Authentic Contextualization in Computer Science Education with Big Data: Pedagogy, Motivation, and Technology

Austin Cory Bart

October 24, 2014

1 Summary

2 Problem Statement

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" [11] . This gap, and the expanding recognition of "Computational Thinking" as a 21st century skill, increasingly requires that computation be positioned in a university's general education curriculum [16]. For instance, Virginia Tech is now formalizing this requirement through a new course ("Introduction to Computational Thinking") that all university students will eventually be required to take [12]. Such a course poses serious pedagogical and motivational challenges: how do we introduce Computational Thinking to students with no prior computing experience and convince them that the field can tangibly benefit their respective disciplines?

We need more students in Computer Science and with Computational skills Describe introductory programming CS-1,2,3: Quick one-sentence descriptions As students progress to CS-3, motivation becomes less critical compared to self-regulation CS-0: "Computational Thinking" Limited time to teach relevant concepts Desire to engage them In order to bring in these students, we must approach this from both the cognitive and motivational sides Want to keep students highly motivated Decontextualized assignments But we also have a new set of skills to teach students Data science Big Data What can technology and pedagogy do to solve this problem In this preliminary proposal, we describe work previously done and future work

Computer Science and Computational Thinking New audiences

Introductory programming CS-1,2,3 Computational Thinking Wing definition Contextualization Abstract Straw-man example of boring lists Constructivism Socio-constructivism Situated Learning Authentication Preauthentication Over-authentication (?) - I like dinosaurs, but you suck at teaching dinosaurs Scaffolding Fading Media Computation Pre-authentication - students won't find it realistic to their experiences Under-authentication - not realistic enough for the students tastes Over-authentication - overly engaging and distracting (Seductive Details)

http://dl.acm.org/citation.cfm?id=2493397 Proposed Theory Retention hypothesis Gender Hypothesis Learning Hypothesis Gains vs. Absolute More-Computing Hypothesis Code Aacdemy, Let's Code Blacksburg

Prior Work Real-time Web Proposed Architecture +Big-Data +Sources Python + Blockly Proposed curriculum Usable as components or a whole Good for students motivated by usefulness Work Plan

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." [11] 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 [11]. Second, postponing these topics until late in the curriculum, as is commonly done [13], 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?

3 Educational Theories of Motivation and Cognition

There are many theories of how people learn. While several modern learning theories are typically applied within Computer Science Education, one of the more popular is Situated Learning Theory [4].

Cognition vs. Motivation Situated Learning

3.1 Situated Learning Theory

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 [10]. 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.

A critical element of these situated environments is the need for social interaction and collaboration, as learners become involved in and acculturated by "Communities of Practice" [5]. Members of a CoP share purposes, tools, processes, and a general direction – they should have a commonly recognized domain. This interaction occurs not only between individuals and the experts of the community (commonly represented by teachers) in an apprenticeship model, but also occurs between different learners as they adapt at uneven paces. This communication between peers leads to growth among both individuals, especially when access to authentic experts is limited. As the learner develops into an expert, they shift from the outside of the community's circle to the center, becoming more and more engaged – this is the process of "Legitimate Peripheral Participation".

Authenticity is another crucial, recurring theme within Situated Learning Theory. All instruction and assessment must be aligned with reality such that success in the former leads to success in the latter. However, there is a subtle nuance here – authenticity is a perceived trait, not an objective one. Students derive value from their learning only if they *perceive* authenticity, regardless of whether the instructor has successfully authenticated the experience.

Situated Learning Theory has been applied to the domain of Computer Science before, with mixed results. A seminal paper by Ben-Ari [4] explores its application and limitations. This paper is somewhat hasty in its application of SL Theory by taking a macro-level view – they narrowly look to Open-Source and Industry Software Development communities as the only potential CoPs and interpret SL Theory as strictly requiring constant legitimacy, largely ignoring the possibility for gradual development of authenticity within individual courses and modules throughout a curriculum:

> What I am claiming is that situated learning as presented in their work cannot be accepted at face value, because it simply ignores the enormous gap between the world of education and the world of high-tech CoPs, which demand extensive knowledge of both CS subjects and applications areas. This gap can only be bridged by classical decontextualized teaching in high schools, colleges and universities.

