Requirements and Design Strategies for Open Source Interactive Computer Science eBooks

Ari Korhonen, co-chair
Aalto University
ari.korhonen@aalto.fi

Thomas Naps, co-chair
U. of Wisconsin Oshkosh
naps@uwosh.edu

Charles Boisvert
Sheffield Hallam University
C.Boisvert@shu.ac.uk

Pilu Crescenzi
University of Florence
pierluigi.crescenzi@unifi.it

Ville Karavirta
Aalto University
ville@villekaravirta.com

Linda Mannila
Åbo Akademi University
linda.mannila@abo.fi

Bradley Miller
Luther College
bmiller@luther.edu

Briana Morrison
Georgia Inst. of Tech.
bmorrison@gatech.edu

Susan H. Rodger
Duke University
rodger@cs.duke.edu

Rocky Ross
Montana State University
ross@cs.montana.edu

Clifford A. Shaffer
Virginia Tech
shaffer@cs.vt.edu

ABSTRACT

Online education supported by digital courseware will radically alter higher education in ways that we cannot predict. New technologies such as MOOCs and Khan Academy have generated interest in new models for knowledge delivery. The nature of Computer Science content provides special opportunities for computer-supported delivery in both traditional and online classes. Traditional CS textbooks are likely to be replaced by online materials that tightly integrate content with visualizations and automatically assessed exercises. We refer to these new textbook-like artifacts as icseBooks (pronounced "ice books"), for *interactive computer science electronic books.* IcseBook technology will in turn impact the pedagogy used in CS courses. This report surveys the state of the field, addresses new use cases for CS pedagogy with icseBooks, and lays out a series of research questions for future study.

Categories and Subject Descriptors

[K.3.1 Computer Uses in Education] Computer-managed instruction, Computer-assisted instruction

General Terms

Design, Human Factors

ITICSE'13 Working Group Reports, June 29–July 3, 2013, Canterbury, UK.

Keywords

Algorithm Visualization, Interactive eBook, Hypertext, Automated Assessment, Digital Education

1. INTRODUCTION

There is a widespread sense that fundamental change is coming to education, even if there is not much consensus yet as to how that change will play out. New internet-based technologies are one driver for this change. Massively Open Online Courses (MOOCs) are this year's most visible face of technology-driven disruption, with new forms of interactive content delivery such as Khan Academy and Code Academy also playing an important role. In this report, we focus on a particular combination of internet technologies that have the potential to especially impact Computer Science education. This is a fusion of visualization, multimedia, and automated assessment that should greatly change the notion of "textbook" and "course materials" in the future, and which has the potential to fundamentally change our approach to CS pedagogy.

The title for this report reflects this emerging fusion. It comprises five notions familiar to all computer scientists — user requirements, design strategies, open source, interactivity, and eBooks. But not all computer scientists have the same perspective on what these terms mean. We clarify below what we mean by the convergence of these five terms. Previous efforts in similar areas have focused on the integration of interactive content with a learning management system [36, 37]. The focus of this research is different, as it targets eBooks, requiring different approaches – and answers – even to similar challenges. To give a shorthand way to refer to the artifacts emerging from this fusion, we coin the phrase *interactive computer science eBook*, hereafter shortened to icseBook and pronounced as "ice book". We feel that there is value to coining a unique name to reinforce in

the reader's mind that we emphatically mean artifacts that go far beyond the text and figures that one typically gets when today's eBook is downloaded from a service like Amazon or Safari. ePub or PDF content is static in the sense that, once received, it offers the reader little chance to customize or interact with it in significant ways. Our concept of an icseBook requires that the content be highly interactive. The icseBook must include, in addition to textbook-quality presentation of content, some or all of the following:

- Interactive visualizations (*i.e.*, the user drives the pacing, and perhaps some aspects of the content)
- Automatically assessed self-quiz exercises, interspersing frequent practice with meaningful feedback so that learners can gauge their own knowledge
- Interactive learning activities appropriate for CS-specific concepts, such as Algorithm Visualizations (AVs), models of computation such as finite state automata, and editing and execution of code segments within the icseBook itself
- Instructor customization that allows selection and ordering of learning modules

While all of the components listed above already exist, they usually exist in isolation. Consequently a learner is required to move between applications to use them. Additionally, developers of the individual components (e.g., traditional textbook authors and algorithm visualization developers) mostly work without coordinating their efforts. Most importantly, there is a synthesis to be gained from tightly integrating content, visualization, and practice exercises. In the icseBook, we no longer think of the interactive components as something that externally supplements the content, and we no longer think of assessment in terms of "homework exercises" that give summative evaluation of the learners' state of knowledge. Instead all of these components collectively comprise the "textbook".

Prototypes for the icseBook are beginning to emerge. Two important examples are referred to frequently within this report: the OpenDSA project[1] [15, 23, 41] and the Runestone Interactive Python project[2] [27]. We encourage you to explore these prototypes before reading the entire report because they illustrate what we describe here much better than any static textual description could. We provide an online version of this report for easy access to such materials[3].

In this report we raise issues that relate to building and deploying the infrastructure (both software and social infrastructure) needed to develop and sustain icseBooks. While tentative solutions to some of the issues are offered by ongoing projects, most of these remain open problems. Thus, Section 5.3 contains a research agenda for the future.

Creating a good icseBook will require far more effort than production of the equivalent content in a traditional textbook. We recognize five stakeholder groups in successful icseBook development — learners, instructors, icseBook content developers, educational researchers, and icseBook infrastructure software developers. Although we hope learners will ultimately be the primary beneficiaries from icseBooks, they are not the audience for this report. We address the issue of requirements in Section 2 by reporting on feedback gathered from a survey of instructors who might potentially use an icseBook. In Section 3 we focus on the instructor as stakeholder by addressing how icseBooks can be used effectively in teaching. Section 4 attends to content developers by describing the types of content required, and how the tools offered by an icseBook development platform should facilitate content creation. Section 5 presents issues related to evaluating an icseBook and its supporting software. We hope that this report can contribute to establishing a broad research agenda that will spur a variety of collaborations between instructors and educational researchers.

Sections 2 through 5 describe *what* an icseBook platform must do to serve the needs of its stakeholder groups. As such, they provide a set of user requirements for the last stakeholder group — infrastructure software developers. In Section 6, we switch our perspective from *what* to *how* by discussing high-level design patterns to guide the development of the software underlying icseBook platforms. Our goal is to facilitate reliable and interoperable software infrastructures for icseBooks, since no single development effort is likely to solve all of the problems involved.

The final term to parse out of our report's title is "open source". We believe that both the instructional content and the infrastructure software for developing icseBooks can benefit from open source distribution strategies. Particularly in Section 4.4, we offer insight into how open sourcing the instructional content of an icseBook might attract a large group of dedicated content developers to an open source icseBook project.

2. SURVEY RESULTS

During May 2013, we developed a survey to elicit opinions and ideas concerning icseBooks from the CS teaching community. An invitation to participate in the online survey was sent to the SIGCSE listserv. We received 75 responses of which 65 participants completed a majority of the questions. The survey consisted of 21 questions broken into four sections: general questions, teaching issues related to an icseBook, instructor customization, and feature requirements. Here we give results for general questions. Results for other questions are discussed in more detail in other sections of the report. The questions consisted of both open-ended (Q4, Q11, Q14, and Q21) and closed-ended questions (the rest). It must be stressed that the respondents to this survey were a self-selecting (and thus an unrepresentative) subset of the SIGCSE listserv, which is in turn a self-selecting subset of the CS education community.

Q1. What is your discipline? (Check all that apply.) Of 75 responses, 93% checked themselves in Computer Science, but also in Software Engineering (18%), Information Science (9%), Information Technology (7%), Mathematics (12%), and/or Computer Game Design and Development (8%). The survey used the term "e-textbook" and defined it to mean the same as an icseBook.

Q2. If you have used an e-textbook for a course, which course(s)? (check all that apply) Of 46 responses, 61% indicated that they have used an e-textbook in a basic course such as CS0, CS1, and Data Structures. 65% had used an e-textbook in some other course, with 17 distinct courses listed. These cover most of the standard computing courses along with various special topics such as Ethics and the Philosophy of Technology.

Q7. Do you plan to teach with an e-textbook in the next two years? Of 72 responses, 36% strongly agree,

[1] `http://algoviz.org/OpenDSA`
[2] `http://interactivepython.org`
[3] `http://icsebooks.appspot.com`

39% agree, 21% are neutral, 4% disagree, and no respondent strongly disagreed.

Q8. For what course would you most like to see interactive e-textbook resources developed? Of 69 responses, 17% indicated breadth-first intro to computing (CS0 in the ACM Curriculum), 33% said intro programming (CS1 in ACM Curriculum, a first programming course), 19% said data structures, and 30% said "Other". Among the 21 respondents who indicated "Other" there were 13 distinct courses listed, spanning the breadth of the CS curriculum.

Q9. For the class you would most like to use an interactive e-textbook, how do you use the current textbook? Of 71 responses, 7% indicated they do not use a textbook, 16% used the textbook for reference only (no assigned readings or problems), 32% assigned readings, 4% assigned problems, 39% assigned readings and problems, while one respondent said that the book was used for reference, readings, and problems.

Q10. In what way(s) do you see yourself using an interactive e-textbook within a class? (Check all that apply.) Of 73 responses, 1% indicated that they did not see themselves using an interactive e-textbook, 29% would use the interactive e-textbook in the same way they use a regular textbook, 73% would assign readings, 73% would assign problems, 59% would use examples during class or lab, 53% would have students use the interactive exercises during class or lab, and 66% would use the data on student use of the e-textbook (problems attempted, solved, etc.).

Q11. What are the main factors that would affect your decision to adopt or not adopt an interactive e-textbook? There were 62 free-response answers. The most common responses concerned the content (17 responses), quality (15), and cost (25) of the icseBook. Usability of the icseBook (7) and hardware requirements (platform availability) (6) were also mentioned. How difficult the icseBook is to customize would also play a role for six respondents. Six responses indicated that how well the automated assessments are done, including the quality and amount of feedback to students (2), and whether the assessments improved learning (3) would be a factor.

Q12. Would class size influence your decision to use an interactive e-textbook? 89 % of 74 responses indicated that class size would not matter.

3. TEACHING WITH AN icseBook

Class contexts and content for even a "traditional" CS course vary a great deal between institutions and between instructors. The teaching strategies that could be applied when using icseBooks might vary even more. In this section we discuss how icseBooks can be used in various educational contexts. An icseBook integrates a range of materials for a course into one product, which can be customized and adapted by instructors according to their specific preferences and needs. The icseBook provides an all-in-one solution for both learners and instructors that is different from the current state-of-the-art in which many separate learning objects (e.g., algorithm visualizations, exercises, multimedia materials, text) are provided.

Q3 asked **If you have used an e-textbook for a course, which type of e-textbook was it?** For CS0, CS1, and data structures courses, 15 of 26 responses (58%) indicate that some level of *interactivity* is involved, while for "other CS courses" only 3 of 29 responses (10%) indicated that the e-textbook was something more than a static ePub-style artifact. While it is natural that basic programming courses adopt new ways of teaching first (if only because they are abundant) this indicates that our discipline is only beginning to taking full advantage of digital learning materials.

3.1 IcseBook: Evolution or Revolution?

Q5 asked **If you have used an e-textbook for a course, how much of the course (relative to the whole course) was covered by the e-textbook?** Of 47 respondents, 21% indicated that the e-textbook covered 100% of the course, 32% that $\geq$ 80% was covered, 17% that $\geq$ 50% was covered, 13% that $\geq$ 33% was covered, and 17% indicated that $<$ 33% was covered. IcseBooks need be not thought of as incompatible with traditional textbooks. The responses indicate that most instructors use icseBooks along with other materials such as textbooks, handouts, etc.

