**Engaged Student Learning (Design and Development I)**

1. Introduction

We propose to enhance the motivation and learning gains that diverse students in diverse STEM majors have in their initial college-level learning experience with computing. While improving learning gains is clearly desired concern for motivation is also critical for several reasons. First, learning core computing concepts often poses significant cognitive challenges for students in non-computing majors. They are asked to learn a new vocabulary and developed new mental frameworks. Second, these students have no commitment to the computing discipline and are often uncertain of, or even fearful about, their ability to succeed. While they use computing-enabled devices daily, their image of computing is of a deeply mysterious and extremely technical nature. Developing and sustaining a high level of motivation and engagement is critical to the learning of this student population. Third, even for computer science majors, motivation is important for retaining students with uncertain commitment to the discipline, especially students from under-represented populations in computing. The key elements of our research are shown in Figure 1 and summarized below.

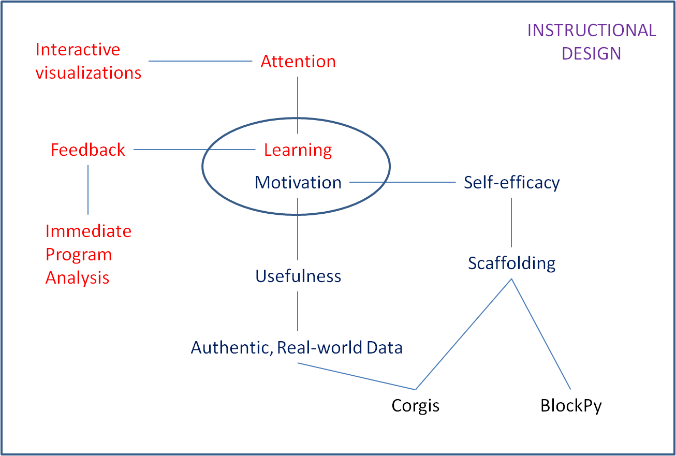
Our approach to improving motivation raises the students’ appreciation of the *usefulness* of their learning and heightens their sense of *self-efficacy*. Usefulness means that students’ perceive that what they are learning has relevance and importance to something they value. What is valued might relate to their professional or disciplinary objectives (this learning will help me in my field of work or study) or to their personal interests (this learning gives me a deeper understanding of an activity that I care about). The sense of self-efficacy arises when students believe that they are capable of achieving success in their learning. A common barrier to self-efficacy is that the material or tools involved in the learning have a gradient that is too steep, causing the students to lose confidence in their ability to succeed.

Figure : Research Approach

Improvements in learning gains, the gold standard for all education research, will be improved by better means for sustaining *attention* and the provision of better *feedback*. The e-book that we developed for the computational thinking course allows us analyze the students’ use of the book text. It is clear that the students, especially those who are struggling, do not focus attention sufficiently on the text and images provided. Our end of term dialog with the students has also provided anecdotal evidence that feedback on programming problems is highly valued and is often a stimulus for continued engagement with the problem.

Our approach to creating a sense of usefulness is by providing authentic, real-world learning experiences based on intrinsically realistic data. Data is intrinsically realistic if it concerns real-world events (e.g., scientific, economic or social), is from authoritative sources (e.g., government agencies, social media), and is of genuine scale and complexity (not a “toy” version). We use data that is big, real time, and/or geo-located. Intrinsically realistic data raises the level of student motivation by engaging students in authentic experiences [3-5]. This work extends an existing framework, Corgis [REF] we have created for big data and real-time data to allow easier generation and curation of data resources by teachers, and greater usability for learners. We are, of course, among several who use realistic data to increase student motivation [3, 6].

The student’s self-efficacy is addressed by providing carefully scaffolded systems that allow incremental exposure of more challenging aspects of the subject matter. Scaffolding is provided in two ways. First, the Corgis framework uses layering. Our own experience has shown that "big data", while highly motivational, is too complex for early assignment where the student’s skill set is still limited. Layering allows the student to work with data at varying levels of complexity as their knowledge and skill increases. Second, we are developing BlockPy [REFS], a programming environment with mutual translation between a visual block-based language, Blockly [REF], and text programming in Python. Students’ initially develop algorithms in Blockly and, as their skill and knowledge increases, gradually begin viewing and creating their algorithms in both forms until they are ready to advance to programming purely in Python. This scaffolded environment allows students to follow the use-modify-create sequence with successively more technical and challenging aspects of computation being exposed at each step (i.e., moving from a point-and-click interface to block-based programming to textual programming).

We address the aspect of attention through interactive visualizations. These interactive elements allows a student to gain insight into the dynamics of algorithms and the manipulations of data. We have developed a significant collection of such visualizations in our OpenDSA framework [61]. We propose to develop corresponding visualizations appropriate for big data (our primary form of realistic data). For example, a visualization that shows operations on a list of ten elements is not sufficient for a big data stream with thousands of elements. We also address the challenge of incorporating visualization capabilities into the block-based language environment (BlockPy).

Immediate feedback is important to improved learning. Many e-book platforms, our own included, provide immediate feedback on questions with highly structured answers (true/false, multiple choice) or on program output (by comparison to hidden correct answers). However, feedback on algorithm design is usually done manually by instructors with the resulting loss of immediacy and degradation of the learning opportunity. We propose to incorporate program analysis into an e-book platform so that immediate feedback on small programing exercises can be provided. We will develop a mechanism for analyzing targeted code that is developed by students as an answer to posed questions (e.g., "Write an algorithm that..."). An important challenge in this work is to develop an authoring tool that allows salient features of the code to be described and related to varying forms of feedback. The provided feedback should be accessible both to the student and to instructors for additional comments. We will make this capability available within the BlockPy environment which, through LTI, can be incorporated into Canvas.

Finally, a well-established, flexible *instructional design method* will be used to create a principled curriculum design for the core computing concepts being taught in our computational thinking course (described next). Using a well-established instructional design method will improve the curriculum we have developed and also provide a guide for how elements of this curriculum could be adopted by others [REF] who are teaching a similar class or an introductory computer science class.

Our primary curriculum target is a recently developed course in computational thinking that has served students from over thirty different majors and with strong gender diversity. This course is a general education course having no prerequisites and making no assumptions about prior computer science or programming experience. This course defines computational thinking as:

*Computational Thinking = Abstraction + Algorithms*

Students learn about these core computing concepts using active learning in peer-learning groups. The groups are assigned to maximize the diversity of majors within each group to promote learning across contexts. Students are gradually introduced to programming through the BlockPy environment. Each student defines and completes a major project in Python using the “big data” resources in the Corgis library. Preliminary assessment of the course is given in Section 2.4. The complete materials for our course can be found at think.cs.vt.edu/book (choose the “browse as guest” option on the login page).

Extensive *multi-methods assessment* of our work will involve two distinct student populations at Virginia Tech and different student populations at four other U.S. institutions and two international universities. At Virginia Tech we will involve: (1) students in a newly created Introduction to Computational Thinking (CT) class that is open to all majors, including majors in all STEM fields and (2) more advanced computer science majors confronting the conceptual and practical intricacies of algorithms and data structures (CS3). These later students are relatively committed to the field but need help in seeing the application of the techniques they are learning to real-world situations and need better help coping with the more challenging cognitive dimensions of the material they are learning. Each course is offered each semester with enrollments ranging from 30 to 80 students. PI Kafura will teach the CT course each semester during the project. Co-PI Shaffer will teach CS3 at least once per year during the project. Also involved in the assessment are four other U.S. institutions (Lehigh University, Virginia Military Institute, University of Texas El Paso, and University of Delaware) and two international institution (Korea University and Escuela Superior Politécnica del Litoral in Ecuador). Support letters from these institutions are provided in the Supplementary Materials. All of the Virginia Tech researchers have experience with human subject research assessment and area currently doing IRB approved research. Our team also includes experts in educational assessment and in instructional design, both from the School of Education.

Our resources will be available in the Canvas learning platform. It is a stable open platform with a large user community and significant developer support. It has a wide variety of pluggable components that can be added through the standard LTI interface.

**Broader Impacts**

Exposure of students to big data provides the "data literacy" described in the National Research Council Workshop Report on "Training Student to Extract Value from Big Data". This form of literacy informs both future computer scientists and future domain specialists. As the report notes: "Students often do not recognize that big data techniques can be used to solve problems that address societal good, such as those in education, health, and public policy". We believe that the use of real world data and its related social impacts, while advantageous for all students, are especially engaging for students in populations currently under-represented in the computing community. Commenting on a number of studies [10-13], Goldweber et.al. write that “there is some evidence to suggest that success in broadening participation may be improved when computing is shown to connect with students’ values rather than their more superficial interests.” [14].

Our dissemination and assessment work will include use of our curriculum and technology at six other sites where faculty have agreed to be engaged with us (see supporting letters) as participants. The variety of contexts, courses, and student populations across these collaborating universities will provide a rich assessment and indicate how further dissemination can be elevated. Local to our institution, we will also use tailored big-data activities in a Women in Computing Day for K-12 students sponsored annually by our department to help inspire young girls toward computing and STEM study and career choices.

