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

Improving Undergraduate STEM Education (IUSE)
Due February 4, 2014

We request support to study the efficacy of and disseminate a STEM learning environment named OpenDSA. OpenDSA is an open-source project with international collaboration. It has the potential to fundamentally change instruction in courses on Data Structures and Algorithms (DSA) and Formal Languages and Automata (FLA). Such courses play a central role in Computer Science curricula. By combining textbook-quality content with visualization and a rich collection of automatically assessed interactive exercises, OpenDSA can solve the following problems with DSA and FLA courses. (1) Students often find this material difficult to comprehend because much of the content is about dynamic processes, such as the behavior of algorithms and their effects over time on data structures. Static media (textbooks) do a poor job of conveying dynamic process. (2) Algorithm visualizations (AVs) have been demonstrated to be pedagogically effective for presenting this material, but adoption has been lower than documented instructor support would indicate. OpenDSA's complete units of instruction help to ease the adoption problems that have plagued previous standalone AV efforts. (3) The greatest difficulty that DSA and FLA students encounter is lack of practice and lack of feedback about whether they understand the material. A typical course offers only a small number of homework problems and test problems, whose results come only long after the student gives an answer. OpenDSA provides a steady stream of exercises and activities with automated grading and immediate feedback on performance. Students (and instructors) can better know that they are on track.

Written using HTML5 standards, OpenDSA can be used on all major browsers, most tablets, and many smart phones. OpenDSA's core infrastructure supports development of a wide range of interactive courseware and exercises, along with support for data collection used to analyze student performance. An initial core of about one semester's worth of instruction has been completed.

This proposal seeks to scale up OpenDSA in a number of ways. The highly successful JFLAP software for interactive instruction on FLA will be redeployed within the OpenDSA framework using HTML5 standards, thereby increasing access. A wide range of colleges and universities will be involved in disseminating OpenDSA and assessing its impact on student learning, and OpenDSA's use in a number of innovative instructional settings will be explored. The OpenDSA infrastructure will be enriched, allowing instructors to more easily tailor the materials to their specific classroom needs, and encouraging new content contributions from these instructors. A number of technical pedagogical experiments will be conducted, such as measuring the effects of augmenting content with audio narration in slideshows, and navigation through topics with concept maps. We will study how these materials can improve teaching in a range of courses for which we create relevant content. Our efforts will impact future active eTextbook projects by demonstrating successful ways to integrate content, interactivity, and assessment in an open-source, creative-commons environment. We will experiment with new models of dissemination for open-source content in conjunction with commercial online content publishers such as Zyante.

Intellectual Merit: The main research objective of this project is to test the efficacy of OpenDSA, and determine the circumstances under which success can be achieved. In particular, we study the effect on student learning of integrating content with visualizations and a rich collection of practice exercises with automated feedback. We also study how using eTextbook materials affects the evolving pedagogical approaches of instructors of DSA and FLA courses.

Broader Impact: Active eTextbooks are valuable beyond Computer Science. Prior research indicates that online instruction in many fields can be enhanced by student interaction with well-designed exercises. Our efforts will provide an exemplar of how collaborative, open-sourced workflows could be used to develop active eTextbooks for many disciplines.

1 Active eTextbooks for DSA

Data Structures and Algorithms (DSA) play a central role in the Computer Science curricula [66], defining the transition from learning programming to learning computer science. However, students often find this material difficult to comprehend because so much of the content is about the dynamic process of algorithms, their effects over time on data structures, and their analysis (determining their growth rates). Dynamic process is difficult to convey using static presentation media such as text and images in a textbook. During lecture, instructors typically draw on the board, trying to illustrate dynamic processes through words and constant changes to the diagrams. Many students have a hard time understanding these explanations at a detailed level or cannot reproduce the intermediate steps to get to the final result. Another difficulty is lack of practice with problems and exercises. Since the best types of problems for such courses are hard to grade by hand, students normally experience only a small number of homework and test problems, whose results come only long after the student gives an answer. The dearth of feedback to students regarding whether they understand the material compounds the difficulty of teaching and learning DSA.

CS instructors have a long tradition of using Algorithm Visualization (AV) [64, 84, 16] to convey dynamic concepts. 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. Such improvements most likely occurr when the level of student engagement and interaction is higher [32, 94, 16]. AV has been used as a means both to deliver the necessary dynamic exposition, and to increase student interaction with the material through interactive exercises [47]. Surveys [64, 83] show that instructors are positive about using AVs in theory, and students are overwhelmingly in favor of using AVs when given the opportunity. However, these same surveys show many impediments to AV adoption, including finding and using good materials, and "fitting them" into existing classes. Fortunately we do have experience with one area of computer science that has had strong AV adoption. That is formal languages and automata (FLA), where the educational software JFLAP [70] has been widely adopted in courses around the world.

The OpenDSA project seeks to address all of the issues listed above. OpenDSA's goal is to build complete open-source, online, configurable eTextbooks for DSA and FLA courses. The content is presented as a series of small modules implemented using HTML5 technology. Thus the modules can be viewed on any modern browser with no additional plugins or software needed, and can even run on tablets and many mobile devices.

OpenDSA modules combine content in the form of text, visualizations, and simulations with a rich variety of exercises and assessment questions. Since OpenDSA modules are complete units of instruction, they are easy for instructors to use as replacements for their existing coverage of topics (similar to adopting a new textbook) rather than including an AV on top of their existing presentation. Since OpenDSA's exercises are immediately assessed, with problem instances generated at random, students gain far more practice than is possible with normal paper textbooks. Since the content is highly visual and interactive, students not only get to see the dynamic aspects of the processes under study, they also get to manipulate these dynamic aspects themselves. Emphasizing student engagement with the material conforms to the best practices as developed through more than a decade of research by the AV research community [63, 56, 42].

Each module includes 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 [50, 3]. We do make use of simple multiple choice and give-a-number style questions, for which we use the

Khan Academy Exercise Infrastructure [38] to generate individual problem instances. We also include many interactive exercises. We make extensive use of "algorithm simulation" or "proficiency" exercises, as pioneered by the TRAKLA2 project [47]. (Note that the TRAKLA2 developers from Aalto University in Helsinki are active participants in OpenDSA, having developed the JSAV graphics library [37, 35] and several OpenDSA exercises. One could view OpenDSA as "TRAKLA3".) In algorithm proficiency exercises, students are shown a data structure in a graphical interface, and must manipulate it to demonstrate knowledge of an algorithmic process. For example, they might show the swap operations that a given sorting algorithm uses. Or they might show the changes that take place when a new element is inserted into a tree structure. Other OpenDSA exercises make use of small simulations for algorithms or mathematical equations to let students see the effects that result from changing the input parameters. Small-scale programming exercises are automatically assessed for correctness. These problems are similar to small homework problems traditionally given in such a course, but which have been hard to grade.

A key feature of OpenDSA is that it is an open-source project (hence the name). This is significant in a number of ways. One is that instructors are supported in configuring OpenDSA to their needs (selecting the modules that they need for their course). They can also revise any aspect of the material as they choose, and contribute such changes back to the project if they choose. We have as a goal to become a community resource with community contributions from a variety of instructors and students. This combination of use and contribution should promote broad "buy-in" by the DSA and FLA instructional communities. Since the early days of Java, developing AVs has been used as training for students seeking to learn web technologies, but unfortunately such independent efforts often are soon lost [84]. The OpenDSA project provides a framework within which students across the world could develop visualizations and contribute them to the broader community.

OpenDSA disrupts the fundamental balance today between students, textbooks, course content and activities, and instructors. This approach of combining tutorial content with automated assessment makes OpenDSA's approach a potential solution to a major problem with another disruptor of education: the MOOC. MOOCs often suffer from a lack of meaningful assessment and coursework. OpenDSA provides a meaningful way to give students practice material and instructors an assessment mechanism in a way that scales up to MOOC-sized classes. Our fundamental research questions relate to studying the effects of this change in the balance, both on the part of students and on the part of instructors. How will OpenDSA impact student learning? How can the OpenDSA environment be used by instructors to support learning? How will eTextbooks encourage instructors to make changes in instruction? How can an eTextbook be disseminated?

