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

1. Introduction

We propose to enhance the motivation and learning gains for diverse students in diverse STEM majors in their initial college-level learning experience with computing. While improving learning gains is clearly desired, concern for motivation is 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 or fearful about their ability to succeed. While they use computing-enabled devices daily, their image of computing is something of a deeply mysterious and technical nature. Developing and sustaining a high level of motivation and engagement is critical to learning for 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. Figure 1 summarizes the pedagogical principles and technology that we bring to bear on these issues.

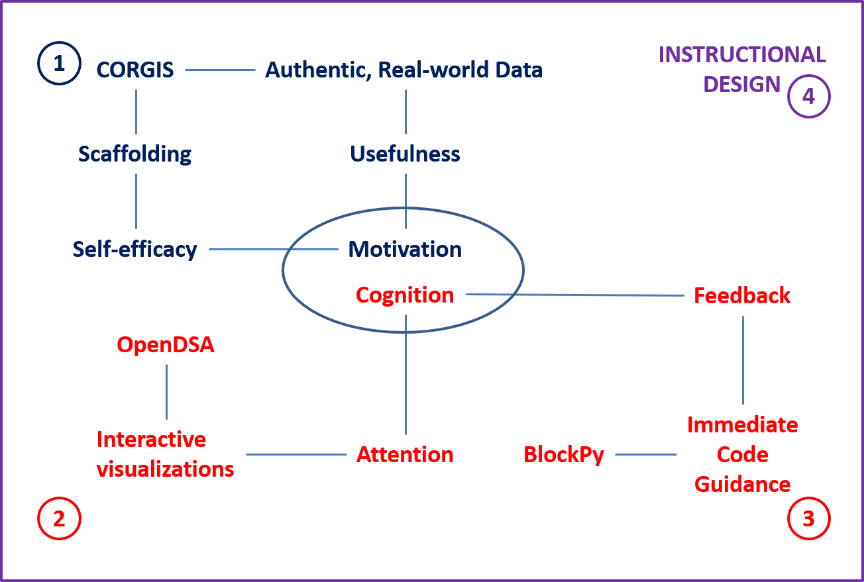
Our approach is to improve motivation by raising the students’ appreciation of the *usefulness* of their learning, and to heighten their sense of *self-efficacy*. Usefulness means that students perceive that what they are learning has relevance and importance to something they value. What is valued might 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 have too steep a learning curve, causing the students to lose confidence in their ability to succeed.

Figure 1: Research Approach

In addition to the positive effects of better motivation learning gains will be improved by better means for sustaining *attention*, and the provision of better *feedback*. The eBook infrastructure that we developed for a computational thinking course allowed us to analyze the students’ use of the book text. It is clear that students, especially those who are struggling, did not focus attention sufficiently on the text. Our end-of-term dialog with students also provided anecdotal evidence that the limited feedback on programming problems currently provided was often a stimulus for continued engagement with the problem and that additional feedback would be highly valued.

Our approach to creating a sense of usefulness is to provide 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 [[1-3](#_ENREF_1)]. Potential problems with this approach are the difficulties of scale and complexity of such data. To address these difficulties we propose to extend an existing framework, Corgis [REF], which we have created for scaffolding big data and real-time data access. The extensions allow easier generation and curation of data resources by teachers, and easier use by learners. While we are not the first to use realistic data to increase student motivation [[1](#_ENREF_1), [4](#_ENREF_4)], Corgis is unique in its ability to provide such real data experiences at the introductory level.

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 allow a student to gain insight into the dynamics of algorithms and the manipulation of data. We have developed a significant collection of such visualizations for our OpenDSA project [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. Also, the visualizations should help students maintain attention while coping with the structural complexity of the data.

Immediate feedback is important to improved learning. Many eBook 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 will develop a mechanism for analyzing and providing immediate guidance on code created by students for small programming exercises (e.g., "Write an algorithm that..."). By guidance we mean feedback that identifies both problems that the student should focus on correcting as well as feedback that identifies desirable aspects of the solution that the student should maintain. An important challenge in this work is to develop an authoring tool for instructors that allows salient features of correct and incorrect code to be described and related to appropriate feedback. The feedback should be accessible both to the student and to instructors for additional comments. We propose to incorporate the analysis and feedback into the BlockPy environment [REF].