However, other researchers have found it a useful lens for analyzing curriculums. For instance, Guzdial and Tew [7] used the theory to innovatively explore and deal with the problem of inauthenticity within their Media Computation project. SL Theory clearly has value, but only as a function of the way that it is applied. In this preliminary proposal, I will take advantage of SL Theory as a generalized tool for exploring the topic of authenticity throughout introductory Computer Science.

3.2 The Application of Situated Learning Theory

The original work in Situated Learning Theory was categorically not about pedagogy or instructional design- it described how people learn and the importance of context and collaboration, but it did not recommend a particular style of classroom. However, subsequent research by Brown [5] and others expanded the theory so that it could be applied to the design of learning experiences. These expansions often naturally dictate the use of active learning techniques, demphasizing the role of lecture in favor of collaborative, problem-based learning activities.

Choi & Hannafin [6] describe a particularly useful, concrete framework for designing situated learning environments and experiences. Because there is no official name given to this framework, as a matter of convenience we will refer to it as the Situated Learning Environment Design Framework (SLED Framework). This framework has four key principles:

Context "... The problem's physical and conceptual structure as well as the purpose of activity and the social milieu in which it is embedded" [15], context is driven not just by the atmosphere of the problem at hand, but also by the background and culture surrounding the problem. A good context enables a student to find recognizable elements and build on prior understanding, eventually being able to freely transfer their learning to new contexts.

Content The information intending to be conveyed to the students. If context is the backdrop to the learning, then content might be seen as the plot. Naturally, context and content are deeply intertwined with each other, and its difficult to talk about one without referencing the other; in fact, content is an abstract entity that needs to be made concrete through contextualization when it is delivered to the learner. If the information is too abstract, than it will never connect with the learner and will not be transferable to new domain. However, if it is too grounded in a domain, then it will not be clear how it can be re-applied elsewhere. Ultimately, content must be given in a variety of forms to maximize transfer. Two useful methods for building content are anchored instruction (exploring scenarios, or anchors, in the context based on the content) and cognitive apprenticeship (mediating knowledge from an expert to the novice learner in a mentoring relationship).

Facilitations The modifications to the learning experience that support and accelerate learning. Facilitations provide opportunities for students to internalize what they are learning by lowering the barriers that can surround situated experiences, possibly at the cost of some amount of authenticity. These modifications might be technological in nature, but they can also be pedagogical. Although there are many different forms that Facilitations can take, Scaffolding is one of the most common. Scaffolding is a form of support that is intended to extend what a learner can accomplish on their own. This support is required at the onset of the learning process, but is unnecessary once a sufficient threshold has been passed; during this transition, the amount of scaffolding can be tuned to the learners understanding. In Computer Science, for instance, students often take advantage of software libraries and frameworks to create sophisticated graphical programs that would be beyond daunting if implemented from scratch.

Assessment The methods used to assess the learning experience and the progress of the student. Choi & Hannafin gives special attention to the teach to the test problem, and how assessment needs to change to measure students ability to solve authentic problem (as opposed to their ability to solve the tests specific problem), and to be able to transfer their understanding when solving different but related problems. It is important that assessment is measured against the individualized goals and progress of a learner, requiring that any standards used be fluid and adaptable to different learners personal situations. Of course, assessment should be an on-going part of the learning process, providing feedback and diagnostics. Ultimately, the learner should join in the process of assessment as they transition to an expert being able to meta-cognitively self-evaluate the effectiveness of ones methods and communicate results to others are key abilities of experts.

vs. Constructivism Learning Theories are lenses - they do not provide all the answers, but they simply give you a way to explore and analyze

Authentication is a tricky thing Hazards of authentication: Pre-authentication - Everyone loves dinosaurs, so today's lesson uses dinosaur examples. What do you mean you don't want to learn about dinosaurs?

3.3 MUSIC Model of Academic Motivation

Situated Learning is a theory of learning, but is not a comprehensive motivational framework – it describes how people learn, but it is limited in explaining why people commit to learning. Instead, it is useful to turn to the dedicated motivational models as a lens to explore why people choose to participate and excel in Computer Science. In this preliminary proposal, we lean on the MUSIC Model of Academic Motivation as a primary framework.

