Curriculum Format White Paper

Date generated: 12/10/2018

People

Here is a list of people with an established interest in this topic:

- Austin Cory Bart (University of Delaware)
- Luke Gusukuma (Virginia Tech)
- Michael Hilton (CMU)
- Phill Conrad (UC Santa Barbara)
- Paul Salvador Inventado (California State University Fullerton)
- Arto Hellas (University of Helsinki)
- Cliff Shaffer (Virginia Tech)
- Bob Edmison (Virginia Tech)
- Michael Stewart (JMU)
- …
- *Feel free to add your name too!*

Introduction

We are interested in the problem of curricular materials "packaging" for CS Ed. Briefly, packaging materials is a generalized term to capture a broad array of tasks (creating, revising, sharing, finding, crediting, etc.) for a broad array of materials (assignments, assessments, teacher notes, evaluation data, etc.). A substantial amount of effort has gone into creating materials over the years, but the community still seems to struggle to find ways to manage it effectively. We are hoping to find others interested in developing a subcommunity to explore what we can do in the future to solve this problem. This subcommunity would join with broader efforts of standardization (e.g., CSSPLICE) in order to build on existing projects and initiatives.

Concretely, we are proposing a SIGCSE Birds-of-a-feather to kickstart further discussion. However, we also have some initial proposals for a Git-based format for curricular material sharing. Exactly what we propose in this document is still on the table.

Prior Work

This problem of curricular reuse is well known in the broader educational community, which has spawned initiatives like Open Educational Resources (OER) (free repositories of useful curricular materials) or the concept of Learning Objects (reusable lessons based on Object-Oriented Programming ideals). Within Computer Science, a number of projects have attempted to solve the problem for CS. For example, a 2013 ITiCSE working group led by Sanders et al has led to the creation of a repository (The Canterbury Question Bank) of multiple choice quiz questions for CS1 and CS2, while the OpenDSA project provides complete eTextbooks and associated assignments. Nifty Assignments is perhaps the most well-known success in the SIGCSE community, growing from a recurring paper to its very own track at the annual SIGCSE conference; the Nifty project is a high-quality, well-curated collection of programming projects. From the research side, Leake and Lewis published a SIGCSE’17 paper with guidelines and notes about developing CS resource sharing sites. This paper covers more than a dozen online repositories such as Ensemble (the NSF Digital Library for CS Education) or Code.org, but was more targeted at the K-12 space.

Problem Statement

All of these efforts, whether they are repositories or guidelines, have helped numerous instructors simultaneously improve their classes while easing the burden of teaching. However, the authors still feel that the problem of Curricular Packaging has not been solved in University-level Computer Science Education. In particular:

1. We need fewer barriers to the creation of high-quality material.
2. We need more wide-spread knowledge of existing repositories and curricular materials
3. We need more evaluation of existing materials.
4. We need a better way to structure and organize our materials (e.g., around a calendar, or topics, or resources)
5. We need better ways to systematically improve our materials.

Here are six concrete scenarios that should be more trivial than they are:

1. The newly hired Dr. X is teaching an Introduction to Algorithms course for the first time. About one-third of the way into the semester, he is teaching a lesson on Heaps and wants to provide some in-class activities. This will require developing a sufficient lecture to prepare students, a series of questions that will test all of the relevant skills and knowledge, and a reference solution. Where can Dr. X go to get all of these?
2. The department of Computer Science at Y University is redesigning their introductory computing course for majors. They know that students struggle with the difference between printing and returning, and want to add a new unit on the subject. From casual discussion at conferences, their faculty has heard that this is a common issue. They want to start organizing the specific misconceptions (along with their frequency and specific examples) that students encounter, and the teaching methods that their TAs have found to be particularly effective at overcoming the misconceptions. How should they collect and share this data in a way that protects student privacy?
3. Dr. Z has created a high-quality collection of lessons on advanced Software Engineering topics, and wants to share the resources with colleagues. This includes not only the teaching plan, but extensive rubrics, student exemplars, and educational video lessons. How does he share his lesson, get external feedback, and gather evidence for his promotion case that he is contributing to the broader CS community?
4. Prof. U has been tasked to teach a course at her new University of A. She wants to use the full material from her colleague at the University of B. The course system that is used at University of B is not, however, the same as in A, and the format in which the course content is stored is quite complex. Thus, she needs to spend significant amount of time to alter the materials into their new current course system.

5. PhD student Q is constructing tools that can automatically identify learning objectives etc from learning materials. Currently, as most of the learning materials follow their own format, she needs to create custom parsers for each set of materials.
6. Professor H talks with colleagues at a conference, and would like to share the material for a class they recently developed on Software Development in Startups. However, because they did not initially plan on sharing the class, they cannot share the current materials which combine both content and results without violating student confidentiality.

These problems are drawn from the authors' own experiences. Although there are partial solutions, they are not really successful enough to consider the problem solved by our standards.

Unique CS Challenges

The field of Computer Science has a number of characteristics that make packaging materials more difficult, particularly at the undergraduate level.

