IUSE: Educational Theories and Technological Tools to Bridge the Big Data Gap

March 24, 2014

Techniques for handling Big Data have become crucial to science, business, and policy making at all levels of government. Frequently in the news, topics related to Big Data provide potentially interesting ideas for projects in Computational Thinking and mainstream Computer Science programming courses. However, technical difficulties have kept Big Data topics out of lower division CS and non-majors courses. This omission deprives most students of the chance to gain an appreciation for the possibilities and risks associated with Big Data, and removes a novel and interesting class of problems from the set of possible assignments. Non-major students, brought to introductory computing classes via the Computational Thinking movement, increasingly have to deal with Big Data problems in their home domains. However, having not encountered these problems in the classroom, these students commonly lack the computational understanding of Big Data analysis tools.

The goal of this project is to provide a framework for creating adaptive Big Data assignments scaffolded with novel technology to apportion and manage the complexity of this challenging subject area. Instructors will be able to easily deploy this framework and its instances to enhance their curriculum with Big Data, bringing in relevant and interesting content from financial data, legal records, social networking information, and many other areas. Our toolchain will make it possible to quickly and seamlessly create new assignments and projects from raw data sets with minimal effort from the instructor.

This proposal builds off the success of the RealTimeWeb project, our framework for rapidly building real-time web-data centered assignments in introductory courses. The RealTimeWeb toolchain employs novel technology to scaffold students, as they work with challenging but motivating dynamically updating data. The framework has been successfully adopted in courses at Virginia Tech and the University of Delaware. Providing technical scaffolding to empower students to work with Big Data will leverage and enhance techniques that we have successfully applied to real time data.

To empirically validate our new educational toolchain, we will introduce new programming projects across an undergraduate program, specifically in the CS 1, CS 2, and data structures courses. We will evaluate Big Data assignments for their impact not only on students' understanding and appreciation for the subject area, but also for their impact on retention rates and students' intrinsic motivation. All tools and infrastructure generated during this project will be publicly available for wide-spread adoption, experimentation, and advancement by peer institutions.

Intellectual Merit: By introducing authentic, massively-sized datasets of relevance into the early undergraduate curriculum, we can create a more engaging experience for students where they develop a foundational knowledge of Big Data concepts. This project will provide an excellent opportunity to improve the theoretical understanding of the problems and best practices of teaching Big Data, even as it offers a practical method for individual instructors. This project will also provide insights about the challenge of designing engaging and relevant contexts for projects that positively impact CS learning outcomes, while simultaneously enabling students to work with one of the most important technologies of the modern era.

Broader Impact: This project will improve recruitment, retention, and engagement of students within the Computer Science disciplines. Prior research indicates that women are more likely to study and excel in Computer Science when content is contextualized and proven useful for solving real-world problems. Non-major students can be given realistic assignments that can more directly relate to their intended line of work, further increasing their motivation to succeed during their time in a CS course. Appropriately redesigning programming projects to involve interesting, contextualized Big Datasets is likely to improve the relevance and attractiveness of Computer Science to a wider community.

1 Problem Statement

Big Data is much in the news these days, from reports of massive data dredging by the NSA to tens of millions of credit cards stolen by hackers from commercial databases. Big Data has become crucial to scientific advances from understanding the genome to predicting climate change. It would do well for anyone these days to appreciate the capabilities and the limits of Big Data processing. Certainly a study of Big Data concepts aligns well with current efforts to acquaint a broad range of students to Computational Thinking, and it certainly would be of broad interest to most of these students.

Leveraging the promise of Big Data offers unprecedented opportunities for improved economic growth and enhanced innovation. Unfortunately, computer scientists in the workforce are woefully unequipped for this shifting paradigm. Indeed, a report by MGI and McKinsey's Business Technology Offices declares that "by 2018, the United States alone could face a shortage of 140,000 to 190,000 people with deep analytical skills as well as 1.5 million managers and analysts with the know-how to use the analysis of Big Data to make effective decisions." [14] In order to close this gap, we need to do more than just improve the education of existing students; we must draw from under-represented and non-traditional sectors. In particular, we need to start reaching out to non-majors through the banner of Computational Thinking.

There are two main obstacles to effectively educating students on Big Data. First, its representation, manipulation, and expression is challenging, with modern curriculums and programming tools being inadequate. In fact, the definition of Big Data is quantities of information that cannot be handled with traditional methods [14]. Second, postponing these topics until late in the curriculum, as is commonly done [15], fails to prepare students to solve problems in this domain; the Computer Science Curricula 2013 recommends 10 core hours minimum, requiring more distribution of the material over the life of an undergraduate program. Moreover, learning how to manage this complexity is central to interdisciplinary work in everything from agriculture to medicine to law, and everything in between [1]. We cannot expect non-majors to progress too far through our own subject, so therefore work must be done to cover this material in introductory courses, as early and often as possible.

Despite the difficulties, there are serious benefits to introducing Big Data problems earlier into the curriculum. By opening an entire new class of problems, assignments can be more varied. By exposing students to Big Data topics earlier, their foundational understanding will be stronger and they will develop a stronger interest in this important subject. And finally, we can introduce non-majors to useful skills that they can immediately apply to their own subjects, fostering interdisciplinary work. But how do we make this complicated and difficult topic accessible?