2 Prior Work

Programmed Learning, Programmed Instruction, Keller plan, and Integrated Learning System are concepts that formed the foundation for Computer Based Training (CBT) in the early to mid 1980s [44]. The concept was originally based on B.F. Skinner's work [90]. These ideas are still used, for example in the National Education Training Group (NETg) series from Thomson/Course Technologies. Typically such implementations are used by industry (HR departments) for employee training, and do not involve a human instructor.

Most effort by CS educators involved with interactive AVs has focused on developing AV technologies and the AVs themselves. At best there have been small, focused tutorial modules that incorporate AVs into a particular topic or small set of topics within a course. For example, [23] reports on web pages integrated with applets that were used in parts of a theory of computation course. Many instructors who teach FLA now use Rodger's JFLAP (as reported in [9], JFLAP is one of the most widely used tools for FLA). In the preface of Rodger's user manual on JFLAP [71], Rodger warns the reader "our book assumes that the reader has read briefly about these topics first in an automata theory textbook or a compiler textbook".

Ross [76] describes an infrastructure that uses Perl and Dreamweaver to create hypertextbooks. Parts of a theory of computing text and a biology book were produced using this technology. Unfortunately, the system requirements for these hypertextbooks hinder their wide adoption due to browser restrictions and use of Java applets. Rößling and Vellaramkalayil [79] report on integrating AVs into Moodle-based lessons, but the emphasis is again on a technology that would support visualization-based hypertextbooks in Moodle. To our knowledge little progress has been made on the actual writing of such a textbook.

Titterton, Lewis, and Clancy [93] have used a lab-centric mode of instruction for introductory CS courses at UC-Berkeley. Their current work is being done in Moodle and uses a small amount of AVs along with many "check point" exercises that students must complete as they progress through the material. It is built upon an earlier technology called UC-WISE that was developed and used at UC-Berkeley. Their materials are designed for introductory CS courses that aimed mostly at developing students' programming skills. Hence they make only limited use of AVs. Alharbi, Henskens, and Hannaford [2] are using eXe, an open-source authoring system that allows instructors to create academic-related web content using XML and HTML (`http://exelearning.org/`). Their efforts intend to help teach an Operating Systems course using visualization. Learners can interact with the AVs, and can take quizzes and tests online. They used the Sharable Content Object Reference Model (SCORM: `http://www.adlnet.gov/`) to support integration of digital teaching materials with a CMS (Moodle in their case).

Karavirta [34] began to integrate AVs with hypertext using HTML and JavaScript, allowing the hypertextbook to be viewed in any browser without additional plug ins. Learners can interact with the animation and draw annotations on it, but his system did not store the annotation, nor support quizzes and tests. Note that Karavirta is now working with OpenDSA. He developed the JSAV support library, described in Section 3.

Miller and Ranum's interactive eTextbook for Python programming [52], and *CS Circles* [68] 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 eBook, and our interactions with their project motivated our use of reStructuredText (reST) [69] and Sphinx (sphinx.pocoo.org) as our authoring system. Miller and Ranum produced a complete book that includes embedded video clips, active code blocks that can be edited within the book's browser page 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. 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. Their grading system is rudimentary and only provides students with simple pass/fail feedback upon completing an exercise. In our opinion, the most important thing missing from this effort is a wider variety of automated assessment with immediate feedback.

CS Circles is an exercise-centric eTextbook for learning Python. It uses the WordPress Content Management System for authoring and the CodeMirror plugin [28] to allow in-browser code editing and automated programming assessment. Both Miller and Ranum's book and *CS Circles* use Guo's online Python tutor [25], an embeddable web program visualization for Python. The online Python tutor takes a python source code as input and outputs an *execution trace* of the program. The trace is encoded in JSON format and sent to the user's browser for visualization via HTTP GET requests. The backend can work on any webserver with CGI support or on the Google App Engine.

Large scale use of educational hypermedia in South Korea provides interesting feedback about the challenges of eTextbooks. Kim and Jung [40] identify usability, portability, interactivity, and feedback as major elements to consider while designing such systems. Learners should be able to ask questions and receive help, as well as control, manipulate, search, and browse the eTextbook content. They also advocate for the development of models to support group collaboration.

To our knowledge, OpenDSA is the first effort to write a complete eTextbook tightly integrated with AVs and automatically assessed exercises that could be used as the primary learning resource in a semester-long computer science course. This is surprising because Marc Brown's groundbreaking dissertation on AV from 1988 [8] states: "Much of the success of the BALSA system at Brown [at the time] is due to the tight integration of its development with the development of a textbook and curriculum for a particular course. BALSA was more than a resource for that course – the course was rendered in software in the BALSA system." Some projects have developed AVs for all the topics of a course, such as JFLAP, which covers topics for a full course on FLA, and JHAVÉ and others for DSA topics. But none of these replace or otherwise integrate with tutorial information (i.e., the textbook). The closest that we are aware of is the AlVIE project [12], which has AVs referenced directly from the (paper) textbook.

Why have AV developers not heeded Brown and authored complete eTextbooks with integrated AVs and exercises? Why have they instead focused on developing AV technologies, separate from the tutorial content? The answer is that creating an active eTextbook is a huge amount of work. DSA and FLA content is especially difficult to develop. Not only have CS educators as yet been unable to author such an eTextbook, but commercial publishers (such as Zyante) that have successfully used interactive technologies for lower-level courses are finding the needs of a DSA course are much more complex. Another impediment had been technical. Java, JavaScript, and Flash collectively were a big step forward in providing cross-platform interactivity. HTML5 provides a further lowering of technical hurdles, making the development task less daunting by permitting one implementation to run on all major browsers, tablets, and even many mobiles. The vision for what an eTextbook can bring to Computer Science courses is laid out in a recent ITiCSE Working Group report [42].

Sophisticated automated exercise environments have been developed to support K12 math instruction, such as IXL [33] and Khan Academy [39]. Many know Khan Academy (KA) for its collection of video lectures on a wide variety of topics. Equally important is the rich infrastructure and large collection of exercises that allow students to practice math problems. The KA infrastructure is especially notable since it is open source, with the exercises themselves implemented in HTML/JavaScript. Simple exercises include multiple choice, fill-in-the-blank, and ordering a set of choices. But the software infrastructure also allows developers to program unique interfaces and activities for specific exercises. The main limitation is only the designer's imagination for how to create an exercise such that the answer can be automatically graded. These capabilities are important to OpenDSA, since they allow us to use the KA exercise infrastructure to create exercises that are relevant to DSA and FLA rather than K12 math. The KA organization has attracted a significant number of volunteers who develop and contribute new exercises. We believe that both the exercise infrastructure and the volunteer contributor model are appropriate for developing OpenDSA.

3 Preliminary Progress on OpenDSA

Project PIs Shaffer, Naps, and Rodger bring unique resources and experience that make this project feasible. We all have extensive experience both with creating relevant content [81, 82] and with AV courseware, including the JHAVÉ [61, 57, 67, 58] and JFLAP [70, 71] systems, and the AlgoViz Portal [1]. Naps has coordinated two ITiCSE Working Groups on AV [64, 59] and two on eTextbooks [77, 42]. Between us we have active collaborations or extensive interactions with most of the major AV developers in the world. We have had some success in leveraging this extensive network of developers to help with OpenDSA (most importantly with the Aalto University group, but the author list of [42] gives a good indication of our current interactions).

Since 2011, Shaffer has worked with Ville Karavirta of Aalto University on JSAV, a JavaScript-based AV library that is meant to be the foundation for building AVs for the active eTextbook [35, 37]. Dr. Karavirta was a major developer for the TRAKLA2 system [47, 41], and TRAKLA2 serves as a key inspiration for our vision with its concept of "proficiency exercises". Under funding from NSF TUES and NSF EAGER, we have already developed a substantial amount of content including class-tested chapters on linear structures, binary trees, sorting, and hashing [10]. We encourage reviewers to look at a sample book instance available at the OpenDSA website, `http://algoviz.org/OpenDSA/Books/OpenDSA`.