The decision to use the MUSIC model was based on several criteria. Although there are many motivational models available, few strive to be holistic models specifically developed for academics. For example, theories like Expectancy-Value and Cognitive Evaluation Theory have a wider scope and have stemmed from other disciplines such as healthcare. The MUSIC Model is derived from a meta-analysis of these other theories, incorporating only the academically relevant components. Further, the MUSIC model is a tool meant for both design and evaluation, allowing it to be used in all phases of this work. Finally, the model and its associated instrument, the MUSIC Model of Academic Motivation Inventory (MMAMI) , has been extensively validated and utilized in other educational domains, making it a reliable device [9].

The MUSIC model identifies five key constructs in motivating students [8]:

eMpowerment: The amount of control that a student feels that they have over their learning – e.g., course assignments, lecture topics, etc..

Usefulness: The expectation of the student that the material they are learning will be valuable to their short and long term goals. There is no clear delineation of the time-scale for these goals, but there is nonetheless a distinction between strategic skills that students need to be successful in careers and personal interests and the tactical skills they need to complete present-day tasks.

Success: The student's belief in their own ability to complete assignments, projects, and other 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. The former covers the aspects of a course related to attention, while the latter covers topics related to the fully-identified areas of focuses of the student.

Caring: The students perception of other stakeholders' attitudes toward them. These stakeholders primarily include their instructor and classmates, but also can be extended to consider other members of their learning experience (e.g., administration, external experts, etc.).

Students are motivated when one or more of these constructs is sufficiently activated. They are not all required to achieve maximal levels, and in fact that is not always desired – it is possible, for instance, for a student to feel too empowered, and become overwhelmed by possibilities. For some of these constructs, a careful balance is required, and it may not be possible to ever achieve a minimal level; no matter how exciting you make your lecture, you may never convince your students it is interesting, although it is possible that they will still consider it useful and stay motivated. Much like in Situated Learning Theory, students' subjective *perception* of these constructs is a defining requirement and is more important than objective reality.

The MUSIC model is often used as an organizational framework and an evaluative tool. As the former, it is a list of factors to consider when building modules, assignments, and content of a course. At all times, instructors can consider whether they are leveraging at least one construct to motivate their students. As the latter, it offers both a quantified instrument (MMAMI) and a structure to anchor a qualitative investigation on. The model has also directly been used tactically in course design: Jones describes a controlled classroom experiment to motivate students by having an experimental group reflect on how a course satisfies the constructs of the MUSIC model (e.g., prompted to answer "How will the material presented here will be useful to you?"). Quantitative data gathered after the experiment indicated a significant increase in motivation.

3.4 Summary

There are many theories of learning, cognition, and motivation that provide different outlooks on educational processes, models, and artifacts. In the context of this paper, we limit ourselves to Situated Learning Theory and the MUSIC Model of Academic Motivation.

- Situated Learning Theory posits that learning inseparably occurs within contexts
- Active and collaborative learning experiences are superior to lectures
- Authenticity is a crucial component of learning and must be carefully considered.
- Curriculum materials can be concretely analyzed by looking to their Context, Content, Facilitations, and Assessment.
- The MUSIC Model posits that academic motivation is a function of a student perceptions of Empowerment, Usefulness, Success, Interest, and Caring within a learning experience.

4 A Theory of Big Data

Big data has been loosely described as quantities of information that cannot be handled with traditional methods [?]. But "traditional methods" is a vague phrase that has different meanings to different learners. To a Humanities major in their first CS-0 course, the traditional method to sum a list is to use Excel. In this scenario, "big data" means anything that won't comfortably fit into Excel's working memory. However, to a third-year Computer Science major, the traditional method would be to write an iterative or recursive sequential loop; being given big data forces them to explore parallel models of execution. Clearly, "bigness" is a function of the learner's experience, but that is still not a solid definition.