Finally, we use the Dick and Cheney instructional design method [REF] to create a principled curriculum design for the core computing concepts taught in our Introduction to Computational Thinking course (described next). Using the Dick and Cheney method both improves the curriculum we are developing, and also provides 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. 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 ID are self-documenting at every phase, so that curriculum adopters are given materials with the context and justifications of the materials. Within the project, instructional design helps to identify targets and requirements for interactive visualizations and immediate feedback on programming exercises.

Our primary curriculum target is an Introduction to Computational Thinking (ICT) course developed with NSF EAGER support. The first three offerings of the course have served students from over thirty different majors and have had strong gender diversity. This course is a general education course with no prerequisites, and it makes 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 so as to promote learning across contexts. Students are gradually introduced to programming through the BlockPy environment [REF]. Each student defines and completes a major project in Python using the “big data” resources in the Corgis library [REF]. 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 students at four other U.S. institutions and two international universities. At Virginia Tech we will involve: (1) students in the Introduction to Computational Thinking 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 they 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 institutions (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 are currently doing IRB approved research. Our team also includes experts in educational assessment and in instructional design, both from VT’s School of Education.

Our resources will be available through the Canvas learning platform. This 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" [REF]. 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 [[5-8](#_ENREF_5)], 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.” [[9](#_ENREF_9)].

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 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 that integrates a 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 assignments and projects.

Our technology work contributes to the Canvas community. Integrating the algorithm and data visualization capabilities developed in OpenDSA into Canvas adds a new tool for constructing dynamic and engaging content. The provision of program analysis and related authoring tools adds a powerful new capability 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 Improving Motivation through Big Data

Our work to create *authentic, real-world learning experiences* is grounded in the well-researched educational theory of Socio-Constructivism. 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 [[10](#_ENREF_10)], 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 by the student, Socio-Constructivism emphasizes the value of culture within the learning process. One way that this culture is made concrete within the learning environment is Anchored Instruction, in which 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.

The MUSIC Model of Academic Motivation [[11](#_ENREF_11)] underpins our work on student motivation. This model 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 [[12](#_ENREF_12)]. The MUSIC model identifies five key constructs [[11](#_ENREF_11)]:

* 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. Interactive Visualization and Automatic Feedback

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 they face is lack of practice with problems and exercises. Since the best types of problems are often 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 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 [[13-15](#_ENREF_13)].

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 [[16](#_ENREF_16), [17](#_ENREF_17)]. We do make use of simple multiple choice and give-a-number style questions. But we also include many interactive exercises. OpenDSA gives practice on algorithm understanding through use of “algorithm simulation” or “proficiency” exercises, as pioneered by the TRAKLA2 project [[18](#_ENREF_18)]. 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.

2.3. Immediate Code Guidance

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. For many kinds of student problems and learning resources, automated feedback on practice exercises can reach the effectiveness of one-on-one human tutoring [REF].

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 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.