2 Solution Overview

Computer Science is highly susceptible to decontextualized learning experiences, particularly in introductory courses. This heavy level of abstraction is at odds with how students learn and develop skills and conceptual knowledge [4]. There has been serious interest in providing relevant, contextualized experiences in Computer Science [7] [24] [25] [2], but it remains a challenge. Other, more traditional engineering disciplines do not suffer from this problem; to demonstrate, mediate on the expected experiences of entry students in Civil Engineering and Computer Science. Although both subjects' introductory courses will likely be project-based, the projects themselves will differ wildly in nature. While Civil Engineering courses typically deal with real-world artifacts (e.g., building a bridge or modeling

a water table), Computer Science assignments are usually oriented around abstract data manipulation (e.g., sorting raw numerical data).

It is important to recognize the difference between studying abstraction as a mental tool and abstraction in the practice of designing instruction – the former is important Content Knowledge of Computer Science that must not be removed from the curriculum, while the latter is a misuse of Pedagogical Content Knowledge that can greatly hamper student learning. Education researchers distinguish these two domains as "what is to be taught" and "how it should be taught" [12]. By relying only on abstractions and decontextualized experiences, students can be greatly confused, and – especially women – even discouraged [5] when they are unable to see the real-world application and relevance of their work. When students compare their projects and assignments to that of their peers in other fields, they may lose motivation to excel or even remain in Computer Science. This proposal aims to address this major problem.

To demonstrate the proposed paradigm shift, consider a novice student learning about list handling. Near the middle of their introductory CS-2 course, students learn that they can sort a list of data using a variety of sorting algorithms. Consider a potential, traditional problem that would students would be assigned for this topic.

Original Problem *Generate a list of 100 random numbers and sort them, first using BubbleSort and then using QuickSort. Now generate a second list of 10000 random numbers and sort them, again using Bubble Sort and then QuickSort. How do the runtimes compare?*

Today's undergraduates are consumers of Big Data, and are already familiar with services such as Facebook that are built on tremendous quantities of data; the artificiality of the data will not ring true with students, leaving them disconnected intellectually from the assignment. We propose leveraging these Big Data sets that are prominent in students lives to authenticate the assignment.

Restructured Problem *Sort a small sample of your Facebook post history using the provided FacebookData API, first using BubbleSort and then using QuickSort. Now sort your entire Facebook post history, again using Bubble Sort and then QuickSort. How do the runtimes of the different algorithms compare?*

As part of their open policy, Facebook makes all of a user's data available on demand through a massive, structured archive. Although this archive is designed to be casually viewed and comprehended by end-users, it is non-trivial for a novice Computer Scientist to programatically access the data within – the student would need comprehension beyond simple file handling, but also understand complicated XML parsing and, most relevant of all, know how to tackle data of such size.

We do not propose trading contextualization for scalability, however. Our software toolchain will manage the complexity and challenges that are irrelevant to the academic exercise at hand. The framework exposes a simple interface for students to access successively larger and larger samples of their Facebook archive, until they eventually reach the original dataset. By applying their operations over these monotonic datasets, they can gain a grounded understanding of how one algorithm can outperform another thanks to the difference in their algorithmic runtime.

There are many other possible assignments that can be demonstrated when students are working with such a novel data set. For instance, they can perform analysis on the frequency of posts – correlating the post's time with the number of "likes" or comments that it attains. Real-world problems with a strong social element are of known interest to students who view technology as a tool. Most relevant data sources in a student's life are Big – including twitter posts, sports data, and much else. Through our framework, instructors can quickly and easily generate assignments relevant to their student's interests and in line

with the instructor's academic goals. In addition to providing a collection of pre-generated tools and assignments, our open-source framework will be designed to be rapidly extended and instantiated.

As students encounter more and more Big Data problems, they will gain an appreciation for the added difficulties and importance of this particular subject. For students within the major, this foundation will prove extremely useful when they reach upper level courses and encounter problems of this scale. Ideally, students will be motivated to take electives in this critically important, under-represented subarea. It will also encourage them to explore related areas, such as parallel processing – most Big Data problems can only be tackled using techniques from multiprocessing and distributed computing, even though most undergraduate curriculum still do not adequately teach these subjects. Even if students do not proceed further in Big Data education, they will have a useful experience in working with Big Data that will serve them whether they stay in Computer Science or not.

From a broader point of view, introducing Big Data into beginner courses provides a stellar opportunity to identify the "Threshold Concepts" of learning how to use Big Data. "Threshold Concepts" are the fundamental ideas of a topic area that, when learned, represent a critical transformation of the student's understanding [17]. Although much research has been done in the past decade to identify the general "Threshold Concepts" of Computer Science, very little is known about what stumbling blocks a student will encounter when learning about Big Data. As students use our proposed framework, we will gain valuable data on what topics students struggle with, and learn about the best practices for bridging their understanding. Threshold Concepts offers a framework that we can build on to improve the theoretical understanding of how to teach Big Data concepts.

3 Authentic, Situated Learning

Situated Learning Theory, originally proposed by Lave and Wenger, argues that learning normally occurs as a function of the activity, context, and culture in which it is situated [13]. Therefore, tasks in the learning environment should parallel real-world tasks, in order to maximize the *authenticity*. Contextualization is key in these settings, as opposed to decontextualized (or "inert") settings. The key difference is that learning is driven by the problem being solved, rather than the tools available – therefore, the problem being solved should lead directly to the tool being taught. Our project builds heavily on this educational theory, and dictates that students should have authentic experiences.

Authenticity has several different, interrelated meanings within this setting: first, that problems are of realistic and massive sizes. Teaching sorting algorithms using a list of 10 elements is not convincing of the utility of this class of tools. Second, that the datasets build off students individual interests and meaningfully relate to the real-world. When a student is assigned to work with an abstract list of numbers, or with a similarly dry source, they will be cognitively disengaged from the task at hand. Third, that the techniques they are learning to tackle their data are techniques that they will require down the road. An English major taking a Computational Thinking class would not find it useful to learn low-level parallelization primitives such as spin-locks; a convincing case would need to be made to the student that this was a useful concept to learn.