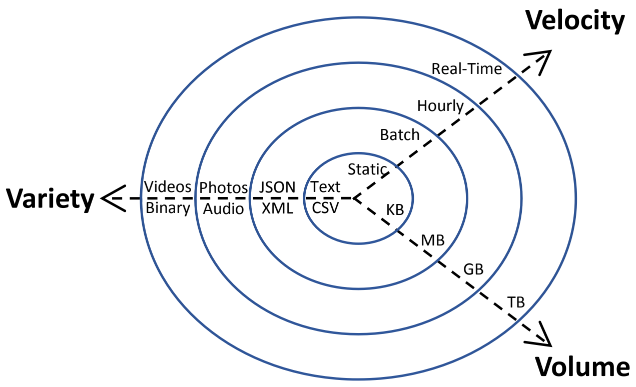

Figure 1. The 3V Model of Big Data

A more precise definition is the "3V Model" [?], which posits that there are three dimensions that distinguish big data from ordinary, run-of-the-mill data:

Volume: The total quantity of the information, usually measured in bytes or number of records. However, this also extends laterally: the number of fields in the structure of the data also impacts the complexity and size. The threshold at which data becomes big is a function of the hardware and software being used – for instance, embedded systems may consider gigabyte-sized files to be big, while modern servers might not struggle until the petabyte level.

Velocity: The rate at which new information is added to the system. High velocity big data implies a distributed architecture, since new data must be arriving from somewhere. The dynamicity of data can vary widely across these architectures, with data updating every year, every day, or even multiple times a second.

Variety: The format or formats of the data. Ideally, data are always distributed in a way that is readily accessible – for instance, simple text-based formats such as CSV and JSON are widely supported, relatively lightweight, and human-readable. More sophisticated data formats for image and audio are also typically well-supported, although still more complicated. However, projects using specialized, compressed binary formats or, more dangerously, multiple formats (e.g., image archives organized with XML files), are more complex.

4.1 Challenges of High Velocity Data

It is not trivial to enable introductory students to work with high velocity data, which is necessarily distributed. Without any scaffolding, it is necessary to delay the use of such data until much later in the course. In a prior paper [2], we outline the biggest barriers to high velocity data as a context:

Access The process of programmatically downloading and parsing a web-based resource is a non-trivial procedure requiring an understanding of both basic concepts (e.g., function calls, data transformation) and specialized web technology (e.g., the difference between GET and POST calls, building query parameters).

Non-Idempotency Because high velocity data is constantly changing, repeated calls to the same URL endpoint can return wildly different results, even over the course of a few minutes. This makes finding errors and testing considerably harder.

Consistency Web-based APIs are controlled and developed by independent entities, which means that changes can occur at any time with little to no notification or time for reaction. This means that students' code can become out of date even during the middle of testing their final project.

Connectivity Although internet speeds for students on a university campus are typically stable, this does not extend to off-campus students or students that are traveling. If the internet connection is down, then students might be completely unable to make progress.

Efficiency Even when the internet connection is stable, it might not always be fast. Requiring a round-trip to a server can greatly drag on the testing and development process, frustrating the student and decreasing the time spent learning.

4.2 Challenges of High Volume Data

In this section, we highlight some of the more challenging aspects of introducing high volume data, similar to how we previously outlined the challenges of high velocity data. Some of these challenges are technical in nature, and some of them of a more pedagogical nature. These challenges lead to certain design requirements that must be satisfied in any scaffolding intended to introduce high volume data.

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 speeds 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 collections to an entire classroom of students.

Storage: High Volume data is not just cumbersome to transmit, but also bothersome to store. Students may find that dealing with gigabytes of data, is an unpalatable experience.

File Management: Although learning how to interact with files on disk is a common topic in CS-1, it is usually reserved for the end of the course, and it may never be covered in a Computational Thinking course.

4.3 Challenges of High Variety Data

Inconsistency of Storage:

Inconsistency of Tools:

Distribution of Data: A common

4.4 General Challenges of Data

Intentionally secured data Differential privacy

Unintentionally secured data Dead URLs, PDF codebooks

Non-uniform topologies High volume data offers different contexts and problems than high velocity data. For instance, high velocity data typically lends itself to small quantities of data that are relevant to the current state of the real world – for instance, students can walk outside and feel the current weather, which should correlate to real-time weather reports made available by a weather library. High volume data, on the other hand, lends itself to large quantities of mostly static data – for instance, crime reports for a long period of time. Although high velocity data gives authentic answers in the here and now, high volume data gives authentic answers for the future through trends. Some fields have both kinds of data available – meteorologists generate forecasts (high velocity, low volume) by studying historical climate data (high volume, low velocity). But some fields are not amenable to both – digital historians typically have large stores of historical information (high volume), but it does not change quickly (low velocity).