2.4 Instructional Design

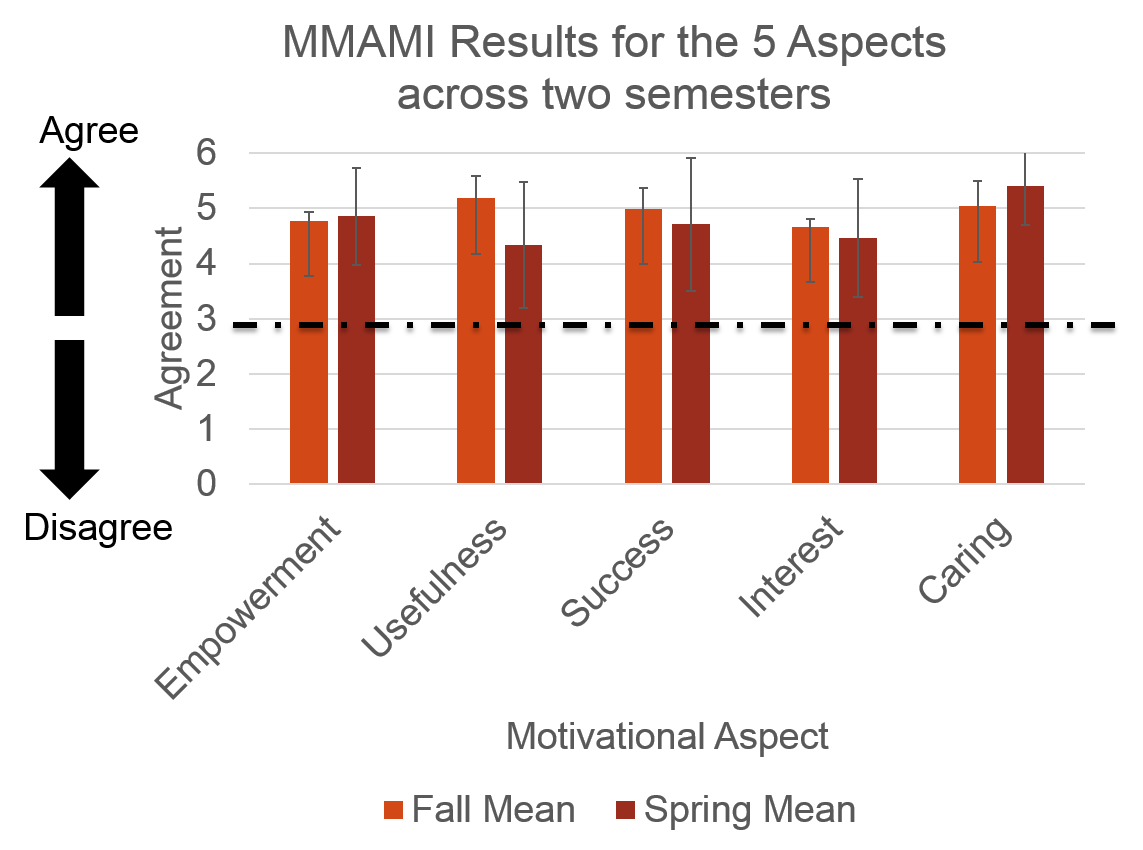
We use well established Instructional Design (ID) approaches during course development. Just as quality software is produced through rigorous software engineering practices, we believe that quality learning environments are produced through rigorous “curriculum engineering” provided by instructional design. Prevailing well intentioned and sometimes successful, but ad-hoc, curriculum design can be dramatically improved by systematically applying known instructional design methods. Instructional Design (ID) is a subfield of education concerned with the systematic design of learning experiences that give measurable results by following a well-defined process. Although ID has existed for decades and is popular in other domains, a survey of the literature has revealed no prior, formal attempts at applying this technique to CS Education. Existing research ironically suggests how software engineering techniques can be applied to ID [3, 4, 7].

We use the Dick and Carey [REF] model of ID 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 [REF] 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 ID methods to create a new unit of the Computational Thinking course, and also to create a five-day introductory workshop on Computer Science for high school students. In the former case, the ID model was applied rigorously to correct a perceived gap in students' learning. In the latter case, the model was applied loosely to create a new instructional unit. In both cases, 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.

2.5. Computational Thinking (CT) Class

A Computational Thinking (CT) class was developed with support from an IUSE Eager award. Preliminary results from the first two offerings of the CT class (Fall 2014 and Spring 2015) include an analysis of student motivation and the use of group-based “cohorts”. There were 65 students in the two classes, 43% female and 57% male. There was a wide variety of 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) [[12](#_ENREF_12)]. 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.

Figure 2 Motivational Data

Forming cohorts with students from different disciplines allowed students to better understand the application and implications of CT across disciplines: *“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.” (Student 5)*

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 how thorough and how much it would tie into my major until I took the class…”(Student 3)*

2.6 Other Related Work

We share with the Media Computation approach [[19](#_ENREF_19)] 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” [[20](#_ENREF_20)]. We share a common goal with courses that use real-world data for motivational purposes. Examples include using on-line data [[21](#_ENREF_21)] and assignments that produce useful tools [[22](#_ENREF_22), [23](#_ENREF_23)]. We also use resources that relate to social impacts. Other work uses a values-oriented approach [[9](#_ENREF_9)] to exploring the social implications in various computational modeling assignments, and [[24](#_ENREF_24)] uses sustainability issues to frame problems used in a data structures and algorithms class. We differ from these in the resource (big data in our case) to which the social concern is connected. We share the most with the work of [[1](#_ENREF_1)] 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. We will integrate and extend these concerns for engagement, realism, social impact, and big data. Integration is achieved by extending the "raw" big data streams with elements connected to a model of social impacts. Extensions include developing interactive visualizations, static analysis for immediacy of feedback, access to big data through two different block-based programming languages, and other supporting technology. We also go outside of mainstream Computer Science education to the general university student population via the Introduction to 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 programming exercises within a “book” format to better integrate learning materials (see Section 4).