Q6. If you have used an e-textbook for a course, did it replace an existing resource? Of 47 responses, 30% indicated that it replaced all other materials on one or more topics, 51% that it was a supplemental along with other resources, and 19% that it was the only resource used for the course. Q6 along with Q3 shows that many paper textbooks have also been replaced with electronic but non-interactive equivalents, either by instructors or at the initiative of learners who find them convenient.

The prevalence of blended learning suggests the following progressive route to adopting icseBooks through an evolution of instructor practice that makes increasing use of an icseBook without requiring a complete, big-bang redesign of the course.

1. Substitute some traditional materials with icseBook content
2. Supplement lecture notes with animated visualizations and materials
3. Substitute content completely for the paper textbook
4. Substitute exercises for the old paper exercises
5. Some student activities use learner tracking and grading features
6. Grading and learner monitoring become an essential part of the learner's grade for the course

An evolutionary route is valuable because the big-bang would be too big an investment for many instructors — far more than adopting a new textbook. Big-bang requires a change of assessment strategy, a reflection on the use of the wider material, and a comprehensive change of teaching style. Should it go wrong, there is a lot at stake. Evolution allows the instructor to introduce icseBook material where it suits the course and gradually adopt more interaction.

IcseBooks can change how learners are assessed, as the final grade can be based on results from different activities ranging from automatically assessed self-quizzes to open assignments, projects, and exams. Recent research addresses improving the quality of feedback given in automatically assessed exercises [19] and developing new types of exercises [24]. Automated data collection can relieve the instructor of some grading work (when exercises can be automatically assessed), as well as support the collection, tracking, and human grading of questions that cannot be automatically marked. In large classes, however, assessment of assignments and projects that cannot be automatically corrected becomes challenging, as no individual instructor has

the time needed to grade and give feedback on hundreds of assignments. Some MOOCs address this challenge by incorporating crowdsourcing techniques, such as peer feedback and collaborative work. In addition, MOOCs commonly have many teaching assistants helping with assessment. Automatically collected data aids in monitoring learner activity and progress (see Section 5.2). The icseBook makes it easy for the instructor to follow what learners are doing through automatic logging. For example, instructors can verify whether students have worked assigned sections. Using formative assessment through regular review exercises, icseBooks allow both instructors and learners to follow up on learner progress.

The use of icseBooks in the classroom may lead to new requirements for the online material, resulting in an iterative process where both the teaching strategy and the material gradually evolve. New teaching tools give rise to new pedagogical models and vice versa. IcseBooks provide opportunities to transform instructor practice. Focusing teaching and learning on the student, rather than on the workflow of the teacher, is not new. But it is difficult to implement properly and requires continual attention. Papert, writing on Logo [30], points out the involuntary reversal of the teaching relationship brought about by poorly designed resources in which we see "the computer programming the student." The student should program the computer.

Technologies such as those found in icseBooks, like the Computer-Assisted Instruction described by Papert, do not, of themselves, ensure student-centeredness. But icseBooks are a disruptive technology, in the sense of [11]: they provide access to the integration of instruction and practice materials, to learner usage and achievement data, and to tools for the integration and the management of these resources, at a higher scale and lower cost than has been available until now. This makes it easy to adopt icseBooks in full, and use them to renew our teaching practice, centering more on learner needs.

3.2 Factors Affecting IcseBook Use

Many factors affect how an icseBook might be used in the classroom, lab sessions, or in any other activity appearing in a course.

Number of learners: Class size can vary widely, ranging from small tutorial groups with fewer than ten learners, to large lecture theatres gathering hundreds of learners and the Open Universities teaching thousands at a time. MOOCs have become popular and can, as their name says, gather very large numbers of students.

On-site vs. online: Teaching strategy is affected by where the learning takes place: face-to-face or online. Most university courses require learners to be present for some of the work, but distance learning plays an increasing role, with MOOCs and the Open University models as examples of off-campus learning.

Content vs. skill: A course is commonly positioned along a continuum from being mainly content and knowledge based (introducing new concepts) to being focused on developing skills. In the case of CS, many courses focus on learning skills related to programming as opposed to a knowledge base.

These three factors can be considered fundamental in the sense that they define the practical course situation and more or less prescribe what kind of icseBook can be used. A large online course focusing on teaching a specific programming skill forces the instructor to choose from a smaller set of materials than are available to an instructor working with a small group face-to-face focusing on knowledge.

In addition to these two factors, we can identify three dimensions that are not directly inherent in a given course, but dependent on choices made by the instructor:

Level of use of the icseBook: Just as instructors can choose to what extent they use a regular textbook, the ways they employ icseBooks can vary. An icseBook is more multifaceted and interactive than a traditional textbook. As described in Section 3.1, there is a wide range of use adoption within the course.

Level of learner engagement: Traditional lectures are commonly considered to be teacher-centered, featuring the instructor as the "sage on the stage" and learners in the audience listening while (hopefully) taking notes. The icseBook can support a more learner-centered approach, putting them in increased control of their learning (making the instructor more of a "guide on the side"). As with any teaching resource, the goal is to maximize learner participation in the learning process. This can be done by increasing the learner's level of engagement with, for example, algorithm visualization (see [29]), or increasing content ownership as suggested by Sorva *et al.* [42].

The dimensions identified above seem to be applicable for categorizing teaching scenarios in any discipline. However, the nature of CS both invites and calls for a large set of different types of interactive exercises (writing or re-arranging code, simulating algorithms, working out solutions to analysis problems). Such CS-specific activities are covered in detail in Section 4.1.1.

3.3 Use Cases

The graph in Figure 1 maps two of the factors discussed above to illustrate some typical teaching practices. The horizontal axis shows the face-to-face vs. online dimension and the vertical axis represents the number of learners. Teaching practices have been roughly divided into three main categories depending upon where on the axes the activities in a class are positioned: *face-to-face learning*, *blended learning* or *online learning*.

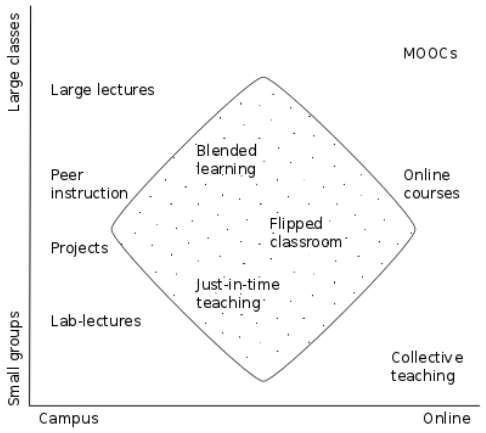

Figure 1: Two dimensions of icseBook use cases

This section describes the use cases found in the figure and discuss the use and benefits of an icseBook in each context. The use cases are not intended to comprise a complete list

of possible teaching approaches, but rather to give ideas for how the icseBook can be used depending on where a given CS class is positioned on the axes.

3.3.1 Face-to-face teaching

IcseBooks can naturally be used for the same purposes as a normal textbook in campus-based teaching scenarios, such as lectures or project-based courses. When it comes to covering content, an icseBook clearly provides learners with a larger selection of instructional resources and activities as compared to a regular textbook. In an icseBook on programming, data structures, or algorithms, material such as algorithm visualizations, programming exercises, and interactive examples of stepping through code offer the learner a larger toolbox both for studying new concepts and reinforcing content already covered.

Another benefit of icseBooks is that they are customizable. For example, the instructor could modify the contents by changing the order of topics, adding specific examples, or controlling the amount of credit given for various exercises. As a lecture aid, the icseBook can be used as the primary lecture material or mainly for showing examples, visualizations, and videos.

With small groups of learners it is possible to integrate lab work into class meetings — for instance by the instructor spending part of the time going through new material while the remaining time is used for hands-on activities in a computer lab. In this lab/lecture blend, the icseBook can be used both by the teacher (showing examples and other material during the lecture) and the learners (working on visualizations and review questions during the lab).

Peer instruction (PI) [34] is a popular interactive teaching strategy focusing on activating students during lectures and creating situations where they can learn from each other. A common way of implementing this scenario is by presenting students with simple multiple-choice questions which they answer using clickers or polling software.[4] The idea is for individual learners to first answer the question on their own, then discuss the question in pairs or small groups, and finally answer the same question again. When everybody has answered the question twice, the instructor shows the results from the two rounds, inviting a class-wide discussion. In PI the learning results become visible, as the results from the two rounds of answering the same question may illustrate a dramatic change in opinion. With an icseBook there is no need for clickers or third party software, and PI can easily be implemented using self-quiz exercises. To accomplish this the same question appears twice and the instructor uses data tracking so that both answer rounds can be stored and displayed to the learners.

3.3.2 Blended learning

Many teaching scenarios involve some kind of blended (or hybrid) model of learning with a combination of face-to-face teaching methods along with computer-mediated or online activities. The aim is to extend the learning opportunities by using online materials such as icseBooks. Here we describe teaching strategies using online material and activities that are suitable even for small class sizes.

Just-in-time-teaching[5] (JITT) refers to a teaching strategy based on student activity outside of lecture hours, helping the instructor prepare the following lecture to suit the needs of the learners. It has been used in Computer Science, as described in [3]. The self-quiz exercises included in icseBooks are automatically corrected, making it easier and faster for the instructor to go through the results. Based on the findings, material is adapted for the upcoming class meeting "just-in-time". This is an advantage over the normal teaching situation, where the instructor must intuit which topics are difficult for the learners. One respondent in our survey described this kind of scenario where interactive exercises in an icseBook were used. At the beginning of each lecture learners were called on (randomly) to present their solutions to the rest of the class. Depending on the types of exercises used, the amount of time needed for an instructor to get an overall picture of the results will vary. Class size might be a restricting factor in determining when this approach can be used. Making automatically corrected exercises part of the grade can also motivate learners to take the pre-class activities seriously.

Similar to the JITT model is the **flipped** (or inverted) classroom, which is rapidly gaining in popularity. In a flipped classroom, content delivery is moved outside the classroom, and contact hours are used for working on what has traditionally been considered homework (e.g. problem solving, labs, discussions, and other types of creative work). The model is not new as it has been used substantially in the humanities, where students have been assigned readings as homework while class time has mainly been used for discussions and debates. Compared to having students read plain text material, an icseBook makes the self-study material interactive and multimodal by integrating text, visualizations, videos, and automatically corrected review exercises. Learners are expected to view the material and take the quizzes before class, and technology makes it easier to hold them accountable for actually doing so. The flipped classroom can be used for classes of various sizes, but the number of students might restrict what can be done during in-class sessions. If the instructor wants to include hands-on activities that require personal assistance, the group size naturally becomes more restricted. With small classes, the pre-class exercises (such as provided by OpenDSA) make it easier for the instructor to prepare for in-class discussions.

In blended learning, interactive icseBooks give new ways of aligning classroom sessions with online learning in large courses. Some lectures, for example, can be replaced by short videos appearing in online material. The face-to-face classroom session can then be used to discuss the questions emerging from the online material just as in JITT, or the students can work in small groups as in PI. The richer the content available in the icseBook, the more numerous the ways that the blended learning model can be applied. Digital material is no longer a supplementary resource, but becomes the main mediator for the content. In this way blended learning approaches MOOCs, which lack instructors in face-to-face sessions (or even lack instructors entirely).

Typically large courses are supported by a learning management system (LMS) such as Moodle[6] or a dedicated learning environment such as TRAKLA2 [24][7]. In addition to learning materials, an LMS can provide features such as on-

[4]For examples of Peer Instruction in CS, see `http://www.peerinstruction4cs.org`

[5]`http://www.jitt.org`

[6]`https://moodle.org/`

[7]`http://www.cse.hut.fi/en/research/SVG/TRAKLA2/`

line discussion forums and extended collaboration possibilities among learners. There are many overlapping features in an LMS and an icseBook. TRAKLA2 is an example of a web-based learning environment that is a subset of an icseBook. TRAKLA2 gives learners the opportunity to work online 24/7 and get automatic summative feedback for their solutions to algorithm simulation exercises. An instructor has the opportunity to monitor students' performance. In addition, the learner benefits from mistakes due to the automatically assessed exercises and instant feedback. The exercise can be tried as many times as required.