To the reader the fundamental unit of OpenDSA is the module, which is a single browser page containing text, graphics, visualizations, code snippets, and exercises of various types. A module corresponds to a section in a traditional textbook or a topic that might be covered in a single lecture or part of a lecture. They are generally equivalent to around 15-30 minutes of instruction, though some modules are much shorter. Modules can be grouped to form "chapters". Modules are authored using reStructuredText [69] (reST). ReST is a so-called "lightweight" markup language, originally designed to produce Python program documentation. Sphinx is the engine that converts reST files to HTML, LaTeX, EPUB, 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. We wrote several directives that augment support for creating book-length documents that Sphinx currently lacks.

OpenDSA includes a configuration mechanism that allows instructors to select from among existing modules to compile into an eTextbook "instance" that contains only those modules that they wish to use for their course. Based on the configuration file, the module source and the indicated collection of independent visualizations and exercises are compiled together to produce a set of HTML pages (one per module) that define an instance of an OpenDSA eTextbook. Navigation through a given eTextbook is currently done through an index of modules that looks much like a traditional table of contents.

OpenDSA uses a client-server architecture. A content server delivers the HTML documents along with embedded visualizations and exercises that make up the eTextbook. The data collection server is a web application that we built using the Django web framework with a MySQL database for data persistence. We designed an API to support communication between the OpenDSA "front end" (the content that runs in the student's browser), and the data collection "back end". The back end collects the necessary information regarding student progress to provide a history of completed exercises for use by the student and the instructor. We also collect fine-grained user interaction data. Such log data can help debug usability or pedagogical problems with the tutorials, or guide redesign to discourage pedagogically poor student behavior [7]. Additional information on the

OpenDSA architecture can be found in [17].

During Fall 2012, Spring 2013, and Fall 2013, OpenDSA materials have been used in classes at five universities in three countries, and OpenDSA is being used in three additional universities during Spring 2014. We have conducted a number of initial evaluations of OpenDSA, both for pedagogical effectiveness and for user satisfaction by students and instructors [26]. OpenDSA was used to replace three weeks worth of standard lecture materials on sorting and hashing during October 2012 by around 60 students in a DSA course at Virginia Tech. Students in the treatment section (that used OpenDSA) scored about one half of a standard deviation higher on the resulting exam than the control group (that did not use OpenDSA), but this was not statistically significant. Student evaluations on OpenDSA from every class over each semester of use have been highly positive. During Fall 2012, mean scores on preference for interactive online tutorials as compared to standard lecture actually went up after the students had experience with OpenDSA materials. Instructors have found that OpenDSA allows them to spend a greater fraction of lecture time on content related to the more abstract and difficult topics, and less on the mechanics of the algorithms. More details on our Fall 2012 evaluation can be found in [26].

4 JFLAP

JFLAP began in 1990 as a tool for pushdown automata written in C++ and X Windows, then added finite automata and Turing machines to become FLAP. Stand-alone tools for topics such as construction proofs, grammars, parsing algorithms, L-systems, and other models of Turing machines were eventually merged in. In 1996 we initiated a Java version of FLAP, called JFLAP. In 2002, a new interface and new material was added such as Moore and Mealy machines, CYK parsing, and graph layout algorithms for automata. In 2011, a redesign of JFLAP was started to allow for a more flexible alphabet and several new features that should be completed in 2014. In addition, Rodger is writing a static FLA textbook that integrates JFLAP into the presentation. Students read the textbook and try JFLAP exercises by loading JFLAP files or creating new examples.

JFLAP is used around the world in FLA courses. The `jflap.org` site has over 460,000 visits since 2005, and by 2006 JFLAP had been downloaded from over 160 countries. JFLAP was one of two finalist candidates in the NEEDS Premier Award for Excellence in Engineering Education Courseware competition in 2007. The submission packet contained letters from over 60 faculty in support of JFLAP. There are nine books that use JFLAP in some way including [43, 53, 22, 5, 20]. There are over fifteen JFLAP papers written by Rodger such as [74, 73], and over twenty-five papers written by others that use JFLAP in some way such as modifying it for blind students [13] or using it in a class [65]. In 2005-2007 Rodger conducted a study on JFLAP with fourteen universities. The majority of students indicated that having access to JFLAP made learning course concepts easier, made them feel more engaged in the course, and made the course more enjoyable [75].

With the software and the learning materials separate, students have to run the software and read along in the book, or switch between an online textual copy and the software. JFLAP is written in Java and does not run on mobile devices or tablets. The logical next step is to fully integrate text with visualizations using HTML5. OpenDSA will provide the infrastructure for completing such an eTextbook. The current static learning materials rewritten using OpenDSA authoring support and automated assessment will be built into JFLAP practice exercises with OpenDSA infrastructure.

We propose a three-year plan for converting the main parts of JFLAP into an eTextbook using the OpenDSA structure. The first year will focus on types of automata, the second will focus on proofs with automata, and the third year will focus on grammars, parsing and associated proofs. In parallel with the HTML5 implementation of JFLAP, each year the learning materials and assessments for the topics will be integrated into OpenDSA.

5 Research Agenda

With existing support from NSF and volunteer efforts from a number of collaborators, OpenDSA is off to a good start in building the base infrastructure and initial content. We are somewhat ahead of the schedule laid out in the proposal that lead to our original TUES award. However, much remains to be done with content, infrastructure, and pedagogical studies, and many new opportunities have arisen as detailed below. The main research objective of the proposed project is to test the efficacy of the integrated OpenDSA courseware and determine the circumstances under which success can be achieved. In this section we describe the specific efforts to be made over the three-year life of this award. We present this in three broad areas of content development, infrastructure development, and pedagogical studies.

Our goals include integrating certain existing AV-related projects into OpenDSA: JFLAP and TRAKLA2 (both described above) and JHAVÉPOP (described below). Doing this goes beyond just improving the original projects or giving them more exposure through reimplementation in HTML5. They have been carefully selected because they provide important activities for students within the context of DSA and FLA courses. Integration of these activities within OpenDSA creates a learning environment that is more than the sum of its parts.

Throughout the development of OpenDSA, we have been guided by the principles of cognitive theory of multimedia presentation espoused by Richard Mayer [48, 49]. Mayer has synthesized from pedagogical literature a set of principles that can inform the design of instructional multimedia such as OpenDSA. Derived from Cognitive Load Theory [91], it aims to maximize the benefits of engaging both the learner's verbal and visual channels, while minimizing distracting and extraneous material. The most compelling aspects of the principles are (1) they provide practical advice that can be used by designers, and (2) they are backed up (collectively) by literally hundreds of individual studies that measure learner gains. Mayer's work directly influences some of the goals listed next.

5.1 Content and Presentation

We have gone through three stages in our philosophy of presentation. When we first began the OpenDSA project, our primary goal was to tightly integrate textbook content with AVs, in order to solve problems with student comprehension of the dynamic processes associated with DSA. We soon realized that making students engage the material through a series of interactive exercises that provide immediate assessment and feedback is even more important than the visualizations. The third stage in our evolution came when we realized that visual presentation of all aspects of the content tends to be more effective than textual presentation. This means that, to the greatest extent possible, the textual content should be shifted into visuals supported by small amounts of text. This is most typically expressed in OpenDSA as a series of "slideshows" embedded within the modules, whose pacing is controlled by the student. This more visual approach aligns well with Mayer's principles. Our evolution can be seen by comparing the current presentations on Sorting (the first OpenDSA chapter that we created) with the chapter on Linear Structures (one of the more recent chapters), which makes far more use of a series of small slideshows.

We believe that we have a good understanding of how to use (algorithm) visualization to present the behavior of dynamic processes as embodied by algorithms and data structures. What we have so far failed to achieve is a better approach to presenting conceptual material, especially the large body of analytical reasoning that can be the most difficult aspects of DSA or FLA courses for students to grasp. In particular, students have trouble understanding the fundamental principles of algorithm analysis theory, or its practical application to analyzing particular algorithms. We believe that we can make good progress by studying various instances of visual proofs that exist in the relevant math and computing literature [21, 92, 6, 89, 27], and extracting a collection of principles that we can use to create effective visualizations of analytical material. We seek to develop a rich collection of techniques and supported graphical primitives to augment our presentations. We seek to address some especially hard topics in a standard DSA course. Of special interest to us are good tutorial presentations for recursion, algorithm analysis, and NP-completeness. In particular, for recursion we have a vision for an interactive tutorial driven by practicing many small programming exercises, under the guidance of an adaptive tutoring system that takes students through a progression of increasingly difficult material as they become ready for it.