1. Proposed Work

3.1 CORGIS for Realistic Data

It is difficult to find, prepare, and deliver realistic data to introductory learners in a pedagogically suitable form. To address these difficulties the PIs developed CORGIS, a collection of highly scaffolded library of over 35 realistic data sources. However, there is a burden on the developer to create these datasets, requiring technical, pedagogical, and domain proficiency. Once a dataset is developed, it 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 datasets must be usable by students no matter what kind of hardware they have and whatever permissions they have on the machine. We propose to overcome these limitations by four major extensions:

* CORGIS Builder to enable developers to prepare data sources efficiently,
* CORGIS Gallery to allow students and instructors to find relevant datasets easily,
* CORGIS Architecture to handle new use cases and resolve technical issues, and
* CORGIS Explorer to allow learners to work with datasets in a highly scaffolded interactive environment.

Each extension is described in more detail in the remainder of this section.

**CORGIS Builder**. There is too high a barrier for instructors to transform a data source into a classroom ready resource arising from three limitations. First, while the CORGIS web-based builder scaffolds the process of 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 standard format, requiring decisions about what fields and instances to keep, how the data should be structured hierarchically, what types the fields should be, and how data should be pre-aggregated for students. Second, there is 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. Third, there is no uniform shape to the data, making it difficult for stoe for students with different datasets.

Our proposed work will add features to the RTW Online Builder 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 “shape” by restructuring the data according to certain common templates and high-level instructions given by the instructor. 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. This use of mocking is a powerful way to provide more controlled learning experiences. The output of the tool will be pluggable datasets and interfaces suitable for the CORGIS architecture, rather than specific language bindings.

**CORGIS Gallery**. The Gallery do not adequately support students’ and instructors’ discovery of relevant data sources. Currently, CORGIS libraries are presented as a flat list of the dataset names in a wiki structure (see examples in <http://think.cs.vt.edu/wiki/index.php/Category:Library>). Each dataset has an ad-hoc page of information that may or may not include code examples, dataset description, and a link to the source code. As dataset selection grows, this informal representation will become inadequate for finding a suitable library and learning more about its nature.

Our proposed work will extend the Gallery 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 dataset. 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 for a dataset in sync with the dataset’s source. In particular, make it easy for developers to update the dataset or the metadata for the dataset.

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.

**CORGIS Architecture**. The current architecture has three problems. First, accommodating changes is difficult. Creating a dataset includes writing a small support library with its own copy of the code needed to access its data. While generated semi-automatically from a specification file, an update to the architecture’s web API requires the developer to make modifications to each of these support libraries. Second, the documentation and metadata is limited. CORGIS 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 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. Third, it is difficult to accommodate hardware with diverse computational power. Several CORGIS libraries suffer from bad caching strategies and poorly sized datasets, resulting in poor performance that can frustrate learners.

Our proposed work is for all the CORGIS libraries to 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 propagate 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 student’s laptop had little RAM and a poor processor, it might decide to sample the dataset, or to process more data on disk.

**CORGIS Explorer**. The scaffolding of a learner’s access to CORGIS datasets is incomplete. Currently, students in the ICT class access the datasets programmatically. While programmatic access is scaffolded through the use of a block-based language in the BlockPy environment, the student must master critical computational concepts (iteration, lists) before they can begin to compose a block-based program that manipulates and visualize the data. More robust scaffolding would permit learners to manipulate and visualize the data using a non-programming approach.

Our proposed work will develop an interactive Explorer for students to interactively select, manipulate, and visualize a CORGIS dataset. Our vision for the CORGIS Explorer is that students will be able to work with a CORGIS dataset in the same way that NetLogo [REF] allows a student to easily work with an underlying agent-based computational model. With the Explorer, a student will be able to select a dataset from the library, access information about the dataset (via the Gallery extensions), select certain data (via the Builder extensions), and visualize the selected data in standard forms (scatter plot, bar charts, histograms). The Explorer will increase student engagement with the datasets by allowing early investigation leading to results which they can duplicate or extend later in programmatic forms.

**3.2 Interactive Visualization and Exercises**

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 are 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.

Through our experience in teaching the Computational Thinking course we have identified an initial set of topics for which we will develop interactive visualizations and accompanying exercises. These topics are both foundational concepts in computing and ones with which students struggle. The initial topics are:

* Abstraction: the idea of representing real-world entities by properties that are relevant to the needs or interests of a given “stakeholder”. This 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).
* Variables: an objective, quantifiable, and measurable characteristic that can be defined in information terms. Again, this is difficult for the students to grasp. 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. This is one of the most fundamental algorithmic concepts. 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. This presents a challenging learning task, especially when multiple layers of abstraction (nested or embedded data structures) are involved.