3.3.3 Online learning

Online courses are at the other end of the spectrum from face-to-face on-campus teaching. Early versions of online courses were basically digital versions of traditional distance (correspondence) learning courses. But as technology has progressed the content, methods, and activities have evolved. Especially in CS, the feedback provided to the learner can be automated in many ways (multiple choice questions, algorithm and program simulation exercises, programming exercises, etc.), which makes it an attractive approach.

Lately MOOCs have received much attention, with the New York Times dubbing 2012 as the year of the MOOC. Compared to earlier online courses, MOOCs aim at large-scale enrollment and are open to everyone, resulting in popular courses offered by Coursera[8], edX[9] and Udacity[10] having tens and even hundreds of thousands of enrolled learners. This naturally creates new challenges with regard to grading, monitoring, and interaction. Most MOOCs are built around video lectures with integrated (multiple choice) review questions, quizzes, peer-reviewed assignments, and discussion forums. The main criteria is that everything must be scalable. An icseBook extended with communications possibilities and supported by a sufficiently large number of instructors or teaching assistants could someday form the basis of a MOOC.

While MOOCs are geared toward large numbers of students, collaborative teaching and connected courses are a way of providing high-quality learning material for small groups of students by combining the efforts of faculty in different institutions. This type of teaching might be applicable in situations where instructors must teach a topic that is compulsory only to a small number of learners. By creating a joint course with other institutions teaching the same topic, we can benefit from economies of scale that make small local versions of a course more feasible. By bringing together learners from several institutions, the course may become large enough for it to be worthy of continuous development. For example, CISCO Systems distributes its professional materials, which include illustrated text, multiple-choice self-quizzes, and simulation software, to many local training centers that teach small numbers[11]. A fuller icseBook lends itself as a platform for mediating material, allowing for learner-learner collaboration and many more functions.

[8] https://www.coursera.org/
[9] https://www.edx.org/
[10] https://www.udacity.com/
[11] http://www.cisco.com/web/learning/training-index.html

3.4 Summary

While many of the components that form an icseBook are well known, their association into an integrated whole enables a transformation of instructor practice. The blend of factors in an icseBook will lead to instructional strategies that differ greatly from a traditional lecture-based course.

This brings many additional questions to be addressed. How should the artifacts in an icseBook be developed and deployed in a class? They will help enable instructors to channel learner behaviors in ways that facilitate learning outcomes; but what behaviors and outcomes are we trying to support? How are exposition, visualization, activity, test, and data collection best assembled? How are they evaluated? The next sections discuss how the tools of a teaching revolution can be created and honed.

4. CONTENT CREATION

IcseBook content developers might create content in a variety of supported mediums: text, pictures, video, audio, algorithm visualizations, self-quiz questions with automated feedback, in-book editing/execution of code segments, algorithm performance studies, algorithm simulation exercises, and other forms of automatically assessed exercises. Contributions might come in the form of brand new content or content that is modified (perhaps slightly, perhaps greatly) from that of another contributor.

Another perspective on contributors is their motivation for contributing. Their willingness to contribute to an icseBook project will be influenced by many factors, including the difficulty encountered in making the contribution, the credit they will receive, and size of the audience they are likely to reach.

4.1 Content Contributors

4.1.1 Types of content contributions

We describe the particular relevance for different types of content within the icseBook context.

Text will appear in an icseBook for the same reasons as it appears in a traditional textbook: to present the topic, the examples, the exercises, and their solutions. However, an icseBook provides more opportunities to enhance the text by integration with other media. For example, the OpenDSA project attempts to integrate text with graphics at a fine-grained level through the frequent use of "slideshows" to present topics. Whereas a typical textbook might use a paragraph of text to describe the data members in a list class implementation, OpenDSA modules might replace such text-only descriptions with slideshows that show code along with the text such that each sentence might trigger a particular line of the code to be highlighted. Good algorithm visualizations routinely juxtapose small amounts of text with each visual step of the algorithm and a highlighted line of the associated code. An alternative to such text snippets is **audio narration** in synch with the visualization.

Traditional textbooks also contain **pictures**. While these could appear in an icseBook as a simple picture, they could also be integrated with the text as just explained, or be made interactive. An example of an interactive picture is an HTML image map in which various areas define clickable "hot spots" within the image. Different hyperlinks are associated with each such hot spot.

A **video** might be included as an aid for explanations and examples. Videos might show an instructor explaining a topic or an example, or contain an algorithm animation/visualization. For example, the video could show an instructor's regular lecture with visuals of the instructor and his/her slides. The video could also be a screen capture of the instructor using algorithm or program visualization software to explain an algorithm, and possibly at the same time demonstrating how to use the software to create or explore a concept [4]. Videos might include embedded questions that automatically pause the video with a question that the learner must answer before the video will proceed. Alternatively, videos might be short, and interactive questions might be embedded in the icseBook after each video. Work presented by Mayer [25] suggests that individual videos should be kept short (less than a minute or perhaps only a few seconds). A longer presentation can satisfy this goal by breaking the video into pieces. An important key is to maximize user interaction, which is why projects such as OpenDSA stress slideshows (perhaps with narration) over videos. Audio in such cases can also increase accessibility of the icseBook for visually impaired students.

Algorithm and Program visualizations (AV and PV) provide visual explanations for dynamic processes. An AV might just explain the algorithm, but it might also involve the learner in stepping through an algorithm or constructing a structure. Similar to this, a PV might provide a visual debugger to step through code. Another approach is to use visualization to focus on an algorithm's development from a partial to a more complete solution [6]. Pacing for the steps of AV and PV should be controlled by the user [39] rather than be an uncontrolled animation.

To make this more engaging, the learner should not only follow the animation, but become an active party in creating the steps. For example, a learner might indicate an understanding of depth-first binary tree traversal by clicking on the nodes in the order that they should appear in a preorder traversal. Such an activity is referred to as either an **algorithm simulation** or an **algorithm proficiency exercise** [23, 24]. In a program simulation exercise [43], students simulate the execution of a program. Moreover, as an aid to understanding the proof that any nondeterministic finite automaton (NFA) can be converted into an equivalent deterministic finite automaton (DFA), the learner might specify an NFA and then follow the steps of the proof to construct the equivalent DFA.

Program exercises involve code editing and execution within the icseBook. They allow learners to experiment and get immediate feedback on the code's syntax and output correctness. The learner should be able to try different inputs. For example, the learner might want to learn how selection sort works. The icseBook can explain the concept with text and/or visualization and then have the learner enter some implementation for all or part of the sort. The icseBook would first let the learner know if there was a syntax error in their code and provide feedback for correcting such errors. When there are no syntax errors, the icseBook would then generate appropriate feedback. The feedback could be textual, or it might be graphical in the form of a **program visualization** for the student's code.

When **questions** and **exercises** are integrated throughout the icseBook they provide the learner with feedback that can indicate whether the learner is prepared to move on to the next module. If not, well-designed exercises provide more practice with the current concept by generating new problem instances. While the icseBook authoring framework might provide direct support for certain question types (such as multiple choice questions), it is probably best if questions and exercises are created as first-class independent components and then assembled along with the rest of the module content as appropriate. The Khan Academy Exercise Framework[12] is an example of such a sub-system that can create questions as separate HTML pages that can then be integrated within an icseBook. An alternative is to include questions within some other component such as an algorithm visualization or Parsons' puzzles. In **Parsons' programming puzzles** [31], students interactively piece together small programs or subroutines from code fragments. A single draggable fragment may contain one or more code lines. Some subset of the given fragments comprise the entire problem solution.

4.1.2 Originality of contributions

We distinguish three types of contributions to an icseBook: creations, modifications, and customizations. **Customizations** involve the least amount of effort. These are defined to be changes for which the icseBook platform provides explicit support. For example, a GUI could be provided that allows an instructor to select the modules that will make up an icseBook instance to be used as the course textbook. This GUI might also allow the instructor to select which particular assessment exercises will appear in a given module, and how much credit it is worth in the semester grade. Customizations can be considered a contribution when a recognizable artifact results. For example, the OpenDSA project controls the definition for an instance of an icseBook through use of a *configuration*. A configuration controls the contents (i.e., the modules), which exercises are included in the various modules, and the point values for the various exercises. In other words, a configuration provides a specification for creating a specific "textbook" instance. A given configuration can be stored, reused, or modified, and as such might prove valuable to other instructors.

Creations are at the opposite end of the contribution spectrum. Here a content developer will create a completely new icseBook content artifact. Depending on what specific artifact types are supported, this might be an entire module, an algorithm visualization, an interactive exercise, or a multiple choice question.

Modification involves taking an existing artifact and changing it to produce a new artifact. For example, a developer might start with the code for a depth-first search AV and modify it to produce a breadth-first search AV. Or he/she might make a small change such as taking the code for a depth-first search AV and changing which colors are used, or the wording of messages displayed during processing.

Contributions might require more or less technical skill, and therefore might be more or less practical for various community members to make. Generally speaking, things like modifying the text or making any configuration changes should be easy to accomplish. Certain changes to other artifacts like changing colors in a visualization or writing a multiple choice question might be easy or hard, depending on the support provided by the icseBook platform. Program-

[12] `https://github.com/Khan/khan-exercises`

ming a significant change in an AV or algorithm simulation exercise might be technically demanding.

Rich customization of existing material requires that the icseBook software platform provide a set of customization tools. In contrast, modification of existing resources and creation of new resources places the focus more on providing clear documentation about how to achieve the modification/creation. When programming is required, a well-designed API must be provided. In Section 4.4 we discuss in more detail the requirements for an icseBook software platform to facilitate a variety of contributions.

4.1.3 Stakeholders

The stakeholders in creating the content of icseBooks are mainly the content developers. A content developer is likely to also be a stakeholder in another way, most typically an instructor whose initial motivation is to create content for the learners in his/her course. Another example of a content developer might be a learner who wants to generate content for a limited group of fellow learners.

4.2 Survey Results on Content Creation

Our survey included questions related to content creation and contribution. Respondents were positive about tailoring an icseBook to their own needs and expectations through creation of their own content and exercises (**Q15**). 35% said they would definitely do this, 46% would likely do it, 13% might depending on the complexity of the changes, only 6% said they would probably not do it, and none said definitely not. Respondents were also positive when asked about whether they would share contributions they had made (**Q16**). 41% would definitely share their contributions and 47% were somewhat likely to share their contributions. 4% said they would probably not share, and 7% said they might but it depended on the nature of the contribution. None said they definitely would not share.

Respondents were given possible content choices for an icseBook and asked to rank the items they would most likely want to tailor to their needs (**Q13**). Three of them appeared to be particularly popular: "selecting the desired subset", "examples", and "programming exercises". Three more choices generally appeared in the middle range of interest: "add quizzes and/or assessments", "add content to existing chapters", and "provide new data sets". Finally, three generated less interest by respondents: "adding chapters", "re-ordering chapters", and "adding videos".

Respondents were mostly positive regarding the possibility of student contributions to an icseBook they were using in a course (**Q17**). Students might contribute examples or test data and then other students could view those examples or data. 35% of respondents said that they would definitely use student contributions, 41% were somewhat likely to use student contributions, 24% would probably not use student contributions, and none said definitely not.