Our most pressing goal in terms of content is to complete a full semester course of OpenDSA materials where the content is delivered using as much visual support as possible. This means re-writing some of our earlier efforts (such as the Sorting chapter), creating new content, and further developing our repertoire of visual presentation techniques and tools.

Mayer's work provides much support for the incorporation of audio narration to replace (note, **not** duplicate!) textual narration of the content associated with the visuals. This would naturally fit with the current slideshow-based presentation where a typical slide presents a (text) sentence with a visual such as the current state of a data structure. Thus, our first experimentation will develop audio narration for aspects of the tutorials.

5.2 Infrastructure Development

OpenDSA content is comprised of a large collection of modules, where a specific module typically has others as a prerequisite. Collectively, the modules and their prerequisite structure defines a directed graph that could be viewed as a concept map. This approach is inspired in part by the Khan Academy (KA) Knowledge Map (`http://www.khanacademy.org/exercisedashboard?k`), which shows a directed graph with the prerequisite relationships for a large body of exercises for K12 mathematics. In our vision, each node on the DSA concept map corresponds to a tutorial where content is integrated with a collection of assessment activities for a particular topic. A typical semester course might include 50-100 modules. We have begun preliminary work toward this goal as part of our EAGER/SAVI award, in conjunction with Sadhana Puntambekar of University of Wisconsin-Madison. Another use for a concept map is to reorganize the glossary, which is currently an alphabetical listing of terms with definitions. Organizing the glossary as a concept map would make it easy for students to explore the connections between related terms.

Given a rich collection of modules and definitions for their prerequisite relationships, an instructor should be able to use a simple interface to select a subset of modules to make up a given course. The selected modules can be processed to generate a "book instance". We currently support this ability through an XML-based configuration file. The next step is to provide a GUI for easily creating the configuration file.

Instructors that have used OpenDSA have requested a mechanism for creating coursenotes for use in class presentations. While the AVs make for good in-class presentation tools, the module format does not directly lend itself to classroom presentation. A mechanism for authoring course slides (in reST) will be created that builds on the existing Sphinx slideshow theme.

We currently generate a massive amount of interaction log data. We need improved tools to turn low-level (syntactic) interactions into semantic meaning. For example, how much time do students spend on various exercises or slideshow presentations? Once we have improved log analysis tools, we seek to take what we learn from the log data to encourage best learning behaviors. Prior experience shows that we must be careful to prevent students from abusing the ability to repeat exercises. If it is possible to achieve completion of a component by guessing, even if it is likely to take additional time, then some students will take that approach [36]. We also need to make sure that students do not find the exercises to be tedious or unreasonably time consuming. There are many fine details to work out. For example, the proficiency exercises typically require students to successfully perform a series of steps correctly. Naively, grading might simply count the number of steps done correctly. Unfortunately, if the student makes a mistake at any point in the process, it might be impossible to recover and get any additional steps correct, leading to a non-representative assessment of their performance. One solution is to always bring the student back "on track" when an incorrect step is made. But if this is not done carefully, it might lead students to guess.

Our approach to key topics like recursion requires that we deliver and automatically assess small programming exercises. Standard test-case driven evaluation is already supported by OpenDSA in rudimentary form. We envision a more visual approach, as implemented in the JHAVÉPOP [19] exercise system developed at University of Wisconsin–Oshkosh. There, the student's solution is presented back to the student in the form of a visualization generated from the student's code. In the original JHAVÉPOP, the student must decide if their answer is correct. We seek to tie this approach in with automated assessment of the exercises. This will be particularly useful for teaching topics like pointer manipulation.

5.3 Pedagogical Studies

OpenDSA provides the opportunity to change pedagogy away from lecture-and-textbook. Making the course more interactive requires mapping the current lecture course into an interactive format to ensure alignment across the associated course objectives. Students and instructors face many challenges when making a transition from lecture-based courses to any format where students are expected to spend significant time on their own working through materials and tutorials. In studies involving similar effort [29], the faculty report some pushback from students, because in a face-to-face course instructors typically do more of the delivery and interaction with the content. We expect that when a course is organized around an active eTextbook, students will spend more time out of class with the material. We want them to become involved in the learning in a deeper and more active way than they were in the past. Success is related to students' level of self-directed learning and self-regulated learning (SRL) [54, 31]. SRL affects how they manage the work required by the new format, how they interpret the new assessment mechanism, and their understanding of the shift in the role of instructor to facilitator.

Henry, et al. [29] showed that students accustomed to traditional instructional strategies often require mentoring about how to direct their own learning of the material and perform the work. How do we build student's ability to self regulate? Self-directed learning represents a paradigm shift for many students since the study habits that have brought them success in traditional learning environments are not always effective in the new settings [30]. Our evaluation of the process therefore has to be formative and promote continued improvement in student performance. We will design effective feedback rubrics and assessment methodologies that are replicable and scalable.

We can hypothesize that instructors will start by using the eTextbook as a direct replacement for text and paper homework, and gradually evolve into larger changes in how they conduct their class. While OpenDSA enables pedagogical changes, does its availability encourage instructors to greater pedagogical experimentation? Does it encourage better teaching practices? Does it lead to more interactive classes? We will study and characterize the evolution of instructors' changing pedagogy with eTextbooks.

6 Implementation and Impact Study

Specific research questions and metrics for the implementation and impact study are discussed in Table 1. OpenDSA courseware will be evaluated in DSA and FLA courses at Virginia Tech (CS3114), Duke (Compsci334), and UW-Oshkosh (CS271), as well as at a number of other institutions. Due to its large enrollment, CS3114 is given to two sections each semester. Similar to experiments in Fall 2012, these class sections can be used to group students into control and experimental cohorts, with control groups receiving traditional, lecture-based instruction while the experimental group uses OpenDSA courseware. The large enrollment provides an opportunity to simultaneously compare new and old strategies while minimizing confounding factors.

From a teaching perspective, OpenDSA tutorials deliver course content incrementally, and balance teaching the content processes with interactive, auto-assessed activities that provide immediate feedback. From a learning perspective, they are intended to move students from a passive stance in a lecture-type classroom setting to an active position of constructing learning and tracking their own comprehension through immediate feedback received from the exercises. The theory of change driving the design and implementation of these tutorials is to encourage students' engagement with the content and involve them in the assessment loop as active participants in such a way that they, as well as their instructors, know that they are learning. According to Bandura's cognitive theory of self-efficacy [4], instruction that allows students to check their own progress at a designated level of proficiency positively impacts outcome. Constructivist theory suggests that timely feedback can encourage students to modify their work. Lovett and Greenhouse [45] show that receiving feedback on the process of learning gives significant improvement as compared to only receiving feedback from the instructor on performance outcomes.

The following research questions will be explored through a study sample of courses taught by instructors recruited at the SIGCSE annual meeting, through the SIGSE listserv, and by personal contacts to project participants. Additionally, a control sample will be selected and matched within institution where number of course sections permit, or alternatively based on an array of population, geographic, and institutional mission indicators identified though features of the study sample. As part of our evaluation, we have requested funding to support instructors from outside the project who will adopt OpenDSA materials for their courses, and collect implementation and impact data for us. We have budgeted $10,000 each year as stipends to support two outside adopters, for a total of six over the three years. While far more instructors than this are expected to use OpenDSA during this time, these instructors will be supported to both tailor OpenDSA content to their classes, and to conduct assessment beyond that normally expected from instructors who use OpenDSA.

Data sources are tagged to key research questions in the table. **RQ1** and **RQ2**: A parallel item cognitive pre-assessment and post-assessment will be administered to the study group and the comparison group to measure degrees of student proficiency concerning DSA and FLA content.