High levels of authenticity are made possible through instructional scaffolding – artificial tools designed to support novices as they develop expertise. Scaffolding is designed to remove certain complexities until a student is prepared to handle them, or in order to reduce the cognitive load of a learner. There are a large number of pedagogical objectives in an introductory course, and it is important to identify what material is important for the student to learn, and what material is only necessary to support that

learning. For instance, introducing the file I/O when attempting to teach list manipulation could be seen as counter-productive; although a student eventually needs to learn how to work with files, the additional material will cognitively overload the learner at this point in time. The need to teach file-handling can be delayed through technical scaffolding that introduces the data through a known construct (e.g., a function that handles the file behind the scene). The scaffolding we propose is not necessary for high-level courses, but it is critical for the introductory level. As the student moves through their program, scaffolding is *faded*; after a student has completed the introductory series of courses, they should be comfortable with manipulating lists stored in files. Until then, our tools fill a critical gap.

4 Prior Work: The RealTimeWeb Educational Toolchain

The foundation of this project rests on prior work developing the RealTimeWeb project, a software architecture framework that provides introductory programming students with an easy way to access and manipulate distributed real-time data [3]. This real-time data offers similar advantages to working with Big Data – contextualized assignments and the experience of working with a novel technology. Our toolchain offers technical scaffolding for the students to gradually ease into, or even totally circumvent, some of the difficulties of distributed computing, including HTTP access, data validation, and result parsing.

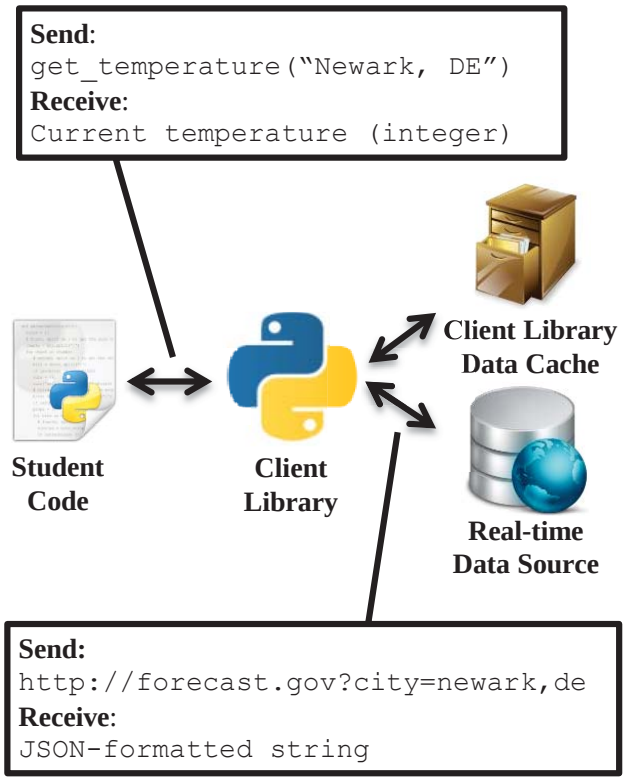

Figure 1. RealTimeWeb Client Library Architecture

At the heart of our project are carefully engineered, open-source client libraries through which students can access the data provided by real-time web services. We provide client libraries for a number of data sources, such as business reviews from Yelp, weather forecasts from the National Weather Service, and social content from link-sharing site Reddit.com. Each of these client libraries is in turn available for three common beginner languages: Java, Python, and Racket. These libraries do more than just streamline the process of accessing distributed data, however; each library is built with a persistence layer that enables the library to work without an internet connection. Not only does this ensure that students without a solid internet connection can maintain productivity, it also simplifies developing unit tests. Figure 1 demonstrates the architecture used in our libraries.

Our client libraries are easily available through a curated, online gallery; each library is designed to be quickly adapted to instructors' specific academic desires. This gallery also provides a tool for rapidly prototyping new libraries based on our framework. As an open-source project, we encourage collaborators to explore and extend the tools that we have created.

The success of the RealTimeWeb project is a large part of the motivation for this project. The lessons learned when integrating real-time data should transfer to this project where we seek to integrate Big Data. We expect similar outcomes and experiences for this new project.

5 Educational Infrastructure System Requirements

Based on our experiences of designing and deploying the RealTimeWeb toolchain, we have devised a list of major requirements that this new educational toolchain for Big Data should satisfy.

Wide Availability: There is no universal agreement within the Computer Science Education community on the perfect introductory language [15]. Indeed, individual instructor's answers will change depending on whether CS-1 or CS-2 is being considered. Python and Racket/Scheme are growing in popularity for CS-1, but Java is still a dominant choice for both CS-1 and CS-2 [8]. One of the great successes of the RealTimeWeb project that we are building off was the availabilty of every API in at least three common languages (Racket, Python, and Java). In order to ensure wide-spread adoption, this new infrastructure must also be available in a number of commonly-used languages.

Data transmission: Internet connections can be difficult and inconsistent, especially for off-campus and non-traditional students. Although most modern universities boast impressive wired connection speeds, these numbers rarely extend off-campus. And even when internet connections are top-notch, they can still be inadequate to serving the needs of transmitting Big Data sized collections to an entire classroom of students. To this end, student-ready instances must be made available through alternative methods: for instance, by streaming the data on demand, or using so-called Sneakerware solutions – copying the entire tool to a USB drive and physically delivering this to the student. By developing a robust software solution, instructors are not at the mercy of a single point of failure.