Another impact comes from the access to big data streams through a programming environment integrating block-based programming language with mutual translation to Python. This work adds value to instructors using block-based languages, allowing them to incorporate more realistic and motivating assignment and projects.

Our technology work also makes two broader contributions, especially to the community of authors in the Canvas community. First, the integration into Canvas of the powerful algorithm and data visualization capabilities developed in OpenDSA adds a new tool for constructing dynamic and engaging content. Second, the provision of program analysis and a related authoring tool adds a powerful new tool for instructors to develop better instructional resources. Finally, the data streams and the visualization tools developed for programming big data are useful not only for those using big data in introductory courses but are also useful to instructors in data science courses.

1. Background – Related and Preliminary Work

2.1 Motivation and Engagement of Big Data

Our work to create *authentic, real-world learning experiences* is grounded in well-researched educational theories of cognitive and motivational concerns. We leverage Socio-Constructivism theories of knowledge for the former, but the latter is more complicated. We hypothesize that introductory students begin with holistic motivational problems and end with more specific self-regulation problems. As student’s progress through a discipline they become more naturally engaged with the material, but still haven't fully developed the needed metacognitive tools. Therefore, we use the MUSIC Model of Academic Motivation [15] and as the underpinning for our work.

Socio-Constructivism is an evolution of Constructivist learning theory that emphasizes the role of context in learning. Constructivism, which has already seen some application within Computer Science Education [17], posits that knowledge is actively and recursively constructed from prior knowledge rather than being passively absorbed through direct instruction and textbook readings. Although both theories suggest the use of Active Learning techniques with rapid feedback and enhanced agency of the student, Socio-Constructivism emphasizes the value of culture within the learning process. This culture can come from the instructor (as both a guiding presence and a source of direct instruction), the classmates (who share the learners inexperience but bring their own skills, history, and understanding to the table), and society at large (with its generations of resources, impetuses, and conventions). One way that this culture is made concrete within the learning environment is Anchored Instruction, an approach where a problem is embedded within a frame story (the anchor). Instead of decontextualized, abstract experiences, students must think critically within realistic scenarios that are easier to construct their knowledge upon. Socio-Constructivism is applied within this proposal to suggest the value of Social Impacts and strongly influences the technology to be developed.

The MUSIC Model of Academic Motivation is specifically designed to explain engagement in education, setting it apart from more domain-unspecific motivational frameworks. Derived from a meta-analysis of other motivational theories, the model is meant for both design and evaluation and has been extensively validated, making it a reliable device [18]. The MUSIC model identifies five key constructs [15]:

* Empowerment: The amount of control that a student feels that they have over their learning.
* Usefulness: The expectation of the student that the material they are learning will be valuable to their short (tactical) and long term (strategic) goals.
* Success: The student's belief in their own ability to complete elements of a course with the investment of a reasonable, fulfilling amount of work.
* Interest: The student's perception of how the assignment appeals to situational or long-term interests.
* Caring: The student’s perception of other stakeholders' (e.g. instructor, classmates) attitudes toward them.

Students are motivated when one or more of these constructs is sufficiently activated. Students' subjective perception of these constructs is more important than objective reality. The MUSIC Model of Academic Motivation Inventory (MMAMI), a well-validated instrument, is used to measure engagement through these five aspects.

2.2. Automatic Feedback and Interactive Visualization

A dynamic process, such has the behavior of an algorithm, is difficult to convey using static presentation media such as text and images in a textbook. During lecture, instructors typically draw on the board, trying to illustrate dynamic processes through words and constant changes to the diagrams. Many students have a hard time understanding these explanations at a detailed level or cannot reproduce the intermediate steps to get to the final result. Another difficulty is lack of practice with problems and exercises. Since the best types of problems for such courses are hard to grade by hand, students normally experience only a small number of homework and test problems, whose results come only long after the student gives an answer. The dearth of feedback to students regarding whether they understand the material compounds the difficulty of teaching and learning computer science.

For this project, we will build several content modules related to ethics and big data using OpenDSA technology [61]. OpenDSA modules combine content in the form of text, visualizations, and simulations with a rich variety of exercises and assessment questions. Since OpenDSA modules are complete units of instruction, they are easy for instructors to use as replacements for their existing coverage of topics (similar to adopting a new textbook). Since OpenDSA’s exercises are immediately assessed, with problem instances generated at random, students gain far more practice than is possible with normal paper textbooks. Since the content is highly visual and interactive, students not only get to see the dynamic aspects of the processes under study, they also get to manipulate these dynamic aspects themselves. Emphasizing student engagement with the material conforms to the best practices as developed through more than a decade of research by the algorithm visualization research community [27-29].

Each module includes mechanisms for students to self-gauge how well they have understood the concepts presented. Self-assessment can increase learner’s motivation, promote students’ ability to guide their own learning and help them internalize factors used when judging performance [30, 31]. We do make use of simple multiple choice and give-a-number style questions. But we also include many interactive exercises. We make extensive use of “algorithm simulation” or “proficiency” exercises, as pioneered by the TRAKLA2 project [32]. (The TRAKLA2 developers from Aalto University in Helsinki are active participants in OpenDSA, having developed the JSAV graphics library [33, 34] and several OpenDSA exercises.) In algorithm proficiency exercises, students are shown a data structure in a graphical interface, and must manipulate it to demonstrate knowledge of an algorithmic process. For example, they might show the swap operations that a given sorting algorithm uses. Or they might show the changes that take place when a new element is inserted into a tree structure. Other OpenDSA exercises make use of small simulations for algorithms or mathematical equations to let students see the effects that result from changing the input parameters. We have experience with small-scale programming exercises that are automatically assessed for correctness. These problems are similar to small homework problems traditionally given in such a course, but which have been hard to grade.

2.3 Instructional Design

The use of well-established instructional design approaches is motivated by two observations. First, computer science education has taken advantage of constructivist theories of education that underlie such pedagogical practices as peer learning. We proposed to take advantage of instructional design methods to similarly improve computer science education. Second, just as quality software is produced through rigorous software engineering practices we believe that quality learning environments are produced through rigorous “curriculum engineering” provided by instructional design. Prevailing well intentioned and sometimes successful, but ad-hoc, curriculum design can be dramatically improved by systematically applying known instructional design methods. Just as software developers use methods of analysis, specification, design, implementation, and testing, curriculum developers should use parallel methods: defining instructional goals and analyzing learners and their context (analysis), creating performance objectives (specification) developing instructional strategies and assessment (design), creating instructional materials (implementation), and conducting assessments (testing). Just as with the software engineering life-cycle has iterative improvement based on feedback from later stages, instructional design also has a “life-cycle” that improves curriculum based on assessment of its use in the classroom

Instructional Design is a subfield of education concerned with the systematic design of learning experiences that give measurable results by following a well-defined process. Although Instructional Design has existed for decades and is popular in other domains, it has seen extremely limited application in Computer Science. Much of this existing research ironically only suggests how Software Engineering techniques can be applied to Instructional Design [3, 4, 7], while the remainder approaches the subfield by focusing on pedagogical tactics [1, 6] rather than holistic, cohesive processes.

Similar to the diverse methods in Software Engineering, there are many models of Instructional Design. The Dick and Carey model is a popular model for introductory Instructional Designers because of its rigorous structure [2]. This model consists of 9 major phases. The first three phases require the designer to analyze and formally specify the instructional goal, the learners, and the context of the learning environment. The next two phases have the designer precisely identify each of the composite performance objectives their learners must achieve, and then develop assessment instruments to concretely measure those objectives (these instruments are used as pretests, practice material, and posttests). Only then can the designer plan out the high level instructional strategy and develop their instructional materials in the sixth and seventh phases. The Dick and Carey model suggests using Gange's Nine Learning Events as a systematic guide to developing these materials, gracefully synthesizing presentation, practice, and feedback; however, there are a wide range of learning and motivational theories that are applicable in this phase. Finally, the last two phases guide the developer in evaluating their materials formatively and then summatively, emphasizing the iterative nature of this process.