Table 1: Research questions and data sources. Pre: Pre-Assessment test. Post: Post-Assessment test. AUD: Analytic User Data. SI: Student Interviews. MI: Midterm Interviews. EI: End-of-Semester Interviews. TL: Teacher Log.

Data Source	Pre	Post	AUD	SI	MI	EI	TL
RQ1 Does OpenDSA increase student proficiencies with core data structures and algorithms content?	X	X					
RQ2 Does OpenDSA student proficiency exceed that of students not exposed to OpenDSA?							
RQ3 How do students use OpenDSA materials?			X				
RQ4 What factors led to instructor selection and implementation of OpenDSA materials?				X			
RQ5 Do instructors change their use of OpenDSA materials over time?					X	X	X
RQ6 Are DSA fundamental course structures altered based on implementation of OpenDSA?							X
RQ7 Does OpenDSA use and/or course success vary across student demographic and social dimensions?	X	X					

OpenDSA materials will consist of an open test bank and a closed test bank. The open test bank can be drawn from OpenDSA exercises, but will also be provided to the control group to use for review and assessment preparation purposes. The closed bank will be electronically administered with a proctor and used to measure progressions in core knowledge. **RQ3**: Analytic user data will consist of access frequency, overall time on task, and individual lesson completion time. These data permit the research team to gauge student use of OpenDSA as well as identify specific content that students collectively find difficult to understand. In addition to user data, randomized study sample site participants will be selected for small group interviews concerning the nature of OpenDSA materials use. **RQ4**, **RQ5**, and **RQ6**: Instructor implementation information will be gathered through a biweekly teacher log, a total of six entries per instructor, as well as midterm and end-of-semester follow-up interviews. Information such as content covered, course format, course learning environment, course activity, and course pace will be collected. **RQ7**: Data sources from **RQ1** and **RQ2** will be used to determine if OpenDSA materials have differential impacts on student academic outcomes and tendencies of integrated courseware use for individuals with different demographic characteristics such as females, underrepresented ethnic minorities, and first generation college students.

Each data source will be collected in each project year. Study sites implementing OpenDSA will contain two student study samples (Spring and Fall semesters) each project year. Pre/post assessment data, analytic user data, small group student interviews, midterm and end-of semester interviews, and teacher logs will be collected or conducted from/for all study sites. Similarly, the control sites will also contain two student study samples each project year where pre/post assessment data and teacher log data will be collected for non-treatment comparison.

The evaluation plan requires systematic data collection at multiple sites, along with planning, merging, analyzing, and documenting the data collected in these sites. We will conduct validity and reliability tests and report the psychometric quality of the instruments developed because aspects of this project require development of new tools and assessment. The plan is sufficiently complex that we have requested support for an external project consultant to execute the plan.

Project evaluation provides continuous, timely, and constructive feedback through multiple evaluations during the course of the project, suggesting program changes, if necessary, to ensure project success. The evaluation plan is guided by these goals:

1. Determine whether project goals and benchmarks were reached concerning materials development, research, and dissemination.
2. Determine whether the specified research methods were employed during the implementation and impact study.
3. Obtain feedback from students and teachers on their perception of the overall effectiveness of the OpenDSA materials. Interviews will be conducted with selected teachers and students.
4. Measure the degree of transfer of the OpenDSA materials.

7 Dissemination Plan

The "open" aspect of OpenDSA means that instructors themselves can become active participants in designing and contributing to the content that OpenDSA offers to their students. This involvement can come in many ways:

- Given that OpenDSA is not a linear sequence of material but rather a "bag of instructional resources", instructors can define their own specific paths through the instructional modules, encapsulated as eTextbook "instances". These instances can then be distributed as OpenDSA configuration files for others to use.
- Instructors are able to contribute their own content to OpenDSA. These contributions might be any artifact type supported by OpenDSA such as text, visualizations, interactive activities, or exercises.
- The contributions might be original (at least to OpenDSA), or modified from materials already existing in OpenDSA.

To encourage faculty to become involved in designing and creating OpenDSA content, we will offer workshops at conferences such as SIGCSE. Such workshops expose participants to materials that we have already developed, which aids adoption. They also show participants how to design and contribute their own materials. We anticipate doing workshops during each year of the project.

In addition to the workshops, we will also hold "advisory panel" meetings at SIGCSE that will be analogous to the stakeholder meetings for our AlgoViz Portal project that we hosted in 2008 and 2009. Such meetings involve individuals who have already become active in the OpenDSA community. The AlgoViz stakeholder meetings turned out to be extremely important in the evolution of that project and in its subsequent success. For OpenDSA, involving key players in the AV development community is an important part of generating a robust community of content developers. We therefore include in our budget support for both the workshops and the stakeholder meetings at SIGCSE during the years of the proposal. We include a request for some international travel support since some of our most important collaborators are located in Europe (see for example the support letter from Aalto University).

One aspect of dissemination is the support that instructors need when using online materials with automated assessment. They need support for collecting and maintaining the (auto) graded material, and the ability to track progress for their students. OpenDSA currently provides this support as part of its server-side infrastructure. However, while it is feasible for an academic project to develop high-quality instructional material, and feasible for an academic project to make such materials available for a period of many years, it is not likely to be feasible for an academic project to sustain the infrastructure needed to support a large body of students (potentially 10s or 100s of thousands per year) and their instructors. This is the sort of service that is more appropriate for a commercial entity, providing hosting and back-end data collection as a service.

Our vision requires that OpenDSA content always be open, and to be viewed as a community-based resource. However, we also seek to work with commercial service providers on a "value added" basis, where the provider hosts OpenDSA book instances, and maintains student progress and instructional gradesheets in a FERPA-compliant manner. To this end, we have begun discussions to partner with Zyante (`http://www.zyante.com/`) to provide materials for their DSA offerings. We have a preliminary analysis of the infrastructure changes that will be needed for OpenDSA materials to communicate with Zyante's infrastructure. While not trivial, the necessary APIs can be designed that allow OpenDSA to both be usable in its open form, or used through Zyante (or potentially other service providers) with the instructor services added at a reasonable cost to students. This model could prove decisive to the future of eTextbooks and online courses in general.

8 Project Management

OpenDSA is a collaboration of several universities, including the three directly supported under this proposal. Dr. Shaffer will be responsible for coordinating the activities of the various participants. Dr. Ernst is experienced in educational research and development initiatives and will be responsible for phases of the project including data collection and reporting. They will work with the other PIs to develop the detailed instruments and collect the necessary data, as described in Section 6. We request funding for 2.5 years of Postdoc support. The Postdoc will assist Dr. Shaffer in directing the software and content development efforts, interacting with outside faculty who will adopt and evaluate OpenDSA, and in conducting assessment activities (under the supervision of Dr. Ernst).

Drs. Rodger and Naps will direct efforts at Duke and UW-Oshkosh, respectively. Dr. Rodger's will oversee reimplementation and deployment within the OpenDSA framework of JFLAP. She will perform evaluations at Duke with parts of the new JFLAP. Dr. Naps will work with Dr. Shaffer to develop OpenDSA content, consult on reimplementation of JHAVÉPOP, perform evaluations at UW-Oshkosh, and coordinate with adopting instructors. We already routinely hold Skype conference calls to coordinate our work, including with our partners at Aalto University. We will formalize this process into a regular monthly Skype discussion of the project PIs and other relevant participants. The annual meetings to precede the SIGCSE conference will serve as additional opportunities to interact in person with key stakeholders.

Key timing aspects are as follows. Each year, we will recruit two outside faculty who will both use OpenDSA in a course, and provide extensive evaluation feedback. These individuals will be compensated by grant funds. We expect that a broad group of faculty will also adopt OpenDSA in their classes, but they will not have the same evaluation obligations. Our postdoc will take on management responsibilities for project development and implementing evaluation tasks. The postdoc should be in place by the middle of the first year. We will have our major stakeholder meetings at the annual SIGCSE conference in early March of years two and three. Some aspects of content development, infrastructure development, and pedagogical studies will be ongoing in each of the three years. But we expect to complete most of the remaining infrastructure development as described in Section 5.2 during year one. Year two will see completion of most of the content-related tasks, with further polishing of materials in year three. Evaluation and dissemination will be ongoing throughout the life of the project, but will be the major focus of year three.