Confidential and Stale data: Big Data sets are often full of confidential data, especially if a student is using one from their own lives. These datasets need to be sanitizable – fields with private data need to be replaced with convincing fake data that protects identities and information. Of course, some students will never feel comfortable using their information; in this case, completely anonymized, stock datasets that match the schema must be provided for students.

Most datasets have some form of time-dependent data, usually the time that it was collected. In this modern age, data can become out of data rapidly. Although some information is timeless and is useful for historical purposes, it can be generally expected that students would prefer working with up-to-date, cutting edge material. Therefore, it is important that datasets either be susceptible to data fudging, or that they are filtered by timeliness.

Large Audience Compliance: Each student in a classroom can have wildly varying hardware; the threshold for data to be Big is different for each of these platforms. A student with a quad-core powered desktop can crunch significantly more data than a student with a five-year-old hand-me-down laptop. Ensuring that two students are having the same experience with a single dataset is a serious challenge. Although one method is to offshore all computation to a single device on the cloud, this is extremely restrictive for students, and brings up serious problems relating to network availability and speed. Instead, a preferred requirement is to ensure that the infrastructure can adapt datasets to the hardware of the system; the quad-core computer should be given a more massive dataset than the laptop.

Authentication: One of the big dangers when attempting to create context is the problem of *Preauthentication*: attempting to design for authenticity without sufficient knowledge of the audience. [16] gives a compelling example:

> The task of balancing a checkbook, for instance, may be an authentic task from the perspective of a 21-year-old, but we would question its authenticity from the perspective of a 5-year-old. But more to the point, even among 21-year-olds, for whom we believe the task should be authentic, there are some who will find any given lesson in personal finance irrelevant, inaccurate, or otherwise inappropriate.

Preauthentication stems from over-generalizations and run-away assumptions; if you attempt to reduce an entire classroom of potential learners to a list of likes and dislikes, you are running a serious risk of ignoring each individual learner's rich history and background that they will be building off. It is difficult to plan for and work against this ever-present danger when designing reusable assignments. [16] recommends that rather than attempting to design around students prior understanding, it is better to simply convince the learner of the authenticity of the problem. But this is extremely limiting, since it completely ignores the prior experiences and understanding that a student brings to their learning. Instead, we propose a middle ground where assignments are built with several possible datasets, and students are allowed to choose the one they find most intriguing; Empowering students with the choice to explore topics of their interest is well understood to be highly motivating [11]. At the same time, convincing rhetoric should be included to emphasize the importance of the problem.

System usability: As the proposed software systems are meant to be used by novices, it is especially critical that all interfaces be as simple and clean as possible. These interfaces will be read-only; there is no reason for the datasets to be mutated, ensuring a consistently correct dataset with idempotent results. The internal structural content of the dataset should be easily accessible to the student using simple method calls, returning an intuitive object model built of native language constructs.

Although we expect all instantiated APIs to have clear instructions and reference materials, significant care must be taken to design the APIs themselves to adhere to battle-tested Software Engineering principles. For example, following conventional naming standards, using strong encapsulation, supporting unit and mock testing, and documenting the code copiously. Finally, example applications using the APIs should be provided, in order to demonstrate the full range of functionality of the APIs.

Supporting automated grading systems: Introductory courses tend to have large enrollments, a trend that will only grow as the Computational Thinking movement spreads. These students present a serious burden for graders. Automated grading systems provide a method for handling the immense quantity of students. These tools can perform enhanced unit testing that provide students with feedback by analyzing the number of failed tests and the overall coverage of the tested code base. Building software compatible with these tools usually requires careful software engineering practices, such as avoiding Singletons in favor of Single Instances, to avoid problems with repeatedly running student assignments. At Virginia Tech, our colleague Steven Edwards leads a highly successful automated testing initiative called Web-CAT [6].

6 Proposed Architecture

In order to satisfy the requirements listed above, we propose a framework for generating APIs between students and datasets. This framework will be configurable not just by the instructor for a dataset, but also automatically attuned to the individual student and their associated hardware. The first time that a particular configuration for a student's platform is generated, it can be cached within the API to enhance future performance. Expected, configurable, primary components of the framework are identified below, although more architectural components may be necessary for different datasets.

Dataset selection: In the simplest case, an API will be connected explicitly to a single dataset from which all data will flow; this simplest form requires no further configuration. However, the API will also expose an interface for manually setting the location of the dataset. This specialization offers a twofold advantage. First, instructors gain the ability to design different datasets with different properties – e.g., an empty dataset or a dataset full of edge cases meant to strain students test cases. Second, students can provide their own copy of the dataset; in the prior Facebook example, for example, students might run their code on their own downloaded Facebook data archive. They might also wish to use a stock data archive if they do not have their own Facebook data, or if they do not wish to access it. As long as the instructor has clearly specified a schema to the API, any schematically-valid data source will be usable in an instance of the API.

Hardware attunement: As previously mentioned, a class of students will utilize an assortment of different kinds of hardware. Each individual instance of an API must be configured for the particular system it is being run on. Internal profiling of relevant operations will be performed in order to determine what the capabilities of this platform are, providing valuable information for future configuration. The exact methods for profiling will vary depending upon the implementation language, but will generally need to empirically measure a variety of actions such as threaded operations and ram size. As these tests could be considered intrusive and resource-intensive, we will offer coarse-grained, precomputed performance profiles that students and instructors can alternatively use, bypassing the profiling phase completely.

Data Massage: The process of moving data is sometimes characterized as "Extract-Transform-Load". This component is concerned with the middle phase, transformation. Often, certain fields of a dataset need to be modified on the fly: time dependent fields might need to be updated to appear more interesting; semi-confidential data might need to be replaced with blank information; an inadequately sized dataset might need to be padded with fake records. Through mocking and filtering utilities, available in most modern languages through libraries, data can be manipulated and altered in accordance with some general patterns to enhance the adaptability of the dataset.