We expect that other topics will be added to this list through additional experience and through the process of applying Instructional Design to the course.

3.3 Immediate Code Guidance

In our piloted prototype systems with code guidance, the automatic analysis of the student’s code is limited to regular expressions, 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. 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.

Our proposed work will create a flexible system that can help instructors define, via an authoring tool, and provide guidance, via automated code analysis, to a student on the code developed by the student in response to a posed exercises. The authoring tool allows the instructor to define and reuse heuristics of the form *(code pattern, feedback)*, where the *code pattern* is defined over parsed abstract syntax trees and the *feedback* is the encouraging or corrective guidance given to the student. Prioritization of the heuristics 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.

A critical asset for instructors is the use of data mining to identify common patterns of student errors. For the past year, we have been recording the entire history of students' programs and the interaction log with our systems. 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.

Our approach avoids the fundamental limitations of program analysis. General code analysis struggles with large code repositories and dynamic language features. However, for beginning exercises, the students' code is a very manageable size and should always resemble some specific form, which is critically useful information to guide the analyzer. Second, instructors can restrict the learner’s code. For instance, we might forbid dynamic retyping - a property can only take on a single type.

We will integrate into BlockPy the automated code analysis and guidance functions described above. An important advantage of this integration is that the instructor authoring and the guidance can be provided for both block-based code (specifically, Blockly) and text-based code (specifically, Python). This dual use is possible because BlockPy translates Blockly into Python.

3.4. Curriculum Development Using Instructional Design

We propose to use the Dick and Carey [REF] model to provide a principled ID process for the computational thinking curriculum. We choose the Dick and Carey model in part because of our previous experience using it. Our research team includes an expert in Instructional Design who has deep understanding of this model (Cennamo), and Kafura has experience with using it to redesign a unit in the ICT course.

Use of the Dick and Carey model affects our project in a number of ways. 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 ID helps us to insure the highest quality of curriculum development. Second, ID helps us with continuous assessment and improvement of the course, because the Dick and Carey model has a clearer connection between assessment and learning objectives. Third, the completed 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 process 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 best practice for curriculum design.

Our initial use of ID will focus on a set of curriculum units for foundational concepts that students in the CT course find the most challenging. The initial threshold concepts are abstraction, variables, iteration, state, and data structures. These concepts were described above (see Section 3.3). We believe that it is important to conduct the instructional design in close connection with the development of the interactive visualizations described above so that the visualizations are deeply connected to the learning objectives and the assessment developed within the instructional design process. We 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

Leveraging the project instructional framework, featuring data rich course experiences, interactive visualizations, and real-time corrective feedback for programming applications, the research team will explore impacts on student learning, motivation, and engagement. The research plan is guided by three questions shown in Table 1.

Table 1 Assessment Questions and Methods

|  |  |  |  |
| --- | --- | --- | --- |
| **Question** | **Pre/Post MUSIC** | **Pre/Post CT/CS** | **Semester**  **Interviews** |
| RQ1 To what extent do big data applications support increased classroom motivation and engagement? | X |  | X |
| RQ2 To what degree do interactive visualizations build student task engagement within the study of data and algorithmic concepts? |  | X |  |
| RQ3 Does instantaneous performance-based task feedback promote increased student learning outcomes of data and algorithmic concepts? |  | 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**: Does instantaneous performance-based task feedback promote increased student learning outcomes of data and algorithmic concepts?

We also will conduct evaluation of the technologies that we are building as part of this project. The evaluation questions and the methods used to perform the assessment are shown in Table 2.