4.3 Facilitating Contributions

4.3.1 Underlying dependencies in modules

The icseBook platform does not simply manage a huge "bag" of unrelated instructional resources. Module topics naturally have other modules as prerequisites, and so the collection of modules can be thought of as forming a dynamic network of learning modules (NLM). Dynamicity comes from the new contributions. Both modification of existing content and creation of new content can potentially modify the topology of the network. This is obvious when creating a new module since the author creates both a new node of the network and new edges specifying which learning modules present in the NLM are prerequisites of the new module. Modification of existing content can also change the structure of the NLM. For example, adding content about the details of analyzing Quicksort to an existing Quicksort module might require that recurrence relations become a prerequisite of the revised module. After performing all desired creations and modifications within the NLM, the authors can finally "instantiate" the sequence of learning modules that define an instance of an icseBook.

The dynamic NLM described above defines the *conceptual* prerequisite dependencies among learning modules. This exists in parallel with another dynamic NLM defining a dependency related to the code snippets, AVs, and exercises shared among the modules. For example, some implementations for Prim's Minimal Cost Spanning Tree algorithm make use of heaps, while others do not. Such re-use of a segment of code or of a visualization creates a *procedural* dependency between the learning modules, thus giving rise to another NLM, different from the conceptual one. Note that the two networks are incomparable. The existence of this procedural NLM means that the icseBook platform must demand content developers be explicit in defining the API of the code they use in developing artifacts such as visualizations, thereby allowing that code to be re-used in future contributions.

4.3.2 Look-and-feel

An icseBook platform should take into account how much the authors are required to respect the "standard" look-and-feel of the learning modules. This is not an easy question. On the one hand, consistency among the look-and-feel of the modules helps learners become familiar with placements and representation of the instructional resources. An inconsistent look-and-feel among instructional resources can lead learners to think that they have left the icseBook they were using. On the other hand, forcing authors to respect a specific look-and-feel can inhibit their willingness to contribute to the icseBook platform if it does not fit well with their preferences. An existing look-and-feel might even cause undesired weaknesses in the usability of a particular type of material. In order to deal with this conflict, the icseBook platform might follow an approach inspired by the *model/view/controller* design pattern [10], which separates the model (instructional resources) from the view (look-and-feel) and thereby allows the authors to develop and integrate new views within the platform itself. Some presentation systems (including Sphinx, which is used by both OpenDSA and Runestone Interactive icseBooks) use the term *theme* to refer to the look-and-feel. It requires significant effort to develop a new theme, but once developed, it can be used by anyone for their modules. It is better if some module can be displayed using different themes with little effort. Even the same icseBook (that is, the same instance of the dynamic network) might be generated easily with a different theme.

- *Learners* might change the look-and-feel of the module they are working on by choosing one specific look-and-feel among available ones. This feature might help

with accessibility problems, such as in the case of visually or aurally impaired learners.

- *Instructors* might define the look-and-feel for an icseBook that they instantiate. They might forbid learners to change the look-and-feel of the modules in order to guarantee consistency in the appearance of all the chapters of the icseBook.
- *Content creators* might set up the look-and-feel of a module they are creating or customizing, and create a new look-and-feel, that is, new views. They might also forbid instructors and learners to change the look-and-feel of the module when the instructional resources included in the module require one specific look-and-feel. For example, theming might be used to ease the conversion of icseBook materials designed for online use into another version designed to be used as course notes (projection slides) during lecture presentation.

4.4 Benefits to Content Contributors

Developing an icseBook is a huge task, far more complex than creating an equivalent paper textbook. Consequently, attracting contributors dedicated to such a project is an important consideration. The likelihood that an instructor (or even a learner) will become a content developer is directly related to the benefit that they see accruing to their efforts in making a contribution. Most contributors are likely to evolve in terms of the contributions they make — starting small and hopefully migrating to more substantive contributions.

How can the icseBook platform encourage such an evolution? In situations where the contributor is an instructor, we envision a two-stage process. Initially, their content will be limited to a group of viewers that the contributor has some control over, most likely students in their class. This will allow them to field test the material being contributed. From a requirements perspective, it means that the software infrastructure must allow contributors to control access to the content. Then, when the contributor feels that the contribution is "publishable" they might wish to make it available to a wider audience.

Dealing with the transition from limited access to "public" dissemination is a fundamental issue that must be dealt with by an icseBook development system. Many models are possible. For example, the Connexions project[13] allows this to happen at the discretion of the author. However, their model is most appropriate to creating complete textbook-level artifacts. For the icseBook, we envision a more granular model of contributions involving alteration to sub-components below the book level. This begs the question of how to treat variations in, for example, single modules, visualizations, or exercises. Some platforms might institute control policies to ensure that their icseBooks, as an entire dynamic network of learning modules, retains a high level of quality. Retaining a high level of quality might naturally go with recognizing contributors for their contributions in a fashion modeled on the traditional academic publication peer review process.

When the contribution is a complete learning module (that has been created new or customized from existing content), the contributor must specify the place that module occupies in the underlying conceptual knowledge map of prerequisites that exist among modules. This is not necessary when the contribution is an artifact that merely plugs into an existing module.

When a particular icseBook is to be created from a large collection of artifacts that are contributed by multiple contributors, there must be a review process to ensure that individual contributions meet the standards that have been established for the icseBook. The managers of a given icseBook development effort might use a formal review process or take advantage of some crowdsourcing methodology. Whatever reviewing methodology is employed, supporting it becomes a user requirement of the software infrastructure underlying the icseBook platform.

Recognition associated with publishing content contributions within an icseBook will likely be directly proportional to the recognition achieved by the entire icseBook. Effective strategies for dissemination and marketing icseBooks will be valuable in attracting a core of dedicated contributors to an icseBook project. Whether the icseBook effort is open source or commercial will affect dissemination and marketing efforts. We focus here on techniques relevant to open source efforts.

Section 5 discusses research questions that will emerge as icseBooks and the platforms that support them evolve. The connection between icseBook development and research is an important consideration because it means that, for those academic contributors and developers who are evaluated based on their research, the standard avenues for disseminating research results will naturally attract them to involvement in icseBooks. If that connection between icseBook development and research is not leveraged, it will be much more difficult to motivitate instructors to become content developers.

Managers of an open source icseBook effort must consider the choice of a copyright under which their content can be cloned and used by others. One nuance in this regard is the blurred line between instructional content and software. Since the instructional content of an icseBook will almost inevitably include Javascript for its interactive components (for example, interactive algorithm visualizations), should stakeholders be thinking in terms of one copyright license that could be used to cover everything — software infrastructure and instructional content — or using separate licenses for the two components? For example, a Creative Commons license would only allow the instructional content of the icseBook to be copyrighted because Creative Commons licenses are not applicable for software source code. On the other hand, licenses such as MIT and GPL would allow everything to be protected under one umbrella license. We also note the curious fact that Creative Commons licenses are not recognized as open source for the purposes of licensing source code, and therefore a project using Creative Commons for its textual component is technically not eligible to use Sourceforge or GitHub as a repository site. If a single umbrella license is chosen, then consideration must be given whether to use a copy-left license such as GPL or a license such as MIT that commercial interests would no doubt find more attractive if they want to leverage the work of the icseBook project. For example, a well-designed platform for the development and the delivery of an icseBook might be of interest to a publisher who would build upon that platform in a fashion analogous to the way that IBM has leveraged the Apache web server or RedHat's leveraging of Linux.

[13] `http://cnx.org`

5. EVALUATION AND CSE RESEARCH

The first question that might come to mind related to assessing icseBooks is: Does more learning occur in classes that use them than in classes that use traditional textbooks? There are a number of difficulties in addressing this question, and perhaps the most important consideration is whether learning gains is the only way to define success for an icseBook. Other concerns that should be factored into a definition of success include retention rates, learning efficiency (time to learn), and resources expended by the instructor or institution [44]. For example, an automated online class without an instructor might be considered a success if it just maintained the learning gains of a traditional class while reducing the per-student resources expended. We might also wish to consider what would be considered a successful learning opportunity for informal learners who would like to study a body of material but are not enrolled in a class.

Perhaps the greatest significance in the long run for education is that using an icseBook might prompt an instructor to radically redesign the learning experience of the class for the better. That is, affordances provided by the icseBook might convince the instructor to change his/her pedagogical approach. This makes evaluation by comparison with the original version of the course difficult. Even now, there is a great range of ways that traditional classes are conducted, both in terms of course content and structure, and in terms of course pedagogy. IcseBooks support potentially greater differences in course pedagogy.

5.1 Evaluation Fundamentals

For now, consider the direct question of how much learning gain comes from a given instructional approach, such as to compare a class that uses an icseBook with one that uses a traditional textbook. Unfortunately, while standard textbook and lecture classes have been in place for many decades, there is little empirical data available to define the learning gains attributed to use of the textbook. Part of the problem is that it is hard to collect data on learning gains associated with traditional textbooks in a natural way. There is no efficient way to accurately measure or evaluate how intensely learners read a traditional textbook, or to capture their fine-grained patterns of behavior (distribution of reading time, etc.) It is not typical practice for instructors to take a pre-test of their students, although one might consider the final exam from the prerequisite course to fulfill this role. So in general we have no formal baseline data for current practice, except when a special educational study has been performed. Fortunately, icseBooks, by their digital nature, provide some advantages to assessment that will be addressed below.

How should we measure learning gain in the first place? Bloom [5] defines the Effect Size of an intervention as

$$\text{EffectSize} = \frac{\text{Average}_{\text{Study}} - \text{Average}_{\text{Conventional}}}{\sigma_{\text{Conventional}}}$$

where we compare the measured outcomes (such as the score on a test) for the study group versus the conventional or control group, as scaled by the standard deviation of the measure on the control group. Typically an effect size over .5 is considered moderate, and effect sizes approaching 1 or more are considered large. Independent of this, we also have to be concerned if the differences are significant (which occurs when there is a large enough absolute size of the effect compared to the standard deviation, where "large enough" depends on the number of subjects involved in the study). For example, a recent study using OpenDSA [15] had a moderate-to-large effect size (.5 standard deviation), but did not have enough subjects (roughly 50 in each section) compared to the standard deviation to reach significance.

One concern is to identify where measured learning gains come from. Do they come from an inherently "better" instructional intervention? Do they come from changing time-on-task (e.g., minor learning gains that come at the cost of compelling students to spend considerably more time)? By investing considerably more resources (e.g., add one-on-one tutoring)? Or are improvements coming simply due to the fact that the new instructional materials are revised to better match learning objectives, something that could have been accomplished in the original control method?

Another issue is the type of knowledge being measured. This might involve improvements in a skill (such as programming), or improvements in knowledge. Within knowledge, we typically recognize a difference between procedural knowledge such as the mechanics for how an algorithm works versus conceptual knowledge such as understanding why the lower bound for sorting algorithms is $\Omega(n \log n)$. Achieving gains for these different types of content might require different types of intervention.

Ideally, we would like our evaluation practices to be efficient, effective, and ethical. All of these dimensions have difficulties. The gold standard is an experimental study, which in education is defined as randomly assigning subjects to control or treatment groups. The next best thing is to take separate, existing sections of a course, or other natural groupings, and use one course section as a control group and another as a treatment group. This is referred to as a quasi-experimental design. It is inferior both because we become less sure that the control and treatment groups are effectively identical, and because the number of differences in the experience of the groups nearly always goes beyond the intervention under study. For example, the sections might have different instructors, the students might be self-selecting based on time of day for the section, or the students might be falling into different groupings because of scheduling constraints from other courses.

Given two interventions, or a control versus an intervention, we have to be concerned that any differences really come from the effect under study, and not from one of a host of other possibilities. Such problems are so prevalent that there are even terms in place for some of the common pitfalls (such as the "Hawthorne effect" [12], which refers to differences that come from novelty or instructor/experimenter enthusiasm for an intervention).

Finally, a number of studies [2, 26, 38] indicate that high quality online education provides the same learning gains as traditional courses. This would seem to bode well for the future of MOOCS. That said, a major problem with MOOCs is that most have limited support for automated assessment. This situation appears to be ideal for the deployment of icseBooks that contain a rich collection of integrated automated assessment materials.