Sampling: Often, students will not want to work with an entire Big Data set when they first start developing their solution. Therefore, the API should be able to return iteratively larger and larger sized samples of the dataset. Ultimately, students will need to test their program against the full-sized dataset, but they can do so on their terms, when they are prepared. In addition, students can see the impact that different sizes of data have on the runtime performance of their algorithms. Critical caching support can be provided within this component in order to improve runtime performance of the API.

Networking: Some Big datasets are so prohibitively large that it is infeasible to have each student run a local version of the API. Instead, a server instance can be provided by the instructor, with each student receiving a thin client library for connecting through the system. This networking API will also take advantage of internal caching in order to enhance runtime and maintain a uninhibited experience during network outages and slow periods.

7 Educational Infrastructure Implementation Plan

We propose to build off the success of the RealTimeWeb project to create a generalized, adaptable framework that can be instantiated with any Big Data source. This API instance will manage complexity while providing students with a uniform experience, while maintaining schematic correctness and student privacy.

To rapidly convert any third-party, massive, dataset into a practical educational resource, the proposed deliverable will utilize standardized Software Engineering methodologies and techniques. To that end, we will generate an extensible software framework. Libraries and frameworks are key building blocks for software development – they provide predefined, extensible software components. Libraries and frameworks are similar, but subtly different in an important way. While both libraries and frameworks offer recyclable elements of functionality, frameworks also define an architecture that the developers can work around to build their software. A typical Object-Oriented software framework is composed of a collection of classes that define some central features. To create application-specific features, the developer can then add custom classes that extend abstract base classes. By developing our project as a framework, instructors that want to work with a new data source can simply extend our architecture. They are free to cherry-pick the features and functionalities that we offer through our system.

In addition to this generalized framework, we also propose to create a collection of pregenerated APIs based on an assortment of existing Big datasets. These pregenerated APIs will be associated with multiple assignments that instructors can rapidly pick up and integrate into their course without extra work. Where possible, APIs will be agnostic to their assignments so that students can swap out an API for one they find more interesting, empowering the student as an active agent in their learning. Designing these API-agnostic assignment must be done exceptionally carefully, since there is a strong penalty on the instructor when grading multiple kinds of assignments. This penalty is critical to avoid, since an important goal of this project is easing the burden on the instructor; having to grade several different kinds of projects is non-trivial.

8 Impacts on the CS Curriculum

In this section, we offer demonstrative scenarios for using the proposed infrastructure in actual introductory Computer Science courses. These scenarios are similar to those perceived during the RealTimeWeb interventions. The RealTimeWeb client libraries were deployed in three different introductory classes at Virginia Tech and the University of Delaware. For example, within the first few weeks of the CS-1 course at the University of Delaware, students created an interactive weather application in Racket using streaming data from the National Weather Service.

A total of 194 students were surveyed afterward on their experience using the libraries. Students reported increased engagement with their assignments and an improved understanding of the challenges of working with Real-Time data, with a majority of students requesting that as many or more assignments integrate RealTimeWeb libraries. Students also reported that the added authenticity of the assignments made it more interesting. This success of the RealTimeWeb offers a strong model to follow as we shift from Real-Time Data to Big Data.

Using our Big Data APIs, students can answer interesting questions as Computational Scientists; for instance, CS-1 students could learn about map-filter-reduce in the context of analyzing their Facebook data. They can also learn about algorithmic efficiency in terms of time and space; picture CS-2 students learning about the benefits of using linear data structures to store data vs. tree data structures. Students could even learn about interesting visualization techniques for large data sets; CS-3 students could learn how to model large web repositories (e.g., Wikipedia offline) as a graph of connected websites.

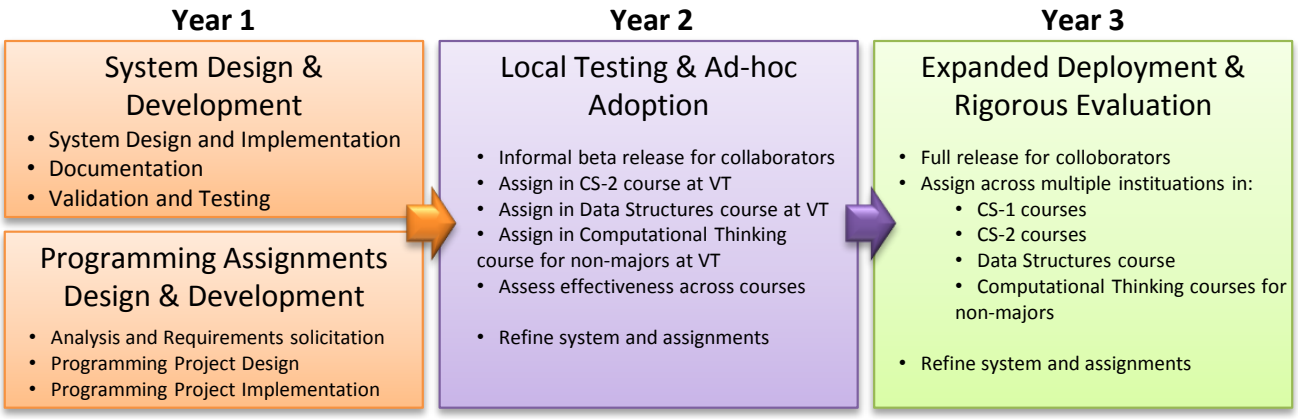

Figure 2. Work Plan

9 Work Plan

Figure 2 shows the proposed three-phase schedule for this project. The first year will be spent primarily designing and developing the proposed educational infrastructure by building off the lessons we learned when developing the RealTimeWeb project. A key element of this phase will be identifying a set of at least 5 representative Big Data sources to instantiate through our framework. This work will be spear-headed by Austin Bart and supervised by Dr. Tilevich, with Dr. Shaffer testing for usability and correctness. In parallel with the development of the instantiated libraries, we will also design and prototype new programming assignments and projects based on each of the data sources. Dr. Tilevich will devise projects for the CS2 course, while Dr. Shaffer will devise projects for the Data Structures and Algorithms (CS3) course. These assignments will be test-run on the graduate research assistants in order to verify correctness, authenticity, and that educational objectives are being covered.