Distribution of Data: A common method when distributing data

5 Reviewing Introductory Computing with Situated Learning Theory

In the following section, we review existing literature on the many forms that introductory computer science has taken. This is not intended to be an exhaustive look at all the ways that computer science has been taught to novices.

Throughout this section, we will refer to the SLED Framework and the MUSIC model in order to analyze existing approaches advantages and limitations.

Younger students tend to have less fully defined domains, which means that it is more important to rely on situational interests and short-term expectations of usefulness.

5.1 The Content of Computer Science

There is a serious, on-going debate about what novice students in Computer Science should learn, with limited consensus. Complicating this discussion is the bifurcation of introductory computing into CS-0 (which studies "Computational Thinking") and CS-1 (which studies pure "Computer Science"). There are key differences in these two educational pathways that strongly influence course design. First, CS-0 is a terminal course intended for non-majors – there is little-to-no expectation that they will take CS-1. By comparison, CS-1 is a foundational course meant for students majoring in Computer Science. Typically, expectations are lower for students in CS-0, and less material is covered – critically, there is a reduced emphasis on programming in favor of higher level knowledge of Computer Science.

However, it seems generally agreed that the CS-2 and CS-3 courses begin to focus on Data Structures and Algorithms.

Computer Science Curriculum 2013 [13] is an attempt to codify the entire breadth of an undergraduate CS education. However, it does not attempt to dictate the exact implementation and order of material.

Traditionally, certain topics are typical – decision, iteration, and the nature of data and control flow.

The "How to Design Programs" curriculum emphasizes a functional programming model, using a LISP-descended language named Racket. The only looping mechanism that students are taught is recursion.

The dominance of the Object-Oriented Model in software engineering usually leads to a strong emphasis in introductory courses – in fact, there is even an "Objects First" curriculum.

Test-Driven development

The strictest interpretation of SL Theory, as might be applied by Ben-Ari in [4], would seem to demand that the content of an introductory course should teach exactly the skills expected in students future careers.

Computational Thinking often strives to teach more than just the practical mechanics of programming and Computer Science. There are a set of cognitive techniques Debugging

5.2 Contexts within Computer Science

As part of the overarching goal to bring more students into Computer Science, a large number of contexts have been explored in Introductory computing. Starting with Seymour Papert's work with robotics and the LOGO programming environment in the 70s, instructors have tried to motivate students first experience.

There is a reciprocal relationship between contexts and content. When students learn programming in the context of game development, they are almost necessarily learning content related to game development that may not be universal to computer science – e.g., how graphical resources are organized and accessed within the game engine. This content may be seen as a distraction by the instructor, or as useful, side knowledge - for example, if a student had to learn how to use a command line in order to compile their game, they would be learning an authentic skill that might not be considered part of the core content, but is nonetheless generally useful. When evaluating a context, it is useful to consider what content it represents, and how authentic and useful it is. The authenticity of content that is attached to a context affects the authenticity of the learning environment as a whole.

Context is where most of the motivation is supposed to come into play, especially with regards to Empowerment, Interest, and Usefulness. Some contexts are entirely Interest-driven – game development, for instance, is often seen as an enjoyable experience for young children. A large number of curriculums use Alice and GreenFoot are both Java-based environments meant for game and animation development. Scratch uses its own event-driven programming language.

Although some of these environments support an authentic language, the content that they mediate is by no means authentic.

Interest-driven contexts

Game Development, Robotics, eTextiles GreenFoot Cowboy coders paper - what have they learned?

Media Computation students won't find it realistic to their experiences overly engaging and distracting (Seductive Details) the entire curriculum is built around this one concept Imagineering - http://dl.acm.org/citation.cfm?i

Simulations Hard to design for success NetLogo

Social Computing for Good, Data Science Authenticity Personalization/Choice - challenging Difficult to bring in real-world data quickly and efficiently

5.3 Facilitations within Computer Science

Programming environments are simultaneously tools to facilitate the learning process, learning environments where students spend a considerable amount of time, and

Introductory learning environments Programming Scratch, Alice Python, Java, Racket

Blockly - bridging the gap

Classroom experience Lecture Lab Online

5.4 Assessment within Computer Science

This is outside the scope.