9 Broader Impacts

OpernDSA provides an interactive, visual presentation for dynamic material (algorithms and operations on data structures) as well as visual presentation of abstract concepts (such as algorithm analysis), combined with immediate feedback through interactive exercises. OpenDSA has the potential to improve the learning of students of DSA and FLA courses in ways never before possible. Beyond Computer Science, the models we provide for architecting, using in class, disseminating, and assessing eTextbook materials are broadly applicable across a range of STEM and non-STEM disciplines.

This project has had many undergraduate students work on AV-related activities over the years. PI Naps managed a site REU project from 2009-2011 that supported 5–10 undergraduates each summer to develop JHAVÉ modules. PI Shaffer has had approximately 20 undergraduate independent study students work on AV projects (9 during Spring 2014 semester alone), and PI Rodger has supervised over 20 undergraduates on JFLAP development. We anticipate that many undergraduates will in the future choose to do independent projects related to OpenDSA, including from other schools not directly supported by the project. Being an open-source project, OpenDSA provides a framework whereby such students can productively contribute. Shaffer has previously involved minority students through Virginia Tech's Multicultural Academic Opportunities Program (MAOP) internship program (equivalent to an NSF REU). We have budgeted money each year at Virginia Tech to support undergraduate researchers, including MAOP interns. Duke and UW-Oshkosh budgets include funds for undergraduate research.

10 Results from Prior NSF Support

NSF TUES Phase I Project (DUE-1139861) *Integrating the eTextbook: Truly Interactive Textbooks for Computer Science Education.* PIs: C.A. Shaffer, T. Simin Hall, T. Naps, R. Baraniuk. $200,000, 07/2012 – 06/2014. **NSF SAVI/EAGER Award (IIS-1258571)** *Dynamic Digital Text: An Innovation in STEM Education*, PIs: S. Puntambekar (UW-Madison), N. Narayanan (Auburn), and C.A. Shaffer (2013). $247,933, 01/2013 – 12/2014. **NSF CCLI Phase 1 Award (DUE-0836940)** *Building a Community and Establishing Best Practices in Algorithm Visualization through the AlgoViz Wiki.* PIs: C.A. Shaffer, S.H. Edwards. $149,206, 01/2009 – 12/2010. **NSF NSDL Small Project (DUE-0937863)** *The AlgoViz Portal: Lowering Barriers for Entry into an Online Educational Community.* PIs: C.A. Shaffer, S.H. Edwards, $149,999, 01/2010 – 12/2011. **Intellectual Merit** The first two projects provided online infrastructure (the AlgoViz Portal (`http://algoviz.org`) and related community development efforts to promote use of AV in computer science courses. This work was an important precursor to OpenDSA, as it allowed us to interact with many CS instructors and AV developers, leading us to an understanding of the fundamental missing parts in existing DSA instruction. They also initiated many of the international collaborations that lead to OpenDSA. Two journal papers [84, 16] and five conference papers [85, 88, 83, 87, 86] have been produced relating to this work. The second pair of (ongoing) awards support the initial phases of OpenDSA, and an active collaboration involving Virginia Tech and Aalto University (Helsinki), among others. Publications related to this work so far include [42, 37, 26, 17]. **Broader Impacts** include dissemination of AV artifacts and DSA courseware to a broad range of CS students, and made them available through the NSF NSDL.

NSF CCLI Phase 1 Awards (DUE-0341148 and DUE-0126494) *Integrating Algorithm*

Visualization into Computer Science Education. PIs: T.L. Naps, S. Grissom, and M. McNally, \$71,993 for 2001–2002 and \$197,118 for 2003–2007. **REU grant (CNS-0851569)** *Exploring Open Source Software: Development and Efficacy of Online Learning Environments in Computer Science*, PIs: T.L. Naps and D. Furcy, \$261,167 for 2009–2012. **Intellectual Merit** The first two grants resulted in significant development on the JHAVÉ algorithm visualization system. A website was established at `http://jhave.org` to disseminate the visualization-based materials that were produced for teaching data structures and algorithms. The following papers were published as a result of the grant, three had undergraduate students as co-authors [18, 46, 62, 51, 58, 60, 24, 55, 78]. **Broader Impacts** The REU grant provided student participants with extensive exposure to open-source software for the delivery of instructional algorithm visualizations, as developers, authors, and educational researchers.

NSF ITEST DRL-1031351 *Collaborative Research: Scaling up an Innovative Approach for Attracting Students to Computing* PIs: S.H. Rodger, S. Cooper (Stanford), W. Dann (CMU), M. Schep (Columbia College), R. Stalvey (College of Charleston), P. Lawhead (U. Mississippi)). \$2,499,870 for 2011–2016. This and award **NSF ITEST 0624642** develop curriculum materials for the Alice programming language, integrating computer science into K-12. **Intellectual Merit:** We developed over 80 tutorials on CS and animation topics for all K-12 disciplines [11, 72], teachers attending our workshops developed over 180 lesson plans (see `www.cs.duke.edu/csed/alice/aliceInSchools`). **Broader Impacts:** Summer workshops taught over 200 K-12 teachers programming and how to integrate computing into their disciplines. Published papers include 7 female undergraduates as co-authors, who participated in teaching and mentoring the teachers.

NSF TUES DUE-1044191 *Integrating Visualization and Interaction into the Formal Languages and Automata Course* PI: S.H. Rodger \$199,996 for 4/2011– 03/2015. **Intellectual Merit:** This and prior award **NSF CCLI 0442513** focus on enhancing JFLAP with new algorithms, designing curriculum materials, and running studies on the usability of JFLAP in learning automata theory. **Broader Impacts:** JFLAP is used extensively around the world in automata theory courses. The published papers [75, 74] include five undergraduate co-authors.

NSF EHR (DRL-1156629) *Transforming Teaching through Implementing Inquiry project*, PIs: J. Ernst, L. Bottomley, A. Clark, V.W. DeLuca, S. Ferguson, \$1,997,532, 2011–2015. **NSF EHR (DRL-1135051)** *Re-designed High Schools for Transformed STEM Learning*, PIs: E. Glennie, J. Ernst, \$1,954,066, 2011–2015. These projects showcase Co-PI Ernst's experience with developing and assessing STEM education initiatives. The first explores the use of cyberinfrastructure tools to improve quality and enhance delivery of professional development materials for grades 8-12 engineering, technology, and design educators. The second is a longitudinal research project to assess 1) North Carolina New Schools Project (NCNSP) STEM strand school student learning over time, using extant data, survey data, and performance assessments, contrasted with student performance in a matched set of comparison high schools using traditional curricula; and 2) School-level policies and instructional practices that schools employ to promote student learning through a series of case studies involving site visits and teacher logs. Publications include [15, 14, 80]. **Broader Impacts:** These projects informed the NCNSP STEM strand schools work with respect to implementation of the empirically re-visioned STEM model. They have provided a cyber system and content, focused on increasing engineering, technology, and design teacher quality, that is readily adaptable to the full breadth of STEM education implementation initiatives.

References

[1] AlgoViz.org. The AlgoViz portal, 2011. `http://algoviz.org`.

[2] 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)*, pages 205–208, September 2010.

[3] Heidi Andrade and Anna Valtcheva. Promoting learning and achievement through self-assessment. *Theory Into Practice*, 48(1):12–19, 2009.

[4] A. Bandura. *Social Foundations of thought and action: A social cognitive theory.* Prentice Hall, Englewood Cliffs, NJ, 1986.

[5] Mordechai Ben-Ari. *Principles of the Spin Model Checker.* Springer, London, 2006.

[6] Don Blaheta. A visual proof of amortised-linear resizable arrays. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '09, pages 338–338, New York, NY, USA, 2009. ACM.

[7] Daniel Aubrey Breakiron. Evaluating the integration of online, interactive tutorials into a data structures and algorithms course. Master's thesis, Virginia Tech, 2013.

[8] M.H. Brown. *Algorithm Animation.* MIT Press, Cambridge, Massachussets, 1988.