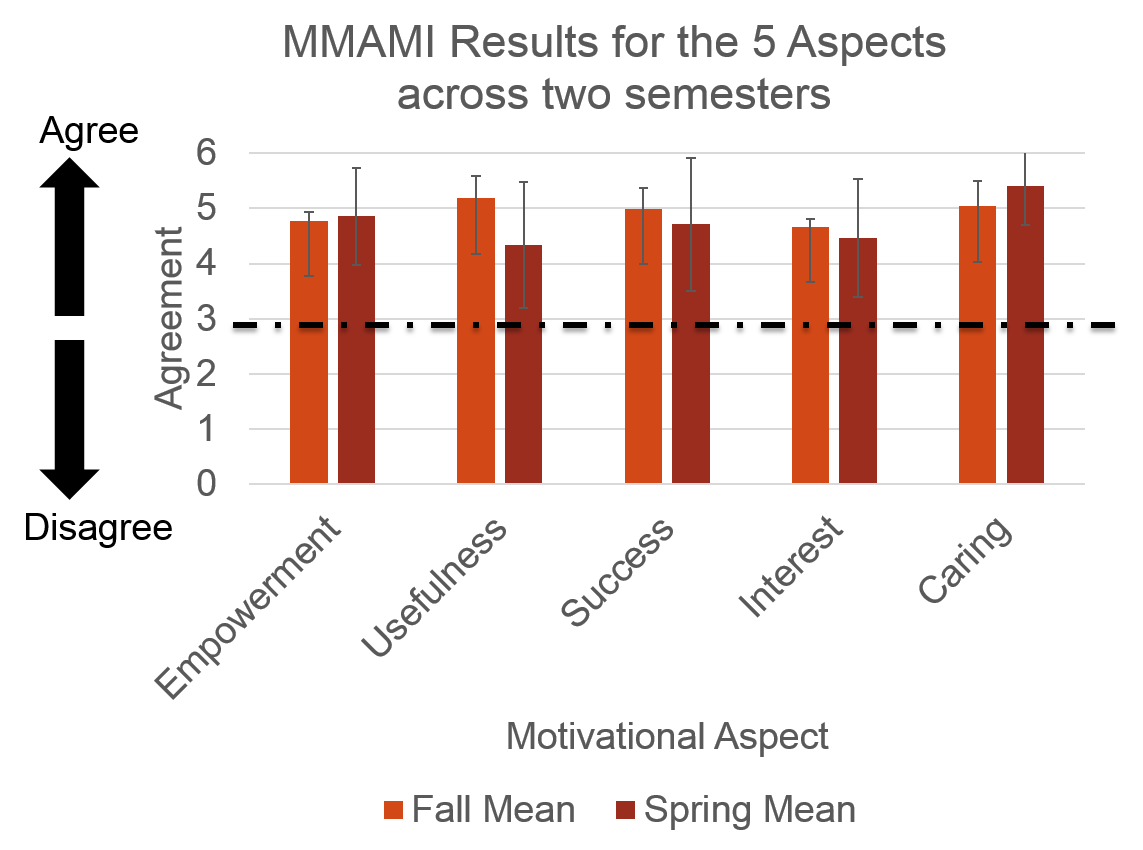
The PIs have previously applied Instructional Design in two trial interventions with excellent results: (1) to create a new unit of an existing Computational Thinking course for non-CS major undergraduates, and (2) to create a new five-day introductory workshop on Computer Science for high schoolers. In the former setting, the model was applied rigorously to correct a perceived gap in students' learning; in the latter setting, the model was applied loosely to create a new instructional experience from scratch. This differentiation represents the degree to which the model can be adapted to the constraints of the instructor. A survey of the literature has revealed no prior, formal attempts at applying this technique to CS Education.

Instructional Design methods not only lead to tight, focused instruction to the benefit of the learners, but also gives detailed metrics on the impact of the course materials. Formative Evaluation is a crucial component of Instructional Design, required at multiple levels using a pre- and post- assessments. In the two case studies, small group and field trials were conducted with real learners to gather data that demonstrates the students had significant learning gains with an average improvement of up to 46% in some areas. However, even more crucially, the thorough planning identified a number of inadequacies that would not have been accounted for otherwise, before developing instructional materials.

A common problem with course dissemination is that the curriculum developers usually only share the course materials (e.g., PowerPoint slides, handouts, project write-ups, etc.) rather than a high-level instructional strategy, learning objectives, and other key resources. The formal methods of Instructional Design are self-documenting at every phase, so that curriculum adopters are given materials with the context and justifications of the materials.

2.4. Preliminary Results from the Computational Thinking class

The preliminary results from the first two offerings of the class in the Fall 2014 and Spring 2015 semesters include an analysis of student motivation and the use of cohorts. Survey results are from XX students in the class, AA% female and BB% male. There was a wide variety among majors, with students in psychology, political science, English, mathematics, theatre, university studies, and other disciplines. To assess the motivational impact of our pedagogical approaches and technological innovations, we surveyed students with the MUSIC Model of Academic Motivation Inventory (MMAMI). MMAMI is a validated instrument for measuring students’ beliefs related to the five key components of the MUSIC model [27]. The version used in our course consists of 26 statements that students responded to on a 6-point Likert scale (ranging from “Strong Disagreement” to “Strong Agreement”). The responses are then averaged into subscales relating to each of the components of the MUSIC model – eMpowerment, Usefulness, Success, Interest, and Caring. Examples of the statements include:*“The knowledge I gain in this course is important for my future.”*  and*“I enjoy completing the coursework.”*

As a baseline measure of success, the results from the MUSIC inventory suggest that students were overall motivated in this course. Students reported high average scores in all five areas of the MUSIC model, with no strong standard deviation. The results, shown in Figure 2, indicate that students “Agreed” in the belief that they were empowered, able to succeed, cared for, and that the course was interesting and useful. Our interpretation of this data is that, at a minimum, this course was successful in engaging students.

To better understand the quantitative data, qualitative data about the class was also collected by observing students working in cohorts during class time and by interviewing 9 students of the class at the end of semester. Group observations (13 hours) and interviews with students (9 students) suggest that working in a cohort was beneficial. While some class activities were designed to be collaborative, students in their cohort usually worked on individual problems. If a student got stuck with a problem s/he would ask other members in their cohort for help. In some cohorts a more active student would inquire if other members were stuck with a problem. Students felt it was easier to ask help from peer members because they were at the same level of learning.

Figure 2 Motivational Data

*“It’s nice to have other people that are in a similar level of learning to you so you can bounce ideas off each other as opposed to get an explanation from someone who already know the materials and is trying remember what it is like not know the material. So it’s getting a better explanation from someone that is closer to where you are… “(Student1)*

Help usually was offered in the form of explanation instead of providing the answer.

*“If we are doing individual work we usually break off and solve the problems. If we have difficulty we ask the other members. Usually if we get an explanation that is more about the concept as opposed to the individual problem we had. So say if we got a problem with ‘if’ statement, we get an explanation on why our ‘if’ statement wasn’t working as opposed to the right way to write that individual ‘if’ statement. There is more learning the ‘whys’ as opposed to the ‘what’ I guess”(Student 1)*

Forming cohorts with students from different disciplines allowed students to better understand the application and implications of CT across disciplines.

*“It offered different perspectives. When we were working with Netlogo and how we chose a view point, like a program that we can relate to our major. I know the biology major did one on AIDS and how it spreads and the other two on voting and voting habits. And I did something on networking… it was good to open up and see different perspectives and how programs can be applied to different focuse” (Student 7)*

*“Since we all are working on different projects it is kind of interesting to see what we can do with the data. So like while my one is working on voting habits and government, I think one of the other guys is comparing literature and it is just like how you can approach problems in different ways…” (Student 4)*

Apart from understanding concepts, students also found cohort members useful in locating technical resources or explaining how to use certain features of a course resource.

*“In the beginning of the Blockly program, the airplane, the diagram, all of that – I really did not know how to do it. It was easy, but I really did not know how to start it. So I asked my team member how to start. He explained to me how to start and after that I was able to do it easily. So it was basically getting to know the basics of how to start the program and then I was able to do it.” (Student 5)*

Students also appreciated the presence of the instructor and co-instructor.

*“The basic understanding, solidifying the basic understanding of the underlying principles of programming –that is not something most people (instructor) will go over, at least at this level I guess. Having that explained with someone there, who knows the material and is willing to explain it further, that was just really helpful…” (Student 1)*

Students of this class stated that taking this CT class has helped them realize the role of computation in their major.

*“Taking this course I now realize how much the modeling that we do in python is being used by people in my major and is seen as a valuable skill to employers …I did not know (before taking this course) how thorough and how much it would tie into my major until I took the class…”(Student 3)*

2.5 Related work

We share with the Media Computation approach [37] the idea of providing a unifying, open-ended resource (images and sound are used in Media Computation vs. big data in our course). However, we believe that big data is seen by students as “useful”---which is more engaging---while media computation is seen as only “interesting” [38]. We share a common goal with courses that use real-world data for motivational purposes. Examples include using on-line data [39] and assignments that produce useful tools [40, 41]. We also share a concern for using resources that relate social impacts. Among this work is [14] that uses a values-oriented approach to exploring the social implications in various computational modeling assignment and [42] that uses sustainability issues to frame problems used in a data structures and algorithms class. We differ from these approach in the resource (big data in our case) to which the social concern is connected. We share the most with the work of [3] that also uses big data in such areas as life sciences, political science, and social media. Like us, they have also explored allowing students to choose the data for a major project. In two of our courses we also have a shared view with courses that used block-based programming (Snap!, Scratch, or App Inventor).

We propose to extend prior work in a number of ways. Our proposed work integrates and extends these concerns for engagement, realism, social impact, and big data. The integration is achieved by extending the "raw" big data streams with elements connected to a model of social impacts. The extension involves the development of interactive visualizations, static analysis for immediacy of feedback, the access to big data through two different block-based programming languages, and other supporting technology. We also extend the application of this approach outside of mainstream Computer Science education to the general university student population via a Computational Thinking course. Finally, our assessment data will add to the body of knowledge on impacts and limits of the big data approach. We add to a block-based programming approach the connection to realistic big data sources and the ability to embed the programming in a “book” form to better integrate learning materials (see Section 4).