5.2 Data Collection for Assessment

There are a number of ways that data can be collected. In this section we attempt to enumerate sources of evaluation

data, and to indicate the roles that the various sources can take in a comprehensive evaluation process. We are mindful of the fact that there are multiple stakeholders (system developers, researchers, content developers, instructors, learners) and there are multiple levels of artifacts to be evaluated (individual visualizations or exercises, modules, icseBooks, icseBook development systems).

Learner performance data The first-order indicator is data on learner performance gathered through assessment instruments. These instruments might be used as either formative or summative assessments for the learner. These instruments might be test questions or various exercises within the icseBook (completed under test conditions, as homework, or voluntarily completed by the learner) — in other words, the score that a learner receives on each exercise. In principle this is no different from similar information gathered in traditional course settings, though the automated assessment capabilities of icseBooks can make this far more convenient for instructors as well as researchers.

Pre-test/Post-test data The gold standard for measuring learning gains is generally considered to be the pre-test/post-test pair. The differences in the observed deltas between control and treatment groups can then be correlated with the differences in the treatment (if we can accept the large assumption that the control and treatment groups are otherwise identical).

Automated log data One of the most important capabilities that icseBooks bring to the table for use in assessment is their natural potential to support automated logging of primitive user interactions with the system. These primitive, timestamped interactions (button clicks, page loads, page focus events) can then be used to build up semantic units of evaluation, such as "watched a slideshow"' or "completed an exercise." With sufficiently complete logging, higher-level action can also be inferred, such as "did this student skip reading the content and go directly to the exercises?". To answer such questions, it is necessary to make good choices regarding what data to log. A typical software developer reaction is to log everything. This can affect storage requirements, time required to conduct analysis, and privacy concerns.

Log data are only as useful as the analysis tools that the evaluator can bring to bear. At some level, custom tools are typically required in order to interpret the logs, though after an initial round of processing it might be possible to transform raw log data into something that conventional data analysis tools can then work with.

Any data collected from individuals raises privacy concerns. So long as icseBook projects are conducted under the auspices of academic research, they are typically covered by the institutions Internal Review Board (IRB) protocols. This might give icseBook users some reassurance that their privacy is protected, but probably not. When icseBooks are implemented and published by commercial publishers, then all bets will be off regarding privacy, as commercial publishers do not undergo the same scrutiny as academics through IRB review. On the other hand, typical users of the internet routinely expect that their actions at commercial websites are being logged. In any case, learners do understand that if they wish to have their performance interpreted (even if just for themselves), their collected actions must be identified to the extent that they can be linked for assessment (e.g., when they return to the site, they need to know what was previously completed). Learners working with the icseBook for credit must have their progress reported to their instructor.

Survey Data Developers of educational systems often rely on survey data for formative evaluation, and in general to get feedback on user satisfaction and preferences. Surveys can be applied by either the system developers or independent evaluators to any of the relevant stakeholder groups (learners, instructors, and content developers). Unfortunately, it is often the case that user opinions do not correlate well with user performance. This might refer to the usability of the system (just because a user claims to prefer some feature does not mean that he or she will actually use it or be more productive because of it). Of greater concern is the fact that learners are often not good judges regarding how great their learning gains are, or of how to maximize their learning gains [28, 33].

Observational Data Finally, evaluators might wish to conduct observations of stakeholder groups (and in this category we include data collected by conducting interviews). The most likely groups to be observed/interviewed are learners (such as observing behavior of students in classes), or instructors (more likely through interviews than through observation). However, if we do not have a good baseline for what was happening in the classroom prior to an intervention, then we will not have any basis to recognize the effects of the intervention. So ideally, advocates of icseBooks will begin studying the existing behavior in courses even before they begin deployment of their materials in order to build such a baseline.

5.3 An Educational Research Agenda

IcseBooks have the potential for huge impact on CS education. With these new capabilities come a host of potential research questions. This section presents several that we believe are worth investigating. We broadly structure them in the context of the stakeholder groups most likely to benefit from the research (Table 1).

Increasing visual presentation In many cases, material can be presented using a more or less visual approach. For example, variables in a code segment could be explained with a text paragraph, or a visual presentation that gives a sentence of text next to a highlighted line for that variable in the code. This fine-grained integration of visuals with short amounts of text is not generally practical in paper textbooks, but is viable in the icseBook (though it requires more effort to create than simple text). How much (if any) benefit is there from the more visual presentation, and how far can we go with minimizing stand-alone text? For example, OpenDSA attempts to use a "slideshow" approach to increase visual presentation as much as possible.

Visual vs. audio presentation Work surveyed by Mayer [25] suggests that the visual presentation of information is better annotated or explained using audio rather than text. (Note that the literature clearly shows that it is not beneficial to augment visuals with the equivalent audio and text at the same time, as any native speaker forced to watch a movie in their language containing subtitles in that same language can attest.) Mayer's findings would argue for the use of audio to narrate "slideshows" or screen capture presentations. On the other hand, in surveys of students using OpenDSA [15], the students have shown relatively little interest in adding audio. Another aspect to this question is to determine what mode is the most appropriate for different

types of material. It is possible that code segments should be presented differently from algorithm explanations to have maximum effect on the learner. Finally, the most effective length for audio presentation in this context should be studied. It is likely to be far less than the length of a discussion topic such as a sorting algorithm, as prior work [25] has indicated that a series of short segments are better than one long segment.

Presentation modes Given the many different possibilities of how the material might be presented (visual diagrams, slide decks, textual prose, video animations, etc.), we need a better understanding of what constitutes the most appropriate presentation mode for different types of information. The presentation mode may differ for content topics as well as the previous knowledge and experience of the learner. That is, the presentation mode for learners in advanced compiler design might not need the same presentation scaffolding as a beginning programming student.

Length of modules, subcomponents What is the most appropriate length for one learning module? Can the prose span more than one screen length? How much time should it take for a learner to complete one module? Within a module, how long should a single visualization last? For example, OpenDSA has experimented with using a single long "slideshow" for a given algorithm versus a series of short slideshows to present the same algorithm.

Automated assessment We envision exercises built into the icseBook providing immediate feedback to learners as they progress through the text. How will this affect learning? How many of the exercises will the learner elect to complete on their own? Will the learner complete the exercises during the reading/exploration of the icseBook or only while studying for an exam? Do students who complete more exercises perform better on exams? Does student performance on the exercises predict exam performance? We see the ability to collect data on learner performance from the automated assessments as a rich area for exploration, and one that was unavailable using traditional textbooks. In addition, when the instructor has immediate access to learner performance data it might change their behavior. Instructors would have the ability to modify lecture content based on student performance on the exercises, without the additional effort of manually grading student submissions. Instructors might also be able to design assessments which specifically concentrate on content topics which students have not shown mastery of in the automated exercises.

Controlling frequency of use As instructors, we often accuse our students of procrastination and "cramming" for exams. Yet we rarely have data to support our implied contention that such procrastination has a negative impact on learning. Data collected within an icseBook gives us the ability to know the exact distribution of time spent. We can determine if learners have completed certain learning modules before class. If we were able to enforce controlling the distribution of time spent using the icseBook, how would this affect learning?

Effect on instructor We anticipate that using an icseBook for a class might affect how the instructor's use of pedagogy evolves over time. We can hypothesize that instructors will start by using the icseBook as a direct replacement for text and paper homeworks, and gradually evolve into larger changes in how they conduct their class. Does the use of an icseBook lead to improvements in teaching practice? Do they encourage better teaching practices? Do they lead to more interactive classes?

Question/exercise type We have identified many different types of exercises and/or question types possible within an icseBook (see Section 4). Are specific question types more effective for certain content topics? Do certain exercise types lead to deeper learning or more near transfer? Perhaps a variety of question types is more efficient than a block of identical type questions. Are certain types of questions ineffective for certain topics?

Question/exercise feedback With the ability to have interactive questions and exercises within the icseBook, it is possible to vary the amount, type, and timing of feedback provided to the learner. Empirical evidence can be used to determine the most effective feedback mechanisms for specific question types. If the question requires multiple steps (as in describing the output of a tree traversal), should the learner be notified at their first incorrect response, so as to get back on track for solving the remaining steps? Should hints be provided for multiple choice questions if the learner selects an incorrect answer? Being able to easily change how a question type delivers feedback to the learner allows researchers the opportunity to study how the changes affect learner performance.

Interaction vs. prediction Research has shown that interactive algorithm visualizations are more effective for learning than passive viewing of animations [17, 29]. One type of interaction involves the learner's prediction of "what happens next." Is this type of interaction effective for learning?

Test bank creation An icseBook allows for the creation of a large number (and type) of questions related to specific content topics. Having a centralized location (within the icseBook content files) and identical format allows for easier sharing and distribution of the questions. It also allows for collection of student performance across institutions and nationalities. Unfortunately, current support for question banks is limited to a small number of question types (multiple choice, T/F and static questions with fixed wording). This is in contrast to dynamic problems where, for example, numbers in the question can be random variables. New question banking facilities would be helpful [32].

Identifying Misconceptions Under the constructivist view of learning, learners do not arrive as blank slates, but rather with ideas of how things work. Some of these ideas are incorrect and are termed *misconceptions*. If multiple choice questions are designed with distractors that represent possible student misconceptions, then analyzing student performance can determine the most common misconceptions from within a group. In addition, learner responses to fill-in-the-blank questions and predictive animations can produce other misconception data. Moreover, simulation exercises can reveal misconceptions that can be automatically identified [22].

Correcting Misconceptions If an instructor can view the most common misconceptions within a class, he/she can address those misconceptions directly. In automatic assessment, however, we can ensure that the learner's submission is checked also against misconceived solutions, allowing accurate and detailed feedback to be provided [40]. Potentially such feedback could even take advantage of adaptive learning techniques [9, 8].

Cheating and Credit Cheating is certainly not new or

	L	I	C	S
Increasing visual presentation	X		X	
Visual vs. audio presentation	X		X	
Presentation modes	X		X	X
Length of modules, subcomponents	X		X	
Automated assessment	X	X	X	X
Controlling frequency of use	X	X		
Effect on instructor		X		
Question/exercise type	X		X	
Question/exercise feedback	X		X	
Interaction vs. prediction	X		X	
Test bank creation		X	X	
Identifying misconceptions	X			X
Correcting misconceptions	X	X	X	
Cheating	X	X		X
Managing undesirable behavior	X	X	X	X
Student attitudes/motivation	X	X		X
Exploration vs. linear traversal	X	X	X	X

Table 1: Potential Participant Input per Research Question. Key: L: Learners, I: Instructors, C: Content Developers, S: Software Developers.

unique to icseBooks. However with new technology comes new ways to cheat and new ways to prevent or deter cheating. How to assign course credit is perhaps the most pressing issue today related to MOOCs, and how to control cheating is a major concern for assigning course credit.

Managing undesirable behavior Some students will look for ways to bypass the standard learning methods and invent shortcuts to obtain their desired outcome. These shortcuts might include viewing only the first (or last) slide from a specific slide deck or guessing at multiple choice questions until a correct answer is obtained, skipping assessments completely. Poorly designed assessment exercises can encourage guessing behavior [21]. An icseBook environment provides new ways to "game the system." But icseBook technology also provides opportunities: first to capture information that allows us to conclusively identify which student behaviors lead to good outcomes and which lead to bad outcomes, and then to design the system to deter or eliminate undesirable student behavior.

Student attitudes/motivation One important aspect of any learning environment is the learner engagement. Research concerning student attitudes toward icseBooks and their specific features should be undertaken. In addition, ways to motivate students toward completion of the material should be considered, including investigation of using gamification aspects such as badging and ranking.