[9] P. Chakraborty, P.C. Saxena, and C. P. Katti. Fifty years of automata simulation: a review. *ACM Inroads*, 2(4):59–70, December 2011.

[10] OpenDSA Contributors. Opendsa project website. `algoviz.org/OpenDSA`, 2013.

[11] S. Cooper, W. Dann, D. Lewis, P. Lawhead, S. Rodger, M. Schep, and R. Stalvey. A pre-college professional development program. In *The 16th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2011)*, pages 188–192, 2011.

[12] P. Crescenzi and C. Nocentini. Fully integrating algorithm visualization into a CS2 course: A two-year experience. In *Proceedings of the 12th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE)*, pages 296–300, 2007.

[13] P. Crescenzi, L. Rossi, and G. Apollaro. Making turing machines accessible to blind students. In *Forty-third SIGCSE Technical Symposium on Computer Science Education*, pages 167–172. SIGCSE, March 2012.

[14] J. Ernst, A.C. Clark, V.W. DeLuca, and L. Bottomley. Professional development system design for grades 6-12 technology, engineering, and design educators. In *Proceedings of the American Society for Engineering Education Annual Conference and Exposition*, June 2013.

[15] J.V. Ernst. Authentic assessment in performance-based stem education activities. In *Proceedings of the Scaling STEM: Transforming Education Matters Annual Conference*, April 2012.

[16] E. Fouh, M. Akbar, and C.A. Shaffer. The role of visualization in computer science education. *Computers in the Schools*, 29:95–117, 2012.

[17] E. Fouh, V. Karavirta, D.A. Breakiron, S. Hamouda, S. Hall, T.L. Naps, and C.A. Shaffer. Design and architecture of an interactive etextbook – the OpenDSA system. *Science of Computer Programming*, 2014. to appear.

[18] D. Furcy, A. Jungwirth, and T. Naps. Blocktree–pedagogical information visualization for heuristic search. In *Proceedings of the Twentieth International Florida Artificial Intelligence Research Society Conference, May 7-9, 2007, Key West, Florida, USA*. AAAI Press, 2007.

[19] D.A. Furcy. JHAVEPOP: Visualizing linked-list operations in C++ and Java. *Journal of Computing Sciences in Colleges*, 25(1):32–41, October 2009.

[20] Wayne Goddard. *Introducing the Theory of Computation.* Jones and Bartlett, Sudbury, MA, 2008.

[21] Michael T. Goodrich and Roberto Tamassia. Teaching the analysis of algorithms with visual proofs. In *Proceedings of the Twenty-ninth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '98, pages 207–211, New York, NY, USA, 1998. ACM.

[22] G. L. Gopalakrishnan. *Computation Engineering: Applied Automata Theory and Logic.* Springer, New York, NY, 2006.

[23] M.T. Grinder, S.B. Kim, T.L. Lutey, R.J. Ross, and K.F. Walsh. Loving to learn theory: Active learning modules for the theory of computing. In *Proceedings of the 33rd ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 2002)*, pages 371–375, Cincinnati, Kentucky, 2002.

[24] S. Grissom, M.F. McNally, and T.L. Naps. Algorithm visualization in CS education: Comparing levels of student engagement. In *SoftVis '03: Proceedings of the 2003 ACM symposium on Software visualization*, pages 87–94, New York, NY, USA, 2003. ACM Press.

[25] 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, New York, NY, USA, 2013. ACM.

[26] S. Hall, E. Fouh, D. Breakiron, M. Elshehaly, and C.A. Shaffer. Education innovation for data structures and algorithms courses. In *Proceedings of ASEE Annual Conference*, page Paper #5951, Atlanta GA, June 2013.

[27] R.H. Hammack and D.W. Lyons. Alternating series convergence: a visual proof. *Teaching Mathematics and its Applications*, 25(2):58–60, 2006.

[28] M. Haverbeke. Codemirror (Version 2.x). `http://codemirror.net/`, 2011.

[29] H. Henry, D. Jonasses, R. Winholtz, and S. Khanna. Introducing problem based learning in a material science course in the undergraduate engineering curriculum. In *Proceedings of International Mechanical Engineering Congress & Exposition*, Vancouver, November 2010.

[30] C. Hmelo-Silver. Problem-based learning: What and how do students learn? *Educational Psychology Review*, 16(3):235–266, 2004.

[31] B.K. Hofer and P.R. Pintrich. The development of epistemological theories: beliefs about knowledge and knowing and their relation to learning. *Review of Educational Research*, 67(1):88–140, 1997.

[32] C.D. Hundhausen, S.A. Douglas, and J.T. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing*, 13:259–290, June 2002.

[33] IXL website. `http://www.ixl.com/`, 2011.

[34] V. Karavirta. Towards seamless merging of hypertext and algorithm animation. In *Proceedings of the Fifth Program Visualization Workshop, PVW'08*, volume 224 of *Electronic Notes in Theoretical Computer Science*, pages 105–114, 2009.

[35] V. Karavirta. Jsav github repository. `github.com/vkaravir/JSAV`, 2013.

[36] V. Karavirta, A. Korhonen, and L. Malmi. Different learners need different resubmission policies in automatic assessment systems. In *Proceedings of the 5th Annual Finnish / Baltic Sea Conference on Computer Science Education*, pages 95–102. University of Joensuu, November 2005.

[37] V. Karavirta and C.A. Shaffer. JSAV: The JavaScript Algorithm Visualization library. In *Proceedings of the 18th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2013)*, pages 159–164, Canterbury, UK, July 2013.

[38] `https://github.com/Khan/khan-exercises`, 2012.

[39] `http://khanacademy.org`, 2012.

[40] J.H.-Y. Kim and H.-Y. Jung. South Korean digital textbook project. *Computers in the Schools*, 27(3 & 4):247–265, 2010.

[41] A. Korhonen, L. Malmi, P. Silvasti, J. Nikander, P. Tenhunen, P. Mrd, H. Salonen, and V. Karavirta. TRAKLA2. `http://www.cs.hut.fi/Research/TRAKLA2/`, 2003.

[42] Ari Korhonen, Thomas Naps, Charles Boisvert, Pilu Crescenzi, Ville Karavirta, Linda Mannila, Bradley Miller, Briana Morrison, Susan H. Rodger, Rocky Ross, and Clifford A. Shaffer. Requirements and design strategies for open source interactive computer science ebooks. In *Proceedings of the ITiCSE Working Group Reports Conference on Innovation and Technology in Computer Science Education-working Group Reports*, ITiCSE -WGR '13, pages 53–72, New York, NY, USA, 2013. ACM.

[43] P. Linz. *An Introduction to Formal Languages and Automata, 5th Edition.* Jones and Bartlett, Sudbury, MA, 2011.

[44] B. Lockee, D. Moore, and J. Burton. Foundations of programmed instruction. In D.H. Jonassen, editor, *Handbook of research on educational communications and technology*, pages 545–569. Lawrence Erlbaum Associates, Mahwah, New Jersey, second edition, 2004.

[45] M.C. Lovett and J.B. Greenhouse. Applying cognitive theory to statistics instruction. *The American Statistician*, 54(3):196–211, 2000.

[46] J. Lucas, T.L. Naps, and G. Rößling. VisualGraph: a graph class designed for both undergraduate students and educators. *ACM SIGCSE Bulletin*, 35(1):167–171, 2003.

[47] 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(2):267–288, September 2004.

[48] R.E. Mayer. Cognitive theory and the design of multimedia instruction: An example of the two-way street between cognition and instruction. *New Directions for Teaching and Learning*, 89:55–71, Spring 2002.

[49] R.E. Mayer. Applying the science of learning: Evidence-based principles for the design of multimedia instruction. *American Psychologist*, 63(8):760–769, 2008.

[50] 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.

[51] M. McNally, T.L. Naps, D. Furcy, S. Grissom, and C. Trefftz. Supporting the rapid development of pedagogically effective algorithm visualizations. *Journal of Computing Sciences in Colleges*, 23(1):80–90, 2007.

[52] B.N. Miller and D.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 (ITiSE'12)*, pages 150–155, New York, NY, USA, 2012. ACM.