1. Proposed Work

3.1 Enhanced Architecture for Realistic Data

Data Science is a non-trivial context for introductory learners, due to the difficulty in finding, preparing, and delivering data to students in a pedagogically suitable form. Integrating it into a learning experience has already shown to be a tricky experience. The previous work of the PIs has led to the CORGIS project, a collection of highly scaffolded libraries for a wide array of different realistic data sources. In this subsection, we outline three challenges related to this approach of scaffolded datasets, and the planned technology to overcome them:

* CORGIS Architecture: An evolution of the existing library architecture that handles new use cases and technical issues.
* CORGIS Gallery: An evolution of the existing library gallery that makes it easier for students and instructors to find relevant datasets.
* CORGIS Builder: An evolution of the existing library builder that makes it easier for developers to prepare data sources.

**Research Question 1: CORGIS Library Architecture**. How can we build and maintain a plethora of divergent datasets, that still ensure students have a uniform experience working with their dataset even in the presence of divergent hardware and datasets?

The primary value of the CORGIS project lies in the diversity of its datasets, giving students an empowered opportunity to find a dataset that appeals to their interests and long-term goals. The CORGIS library currently has over 35 different datasets, including animal feed data, weather reports, historical disease tracking, and much more. However, there is a large burden on the developer to create these datasets, requiring technical, pedagogical, and domain proficiency. And once these libraries are developed, they must be maintained: web-based libraries need to stay current with their API, local libraries need to stay compliant with new hardware, and every library must be amenable to bug fixes and refinements. Finally, these libraries have to be usable by students no matter what kind of hardware they have and whatever permissions they have on the machine.

Although the earlier work of this project greatly simplified the process of creating web-based libraries by using configuration files to generate libraries, this approach was meant as a first step in development rather than a terminal step. Once a library has been generated, it becomes an independent code base with its own copy of the structure needed to access its data – in-lining a tremendous amount of code. Worse, the novel features of the library become mired in boilerplate and library specializations. For example, consider the differences between the Gutenberg Books library and the Weather library, both of which expose a single function that connects to an online data source and returns a data structure mixing lists and maps: these two libraries are significantly similar except for their initial method to submit the web request and their final method that processes the retrieved data into the proper form. When an update is made to the web API, the developer must hunt down these two functions and make modifications, navigating a mess of boilerplate. Even worse is when improvements are needed to the core architecture. Despite sharing a common general architecture, each library is an independent code base with minute modifications. Therefore, a change to the architecture must be percolated to three dozen other codebases.

A second problem with the current architecture is the disorganization of the documentation and metadata that is associated with each library. Getting to know an API is a difficult process akin to learning to a new language. Supplementary documentation, including tutorials and API references, are necessary. The project provided tools for documenting the libraries it generated, but these were limited to creating simple API reference materials that were not adaptable to different levels of learners and did not instruct the learner on its use; creating tutorials to use the libraries was a manual, cumbersome effort that was redundant across similar libraries. Further, there was absolutely no tooling to generate supporting documentation related to metadata for the library – information such as origin of the data, explanation of terminology used, and terms of its use had no structured representation.

The third major problem is that students using our software have diverse computational power: seniors might have a laptop from their freshman year, or run an older version of MacOS, or make do with very limited memory. Ideally, the students should always be able to run their code quickly and efficiently while developing, without noticeable lag from their programs. However, several existing CORGIS libraries suffer greatly from bad caching strategies and poorly sized datasets, resulting in poor performance that can frustrate beginners. A 100 MB library that runs fine on a developer’s new machine can be treacherously slow on a students’ ancient laptop.

**Research Question 1 Proposed Work**: Create a new architecture that simplifies the creation of dataset libraries (whether web-based or local), while being highly maintainable to developers and performant for students.

In effect, all of the CORGIS libraries will be generated from high-level specifications; generated libraries will be treated as read-only codebases, with all refinements and errors being made in the source specification. The advantage of this approach is the assurance that improvements percolate across all instances of the library, across languages and formats. These specifications incorporate structured rich metadata and work against prepared data files, with an interface specification to indicate how students can access the data. Further, the library will be able to analyze the architectural suitability of the host machine and make intelligent decisions to adapt to the hardware. For example, if the software found that the students laptop had little RAM and a poor processor, it might decide to sample down the dataset, or to process more data on disk.

**Research Question 2: CORGIS Builder Architecture**. How can we lower the barrier for instructors and domain experts to transform a data source into a classroom ready resource?

There is still too high a barrier for instructors to transform a data source into a classroom ready tool. Currently, the CORGIS web-based builder scaffolds the process writing a specification file, it has no features for preparing local datasets or inferring the structure of web-based APIs. Preparing a dataset is an ad hoc process of converting between data formats (e.g., JSON, CSV, SQL, etc.) into something manageable, requiring decisions about what fields and instances to keep, how the data should be structured hierarchically, what type fields should be, and how data should be pre-aggregated for students.

A further limitation is that the RTW Builder has no support for the process of building data caches. Instead, the instructor has to use the individual library to build up data caches using a poorly documented internal tool. This tool works in a cumbersome “VCR recording”-style, where the user runs the queries they’re interested in retaining in real-time. There is no way for the instructor to create artificial data caches matching their use case, without resorting to writing their own completely custom scripts.

Finally, different datasets have wildly varying structure depending on the nature of their data. A student working with social media data may find the data to be recursive or tree structured, as opposed to a student with more tabular data working on sports statistics. Although this may be expected and natural, it is not desirable or necessary for students to have wildly divergent experiences with learning the structure of their data. If there is a uniform shape to the data, the instructor can have confidence and knowledge in providing technical and pedagogical support, no matter which student they are helping. Inversely, they can give a more uniform lecture that is accurate for all of the students.

**Research Question 2 Proposed Work.** An evolution of the RTW Online Building tool to make it even easier to prepare a JSON/CSV data source into a library.

This new version of the online building tool will have features to reshape and organize a dataset, including ways to create data caches and artificial reworking of the dataset according to instructor-supplied constraints. Specifically, the tool will be able to work with several different data formats, including CSV, JSON, SQL, and TXT, and be able to infer definitions to connect to online APIs. Instructors will be able to write commands and queries to transform the data according to certain common functions or by using a regular query syntax. Additionally, the tool will be able to manipulate datasets to have a desired shapes by restructuring the data according to certain common templates and high-level instructions given by the instructor. For example, consider a dataset of car makes and models over years with the columns ("year", "make", "model", "company"); this dataset could be grouped by year in order to make it easier for students to create bar charts in Matplotlib (which would otherwise require the student to write grouping code). The other crucial new feature of the builder will be the ability to specify constraints and rules to generate artificial data for testing or to expand a data source, such as fake weather reports for the weather library. This use of mocking is a powerful way to provide more controlled learning experiences for learners. The output of the tool will be pluggable datasets and interfaces suitable for the CORGIS architecture, rather than specific language bindings.

**Research Question 3: CORGIS Gallery**. How can we support students’ and instructors discovery of their data source?

Instructors have a choice to assign students a specific library, a choice of libraries, or to give students freedom to choose their own library. There are motivational and pedagogical trade-offs to consider, but this decision lies with the instructor. The goal of the CORGIS library is to allow the instructors to be as flexible as they want in assigning a dataset. Currently, the list of CORGIS libraries is represented as a flat list of the libraries’ names in a wiki structure ^1. Each library has an ad-hoc page of information which may or may not include code examples, library description, and a link to the source code. As the selection grows, this informal representation becomes more and more inadequate for finding a suitable library and learning more about its nature. Originally, the gallery was a dynamically generated page based on a separate specification of the libraries (separate from the libraries themselves) – this was too difficult to keep in sync with the libraries as they changed. The Wiki technology was adopted to make it simpler to make quick updates, but that just exacerbated the problem of keeping the library documentation up-to-date.

**Research Question 3 Proposed Work:** An enhancement to the RTW Online Gallery to make it more interactive and guiding for students to discover their datasets

We propose to make a new version of the gallery for the CORGIS project with three design goals:

1. Support instructors and students finding a suitable dataset. Provide features for both browsing and searching for libraries, especially for students who might have limited domain knowledge.
2. Support students looking up information about a library. Provide accessible information about the origin of the data source, the abstractions that it uses, citation data, information about the data’s structure and fields, the interface exposed to access the data, any important limitations and features of the dataset, and other metadata relevant to the learner.
3. Keep the publicly available information of a dataset in sync with the datasets source. In particular, make it easy for developers to update the dataset or the metadata for the dataset, without requiring interaction with the server.

Once the solution is implemented, it will be evaluated based on case studies of creating new datasets and analyzing the work required to update existing datasets. The impact on the students’ experience will be analyzed through usability studies: students will be interviewed about their experience learning about the metadata and problems that they encountered while getting to know their dataset. More information on these usability studies is given in Research Question 2.1.

1 http://think.cs.vt.edu/wiki/index.php/Category:Library

3.2 Immediate Feedback and Interactive Visualizations

We discussed above the value of automatically assessed practice exercises. In a classroom of many students and minimal instructors, it is a challenge to keep students supported with timely feedback and engaging content as they work. Most students lack the programming knowledge to identify their own errors, let alone diagnose and fix them. Further, they often disengage with static material such as textual prose. Although 1-1 human tutoring is ideal for learners, there are simply not enough staff resources. For many kinds of student problems and learning resources, however, the system can support and engage the student.