As the project enters the second year, we will begin deploying the libraries in classrooms at Virginia Tech and collaborating institutions. Positive feedback at conferences has already spread adoption of RealTimeWeb resources, which will provide a channel for deploying our new Big Data-oriented resources. The scale and quantity of courses and sections that use the framework will increase during the spring, in order to provide control data on the impact of the learning objectives. Feedback from each semester will inform design changes in successive semesters. Throughout the second year we will assess and summarize our findings regarding the effectiveness of the created research infrastructure and their impacts on the courses and the overall curriculum. These preliminary findings will be published and used to advertise for further potential collaborators.

By the third year of the project, we will have implemented at least 5 new additional bindings for the framework. More importantly, however, we will find 3 collaborating peer institutions to deploy our assignments and gather extensive data on their effectiveness. This will generate highly useful data on how the assignments and tools can be generalized to a wider audience of instructors while retaining the powerful contextualization to individual students. The extensive amount of data that can be collected at this scale will contribute solid evidence to the educational theories that this project rests on.

10 Evaluation Plan

This project has two results: a new software framework for rapidly incorporating Big Data sources into programming assignments, and a set of introductory assignments using these scaffolded Big Data sources. We plan to empirically determine the success of these resources by answering the following questions:

1. Does working with these Big Data sources have a positive impact on student retention in Computer Science?
2. Are students more engaged during assignments that use Big Data sources?
3. Is our scaffolding easy-to-use and reliable?
4. Do students have a better understanding of the challenges, value, and concepts of Big Data?
5. Are students more likely to engage in computing related research and careers?

To answer the first question, we will use historical data collected by the department on retention and drop-out points in the program. These statistics will be compared before and after the intervention in order to determine the success of this approach. We will show empirically whether realistic computing experiences have a significant, positive impact on student retention. Additionally, we will conduct qualitative interviews with students to determine how contextualization has impacted their decision to stay or leave Computer Science.

In addition to boosting retention, it is expected that our additions will engage students in the projects and motivate them to work harder and learn skills more deeply. Relevancy and interest are known contributing factors for engaging students [10]. In order to measure these and other factors of motivation, we will use the MUSIC model of academic motivation as our theoretical foundation. The MUSIC model is a well-supported theory that identifies five key constructs in motivating students:

eMpowerment The amount of control that a student feels that they have over an assignment.

Usefulness The expectation of the student that the material they are learning will be valuable to their short and long term goals.

Success The student's belief in their own ability to complete an assignment.

Interest The student's perception of how the assignment appeals to situational or long-term interests.

Caring The students perception of their professor's and classmates attitudes toward them.

The MUSIC model has compelling evidence supporting the validity of its associated instrument, the MUSIC Model of Academic Motivation Inventory [11]. This instrument will be deployed before, during, and after courses that use our resources, alongside qualitative questions that will isolate students' perceptions of our curriculum in light of these five elements. This data will indicate whether psychological constructs related to motivation can be affected through our assignments.

The software itself must be also evaluated for its ease-of-use. The datasets and associated technology scaffolding must be understandable, reliable, and yet still appropriately challenging. We will include

questions on our survey that gauge students difficulty and comprehension of the systems. Qualitative data will also be gathered on the students experiences and used as formative and summative feedback to improve the quality of the end-product. Ultimately, we will perform requirements analysis to evaluate how the software achieves each of the stated requirements for our software.

We also intend to show that students gain an increased understanding and appreciation for the scale of Big Data problems and the methods required to solve them. The depth that individual classes will integrate Big Data concepts will vary; earlier classes may seek to only use the problems to contextualize assignments, while later classes will require students to develop a fuller understanding that can be expanded upon in future courses involving Big Data. However, since all students will receive some level of instruction on the nature of Big Data, they should be able to answer questions related to it. For instance, non-majors that use our tools should be taught and assessed on WHY the tool is necessary - that is, what some of the specific technological factors are that make Big Data difficult to work with. More advanced students should also be able to explain how our abstractions work, and how they could implement them themselves.

As a 3-year project, longitudinal data must be collected to show long-term behavior changes in the students that use our tools. Students will be tracked as they move through their undergraduate program. Follow-up evaluations of these students as they progress can shed further light on whether foundational principles were successfully established. Additionally, we can identify whether students were more likely to work on research problems or take internships that involve Big Data-related challenges.

The above measures will be used in symphony with each other during each of our deployments in the second and third phase of our work plan. Usability studies, and preliminary forms of the other measures, will be used during the first year to guide development. However, a final measure will be used to determine the impact of this project as a whole: the rate of adoption. During the second and third phases, we will be deploying these resources in partner universities through the same channels as our RealTimeWeb project. We will consider this a success if we have at least three serious external adopters that also use our evaluation process. This will ensure validity of our data in a wider context.