[53] Maxim Mozgovoy. *Algorithms, Languages, Automata, and Compilers.* Jones and Bartlett, Sudbury, MA, 2010.

[54] K.R. Muis. The role of epistemic beliefs in self-regulated learning. *Educational Psychologist*, 42(3):173–190, 2007.

[55] T. Naps and S. Grissom. The effective use of quicksort visualizations in the classroom. *Journal of Computing Sciences in Colles*, 18(1):88–96, 2002.

[56] Thomas Naps, Stephen Cooper, Boris Koldehofe, Charles Leska, Guido Rößling, Wanda Dann, Ari Korhonen, Lauri Malmi, Jarmo Rantakokko, Rockford J Ross, Jay Anderson, Rudolf Fleischer, Marja Kuittinen, and Myles McNally. Evaluating the educational impact of visualization. In *ITiCSE-WGR '03: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 124–136, 2003.

[57] T.L. Naps. *Introduction to Data Structures and Algorithm Analysis.* West Publishing Co., St. Paul, MN, USA, second edition, 1992.

[58] T.L. Naps. Jhavé: Supporting algorithm visualization. *IEEE Computer Graphics and Applications*, 25:49 – 55, September 2005.

[59] T.L. Naps, S. Cooper, and twelve more authors. Evaluating the educational impact of visualization. In *ITiCSE-WGR '03: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 124–136, 2003.

[60] T.L. Naps, M. McNally, and S. Grissom. Realizing XML driven algorithm visualization. In *Proceedings of the Fourth Program Visualization Workshop*, pages 1–5, Florence, Italy, June 2006.

[61] T.L. Naps and G.J. Pothering. *Introduction to Data Structures and Algorithm Analysis with Pascal, 2nd ed.* West Publishing Co., St. Paul, MN, USA, 1992.

[62] T.L. Naps and G. Rößling. JHAVÉ – More Visualizers (and Visualizations) Needed. In *Proceedings of the Fourth Program Visualization Workshop*, pages 112–117, Florence, Italy, June 2006.

[63] T.L. Naps, G. Rössling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J.A. Velázquez-Iturbide. Exploring the role of visualization and engagement in computer science education. In *ITiCSE-WGR '02: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 131–152, 2002.

[64] T.L. Naps, G. Rössling, and nine more authors. Exploring the role of visualization and engagement in computer science education. In *ITiCSE-WGR '02: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 131–152, 2002.

[65] N. Neff. Problem-directed discrete structures course. In *Fourty-first SIGCSE Technical Symposium on Computer Science Education*, pages 148–151. SIGCSE, March 2010.

[66] ACM/IEEE-CS Joint Task Force on Computing Curricula. Computer science curricula 2013. Technical report, ACM Press and IEEE Computer Society Press, December 2013.

[67] G.J. Pothering and T.L. Naps. *Introduction to data structures and algorithm analysis with C++.* West Publishing Co., 1995.

[68] 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, New York, NY, USA, 2013. ACM.

[69] Markup syntax and parser component of docutils. `http://docutils.sourceforge.net/rst.html`, 2013.

[70] S. H. Rodger. Jflap web site. `www.jflap.org`, 2013. [online; accessed 5-May-2013].

[71] S.H. Rodger and T.W. Finley. *JFLAP – An interactive formal languages and automata package.* Jones & Bartlett Learning, 2006.

[72] Susan Rodger, Melissa Dalis, Chitra Gadwal, Jenna Hayes, Peggy Li, Liz Liang, Francine Wolfe, and Wenhui Zhang. Integrating computing into middle schools disciplines through projects. In *Fourty-third SIGCSE Technical Symposium on Computer Science Education*, pages 421–426. SIGCSE, March 2012.

[73] Susan H. Rodger, Jinghui Lim, and Stephen Reading. Increasing interaction and support in the formal languages and automata theory course. In *The Twelfth Annual Conference on Innovation and Technology in Computer Science Education*, pages 58–62. ITiCSE, June 2007.

[74] Susan H. Rodger, Henry Qin, and Jonathan Su. Increasing the use of jflap in courses. In *Sixth Program Visualization Workshop*, pages 53–56. PVW, June 2011.

[75] Susan H. Rodger, Eric Wiebe, Kyung Min lee, Chris Morgan, Kareem Omar, and Jonathan Su. Increasing engagement in automata theory with jflap. In *40th SIGCSE Technical Symposium on Computer Science Education*, pages 403–407. SIGCSE, March 2009.

[76] R.J. Ross. Hypertextbooks and a hypertextbook authoring environment. In *ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 133–137, Madrid, Spain, 2008. ACM.

[77] G. Rößling, T. Naps, M.S. Hall, V. Karavirta, A. Kerren, C. Leska, A. Moreno, R. Oechsle, S.H. Rodger, J. Urquiza-Fuentes, and J.A. Velázquez-Iturbide. Merging interactive visualizations with hypertextbooks and course management. In *ITiCSE-WGR '06: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, pages 166–181, 2006.

[78] G. Rößling and T.L. Naps. A testbed for pedagogical requirements in algorithm visualizations. In *ITiCSE '02: Proceedings of the 7th annual conference on Innovation and technology in computer science education*, pages 96–100, New York, NY, USA, 2002. ACM Press.

[79] G. Rößling and T. Vellaramkalayil. First steps towards a visualization-based computer science hypertextbook as a moodle module. In *Proceedings of the Fifth Program Visualization Workshop, PVW'08*, volume 224 of *Electronic Notes in Theoretical Computer Science*, pages 47 – 56, 2009.

[80] L. Segedin, J.V Ernst, and A.C. Clark. Transforming teaching through implementing inquiry: A national board-aligned professional development system. In *Proceedings of the Association for Career and Technical Education Research*, 2013.

[81] C.A. Shaffer. *Data Structures and Algorithm Analysis.* Dover Publications, third edition, 2011.

[82] C.A. Shaffer. Data structures and algorithm analysis, Edition 3.2, 2011. Published online at `http://people.cs.vt.edu/~shaffer/Book/`.

[83] C.A. Shaffer, M. Akbar, A.J.D. Alon, M. Stewart, and S.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, 2011.

[84] C.A. Shaffer, M.L. Cooper, A.J.D. Alon, M. Akbar, M. Stewart, S. Ponce, and S.H. Edwards. Algorithm visualization: The state of the field. *ACM Transactions on Computing Education*, 10:1–22, August 2010.

[85] C.A. Shaffer, M.L. Cooper, and S.H. Edwards. Algorithm visualization: A report on the state of the field. In *SIGCSE '07: Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, pages 150–154, March 2007.

[86] C.A. Shaffer, V. Karavirta, A. Korhonen, and T.L. Naps. OpenDSA: Beginning a community hypertextbook project. In *Proceedings of the Eleventh Koli Calling International Conference on Computing Education Research*, pages 112–117, Koli National Park, Finland, November 2011.

[87] C.A. Shaffer, T.L. Naps, and E. Fouh. Truly interactive textbooks for computer science education. In *Proceedings of the Sixth Program Visualization Workshop*, pages 97–103, Darmstadt, Germany, June 2011.

[88] C.A. Shaffer, T.L. Naps, S.H. Rodger, and S.H. Edwards. Building an online educational community for algorithm visualization. In *Proceedings of the 41st ACM technical symposium on Computer science education - SIGCSE'10*, pages 475–476, Milwaukee, Wisconsin, USA, March 2010.

[89] David B. Sher. A visual proof for an average case of list searching. *SIGCSE Bull.*, 40(2):74–78, June 2008.

[90] B.F. Skinner. The science of learning and the art of teaching. *Harvard Educational Review*, 24:86–97, 1954.

[91] J. Sweller. *Instructional Design in Technical Areas.* ACER Press, 1999.

[92] Hussein Thompson and Pranay Chadhuri. An alternative visual analysis of the build heap algorithm. *ACM Inroads*, 2(3):31–32, August 2011.

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

[94] Jaime Urquiza-Fuentes and J. Ángel Velázquez-Iturbide. A survey of successful evaluations of program visualization and algorithm animation systems. *Transactions on Computing Education*, 9:9:1–9:21, June 2009.