A specific type of exercise that we focus on will be small programming exercises. There is considerable work on automated assessment of programming exercises via testing [51-53]. Most take the student's solution to a "sandbox" where the solution is compiled and executed. A standard approach is to use something equivalent to unit tests to make sure that the student's solution has the correct behavior. WebCAT [18] is a well-known tool for solving this problem – it uses style checks and unit testing to identify mistakes and make suggestions to the student. A limitation of systems like WebCAT is that they often depend on a highly restrictive code shape: usually, they operate over explicit function contracts. They are also unable to collect data about the internals of the students' code, treating the program as a simple black box.

We have found that it is also helpful to give students incremental feedback on much more ad-hoc programs, beyond writing functions against a contract. For example, consider tasking students to average a list of numbers through iteration: students should know to use certain constructs including an external variable, a loop, and mathematical division. In such a problem, static analysis heuristics could be given by an instructor that specify those constructs must be present – if the student tries to compile their program without it, they are directed to “just-in-time” instructional materials regarding the constructs. OpenDSA and BlockPy both already have limited support for static analysis heuristics of this nature.

In our piloted prototype systems, the automatic analysis is limited to regular expressions that are checked against the student’s code, e.g., requiring iteration by checking that the word “for” is present in the text of the program, at the start of a line, and with whitespace immediately afterwards (signaling its use as a programming identifier). This kind of analysis is limited – even complicated regular expressions are unable to account for constraints such as “initialize a variable outside of a ‘for’ loop, and then modify that variable inside the loop’s body”. Worse, it is delicate and unsustainable, since every exercise requires unique analysis that is difficult to modify.

We propose a flexible system of reusable heuristics that can help instructors define and then enforce constraints on students’ code by indicating specific errors and giving targeted feedback individualized for the student. These heuristics will operate over parsed abstract syntax trees, so that the program can be intelligently analyzed. Prioritization of the suggested heuristics, matched with targeted support, can provide a guiding experience for students. For example, the instructor might specify the following constraints and priorities in a problem requiring the student to write an average function:

1. High priority: Use a ‘for’ loop in their code. Link to section in textbook on iteration – unless they’ve already read that section recently, in which case directly state that they haven’t used a ‘for’ loop.
2. Medium priority: Define a variable X outside a ‘for’ loop, and then modify X inside the ‘for’ loop’s body. Link to section in textbook on using variables for state – unless they have been struggling a lot with this problem, in which case suggest they ask an instructor.
3. Low priority: Ensure that the variable X is used after the ‘for’ loop in a mathematical division operation. Link to Wikipedia article on the formula for finding an average – unless they have already been given that hint, in which case explicitly suggest that they still need to divide.

This API could incorporate Intelligent Tutor approaches by modelling the learners (e.g., prior mistakes, time on task, estimated motivation, etc.) to provide more contextualized assistance – if a student consistently iterated over an empty list across multiple different problems, for instance, this could be representative of a greater misunderstanding rather than a simple typo, and small hints would be less useful to their long-term learning than instructor intervention. To improve this API further, we intend to identify common misconceptions and negative patterns of student behavior, informed through data mining collections of student code. For the past year, we have been recording the entire history of students' programs and the interaction log with our systems. Our hope is that by investigating this repository of code, we can infer common patterns that suggest undesirable learner behavior and suggest interventions to guide the student to success. For example, users that frequently move the same blocks without progressing in the problem objectives (``churning'' the interface) might be indicative of taking longer on the problem than other users. Alternatively, students who pick a decision block to complete a problem about iteration might commonly fail to complete a problem.

Program analysis informs the nature and extent of the immediate feedback to support students. Although hints and suggestions are obvious kinds of feedback, there are other mechanisms too. Students can be reminded of course goals that they have established (appealing to their long-term objectives). Automated, self-regulatory suggestions can be sent to the learners when they have not engaged with the materials in a while, suggesting that they try out a prob-lem for a while to see if they can make some progress (or at least identify their misconception) or move onto lateral material so they can make headway elsewhere. Staff can be notified of a particularly struggling student, in order to encourage them to reach out with suggestions or encouragement. Finally, situational interest could be appealed to with motivational content such as inspiring quotes, amusing images, or other rewards.

There are other techniques that can be applied to give feedback; for instance, the system can infer the types of most variables. Simple dynamic analysis of program trace profiles can be used to infer the type of a property over the course of the programs execution. Certain restrictions can be made towards beginner’s codes that, while draconian to an experienced developer, are reasonable for someone starting out. For instance, we might forbid dynamic retyping - a property can only take on a single type. In general, code analysis struggles with dynamic language features and large code repositories. However, in an educational setting, the students' code is a very manageable size and should always resemble some specific form, which is critically useful information to guide the analyzer.

In addition to the type inferencing, symbolic execution can also be used to support output and unit testing. The data libraries of BlockPy provide a useful mechanism for conducting behind-the-scenes testing by operating on a fixed domain of outputs. A student submits code such as in 10. This code depends on a special CORGIS library that returns weather data, in particularly retrieving the weather for a specific city. When the student’s code is evaluated by the system, this libraries functions' can be evaluated symbolically, ensuring that the students’ code returns the proper value integrating this symbolic value. This approach avoids more easily-abused output-checking, which students can fool by printing literal values as opposed to computed values. This approach provides the validity of unit testing without the resulting inflexibility.

To improve the interactivity of content, the OpenDSA project has developed a sophisticated support system for developing rich interactive visualizations in HTML5 [33]. We have already developed algorithm visualizations and exercises for much of undergraduate data structures and algorithms content, and will be expanding this content to more advanced topics in programming languages, finite automata, and complexity theory in the coming year under other NSF supported projects. This rich body of materials and expertise in their creation will allow us to create a collection of materials appropriate to the non-major students in the CT course. Many of the fundamental data structures and themes relevant to big data are already covered within the existing collection of materials. We will be able to tailor those materials and create new materials more appropriate for this constituency. The OpenDSA infrastructure allows us to create small-scale simulation environments and interactive exercises that will let students "play with" big data streams and principles.

3.4. Curriculum Development Using Instructional Design

We propose to use the Dick and Carey [REF] model to provide a principled instructional design for the computational thinking curriculum. We choose the Dick and Carey model because it is one with which we have some exposure and for which there is expertise available at Virginia Tech. Included on the research team is an expert in Instructional Design who has deep understanding of this model.

The proposed instructional design is done for four reasons. First, we are introducing the fundamental computing concepts in the context of real-world “big data”. While not unique, the preponderance of computing curriculum have been developed with other points of view (e.g., “toy” examples, games, multimedia) or no clear overarching theme. Thus, applying the disciplined approach of instructional design will help us to insure the highest quality of curriculum development. Second, instructional design will help us to continue the continuous assessment and improvement of the course because the model has a clearer connection between assessment and learning objectives. Third, the completed instructional design for a concept (e.g., “properties” or “abstraction”) provides a guide for how this design (including its materials, methods, and assessment) can be adopted and adapted by others [REF]. Such dissemination is facilitated because the design exposes the deeper instructional rationale to potential adopters. In contrast, merely providing data streams, project ideas, or technology does not provide an adopter with a framework to understand whether or how these materials would improve the learning environment of their students. Fourth, we offer our experience as an exemplar for others in the computer science education community to emulate. In this way we help to spread the best practice of course design.

We specifically propose to use the Dick and Carey model to develop curriculum components for a list of key “threshold concepts” in computing. These concepts have two characteristics. First, each concept is fundamental to understanding computing. Second, they are concepts that are difficult ones for our students in the Computational Thinking class. The initial threshold concepts are:

* Abstraction: the idea of representing real-world entities by properties that are relevant to the needs or interests of a given “stakeholder” is not something that our students find natural. Currently, a table is used to concretely express an abstraction where a column corresponds to a property and a row corresponds to an instance (the collection of properties that represent a specific real-world entity).
* Properties: an objective, quantifiable, and measurable characteristic that can be define in information terms is not intuitively clear to the students in the computational thinking class. Properties include both those defined for an abstraction (e.g., the temperature forecast for a given city by the weather service) but also a computed result (e.g., the average of the forecast temperature over a set of cities). Our students struggle with both the fundamental nature of properties (what are the relevant properties, the distinction between a property and its value) as well as the mechanical aspects (e.g., how should a property be named).
* Iteration: the idea of repetitive execution of a set of steps until some condition is satisfied is one of the most difficult algorithmic concepts for our students. The changing value of the iteration variable and its relationship to the entire collection of values is difficult for our students to grasp.
* State: the idea that the values of properties change over time corresponding to the actions of the algorithm is a subtle relation. The difficulty of explaining or debugging algorithms often traces back to a lack of understanding of this fundamental idea.
* Data structures: the different ways of organizing and accessing data with complex relationships is a difficult learning task. This is especially the case when multiple layers of abstraction (nested or imbedded data structures) are involved.

We believe that these initial threshold concepts are important ones for all introductory courses in computing. We also anticipate that the initial list will be expanded as we learn from the experience with using instructional design and we gain additional assessment data.