Exploration vs. linear traversal With many courses, it is possible to learn the material in many different orders, as long as prerequisite information is identified and enforced. Rather than presenting the typical "table of contents" (which represents a single linear traversal of the material), learners might be presented with a concept map of all the topics to be learned within the course. The learner might explore the topics in any desired order, as long as prerequisite knowledge is enforced, and one could research whether this exploratory version of learning is more effective than a traditional linear approach.

6. IcseBook SOFTWARE DESIGN

The previous sections have identified requirements for icseBook design and functionality. This section discusses icseBook platforms—software systems that meet the requirements of icseBooks and their stakeholders: learners, content developers, instructors, and education researchers, as well as icseBook platform designers themselves. Ideally an icseBook platform would be an environment that allows content developers to create content (an entire icseBook, individual modules for an existing icseBook, visualizations, exercises) without concern about the underlying software required to deploy the icseBook. This section provides direction to those who would design and implement such icseBook platforms. Our chief goal is that developers of icseBook platforms (or of individual components that might be used as part of an icseBook platform) will work to support a good degree of interoperability. No one group of developers is likely to solve all of the problems involved, and creating interoperable systems will allow the community to achieve faster progress than a plethora of platforms with similar functionality but little interoperability.

Any icseBook authoring platform (as well as any icseBook itself) will be an incremental work in progress as modifications and new features are implemented based on desirability and feedback from stakeholders. Thus, we focus only on the primary features necessary for an icseBook platform.

6.1 The icseBook Platform Architecture

We assume that an icseBook will be an HTML5-based web application. While Java applets and Adobe Flash have been popular in the past for producing interactive web content, the rise of HTML5 combined with the rapid deprecation of both Java applets and Flash due to a host of security and compatibility concerns makes HTML5 the only plausible alternative from among these three. It may be desirable at some point to deliver content in a format other than HTML, such as ePub, which now supports interactivity. But as of this writing, web technologies are the best option because of their ubiquity, standards, and capacity for interactive engagement with users.

A proposed architecture of an icseBook platform is shown in Figure 2. The icseBook platform is divided into two main components—a renderer and a web application server.

The *renderer* is the component that is intended to make the task of content developers as extensive, powerful, fluid, transparent, and painless as possible. It should be designed in a way that allows for all the various content artifacts—text, externally referenced content (e.g., Khan Academy exercises), images, slideshows, movies, interactive quizzes, programming examples, algorithm visualizations, models (e.g., of computation)—to be included (and/or adapted for inclusion) by an icseBook content developer while demanding the least amount of underlying knowledge of icseBook platform components. The output of the renderer is an HTML, CSS, and JavaScript icseBook that can be deployed to a web application server.

The *Customization Interface* allows an icseBook creator to rearrange or remove pages, sections, chapters, modules, and so forth to customize an icseBook as desired, while leaving the calculation of the internal hyperlinking (i.e., for navigation menus) up to the renderer.

The *web application server* provides services for serving the content to learners, as well as managing icseBook information gathering, including

- data collection and storage for the icseBook,

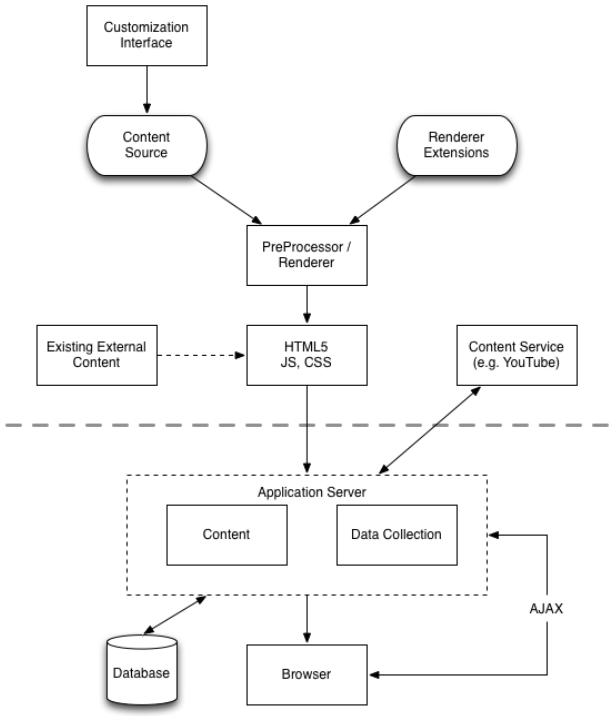

Figure 2: The architecture of the icseBook software platform. The authoring platform is above the dashed line, the deployed book application below.

- logging of learner activities,
- permanent storage for student assignments and grades,
- an interface for the instructor to grade assignments and view grades, and
- a customization interface (for the instructor).

Figure 2 is not meant to imply a "from-scratch" or monolithic implementation of all functionality. We see it as desirable to use existing software and third-party services for some of the platform features (e.g., using a learner's social media login credentials for authentication). Throughout this section, we give examples of such services and software.

6.2 Support for Content Creation/Inclusion

6.2.1 *Support for authoring*

An icseBook platform should support various levels of content creation, matching the contributions described in Section 4.1.2. Example use cases include:

1. Authoring content with artifacts (e.g., media, quizzes, visualizations, etc.). The platform would ideally allow authoring without requiring detailed knowledge of HTML, CSS, or JavaScript.
2. Extending content artifacts, which may require programming expertise to write such things as "macros" or simple commands that expand to complex combinations of HTML, JavaScript, and CSS, thereby allowing a content developer to avoid this complexity.
3. Customization, which allows, for example, instructors to create an icseBook by customizing the work of others, or adding additional exercises to a part of an existing icseBook.

Support for embedding content occurs widely in web design systems. For example, when a video is to be inserted, a dialog might ask what the width of the video should be and allows the content developer to browse to the location of the video on the local disk. A program behind the scenes can then generate the proper HTML link to the video, directly inserting the result into the web page. Similar interfaces can be implemented by an icseBook platform for inclusion of other, more complex artifacts as well. These might include automatic transcoding of media to web-ready formats.

Examples of authoring languages that can be used to create web-accessible textual content include reStructuredText[14], Markdown[15], and comprehensive WYSIWYG editing systems for HTML (e.g., Adobe Dreamweaver). While tools like Dreamweaver or iBooks Author[16] provide "macro" support, we know of no WYSIWYG-only editors that provide this functionality. Runestone Interactive and OpenDSA have both adopted reStructuredText due to its "lightweight markup" syntax, support for embedding media, and its flexible customization of the authoring language through macro definition. Sphinx[17] provides extensions to reStructuredText and can produce HTML (among other formats) as output. Such features ensure that content developers will not get mired in the details of custom JavaScript coding of each interactive element of an icseBook.

An Example. To get a better idea of how reStructuredText macros (called "directives") can work, we give an example that provides for interactive code execution and editing [27]. The directive is called `activecode`. To a content developer, the `activecode` directive in the content source would look as follows:

```
.. activecode:: divid

   print "hello world, I'm a Python example"
```

The `activecode` directive causes two things to happen. To start, several JavaScript files are added to the list of JavaScript libraries referenced on the web page under construction. Then a block of HTML is created that has the structure shown in Figure 3.

The output includes an HTML `div` element that contains a unique identifier (needed to ensure that multiple instances of an `activecode` directive can be contained on the same page). Inside the div is a `textarea` element which gets embedded in a JavaScript editor when the page is loaded. In addition some HTML buttons, a canvas, and an HTML **pre** tag are generated. The canvas is needed for graphical output by the program, and the **pre** tag is there for any textual output from the program. One of the buttons generated by the directive is a run button that allows execution of code in the editor inside the div. Figure 4 illustrates what an `activecode` directive looks like when rendered to HTML.

Most of the functionality described above is also possible in Dreamweaver, although the only known Dreamweaver platform specific for icseBook creation has been designed for supporting interactive models of computation rather than interactivity that requires model extension or interactive program source code editing and compiling [35].

[14] `http://docutils.sourceforge.net/rst.html`
[15] `http://daringfireball.net/projects/markdown`
[16] `http://www.apple.com/ibooks-author/`
[17] `http://sphinx.pocoo.org`

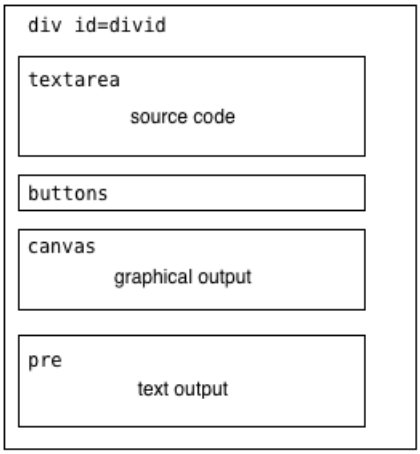

Figure 3: Example of the HTML structure generated by the activecode directive.

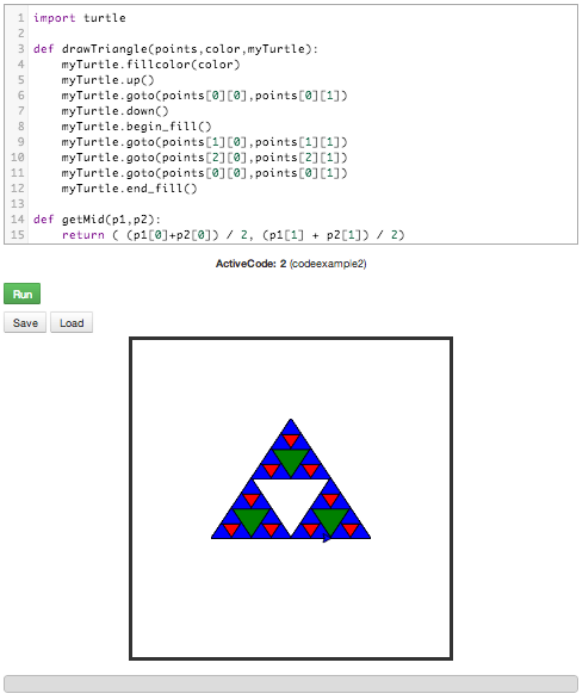

Figure 4: The activecode directive fully rendered to HTML.

6.2.2 *Support for interactive content creation*

Any authoring system should support creation of the following elements of interactive content (introduced in Section 4.1.1).

Self check questions The platform should support generating self-check questions. Question types should include fill-in-the blank, multiple choice, Parsons' problems [18], and more advanced interactive questions that ask a user to manipulate a particular data structure or object. OpenDSA and Runestone Interactive both provide a variety of these kinds of questions. Of course, one good way to "support" this is by relying on a third party. For example, OpenDSA supports embedding questions created with the Khan Academy Exercise Framework. This system supports both embedding individual questions, and embedding collections of questions from which instances are selected at random.

Visualizations Extensive visualization of data structures and algorithms can be provided in a variety of ways. For example, OpenDSA supports visualizations of sorting, searching, trees, and graph algorithms using the JSAV library [23]. The Runestone Interactive Project uses the Online Python tutor system [14], allowing learners to step through their code interactively and view common Python data structures.

Editing and execution of code This was covered extensively in our preceding example of an **activecode** directive. While the `activecode` directive is tied to the Python language through a JavaScript implementation of Python known as Skulpt[18], other languages could be supported by JavaScript implementations including Scheme[19], Processing[20], and JavaScript itself. The Emscripten project [45] might allow many interpreted languages implemented in C to be compiled to JavaScript.

Programming assignments Programs can be assigned to students either in an open ended form or having a defined set of inputs and outputs. For example the Runestone Interactive project has a unit-test framework that the content developer can use to provide an invisible set of test cases, giving the learner feedback on their programming assignment. JhavePOP[21] supports small-scale programming exercises on pointer operations, and provides visualizations of the student's answer. This is being ported to OpenDSA, where it will also include automated assessment of the answers.