6 Existing work

The foundation of this proposal 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]. Real-time data is another This real-time data offers similar advantages to working with Big Data – contextualized assignments and the experience of working with a novel technology. The 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.

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 **??** 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.

CORGIS Quickly leveraging real-world data Big Data definition challenges Real-time Web Solves High Velocity Challenges

Careful consideration must be made when choosing problems and designing contexts so that the data leads to optimally authentic learning experiences. One of the big dangers when attempting to create meaningful context for learners is the problem of *Preauthentication*: attempting to design for authenticity without sufficient knowledge of the audience. This is a problem shared by any approach to introductory material. Petraglia gives a compelling example [14]:

> 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 to a list of likes and dislikes, you run the risk of ignoring each individual learner's rich history and background that they will be building from. It is difficult to plan for and work against this ever-present danger when designing reusable assignments. Petraglia [14] 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 limiting, since it ignores the prior experiences and understanding that a student brings to their learning. Instead, it would be better to find a middle ground where students are given flexibility while maintaining a relatively uniform experience for students.

Eve Solves High Volume challenges High Volume challenges: Too large for the HD Inconsistent experience for students High Variety challenges: Too many fields Non-text/csv/json/xml data

7 Proposed Research

At this point, the largest job still ahead is data collectin to test the hypotheses.

Authentic Contextualization is important for learning Learning gains have gone up - Concept Inventory? Grades? Test Scores? Gather some data in CS-2 (to indicate transfer)? Authentic Contextualization is important for motivation Retention has increased Behavioral outcomes - retention increases, students report they do follow-ups (code academy, let's code blacksburg), students continue work in their free time, students start earlier Self-report outcomes - pre/post survey on attitudes of computing improve, do they feel motivated during the course (MUSIC model might require some modifications to distinguish WHERE the motivation is coming from) Our software will enhance this authentic contextualization Through Big/Real-Time, Situated Data

Best practices for data source organization.

Add more Sources Expanding to cover most majors Simulation sources Blockly integration Blocks for real-time data Python translation CompThink book Curriculum Units In addition to libraries/data sets, we also provide problems Set a clear path for ”cracking them open” - useful for CS-3 professors interested in making their students tackle things more in-depth

First semester Second semester Third semester Evaluation Plan Pre-survey Post-survey

8 Proposed Technology

9 Work Plan

First semester More data sources Computational Thinking course data collection Were students more engaged? Self-report (MUSIC model) Behavioral more time on task getting started earlier further projects integration into their own major work Did students learn more? (UNSURE) Improved assessments? Not really sure this can be proven empirically from the data without some kind of standard measure. Second semester More data sources Repeate measures for Computational Thinking CS-1114 (CS-1) Learning outcome changes Behavioral Increased retention More time on task Self-report (MUSIC Model) Did they want more game dev? More media computation? Less RealTimeWeb? Third semester Further data collection

10 Publication Plan

SIGCSE TOCE

11 Conclusion

Summarize the things we wish to prove Summarize how we’ll collect it Summarize why this is important

References

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

[2] A. C. Bart, E. Tilevich, S. Hall, T. Allevato, and C. A. Shaffer. Using real-time web data to enrich introductory computer science projects. In *Splash-E '13*, Oct 2013.

[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] M. Ben-Ari. Situated learning in computer science education. *Computer Science Education*, 14(2):85–100, 2004.

[5] J. S. Brown, A. Collins, and P. Duguid. Situated cognition and the culture of learning. *Educational researcher*, 18(1):32–42, 1989.

[6] J.-I. Choi and M. Hannafin. Situated cognition and learning environments: Roles, structures, and implications for design. *Educational Technology Research and Development*, 43(2):53–69, 1995.

[7] M. Guzdial and A. E. Tew. Imagineering inauthentic legitimate peripheral participation: An instructional design approach for motivating computing education. In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, pages 51–58, New York, NY, USA, 2006. ACM.

[8] 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.

[9] 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.

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

[11] 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. *McKinsey Global Institute*, 2011.

[12] O. of the Senior Vice President and Provost. Academic implementation strategy for a plan for a new horizon: Envisioning virginia tech 2013-2018. Technical report, Virginia Tech, 2013.

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

[14] 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.

[15] B. E. Rogoff and J. E. Lave. *Everyday cognition: Its development in social context.* Harvard University Press, 1984.

[16] J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.