Table 2: Evaluation Questions and Methods

|  |  |  |
| --- | --- | --- |
| **Evaluation Question** | **Usability Study** | **External Consultants** |
| Are the extensions to Corgis effective in building and maintaining collections of data sets? | X | X |
| Do to extensions to Corgis lower the barrier to adoption for instructors? |  | X |
| Do the extensions to Corgis support student discovery of relevant data sets? | X |  |
| Are the interfaces and tools for creating automatic feedback effective? |  | X |

1. Project Organization and Team

The proposed work is organized as shown in the Table 3 that identifies the major goals to be accomplished in each year of the project and the project team member involved in each activity

Table 3: Project Organization

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Year** | **Goals** | **Kafura** | **Shaffer** | **Tilevich** | **Ernst** | **Cennamo** |
| **1** | Apply instructional design for units on abstraction and iteration  Create interactive visualizations for abstraction and iteration  Develop prototype of immediate feedback code analyzer  Extend Corgis Architecture and Gallery  Conduct initial evaluation of   * Prototype for code analyzer * Corgis architecture and gallery   Assessment: RQ1 (motivation), RQ2 (vis), RQ3 (feedback)  External consultants: feedback on   * Prototype code analyzer * Visualizations * Corgis architecture and gallery | X  X  X  X  X | X  X  X  X  X  X | X  X  X | X  X | X |
| **2** | Apply Instructional Design for units on state, variables, data structures  Create interactive visualizations for state, variables, data structures  Evaluate and improve instructional design units from year 1  Develop authoring tool for immediate feedback system  Extend Corgis Builder and develop Corgis Explorer  Conduct intermediate evaluation of code analyzer  Conduct initial evaluation of   * Authoring tool for immediate feedback * Corgis Builder and Explorer   Assessment: RQ1 (motivation), RQ2 (vis), RQ3 (feedback)  External consultants: feedback on :   * Visualizations * Immediate feedback authoring tool * Corgis Builder and Explorer | X  X  X  X  X  X  X  X  X | X  X  X  X  X  X  X | X  X  X | X  X |  |
| **3** | Evaluate and improve instructional design units from years 1 and 2  Assessment: RQ1 (motivation), RQ2 (vis), RQ3 (feedback)  Conduct final evaluations of all developed technologies  External consultant feedback on all developments | X  X  X  X | X  X  X |  |  |  |

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 already collaborate on the OpenDSA 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. One GRA will be assigned to work with the tasks related to visualization and instructional design, and one GRA will be assigned to work with the tasks related to the immediate feedback and Corgis development.

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. 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-12 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 starting 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 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 project examined the use of cyberinfrastructure tools and their impacts on delivery and quality of professional development for engineering, technology, and design educators working with student populations in grades 8-12. Eighteen teacher learning objects were developed, tested, and published (STEM Teacher Learning, 2016). Data indicate that the materials can support teacher proficiency in managing and monitoring classroom environments, adjusting instruction per student needs, self-assessing, and contributing to a community of learning.  To date, this project has also produced three research papers (Ernst, Clark, DeLuca, & Bottomley, 2013; Ernst, Segedin, Clark, & DeLuca, 2014; Segedin, Ernst, & Clark, 2013), with two additional research manuscripts currently under review. **Broader impacts:** The project supported four doctoral students, one masters student, and three research associates. The teacher professional development materials have been implemented in over 40 schools and are now commercially available to STEM educators nationwide.

**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**: 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 Virginia Tech, the 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 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. We 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] R. E. Anderson, M. D. Ernst, R. Ordonez, P. Pham, and S. A. Wolfman, "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.

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

[3] 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.

[4] 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.

[5] L. J. Barker, C. McDowell, and K. Kalahar, "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.

[6] J. Carter, D. Bouvier, R. Cardell-Oliver, M. Hamilton, S. Kurkovsky, S. Markham*, 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.

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

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

[9] M. Goldweber, J. Barr, T. Clear, R. Davoli, S. Mann, E. Patitsas*, et al.*, "A framework for enhancing the social good in computing education: a values approach.," *ACM Inroads,* vol. 4, pp. 58-79, 2013.

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

[11] 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.

[12] 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.

[13] A. Korhonen, T. Naps, C. Boisvert, P. Crescenzi, V. Karavirta, L. Mannila*, 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.

[14] T. Naps, S. Cooper, B. Koldehofe, C. Leska, G. Rossling, W. Dann*, 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.

[15] T. L. Naps, G. Rossling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen*, 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.

[16] 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.

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

[18] L. Malmi, V. Karavirta, A. Korhonen, J. Nikander, O. Seppala, and P. Silvasti, "Visual algorithm simulation exercise system with automatic assessment: TRAKLA2," *Informatics in Education,* vol. 3(2), pp. 267-288, Septmenber 2004 2004.

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

[20] 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.

[21] 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.

[22] D. Stevenson, E., M. R. Wick, and S. J. Ratering, "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.

[23] 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.

[24] A. Erkan, T. Pfaff, J. Hamilton, and M. Rogers, "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.