1. Assessment and Evaluation

Progressions of learner motivation and engagement, outcome proficiency, and self-regulated learning behaviors, in the context of data rich course experiences, will be assessed in Virginia Tech’s CT, CS1, and CS3 courses. Specific research questions and metrics for the implementation and impact study are discussed in Table 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 2: Research questions and data sources** | | | | |
| Data Source | Pre/Post MUSIC | Pre/Post CT/CS | Pre/Post MSLQ | Semester  Interviews |
| RQ1 Do big data applications support increased classroom motivation and engagement? | X |  |  |  |
| RQ2 viz/feedback |  | X |  |  |
| RQ3 Does Instructional Design improve student learning of threshold concepts in computing? |  | X |  |  |
| RQ4 Does classroom motivation and engagement vary across student populations (CS major and non CS major), student demographical categories and social dimensions? | X |  |  | X |

**RQ1**: A motivational survey based on the MUSIC Model, developed in conjunction with the original creator of the MUSIC Model Dr. Brett Jones, will be administered in a pre-measure/post-measure format to gauge student increases in motivation and engagement. The specific constructs measured through the instrument are: 1) Empowerment, 2) Usefulness, 3) Success, 4) Interest (situational), and 5) Caring. This inventory consists of 5 open-ended, qualitative questions about the whole course, and 25 multiple-choice prompts relating the motivational constructs with high-level course components. Students select from response options on a 7-point Likert Scale, with options ranging from 1 – Strongly Disagree to 7 – Strongly Agree. Finally, the survey includes a small number of questions about students long-term plans related to computing and demographic questions (including prior computing experience). This instrument is used in favor of the original MUSIC Inventory instrument since it can be highly targeted to the course material. Study group students will be administered the instrument within the first full week of class and then again immediately after the intervention has been completed. Analyses of the repeated inventory measures will be conducted to determine significant changes within-subject comparison.

**RQ2**: Student achievement item-level outcomes from the 2014-2015 academic year will be examined in the CT, CS 1, and CS 3 course offerings. This measure is to determine intact degree of validity concerning measurement and reliability of result consistency for currently deployed student content evaluation. Also, this measure will assist in the formation of a non-treatment achievement baseline enabling further characterization of study group achievement progressions. Once item-level analyses have been conducted, a CT, CS1, and CS3 panel team (comprised of instructors of these designated courses) will be convened for the purposes of reviewing their specified content assessments.  After panel team content examination, each team will then build parallel items that will serve as project pre-assessments. The cognitive content achievement pre-measures will be administered to the study group prior to the onset of the intervention-based instruction and activities and then the post-measure administered at the completion of the planned intervention-based instruction and activities for the CT group, the CS1 group, and the CS3 group. Analyses of the measures will be conducted to determine significant changes within-subject comparison and non-treatment baseline comparison.

RQ3:

**RQ4**: Data from RQ1, paired with student major, level, demographics, and social dimensions will be used to determine if course experiences have variability in impact on student motivation and engagement.

Evaluation

CRQ1: Corgis build and maintain librarires [usability study] [partner feedback]

CRQ2 lower barrier for instructors [partner feedback]

CRQ3 Support student discovery [usability study]

Feedback: Evaluation of API/tools for creating automatic feedback [partner feedback]

1. Project Organization

The proposed work is organized as shown in the following table that identifies the major goals to be accomplished in each year of the project.

|  |  |
| --- | --- |
| **Year** | **Goals** |
| 1 | Develop additional big data resources  Develop (Fall) and trial run (Spring) of scaffolded environment for CT class  Refine Social Impacts model (Spring)  Develop automated feedback support and authoring environment  Port existing components to Open edX (Fall) and integrate (Spring) automated feedback support  Develop (Fall) and trial run (Spring) of visual environment (Greenfoot) integration  Dissemination activities |
| 2 | Assessment of motivation and self-regulation in CT class  Assessment of motivation and self-regulation in CS3 class  Develop enriched data streams by adding social impacts  Develop taxonomy of data streams  Develop interactive visualizations  Improvement of visual environment; assessment of motivation and self-regulation in CS1 class  Dissemination activities |
| 3 | Assessment in CT, CS2 and CS3  Analysis of data from multi-methods studies  Dissemination activities |

The project team is well qualified to carry out the proposed work. The team has a track record of collaborative research. Team members Shaffer and Ernst have collaborated on a previous educational project. Team members Kafura, Shaffer and Tilevich are currently collaborating on the use of big data in various courses. The team has extensive expertise with algorithm and data visualization (Shaffer), frameworks for big data and real time data access (Tilevich), computational thinking (Kafura), and web-based frameworks for course materials (Shaffer, Kafura). We also include on our team an expert in instructional design, Dr. Katherine Cennamo, with whom we have worked closely in the development of this proposal.

1. Dissemination

Our dissemination plan is organized around community-building activities supported by technology practices that facilitate adoption.

*Community-building Activities*. Our primary means of dissemination is through the four national partner institutions (Lehigh University, University of Delaware, Virginia Military Institute, University of Texas El Paso), and two international partner institutions (Korea University, and Escuela Superior Politécnica del Litoral in Ecuador). Letters of support are included in the Supplementary Documents section of the proposal. We have worked closely before with the faculty at Lehigh, Delaware, and UTEP. We will create a wiki-based web site as a central point for the distribution of all resources developed in the project. The wiki-based nature will also encourage contributions from the community of adopters. Natural ways to contribute are through the addition of new data streams, and new projects. We will promote awareness of the developed resources and the practice of instructional design through BOF sessions and workshops at primary computer science education conferences. Publication of the results demonstrating the benefits of the resources will be done in venues with wide visibility in the computer science education community. . We will also participate in an annual Women in Computing Day sponsored by our department for K-121 girls. By engaging young girls with realistic and socially-meaningful computing experiences we hope to positively affect their subsequent study and career choices toward computing and STEM fields.

*Technology Practices*. A key element of our dissemination activity is the use of a "componentized" approach to the design and development of the resources created during the project, allowing adopters wide latitude in choosing which and how much of the developed resources to adopt. For example, an adopter may:

* use some or all of the curriculum components developed through instructional design,
* use only one (or a small number) of the raw big data streams for use in an existing assignment ,
* use the entire library of raw big data streams to give students enhanced self-direction ,
* use the entire library of data streams and the scaffolded execution environment so that a progressive exploration of algorithmic manipulation of the data streams can be done in a supportive environment, or
* use all of the artifacts together with the learning resources in the Canvas framework.

Alternatively, an adopter may use only the scaffolded execution environment with their own data streams. Other combinations are also possible. Adoption is facilitated by the use of open-source standards and widely used tools (e.g., Sculpt, Canvas).

1. Results from Prior NSF Support

**Award (TUES-1444094)**: PI Dennis Kafura, 2014-15; Co-PIs Cliff Shaffer and Eli Tilevich. *Scaffolding Big Data for Authentic Learning of Computing*; $97,658. **Intellectual Merit:** This award supported the design, construction of technology support for, and assessment of a course in Computational Thinking. The course was offered in Fall, 2014. An electronic book was created (see think.cs.vt.edu/book) that included immediate feedback questions, features for group collaboration, and interactive use of the block-based programming language (Blockly). The initial results show that the students were highly motivated and engaged by the course design and the use of real-world data. **Broader Impact:** The project is a model of a university-level course that engages students from a wide variety of disciplines in computational thinking. It also provides additional evidence and available resources for using a "big data" approach to instruction in computer science courses. This heightened engagement and real-world connections help to attract and retain students from under-represented populations.

**NSF TUES Phase I Project (DUE-1139861** *Integrating the eTextbook: Truly Interactive Textbooks for**Computer Science Education.* PIs: C.A. Shaffer, T. Simin Hall, T. Naps, R. Baraniuk. $200,000, 07/2012-06/2014.**NSF SAVI/EAGER Award (IIS-1258571** *Dynamic Digital Text: An Innovation in STEM Education,* PIs: S. Puntambekar (UW-Madison), N. Narayanan (Auburn),and C.A. Shaffer (2013). $247,933, 01/2013 - 12/2014.**NSF CCLI Phase 1 Award (DUE-0836940**) *Building a Community and Establishing Best Practices in Algorithm Visualization through the AlgoViz Wiki*. PIs: C.A. Shaffer, S.H. Edwards. $149,206, 01/2009-12/2010. **NSF NSDL Small Project (DUE-0937863)** *The AlgoViz Portal: Lowering Barriers for Entry into an Online Educational Community*.PIs: C.A. Shaffer, S.H. Edwards, $149,999, 01/2010-12/2011. **NSF IUSE (DUE-1432008** *Collaborative Research: Assessing and Expanding the Impact of* *OpenDSA, an Open Source, Interactive eTextbook for Data Structures* *and Algorithms*. PIs: C.A. Shaffer, J.V. Ernst, T.L. Naps (U Wisconsin-Oshkosh), S.H. Rodger (Duke U), $998,402, 01/01/2015-12/31/2017.**Intellectual Merit**: The first two projects provided online infrastructure (the AlgoViz Portal: http://algoviz.org) and related community development efforts to promote use of AV in computer science courses. This work was an important precursor to OpenDSA, as it allowed us to interact with many CS instructors and AV developers, leading to understanding of the fundamental missing parts in existing DSA instruction, and initiating many of the international collaborations that lead to OpenDSA. Three journal papers~\cite{Shaffer10,Fouh:Av11,Cooper14} and three conference papers~\cite{ShafferSIGCSE07,ShafferSIGCSE10,ShafferSIGCSE11} came from this work. The last three awards supported OpenDSA, and active collaborations involving Virginia Tech, and Aalto University (Helsinki), Duke University, and U Wisconsin at both Madison and Oskhosh, among others. Publications related to this work so far include~\cite{ShafferKoli11,ShafferPVW11,Hall13,Karavirta:ITiCSE13,Fouh14CHB,Fouh14SCP,Karavirta16}. **Broader Impacts** include dissemination of AV artifacts and eTextbooks to a broad range of CS students.