Audio and Video HTML5 includes new elements for embedding video and audio directly into the page without plugins. However, the supported audio and video formats vary among browsers, currently requiring content developers to transcode original videos into both of the supported formats (mp4 and webm) using freely available software such as Handbrake[22] and Firefogg[23]. Ultimately, the authoring platform should handle resizing and transcoding automatically. Another approach is to host the videos on platforms such as YouTube and embed them into the icseBook.

Social Aspects There are many possibilities for incorporating social sharing into an icseBook. Some might simply serve a marketing purpose, such as "like me", "follow me", and "share me" links. Facebook, Twitter and Google all have code for incorporating such links. Furthermore, in case of *gamification* features, the platform could enable sharing achievements (such as badges gained) to social media.

Another possibility is to use social media services to encourage discussion and sharing among students. For example, Runestone Interactive provides a directive that allows a disqus[24] discussion box to be placed at any point on the page. Similar services include IntenseDebate[25] and LiveFyre[26], both of which provide a discussion forum service for

[18] `http://www.skulpt.org`
[19] `http://www.biwascheme.org/`
[20] `http://processingjs.org/`
[21] `http://jhave.org/jhavepop`
[22] `http://handbrake.fr/`
[23] `http://firefogg.org/`
[24] `http://disqus.com/`
[25] `http://www.intensedebate.com/`
[26] `http://web.livefyre.com/`

free. We encourage developers to make use of these web services rather than inventing their own.

Learner Driven Algorithm and Program Simulations In algorithm simulation exercises, learners are the ones describing the behavior of an algorithm by modifying a data structure visualization. For example, this can be done by clicking nodes within a binary search tree. Systems supporting such exercises include TRAKLA2 [24] and JSAV [23]. Algorithm simulations require rather specialized support, such as a mechanism for storing the key steps of an algorithm, generating model answer visualizations, comparing student and model answers, providing feedback in the event of mistakes, and capturing scoring information. Thus, it is significant that high-quality open-source libraries exist as exemplars for algorithm visualization support, rather than requiring individual development teams to create their own. Program simulation systems such as UUhistle [43] provide some similar features, except that the student is simulating the execution of a program and getting feedback in the event of mistakes.

External Content A crucial aspect of interoperability is the ability of the icseBook platform to support embedding interactive content from third party services. At the simplest, this can be embedding content from a URL into an iframe on a page[27]. For more complex cases, the OEmbed standard[28] used by many online services (e.g. Flickr[29]) should be supported by the platform. Another example of a lightweight service embedding protocol is implemented in the A+ learning platform [20].

While the actual process for authoring interactive content depends heavily on the type of content and the software system used, we can give some general guidelines for content developers. In general, all software support for interactive content should be designed to be *extensible*, since the original content developer is not going to be able to implement support for all future uses. Also, the style of the content created with the system should be separated from the content in order to allow changing the look-and-feel of all learning artifacts created with the same system (as discussed in 4.3.2). Finally, content developers writing code should consider *reusability* of their code and *expose and document APIs* in order to help both themselves and other content developers to build on their efforts.

6.2.3 Support for icseBook customization

The survey tells us that some amount of customization is moderately important to most instructors, as discussed in 4.2. Listed as important (**Q18**) were the ability to delete or reorganize modules, add a new module, modify existing text, and add exercises. To address the key requirement of reordering content, the software system should support a table of contents that can be reordered. For example, an icseBook generated from restructuredText has a single master file called `index.rst`. This file defines the table of contents for the book and controls the order in which the material is presented. In order to create a customized version of the book, an instructor can simply create a new index.rst file. This can be done using a traditional text editor, or through a more friendly user interface that allows the instructor to drag and drop sections or chapters into an appropriate order, letting the tool write the new index file. The Dreamweaver approach uses a similar table of contents file to automatically organize a group of HTML files.

To support instructors who want to customize the order of the learning modules in their icseBook, there must be some way of describing the data required to build the dynamic network of learning modules (see Section 4.3). This might be as simple as specifying all of the prerequisites for a particular module. This means that each module includes metadata describing both the content, perhaps in the form of tags, as well as prerequisites. This could be done using comments at the top of each source file, or a with a restructuredText directive. This information can be conveyed to other instructors using a standard such as LOM[30] or Dublin Core[31]. A standard approach using JSON or XML would make prerequisite structures interoperable.

To support the customization of the look-and-feel, it is strongly advised that content and presentation be separated so that the presentation can be changed without affecting content in a manner similar to the model-view-controller software design pattern [10]. This applies to the entire icseBook infrastructure as well as to the design of individual modules, especially interactive artifacts.

Finally, customization can itself generate a first-class artifact in the system, perhaps in the form of a "configuration file" such as OpenDSA uses. This allows instructors to define the contents for a textbook in a way that other instructors can reuse, allowing modifications of their own without having to build the configuration from scratch.

6.2.4 Support for peer review

Providing for comments and peer review on the contents of the book is an important feature that has been solved by several systems. For example *djangobook*[32] and *medium.com* allow readers to click on a paragraph and leave a comment. A small icon appears next to any paragraph that has been commented on. This feature should be able to be turned off. Once a book has been reviewed and is available for learners, peer feedback may not be helpful.

Peer review data should be available in an XML format, or through a web-based interface. Ideally this availability would help make the case that contributions to an open source icseBook should count for tenure and promotion.

6.2.5 Support for versioning

Versioning is a huge topic, but what is important for this discussion is that there should be some way to make old versions of a text available. For example, an instructor might generate a new instance of an OpenDSA textbook each semester as he/she makes minor changes in the content to be covered. Without some sort of versioning support, a great deal of confusion might result. One way to accomplish this is through a simple URL-based approach as follows:

- ...BookName – lets the user access the most current revision of the icseBook.
- ...BookName/edition – lets the user access the most recent revision of an edition of an icseBook

[27]This is how some JSAV-based AVs and simulation exercises are actually embedded into an OpenDSA icseBook.
[28]`http://oembed.com/`
[29]`http://flickr.com/`
[30]`http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf`
[31]`http://dublincore.org`
[32]`http://djangobook.com`

- ...BookName/edition/revision – lets the user access a specific edition and revision of an icseBook.

6.2.6 Support for navigation (UI design)

To aid learners with navigating the icseBook, it is desirable for the software system to automatically generate a navigation bar that is always visible. This navigation bar should provide links to the table of contents, an index, and the next or previous module. In addition, the navigation bar should provide access to a search box for the book.

6.2.7 Internationalization

A complex topic with an icseBook is its internationalization, that is, providing the book and the content in multiple languages. We see different levels of internationalization: translation of text content, changing the UI language in the book and interactive modules, and translating the interactive content. The main point we want to make here is that developers of systems for interactive content should make it possible to translate the UI and the module content without copying significant parts of the implementation code. Furthermore, since a lot of the book content deals with programs, it should be possible to change the programming language used in the book (in topics where this makes sense, such as in a data structures and algorithms books).

6.3 Support for Teaching with an icseBook

6.3.1 Content use scenarios

We described earlier the reasons why HTML5 is currently the only viable solution in our view for icseBook presentation. However, other formats with the necessary functionality are likely to appear in the future. One such potential format is ePub. Version 3 of ePub supports interactive content using JavaScript[33]. The ePub 3 format is supported by many reading devices and applications (such as Apple iBooks). Creating an interactive paper copy of any icseBook is not possible, though a static print version could exist that references online dynamic content. This approach was taken in the textbook by Crescenzi [13] that includes notes for readers to use the AlViE visualization system[34]. However, while many teachers adopted the book for their teaching, many of them did not use the accompanying visualization tool. We conclude that integrating visualizations, exercises, and textbook content is one of the most important features of an icseBook, providing crucial synergies that cannot otherwise be achieved.

Another major factor in usage scenarios are the devices used to access the material. In the survey results (**Q19**), "must run on mobile devices" was rated the single most important requirement by 48% of the respondents. Students today might be using their laptop at school, a smartphone on the bus, and a tablet on the couch at home. The assumption of modern web applications is that they work on all these devices, and that student progress is synced across devices and browsers. All modern mobile devices include technically sophisticated web browsers, so the material that works in a desktop browser should work on smartphones as well. However, when designing interactive content and the layout of the icseBook, the limitations and capabilities of target devices should be taken into account:

- Content should adapt to different screen sizes. Technically, this can be achieved with CSS using sizing relative to the window as well as CSS3 media queries to style content differently based on the screen size. Content developers might decide not to support below a certain resolution (for example, cutting out smartphones while still supporting tablets), but they should be aware of the consequences.
- The correct input method for the interactive content should be carefully considered. For example, JSAV [23] does most interaction through clicks instead of the drag-and-drop approach used in the older TRAKLA2.
- Interaction targets should be large enough to be usable on touch devices. For a detailed discussion of touch and touch targets, see [16].
- The content should not depend on functionality that is not practical on touch screens, such as hover.

6.3.2 Authentication and authorization

For learners in a specific course who aim to complete a set of modules assigned to them by the instructor, *authentication* is an important consideration. Learners (as well as instructors) want to ensure that their achievements are properly recorded. Thus, the icseBook must support user registration and login/logout.

To ease the registration process, the icseBook should support authentication with existing user accounts such as Facebook, Twitter, or Google as well as single sign-on systems provided by universities. A useful service for integrating with multiple identity providers is the Janrain User Management Platform[35], which is used by Runestone Interactive. Another similar platform is the Gigya Social Login[36].

For voluntary learners not tied to a course, registering with the icseBook can simply be an annoying additional step to accessing the content. Instructors who might consider adopting an icseBook are also likely to be put off by requirements to register just to try out exercises or view content. So while registration and login must be supported, developers should carefully consider what limitations are placed on anonymous users. We consider it to be good practice to allow anonymous users to access the content and complete the exercises. This permits interested parties to "try out" the icseBook before committing to its use. To protect registered users who forget to log in, the interaction log data and results can be stored in the browser's local storage in case the user decides to register or log in later on. When registering, the results from the local storage can be attached to the created user profile. This approach is already implemented in the OpenDSA backend [7].

In addition to attaching grades and log data to the correct learner, authentication provides at least a rudimentary support against spamming and attempts to hack the system to, for example, get credit without solving the exercises.

For authorization, we see at least two levels of users with access to different data. In general, learners should only be able to access the material and their own results, whereas instructors need access to the data of all of their students. However, as social aspects become more widely used, the

[33] `http://www.idpf.org/epub/30/spec/epub30-overview.html`

[34] `http://alvie.algoritmica.org/alvie3`

[35] `http://janrain.com/`

[36] `http://www.gigya.com/social-login/`

learner might wish to make part of his/her profile public and thus accessible to other learners.

6.3.3 Instructor's version

Traditional textbooks typically come with an instructor's version which includes ready-made lecture slides and solutions to exercises. According to the survey (**Q20**), 40% of the respondents do not require an instructor's version of an icseBook whereas only 14% would definitely require it. When asked which instructor content they must have, the most common answers were solutions to all exercises, test banks of questions, and coursenote slides.

Solutions to exercises can be provided on a separate page that requires instructor authentication. Examples include multiple choice or fill-in-the-blank questions with automatic feedback as well as the algorithm simulation exercises. For icseBooks that are built entirely with automatically assessed exercises, explicit solutions might not be required. In fact, dynamic exercises where various parameters of the question are selected at random cannot have "an answer", though an instructor version could provide an answer for specified problem instances. For many interactive exercise types, feedback about the correct answer is built into the activity.

For generating lecture slides, the content renderer might extract specifically marked content to slides. While there has been research on how to automatically generate lecture slides from visualizations, we think these slides should also be in HTML5 in order for the interactive content to work in them. This functionality is already supported, for example, by Sphinx (but it does require that the content developer explicitly redesign the content to achieve an effective presentation).

In addition to the typical instructor material, an icseBook can offer supporting functionality right within the content. For example, in Runestone Interactive, special links next to each exercise can appear only for instructors. When an instructor clicks on an exercise, they might see the aggregate results for their class. When an instructor clicks on a programming assignment, they might have the ability to see their students' submissions and provide feedback.