11 Dissemination Plan

In order to distribute the resources created during this project, we will reserve the domain www.BigDataEd.edu. This website will host all of the content we've created in a form easily consumed both by VT students and eventually colleagues. A primary design goal is to make our software portable and adaptable, so that other instructors can rapidly integrate our existing and materials designed with our tools into their introductory classrooms. In addition to hosting the latest iteration of our educational framework, we will also keep it updated with news on the project and resources for teaching and learning about Big Data.

This website also offers an excellent opportunity to showcase not only completed instantiations of our framework, but also exemplary student projects created using our tools (with the agreement of willing students). Particularly spectacular projects can be encouraged to be developed into professional software applications. Motivated students can be guided through formalized Independent Studies, summer research and internship opportunities, or even through non-formal code jams – events of rising popularity in the Computer Science Education community. When fully realized, these projects will be included in a special section of the website, along with educational explanations of the design team's process.

All software created for this project will be open-source and liberally licensed under an MIT license. We have had great success hosting our RealTimeWeb framework and instantiated client libraries on

open project hosting services such as GitHub, which allows potential collaborators to browse, fork, and contribute to our work.

In the third phase of the project, we will target a useful sample of colleagues at peer institutions to pilot and test our software toolchain and the associated instantiations. During the RealTimeWeb project, despite being only a TUES I project, we successfully collaborated with the University of Delaware to use two of our client libraries, with great results. By offering constant technical support, we were able to rapidly solve any problems that the instructors met, and provided students with an exciting and enriching experience. This same level of technical assistance will be available to all colleagues who volunteer to deploy and pilot the materials developed during this project.

All theoretical and evidence-based conclusions researched during this project will be disseminated through publications and workshops at relevant international and national conferences. There are a large number of appropriate, recognized conferences that cover topics in Computer Science Education, including SIGCSE, ITiCSE, FIE, CCSC and/or ASEE. Relevant journals include ACM Transactions on Computing Education (previously, JERIC), Elseviers Computers & Education: An International Journal, IEEE Transactions on Education, and Taylor & Francis Computer Science Education. Finally, tutorials, API references, and technical reports (regarding software analysis, implemented libraries, and infrastructure development) will all be made available through our website and open source repositories.

In order to advertise and promote our toolchain, we will publish through highly visible communication channels within the CS Education community, such as the SIGCSE listserv. We intend to host a formal workshop on using our materials at a conference such as SIGCSE; this will build on the success of the RealTimeWeb project's workshop at SIGCSE, the acceptance of which indicates the demand from researchers for such resources.

12 Broadening Participation: Minorities and Undergraduates

In their education-related projects, both Dr. Tilevich and Dr. Shaffer have a history of regularly providing opportunities for undergraduates. The VT Bus Tracker project was essentially started by undergraduates in Dr. Tilevichs class. The work has continued as independent research projects. Dr. Shaffer has had many students work on algorithm visualizations over the past 10 years, including both undergraduate independent study students during the academic year and summer intern students. The VT CS department generally has had success recruiting African-American female students from neighboring Historically Black College and Universities (HBCU). We have three recent female African American PhDs in computing, (Tracy Lewis, at Radford University, Cheryl Seals at Auburn University, and Jamika Burge at Penn State). We currently are recruiting African-American students from Auburn University, as part of our collaboration with Dr. Juan Gilbert. The success of women in Computer Science in the department is reflected by a high percentage of Ph.D. graduates. Specifically, 10 out of 50 Ph.D. graduates from Spring 2008 to Spring 2011 have been female, which is higher than the national average. We currently have four Latino students in our graduate program (2 PhD students and 2 Masters). One of them, Ricardo Quintana-Castillo, is a recipient of an NSF Graduate fellowship. Dr. Shaffers NSF-funded AlgoViz projects have supported four graduate students: A.J. Alon (minority), Monika Akbar (female), Michael Stewart (first-generation college graduate), and Eric Fouh (minority). The CS department participates in the NSFs Research Experience for Undergraduates (REU) and Broadening Participation in Computing (BPC) program, which attracts talented and enthusiastic undergraduate students from our partner minority serving institutions during summer sessions. As part of this program, Dr. Tilevich

mentored a female student, SooBeen Kim, and the results of her research project led to a full research paper presented at ICPC 2009 [9]. We will continue to mentor CS@VT REU/BPC students as well as VT MAOP1 [23] students. Both programs are nationally recognized for involving women and minority students in quality research programs. PI Tilevich has been mentoring a female CS undergraduate researcher, Kristin Whetstone, who has won two prizes for her research in the 2010 VT Department of Computer Science annual undergraduate research competition. Her research abstract based on the work she has been pursuing under Tilevichs supervision was accepted to the poster session at the Grace Hopper Celebration ofWomen in Computing 2010, and also selected for the ACM Student Research Competition (SRC).

13 Key Staff

PI: Eli Tilevich, ¡continued¿

Co-PI: Clifford A. Shaffer, ¡continued¿

Graduate Research Assistant: Austin Cory Bart, Doctoral Student in Computer Science at Virginia Tech, is a member of the Software Innovations Lab. In addition to his PhD, Bart is concurrently pursuing a certification in the Learning Sciences to augment his research and technical expertise with a solid foundation of educational theory and practice. In addition to the two publications from the RealTimeWeb project at SIGCSE and SPLASH-E, Bart was awarded an NSF Graduate Research Fellowship Honorable Mention. As the research assistant for the RealTimeWeb project, he was the lead architect of the software and helped to design the associated pedagogical materials. As exemplified by his work with the local chapter for the Association for Women in Computing, he is committed to creating educational tools for broadening participation and closing the gender gap in computer science. He will be in charge of developing the materials for this new project and organizing their deployment in Computer Science classrooms.