**NSF DUE Project (DUE-1140318)** TUES-Type1:"Transforming Introductory Computer Science Projects via Real-Time Web Data" PI: E. Tilevich. Co-PI: C. A. Shaffer. $200,000.00 for 07/2012 to 06/2015. This project creates an educational software infrastructure to support computer programming projects that use real-time web-based data to better engage and better train introductory computer science students. The project has led to research papers presented at SIGCSE 2014 [BartSIGCSE14] and SPLASH-E 2013 & 2014 [BartSPLASHE13, BartSPLASHE14]. **Intellectual Merits** include validation of the theory that contextualization can provide more engaging introductory programming experiences that also improve student comprehension of real-time technology. **Broader Impacts** include workshops offered at SIGCSE 2014 and SIGCSE 2015 to introduce the developed technology to our peers in other institutions [WorkshopSIGCSE14,WorkshopSIGCSE15]. In addition, the curricula of CS1 and CS2 classes at Virginia Tech the University of Delaware were enhanced with the projects developed under the auspices of this project.

**NSF EHR Project (DRL-1156629) *Transforming Teaching through Implementing Inquiry (T2I2)* project. PI: J. Ernst, Co-PIs: L. Bottomley, A. Clark, V.W. DeLuca, S. Ferguson. $1,997,532 for 09/2011-8/2015. Intellectual Merit**: This full research and development project explores the use of cyber-infrastructure tools to significantly enhance the delivery and quality of professional development for grades 8-12 engineering, technology, and design educators. The goal is to study whether the use of highly interactive cyber-infrastructure tools increases the educators’: 1) understanding of how to address student learning needs 2) ability to manage, monitor, and adjust the learning environment 3) use of self-assessment to enhance teaching ability and 4) engagement in a community of practice. Results to date have shown that sixteen teachers from five states (teaching grades 6-12) have attained satisfactory competency on the learning objects [72-74]. Broader impact: The focus on using an object-oriented system design enables the cyber-infrastructure to be reusable, adaptable, and scalable.

**NSF Award ID: 0725290,** *Collaborative Research: Investigating and Refining the Studio Experience as a Method for Teaching Human Computer Interaction*, $494,818.00, August 7, 2007- July 31. 2012. **Intellectual Merit**: Computer science is, in many ways, a design discipline. The goal of this project was to leverage knowledge about design education from Architecture and Industrial Design to develop new educational models for the design of software-intensive systems, specifically in the area of Human Computer Interaction (HCI). In a 3-year collaborative research and education project involving three universities (Virginia Tech, University of Montana, and the University of Oregon), experts in HCI, architecture, industrial design, and educational theories engaged in empirical research to explore and refine the studio method as a means for preparing future computer science professionals. The first two years of the project focused on research involving data collection and analysis that lead to the development of curriculum guidelines for use in HCI courses in the third year. The effectiveness of the curriculum guidelines was evaluated through pilot testing in HCI classrooms, and revised based on the results of the formative evaluation. Publications resulting from this work include [Cennamo1-8]. **Broader Impact**: We have advanced software design research and education through the development of new models and methods that are supported by empirical evidence and that are teachable. The research outcomes and curriculum guidelines have been disseminated through presentations, publications, and a project web site. In this way, have contributed to a strong intellectual foundation for teaching software design, which has the potential to transform the teaching of Human Computer Interaction, ultimately improving the processes of constructing, evaluating, and modifying software-intensive systems across a variety of computer science specialty areas.

**REFERENCES**

[1] E. Fouh*, et al.*, "Design and architecture of an interactive eTextbook – The OpenDSA system," *Science of Computer Programming,* vol. 88, pp. 22-40, 2014.

[2] Open edX. (2015. Available: <http://code.edx.org/>

[3] R. E. Anderson*, et al.*, "Introductory programming meets the real world: using real problems and data in CS1," presented at the Proceedings of the 45th ACM technical symposium on Computer science education, Atlanta, Georgia, USA, 2014.

[4] A. E. Egger, "Engaging student in earthquakes via real-time data and decisions.," *Science,* vol. 336(6089), pp. 1654-1655, 2012.

[5] M. Waldman, "Keeping it real: utilizing NYC open data in an introduction to database systems course," *J. Comput. Sci. Coll.,* vol. 28, pp. 156-161, 2013.

[6] T. T. Yuen and K. A. Robbins, "A Qualitative Study of Students' Computational Thinking Skills in a Data-Driven Computing Class," *Trans. Comput. Educ.,* vol. 14, pp. 1-19, 2014.

[7] College Board. (2014, December 2014). Computer Science Principles Performance Assessment. Available: <http://media.collegeboard.com/digitalServices/pdf/ap/ap-computer-science-principles-performance-assessment.pdf>

[8] A. C. Bart*, et al.*, "Using real-time web data to enrich computer science projects," in *Splash-E*, Indianapolis, Indiana, USA, 2013.

[9] A. C. Bart*, et al.*, "Transforming introductory computer science projects via real-time web data," presented at the Proceedings of the 45th ACM technical symposium on Computer science education, Atlanta, Georgia, USA, 2014.

[10] L. J. Barker*, et al.*, "Exploring factors that influence computer science introductory course students to persist in the major," presented at the Proceedings of the 40th ACM technical symposium on Computer science education, Chattanooga, TN, USA, 2009.

[11] J. Carter*, et al.*, "Motivating all our students?," presented at the Proceedings of the 16th annual conference reports on Innovation and technology in computer science education - working group reports, Darmstadt, Germany, 2011.

[12] B. DiSalvo and A. Bruckman, "From interests to values," *Commun. ACM,* vol. 54, pp. 27-29, 2011.

[13] A. Fisher and J. Margolis, "Unlocking the clubhouse: the Carnegie Mellon experience," *SIGCSE Bull.,* vol. 34, pp. 79-83, 2002.

[14] M. Goldweber*, et al.*, "A framework for enhancing the social good in computing education: a values approach.," *ACM Inroads,* vol. 4, pp. 58-79, 2013.

[15] B. D. Jones, "Motivating Students to Engage in Learning: The MUSIC Model of Academic Motivation," *International Journal of Teaching and Learning in Higher Education,* vol. 21, pp. 272-285, 2009.

[16] A. Bandura, "Social cognitive theory of self-regulation," *Organizational behavior and human decision processes,* vol. 50, pp. 248-287, 1991.

[17] M. Ben-Ari, "Constructivism in computer science education," *Journal of Computers in Mathematics and Science Teaching,* vol. 20, pp. 45-73, 2001.

[18] B. D. Jones and G. Skaggs, "Validation of the MUSIC Model of Academic Motivation Inventory: A measure of students’ motivation in college courses," in *International Conference on Motivation*, Frankfurt, Germany, 2012.

[19] P. Steel, "The nature of procrastination: A meta-analytic and theoretical review of quintessential self-regulatory failure.," *Psychological Bulletin,* vol. 133(1), pp. 65-94, 2007.

[20] R. M. Klassen*, et al.*, "Academic procrastination of undergraduates: Low self-efficacy to self-regulate predicts higher levels of procrastination. ," *Contemporary Educational Psychology,* vol. 33, pp. 915-931, 2008.

[21] P. R. Pintrich*, et al.*, "Reliability and predictive validity of the Motivated Strategies for Learning Questionnaire (MSLQ). ," *Educational and Psychological Measurement,* vol. 53, pp. 801-813, 1993.

[22] Y. Lambrinidou*, et al.*, "Ethnography in Engineering Ethics Education: A Pedagogy for Transformational Listening," presented at the 121st American Society for Engineerng Education Annual Conference & Exposition, Indianapolis, IN, 2014.

[23] ABET Engineering Accreditation Commission. (2014, December 2014). Criteria for Accrediting Engineering Programs. Available: <http://www.abet.org/uploadedFiles/Accreditation/Accreditation_Step_by_Step/Accreditation_Documents/Current/2015-2016/E001%2015-16%20EAC%20Criteria%2011-7-14.pdf>

[24] Association for Computing Machinery. (2014, *Code of Ethics*. Available: <http://www.acm.org/about/code-of-ethics>

[25] Association for Computing Machinery. (2014, December 2014). *Software Engineering Code of Ethics and Professional Practice*. Available: <http://www.acm.org/about/se-code>

[26] E. A. Cech, "Culture of Disengagement in Engineering Education?," *Science, Technology, and Human Values,* vol. 39 (1), pp. 42-47, 2014.