1) **Non-standardized Goals:** As a younger, less-established discipline, CS has a less-established common core - instructors are relatively unlikely to be working from the same list of learning objectives. Fields like Biology and Chemistry are more likely to have classes that are consistent across universities, compared to CS.
2) **Novice Designers:** There is little professional development of pedagogical training: most instructors in CS have no knowledge of practices like instructional design or best practices, and build up resources without being systematic or comprehensive.
3) **Tool Lock-in:** Because computer science as a field is deeply wrapped in technology, many curricular materials end up locked to a particular tool: whether that tool is a particular programming language, or a platform, or a learning management system, this has implications for the long-term sustainability of the material surrounding that tool. Growing interest in data interchange formats and standard protocols could present an opportunity, however (i.e., CSSPLICE). It would be beneficial if the materials we create are not tied to a specific technology, but instead are lightweight formats.
4) **Lack of Incentives:** At the undergraduate level, there are few incentives for instructors to package up their materials and share them. With the growing importance of teaching-focused professionals in CS, however, there might be a new opportunity to change this.
5) **Misguided HCI**: Computer Scientists are in a unique position to develop and adopt materials of a more technical nature. However, we still need high usability in our interfaces. This requirement is easily overlooked because of our strong technical skills: developers may choose to cater to only "technically adept" adopters (perhaps better described as adopters who are particularly enthusiastic, or who have particular amounts of time and energy). Of course, good user experiences for those sharing materials is extremely difficult, and the general challenges should not be minimized. This is also an opportunity to bake-in accessibility from the start.

What are Curricular Materials?

Curricular materials in Computing Education come in many shapes and sizes.

- Public student-facing artifacts: slide decks, textbooks, video lessons, essay questions, programming exercises, lab write-ups, projects, reference sheets
- Semi-private instructor materials: However, they also include instructor notes, grading rubrics, reference solutions, common misconceptions, TA training handouts and videos
- Private Evaluation data: summarized survey statistics from student surveys, usage and behavior patterns information that describe the effectiveness of a tool, difficulty and discrimination data for quiz questions, reflections from other adopters on limitations and variations of a lesson plan

Notice that much of these materials are not complex - they come in standard file formats like plain text, HTML, markdown, slides, etc. However, the range of these materials complicates the development of high-quality repositories.

What is Curricular Packaging?

These curricular materials are involved in general processes related to “packaging material”. We use this phrase to capture a very wide array of actions related to these materials:

- Creating: When instructors want to create new lessons from scratch.
- Fixing: Fixing typos in write-ups, correcting bad information, or making small improvements to UI.
- Revising: Updating material with new information or to account for new understanding in the field. For example, changing a lesson involving Java Observers (Java <9) to the more modern idiom of Java Listeners would be a substantial change.
- Reusing: The same instructor teaching with the same materials again, to a new batch of learners.
- Sharing: Giving the materials to someone else, or making it publicly available.
- Finding: Searching and navigating through repositories to find specific curriculum, or perhaps just browsing for things of interest.
- Tracking: Hearing about updates on materials in use
- Pulling: Accepting revisions and updates made by others
- Pushing: Notifying adopters of changes to the materials.
- Evaluating: Gathering data from learners or adopters about the efficacy of the materials.
- Crediting: Getting recognition or rewards for any of the above

Subcommunity Formation

The first concrete “next step” for this initiative is a Working Group within CSSPLICE and the submission of a SIGCSE BOF on curriculum packaging.

Git-based Format

Our second concrete "next step" is the proposal of a Git-based Format.

A format for curricular resources revolving around Lessons, Modules, and Courses. These materials need to be created, edited, revised, shared, and have edits incorporated later. Many of these verbs identified seem to correspond to Git verbs, which makes it seem like a natural fit for the repository.

A valuable component of this format would be to establish a bunch of "Nouns" that courses should be collecting and organized around. These nouns refer to the curricular materials that we described before. Ideas for metadata around nouns like Learning Outcomes, Programming Assignments, Essay Rubrics, etc. could help instructors develop more high quality material. Having more standardized materials make them easier to compare, easier to revise, etc.

Some of these materials need varying levels of privacy and security. Best practices could be established to help instructors work in the bounds of IRB/FERPA. If you are collecting evaluation data, how much does it need to be anonymized/summarized/filtered before it can be publicly shared? How can you organize a Git repo to prevent non-teachers from accessing your instructor reference solutions, even when you want to let casually interested potential adopters to peruse the materials?

If the format is precise enough, it could lead to tools that automatically convert/prepare materials for insertion into popular LMS like Canvas or Moodle. Using tools like GitHub Pages, materials could be served more directly too.

Some theoretical long-term steps

- Extension/Adoption/Cheerleading of existing or new websites for sharing materials
- More internal and external incentives for sharing/revising high quality material (e.g., grants, glamour).

Potential Criticisms

Here are some arguments that can be made in response to this initiative:

- *"If so many repositories have attempted to solve this problem, why should this effort succeed?"* - We're not proposing a repository, we want to propose a simple format that could be compatible with existing repos.
- *"Isn't this just Learning Objects? Hasn't this been solved?"* LOs and OER seems to have been fairly effective in other domains. CS Ed has a unique opportunity to build on that success, and yet we still haven't really seen the delivery of that promise. We hope a new direction will help with that.
- "*Why should people care if you make a curriculum format? They're busy, they won't suddenly have more time to package materials.*" Quite possibly. This might have more limited adoption by those who are most excited at first. The trick is hopefully making the

format immediately useful and long-term useful for instructors. The immediate problem being solved is the need for instructors to have ways of reusing and sharing their materials with immediate colleagues. If the format is simple enough, and there is tooling support, it might be useful enough to justify its extra work.

BOF Submission:
https://docs.google.com/document/d/1XngZ6PzI5-sM9H3ECD616eIei1SUbINApyF5OI-iZBM/edit?usp=sharing