Graduate, undergraduate, and intern student: In addition to Bart, this project will hire another graduate research assistant to perform mission critical development. Virginia Tech's Computer Science department has had great prior success in employing undergraduate students to assist in educational research projects, to the mutual benefit of both the student and the project. This undergraduate assistant-ship is formalized through the VTURCS program. The foundation of the RealTimeWeb project was co-implemented by undergraduate volunteers, and this tradition will be continued by introducing a number of undergraduates through independent study projects and summer research opportunities.

14 Results from Prior NSF Support

NSF CCLI Phase 1 Award (DUE-0836940): *Building a Community and Establishing Best Practices in Algorithm Visualization through the AlgoViz Wiki*. PIs: Clifford A. Shaffer and Stephen H. Edwards. $149,206. for 01/2009 12/2010. **NSF NSDL Small Project (DUE-0937863)**: *The AlgoViz Portal: Lowering Barriers for Entry into an Online Educational Community*. PIs: Clifford A. Shaffer and Stephen H. Edwards, $149,999 for 01/2010 12/2011. These projects seek to developed online infrastructure (a website and related community development efforts) to promote use of algorithm visualization (AV) in computer science courses. This is achieved by building a community of AV users and developers. So far, one journal paper [20] and four conference papers [18] [19] [21] [22] have been published relating to this work.

References

[1] C. Anderson. The end of theory. *Wired magazine*, 16, 2008.

[2] R. E. Anderson, M. D. Ernst, R. Ordóñez, P. Pham, and S. A. Wolfman. Introductory programming meets the real world: Using real problems and data in cs1. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 465–466, New York, NY, USA, 2014. ACM.

[3] A. C. Bart, E. Tilevich, S. Hall, T. Allevato, and C. A. Shaffer. Transforming introductory computer science projects via real-time web data. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 289–294, New York, NY, USA, 2014. ACM.

[4] J. D. Bransford, A. L. Brown, R. R. Cocking, et al. *How people learn*. National Academy Press Washington, DC, 2000.

[5] L. Carter. Why students with an apparent aptitude for computer science don't choose to major in computer science. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '06, pages 27–31, New York, NY, USA, 2006. ACM.

[6] S. H. Edwards and M. A. Perez-Quinones. Web-cat: Automatically grading programming assignments. *SIGCSE Bull.*, 40(3):328–328, June 2008.

[7] A. E. Egger. Engaging students in earthquakes via real-time data and decisions. *Science*, 336(6089):1654–1655, 2012.

[8] M. Hertz and S. M. Ford. Investigating factors of student learning in introductory courses. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 195–200, New York, NY, USA, 2013. ACM.

[9] K. Hussein, E. Tilevich, I. I. Bukvic, and S. Kim. Sonification design guidelines to enhance program comprehension. In *Program Comprehension, 2009. ICPC'09. IEEE 17th International Conference on*, pages 120–129. IEEE, 2009.

[10] B. D. Jones. Motivating students to engage in learning: The MUSIC model of academic motivation. *International Journal of Teaching and Learning in Higher Education*, 21(2):272–285, 2009.

[11] B. D. Jones and G. Skaggs. *Validation of the MUSIC Model of Academic Motivation Inventory: A measure of students' motivation in college courses*. Research presented at the International Conference on Motivation 2012, 2012.

[12] M. Koehler and P. Mishra. What is technological pedagogical content knowledge (tpack)? *Contemporary Issues in Technology and Teacher Education*, 9(1):60–70, 2009.

[13] J. Lave and E. Wenger. *Situated learning: Legitimate peripheral participation*. Cambridge university press, 1991.

[14] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers. Big data: The next frontier for innovation, competition, and productivity. 2011.

[15] A.-C. J. T. F. on Computing Curricula. Computer science curricula 2013. Technical report, ACM Press and IEEE Computer Society Press, December 2013.

[16] J. Petraglia. The real world on a short leash: The (mis) application of constructivism to the design of educational technology. *Educational Technology Research and Development*, 46(3):53–65, 1998.

[17] K. Sanders, J. Boustedt, A. Eckerdal, R. McCartney, J. E. Moström, L. Thomas, and C. Zander. Threshold concepts and threshold skills in computing. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research*, ICER '12, pages 23–30, New York, NY, USA, 2012. ACM.

[18] C. A. Shaffer, M. Akbar, A. J. D. Alon, M. Stewart, and S. H. Edwards. Getting algorithm visualizations into the classroom. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 129–134. ACM, 2011.

[19] C. A. Shaffer, M. Cooper, and S. H. Edwards. Algorithm visualization: a report on the state of the field. In *ACM SIGCSE Bulletin*, volume 39, pages 150–154. ACM, 2007.

[20] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce, and S. H. Edwards. Algorithm visualization: The state of the field. *ACM Transactions on Computing Education (TOCE)*, 10(3):9, 2010.

[21] C. A. Shaffer, T. L. Naps, and E. Fouh. Truly interactive textbooks for computer science education. In *Proceedings of the 6th Program Visualization Workshop*, pages 97–103, 2011.

[22] C. A. Shaffer, T. L. Naps, S. H. Rodger, and S. H. Edwards. Building an online educational community for algorithm visualization. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 475–476. ACM, 2010.

[23] V. Tech. The multicultural academic opportunities program (maop), dec 2009.

[24] L. Torgo. *Data Mining with R, learning with case studies*. Chapman and Hall/CRC, 2010.

[25] M. Waldman. Keeping it real: utilizing NYC open data in an introduction to database systems course. *J. Comput. Sci. Coll.*, 28(6):156–161, June 2013.