[27] A. Korhonen*, et al.*, "Requirements and design strategies for open source interactive computer science eBooks," presented at the Proceedings of the ITiCSE working group reports conference on Innovation and technology in computer science education-working group reports, Canterbury, England, United Kingdom, 2013.

[28] T. Naps*, et al.*, "Evaluating the educational impact of visualization," presented at the Working group reports from ITiCSE on Innovation and technology in computer science education, Thessaloniki, Greece, 2003.

[29] T. L. Naps*, et al.*, "Exploring the role of visualization and engagement in computer science education," presented at the Working group reports from ITiCSE on Innovation and technology in computer science education, Aarhus, Denmark, 2002.

[30] J. H. McMillan and J. Hearn, "Student self-assessment: The key to stronger student motivation and higher achievement," *Educational Horizons,* vol. 87 (1), pp. 40-49, 2008.

[31] H. Andrade and A. Valtcheva, "Promoting learning and achievement through self-assessment," *Theory Into Practice,* vol. 48(1), pp. 12-19, 2009.

[32] L. Malmi*, et al.*, "Visual algorithm simulation exercise system with automatic assessment: TRAKLA2," *Informatics in Education,* vol. 3(2), pp. 267-288, Septmenber 2004 2004.

[33] V. Karavirta and C. A. Shaffer, "JSAV: the JavaScript algorithm visualization library," presented at the Proceedings of the 18th ACM conference on Innovation and technology in computer science education, Canterbury, England, UK, 2013.

[34] V. Karavirta. (2013, *JSAV github repository*. Available: github.com/vkaravir/JSAV

[35] B. Friedman*, et al.*, "Value Sensitive Design and information systems," in *Human-computer interaction in management information systems: Foundations*, P. Zhang and D. Galletta, Eds., ed Armonk, New York; London, England: M.E. Sharpe, 2006, pp. 348-372.

[36] U. Wilensky. January 2015). *NetLogo*. Available: https://ccl.northwestern.edu/netlogo/

[37] M. Guzdial, "A media computation course for non-majors," *SIGCSE Bull.,* vol. 35, pp. 104-108, 2003.

[38] M. Guzdial and A. E. Tew, "Imagineering inauthentic legitimate peripheral participation: an instructional design approach for motivating computing education," presented at the Proceedings of the second international workshop on Computing education research, Canterbury, United Kingdom, 2006.

[39] P. DePasquale, "Exploiting on-line data sources as the basis of programming projects," presented at the Proceedings of the 37th SIGCSE technical symposium on Computer science education, Houston, Texas, USA, 2006.

[40] D. Stevenson, E.*, et al.*, "Steganography and cartography: interesting assignments that reinforce machine representation, bit manipulation, and discrete structures concepts," presented at the Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA, 2005.

[41] D. E. Stevenson and P. J. Wagner, "Developing real-world programming assignments for CS1," presented at the Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education, Bologna, Italy, 2006.

[42] A. Erkan*, et al.*, "Sustainability themed problem solving in data structures and algorithms," presented at the 43rd ACM technical symposium on Computer Science Education, Raleigh, North Carolina, USA, 2012.

[43] A. C. Bart*, et al.*, "Motivating Students with Big Data: CORGIS and MUSIC," in *Splash-E*, Portland, Oregon, USA, 2014.

[44] H. Abelson*, et al.*, *Blown to Bits*. Upper Saddle River, NJ: Addison-Wesley, 2008.

[45] D. Rushkoff, *Program or Be Programmed*. New York, NY: OR Books, 2010.

[46] Executive Office of the President. (2014, Decmber 2014). Big Data: Seizing Opportunities, Preserving Values. Available: <http://www.whitehouse.gov/sites/default/files/docs/big_data_privacy_report_5.1.14_final_print.pdf>

[47] K. Davis, *Ethics of Big Data*: O'Reilly Media, 2012.

[48] ACM Committee on Computers and Public Policy and Peter G. Neumann (Moderator). Forum on Risks to the Public in Computers and Related Systems [Online]. Available: <http://catless.ncl.ac.uk/Risks/>

[49] K. Bowyer. December 2014). *Ethics and Computing*. Available: <http://www3.nd.edu/~kwb/nsf-ufe/>

[50] Center for Engineering Ethics and Society. (2007, January 2015). *Online Ethics Center for Engineering and Science*. Available: <http://www.onlineethics.org/>

[51] C. Douce*, et al.*, "Automatic test-based assessment of programming: A review," *J. Educ. Resour. Comput.,* vol. 5, p. 4, 2005.

[52] S. H. Edwards, "Using software testing to move students from trial-and-error to reflection-in-action," presented at the Proceedings of the 35th SIGCSE technical symposium on Computer science education, Norfolk, Virginia, USA, 2004.

[53] S. H. Edwards*, et al.*, "Experiences using test-driven development with an automated grader," *J. Comput. Sci. Coll.,* vol. 22, pp. 44-50, 2007.

[54] B. D. Jones, "The MUSIC model of academic motivation: User guide to assessing the MUSIC model components," ed. Blacksburg, VA., 2014.

[55] B. D. Jones, "Strategies to implement a motivation model and increase student engagement," presented at the annual meeting of the International Society for Exploring Teaching and Learning, Nashville, TN, 2010.

[56] B. D. Jones*, et al.*, "The effects of a collaborative problem-based learning experience on students’ motivation in engineering capstone courses," *Interdisciplinary Journal of Problem-based Learning,* vol. 7(2), 2013.

[57] B. D. Jones and G. E. Skaggs, "Measuring students’ motivation: Development of and validity evidence for the MUSIC Model of Academic Motivation Inventory," *The International Journal of Educational and Psychological Assessment,* In press.

[58] B. D. Jones and J. L. Wilkins., "Validity evidence for the use of a motivation inventory with middle school students," presented at the annual meeting of the Society for the Study of Motivation (Poster), Washington, D.C., 2013.

[59] P. R. Pintrich, "A conceptual framework for assessing motivation and self-regulated learning in college students," *Educational Psychology Review* vol. 16, 385–407 2004.

[60] A. R. Artino. (2005, September 28, 2012). Review of the motivated strategies for learning questionnaire. Available: <http://search.proquest.com/docview/62004605?accountid=14826>

[61] M. L. Cooper*, et al.*, "Open source software and the algorithm visualization community," *Science of Computer Programming,* vol. 88, pp. 82-91, 2014.

[62] E. Fouh*, et al.*, "The role of visualization in computer science education," *Computers in the Schools,* vol. 29, pp. 95-117, 2012.

[63] C. A. Shaffer*, et al.*, "Building an online educational community for algorithm visualization," presented at the Proceedings of the 41st ACM technical symposium on Computer science education, Milwaukee, Wisconsin, USA, 2010.

[64] C. A. Shaffer*, et al.*, "Algorithm visualization: a report on the state of the field"," presented at the SIGCSE '07: Proceedings of the 38th SIGCSE technical symposium on Computer science education,, New York, NY, USA, 2007.

[65] C. A. Shaffer*, et al.*, "Getting algorithm visualizations into the classroom," presented at the Proceedings of the 42nd ACM Technical Symposium on Computer Science Education - SIGCSE '11, Dallas, TX, USA, 2011.

[66] C. A. Shaffer*, et al.*, " Truly Interactive Textbooks for Computer Science Education," presented at the Proceedings of the Sixth Program Visualization Workshop, Darmstadt, Germany, 2011.

[67] C. A. Shaffer*, et al.*, "OpenDSA," presented at the Proceedings of the 11th Koli Calling International Conference on Computing Education Research - Koli Calling '11, Koli, Finland, 2011.

[68] E. Fouh*, et al.*, "Exploring students learning behavior with an interactive etextbook in computer science courses," *Computers in Human Behavior,* vol. 41, pp. 478-485, 2014.

[69] S. Hall*, et al.*, "Education Innovation for Data Structures and Algorithms Courses," presented at the Proceedings of ASEE Annual Conference, Atlanta, GA, 2013.

[70] E. Tilevich*, et al.*, "Creating stimulating, relevant, and manageable introductory computer science projects that utilize real-time web-based data (abstract only)," presented at the Proceedings of the 45th ACM technical symposium on Computer science education, Atlanta, Georgia, USA, 2014.

[71] E. Tilevich*, et al.*, "Creating stimulating, relevant, and manageable introductory computer science projects that utilize real-time large, web-based datasets (abstract only)," presented at the Proceedings of the 46th ACM technical symposium on Computer science education, Kansas City, Missouri, USA, 2015.

[72] J. V. Ernst*, et al.*, "Professional development system design for grades 6-12 technology, engineering, and design educators," presented at the American Society for Engineering Education Annual Conference and Exposition, 2013.

[73] J. V. Ernst, "Authentic assessment in performance-based stem education activities," presented at the Scaling STEM: Transforming Education Matters Annual Conference, 2012.

[74] L. Segedin*, et al.*, "Transforming teaching through implementing inquiry: A national board-aligned professional development system," presented at the Association for Career and Technical Education Research, 2013.