6.3.4 Deployment practicalities

Open Source and Licensing Software projects in academia often end once a few papers about the software are published. At this point, many useful systems end up being "wasted" instead of released for others to benefit and build on. Luckily, all of the systems mentioned above have been released as open source. We highly encourage developers of future systems to publish their code, even if it is not highly polished. Furthermore, the source code for the system should have explicit, easy-to-find license information.

Hosting There are two models to consider for hosting. First, to deploy an icseBook an instructor could download the source from a public repository such as GitHub[37], then compile, configure, and host the icseBook on a local server. The advantage is that it is easy for the instructor to customize any aspect of the icseBook and to keep any student data collected within the walls of the university. The disadvantage is that building and hosting the platform could be a lot of work. The second option is to design the icseBook platform so that hosting courses is a service. This is the approach taken by the Runestone Interactive project at their *interactivepython.org* site. In this approach, an instructor simply goes to the site and creates a course. At the time of book creation, the instructor might customize the book using a drag-and-drop interface to select various chapters or sections, or they might simply adopt an entire book. After the book is built, any online activities, grades, exercises, etc. are tied to this newly built book instance.

Online vs. Offline Use In some designs, content in an icseBook executes on the client side without further intervention from a server once the page is loaded. This allows the book to be packaged and distributed to be loaded to, say, a mobile device. While we realize that some content requires communicating with a server, we encourage as much of the functionality to be implemented so that it works without an internet connection once downloaded. As in the case of anonymous users, the interaction data and learner results can be cached locally and synchronized with the server later.

6.4 Support for Evaluation

We have identified six general areas that would benefit from evaluation: learning, instructing, developing content, usability, icseBook platform development, and education research. Each of these requires the collection, archiving, retrieving, processing, and reporting of data. This, in turn, implies that an icseBook platform be a Web application that not only delivers icseBook content but can also be formulated to acquire and process data from icseBook use. In the interests of interoperability, it is strongly recommended that all such data be represented in a standard format, such as XML or JSON. A list of results to be achieved, and the data to be gathered for computing results would be both too long and incomplete. We only provide flavors of these below.

6.4.1 Learner interaction data collection

For instructors, content developers, or researchers to find answers to the learner interaction questions listed in Section 5, collecting rich interaction data is essential. Examples of systems that currently do much of this are OpenDSA and Runestone Interactive [27]. However, these do not store data in a standard form.

As an example of learner interaction data collection, an icseBook should be able to record the learner, course, learning object with which the learner is interacting, and timestamp of the activity. Furthermore, each logged event should include the event type and a description of the activity. Different content will need to store different kinds of information, so this payload should allow storing free text. To submit interaction data to the server, the backend should expose a URL where a log event can be posted as JSON or XML using AJAX.

One standard that could be adopted for interaction data collection is the Tin Can API[38]. While the specification looks promising, it is new and its benefits and popularity remain unknown.

6.4.2 Teaching data collection

There are two distinct goals to data collection related to teaching: (1) results that can lead to improvements in instruction, and (2) results that allow student assessment (e.g., grades, progress through assigned icseBook material, and so forth). Point 1 can be determined through recording and processing learner interaction (discussed above). It can also

[37] `http://github.com/`

[38] `http://tincanapi.com/`

be done through periodic student surveys that are recorded for analysis. Point 2 implies that student scores of interactive tests and exercises must also be saved, and analysis and integrated display software must exist that allows an instructor to view scores as well as allowing individual students to view their results.

In the future, it would be advantageous to integrate grade data information with various learning management systems such as Moodle. Standard specifications for this are available—for example, the IMS Learning Tools Interoperability framework[39], which has been seen as promising by eLearning system vendors [1].

6.4.3 Data collection for other stakeholders

The collection of learner interaction and instructor data will be invaluable for other stakeholders as well. For content developers, the data can provide insight into the usability of the content and specific features. For platform developers, use statistics can provide formative guidance and polishing of the most-used features, and identification of under-used features. For educational researchers, data collection works as a base for analysis. However, the data gathered from users and instructors might need to be supplemented with more targeted surveys specific to the questions being addressed, possibly including mechanisms whereby learners, instructors, and peer reviewers can provide feedback on the icseBook content.

7. CONCLUSION

To borrow a term from the world of music, the vision for an icseBook we have presented here is that of a *remix*, blending components from instructional technologies that already exist in isolation into a whole that exceeds the sum of its parts. Our hope is that the report will offer inspiration for content developers to begin actively contributing to such projects, particularly those that are open source, thereby leading to their widespread use by Computer Science instructors and students. Toward that end, Sections 2, 3, and 4 have presented a set of icseBook user requirements. We also want to ensure that, from a software engineering perspective, such open source icseBook projects themselves are developed using guidelines that ensure standards for interoperability (Section 6). This will afford education researchers the opportunity to explore more deeply than ever before the variety of interactions between factors that lead to effective learning (Section 5).

8. REFERENCES

[1] C. Alario-Hoyos and S. Wilson. Comparison of the main alternatives to the integration of external tools in different platforms. In *Proceedings of 3rd International Conference of Education, Research and Innovation*, pages 3466–3476. IATED, 2010.

[2] E. Allen and J. Seaman. Going the distance: Online education in the united states. Technical report, The Sloan Consortium, November 2011.

[3] T. Bailey and J. Forbes. Just-in-time teaching for CS0. *ACM SIGCSE Bulletin*, 37(1):366–370, Feb. 2005.

[4] J. Bennedsen and M. E. Caspersen. Revealing the programming process. *ACM SIGCSE Bulletin*, 37(1):186–190, 2005.

[5] B. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-on-one tutoring. *Educational Researcher*, 13(6):4–16, 1984.

[6] C. R. Boisvert. A visualisation tool for the programming process. *ACM SIGCSE Bulletin*, 41(3):328–332, 2009.

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

[8] P. Brusilovsky, G. Chavan, and R. Farzan. Social adaptive navigation support for open corpus electronic textbooks. In *Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 24–33. Springer, 2004.

[9] P. Brusilovsky, J. Eklund, and E. Schwarz. Web-based education for all: a tool for development adaptive courseware. *Computer Networks and ISDN Systems*, 30(1):291–300, 1998.

[10] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*. Wiley, 1996.

[11] C. M. Christensen, M. B. Horn, and C. W. Johnson. *Disrupting class: How disruptive innovation will change the way the world learns*. McGraw-Hill, 2008.

[12] D. L. Cook. The Hawthorne effect in educational research. *The Phi Delta Kappan*, 44(3):116–122, 1962.

[13] P. Crescenzi and C. Nocentini. Fully integrating algorithm visualization into a CS2 course. a two-year experience. In *ITiCSE '07: Proceedings of the 12th annual conference on Innovation and technology in computer science education*, pages 296–300, 2007.

[14] P. J. Guo. Online Python Tutor: Embeddable Web-Based Program Visualization for CS Education. In *SIGCSE '13: Proceeding of the 44th ACM technical symposium on computer science education*, pages 579–584, 2013.

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

[16] N. Henze, E. Rukzio, and S. Boll. 100,000,000 taps: analysis and improvement of touch performance in the large. In *MobileHCI '11: Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 133–142, 2011.

[17] C. D. Hundhausen, S. A. Douglas, and J. T. Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, 2002.

[18] P. Ihantola and V. Karavirta. Two-dimensional parson's puzzles: The concept, tools, and first observations. *Journal of Information Technology Education: Innovations in Practice*, 10:1–14, 2011.

[19] P. Ihantola, V. Karavirta, and O. Seppälä. Automated visual feedback from programming assignments. In *Proceedings of the Sixth Program Visualization Workshop*, pages 87–95, Darmstadt, Germany, 2011.

[20] V. Karavirta, P. Ihantola, and T. Koskinen. Service-oriented approach to improve interoperability

[39] `http://www.imsglobal.org/lti/`

of e-learning systems. In *13th IEEE International Conference on Advanced Learning Technologies*, page 5, 2013.

[21] V. Karavirta, A. Korhonen, and L. Malmi. On the use of resubmissions in automatic assessment system. *Computer Science Education*, 16(3):229–240, Sept. 2006.

[22] V. Karavirta, A. Korhonen, and O. Seppala. Misconceptions in visual algorithm simulation revisited: On ui's effect on student performance, attitudes and misconceptions. In *LaTiCE'13: Learning and Teaching in Computing and Engineering*, pages 62–69, 2013.

[23] V. Karavirta and C. A. Shaffer. JSAV: The JavaScript Algorithm Visualization library. In *ITiCSE'13: Proceedings of the 18th annual conference on Innovation and technology in computer science education*, pages 159–164, 2013.

[24] 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, 2004.

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

[26] B. Means, Y. Toyama, R. Murphy, M. Bakia, and K. Jones. Evaluation of evidence-based practices in online learning. a meta-analysis and review of online learning studies. Technical report, U.S. Department of Education, September 2010.

[27] B. N. Miller and D. L. Ranum. Beyond PDF and ePub: toward an interactive textbook. In *ITiCSE'12: Proceedings of the 17th annual conference on Innovation and technology in computer science education*, pages 150–155, 2012.

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

[29] 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 *Working group reports from ITiCSE'02 on Innovation and technology in computer science education*, pages 131–152, 2002.

[30] S. Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., 1980.

[31] D. Parsons and P. Haden. Parson's programming puzzles: A fun and effective learning tool for first programming courses. In D. Tolhurst and S. Mann, editors, *Eighth Australasian Computing Education Conference (ACE2006)*, volume 52 of *CRPIT*, pages 157–163, 2006.

[32] A. Paul. Qbank: A web-based dynamic problem authoring tool. Master's thesis, Virginia Tech, June 2013.

[33] G. Pike. Students personality types, intended majors, and college expectations: Further evidence concerning psychological and sociological interpretations of Holland's theory. *Research in Higher Education*, 47(7):801– 822, 2006.

[34] L. Porter, C. Bailey Lee, and B. Simon. Halving fail rates using peer instruction: a study of four computer science courses. In *SIGCSE '13: Proceeding of the 44th ACM technical symposium on Computer science education*, pages 177–182, 2013.

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

[36] 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. *SIGCSE Bull.*, 38(4):166–181, June 2006.

[37] G. Rößling and T. Vellaramkalayil. A visualization-based computer science hypertextbook prototype. *Trans. Comput. Educ.*, 9(2):11:1–11:13, June 2009.

[38] T. Russell. *The No Significant Difference Phenomenon: A Comparative Research Annotated Bibliography on Technology for Distance Education*. IDECC, 5th edition, 2001.

[39] P. Saraiya, C. Shaffer, D. McCrickard, and C. North. Effective features of algorithm visualizations. In *SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, pages 382–386, March 2004.

[40] O. Seppälä, L. Malmi, and A. Korhonen. Observations on student misconceptions – a case study of the build-heap algorithm. *Computer Science Education*, 16(3):241–255, September 2006.

[41] C. Shaffer, V. Karavirta, A. Korhonen, and T. Naps. OpenDSA: beginning a community active-ebook project. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, pages 112–117. ACM, 2011.

[42] J. Sorva, V. Karavirta, and L. Malmi. A review of generic program visualization systems for introductory programming education. *ACM Transactions on Computing Education (TOCE)*, To appear.

[43] J. Sorva and T. Sirkiä. UUhistle – A Software Tool for Visual Program Simulation. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, pages 49–54, 2010.

[44] H. Wasfy, T. Wasfy, J. Peters, and R. Mahfouz. No skill left behind: Intelligent tutoring systems enable a new paradigm in learning. *Computers in Education Journal*, 4(2):2–10, April 2013.

[45] A. Zakai. Emscripten: an LLVM-to-JavaScript compiler. In *SPLASH '11: Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pages 301–312, 2011.