 Turner Street NW
Blacksburg, Virginia 24061
540/231-4606 Fax 540/231-0959
email irb@vt.edu
website http://www.irb.vt.edu

MEMORANDUM

DATE: June 5, 2014

TO: Cliff Shaffer, Jeremy V Ernst, N. Dwight Barnette, Susan Rodger

FROM: Virginia Tech Institutional Review Board (FWA00000572, expires April 25, 2018)

PROTOCOL TITLE: Collaborative Research: Assessing and Expanding the Impact of OpenDSA, an Open-Source, Interactive eTextbook for Data Structures and Algorithms

IRB NUMBER: **14-623**

Effective June 5, 2014, the Virginia Tech Institution Review Board (IRB) Chair, David M Moore, approved the New Application request for the above-mentioned research protocol.

This approval provides permission to begin the human subject activities outlined in the IRB-approved protocol and supporting documents.

Plans to deviate from the approved protocol and/or supporting documents must be submitted to the IRB as an amendment request and approved by the IRB prior to the implementation of any changes, regardless of how minor, except where necessary to eliminate apparent immediate hazards to the subjects. Report within 5 business days to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.

All investigators (listed above) are required to comply with the researcher requirements outlined at:

http://www.irb.vt.edu/pages/responsibilities.htm

(Please review responsibilities before the commencement of your research.)

PROTOCOL INFORMATION:

Approved As:	**Expedited, under 45 CFR 46.110 category(ies) 5,7**
Protocol Approval Date:	**June 5, 2014**
Protocol Expiration Date:	**June 4, 2015**
Continuing Review Due Date*:	**May 21, 2015**

*Date a Continuing Review application is due to the IRB office if human subject activities covered under this protocol, including data analysis, are to continue beyond the Protocol Expiration Date.

FEDERALLY FUNDED RESEARCH REQUIREMENTS:

Per federal regulations, 45 CFR 46.103(f), the IRB is required to compare all federally funded grant proposals/work statements to the IRB protocol(s) which cover the human research activities included in the proposal / work statement before funds are released. Note that this requirement does not apply to Exempt and Interim IRB protocols, or grants for which VT is not the primary awardee.

The table on the following page indicates whether grant proposals are related to this IRB protocol, and which of the listed proposals, if any, have been compared to this IRB protocol, if required.

Date*	OSP Number	Sponsor	Grant Comparison Conducted?
05/30/2014	14163802	National Science Foundation	Compared on 06/05/2014

* Date this proposal number was compared, assessed as not requiring comparison, or comparison information was revised.

If this IRB protocol is to cover any other grant proposals, please contact the IRB office (irbadmin@vt.edu) immediately.