Design and Preliminary Results from a Computational Thinking Course

Dennis Kafura
Virginia Tech
kafura@cs.vt.edu

Austin Cory Bart
Virginia Tech
acbart@vt.edu

Bushra Chowdhury
Virginia Tech
bushrac@vt.edu

ABSTRACT

This paper describes the design and initial assessment of a general education course in computational thinking for non-computer science majors. The key elements of the course include multidisciplinary cohorts to achieve learning across contexts, multiple languages/tools, including block-based and textual programming languages, repeated exposure to the underlying computational ideas in different forms, and student-defined projects using real world ("big") data to heighten motivation through self-directed contextualized learning. The preliminary multi-methods assessment shows that the course engendered high levels of motivation, achieved key objectives for learning in and across contexts, largely affirmed the choice of languages/tools, and supported, though less strongly than anticipated, the motivational effects of real-world data.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer Science education, Curriculum – *computational thinking*, *problem-based learning*, *engagement, group work.*

General Terms

Measurement, Design, Experimentation.

Keywords

Computational Thinking; Big Data; Student Engagement; Interdisciplinary Cohorts.

1. INTRODUCTION

Believing that a computational mode of thought is valuable to all members of society, our university recently included learning objectives for computational thinking in its general education requirements that must be met by all graduating students. Informed by a national report [1] , Wing's writing [2], and the Computer Science Teachers Association (CSTA) [3] among others, the university considered computational thinking to be:

The intellectual skills rooted in the ability to conceive of meaningful information-based representations that can be effectively manipulated using an automated agent (e.g., a computer).

In this paper we describe the innovative pedagogical approach and technology support for an Introduction to Computational Thinking course and present early results, both quantitative and qualitative, from a first offering of the course in Fall 2014. In particular, we focus in this paper on:

- innovative progressive course design in which core concepts are encountered several times in different contexts (Section 2),
- multidisciplinary "cohorts" to foster collaborative learning and learning across contexts (Section 3),
- "big data" to enhance the students' sense of realism and utility, not just interest (Section 4),
- technology support for cohort interaction and access to big data through block-based programming, all in an extended web-based ebook framework (Section 4), and
- preliminary multi-methods assessment of motivation and how helpful, useful, and interesting different elements were to students (Section 5).

We believe that the conclusions (Section 6) are valuable to others in computer science education, especially to those teaching computational thinking or introductory computer science courses.

The course we created both draws on and is distinct from other approaches that teach computational thinking, offer an introduction to computer science, or use big data. As a single course our approach differs from the inclusion of computational thinking modules into several required courses in a curriculum (e.g. general education[4], architecture [5], or the humanities [6]). As a general education course we differ from the inclusion of computational thinking into courses that target a specific discipline (e.g., sciences [7], computer science [8], humanities [9], and biology [10]). We draw on the ideas of other computational thinking or introductory computer science courses intended for all students. Some courses used teams (e.g., [11]), included social impacts (e.g.,[12]), or employed block-based programming (e.g., [13]). All of these elements are integrated in our course. We share the spirit of the "fluency" approach [14] but trade depth for breadth (e.g., we include only three of the ten "fluency" concepts.) We share with the media computation approach [15] the idea of providing a unifying, open-ended resource (images and sound in media computation vs. big data in our course). However, we believe that big data is seen by students as "useful" which is more engaging than media computation which is seen as "interesting" [16]. Our course shares with [17] the sense of engaging students with real world data. What we offer is that the data and questions to be answered are decided by the student and are not pre-determined assignments. Though we are using big data our goal is to teach computational thinking *using* big data as opposed to the goal of "data science" courses where big data is itself the object of study. We have in common with [18] the assessment of big data approaches though we also include quantitative methods and are focused on computational thinking rather than an introduction to computer science course. We also have a shared view with courses that used block-based programming (Snap!, Scratch, or App Inventor). What we add to a block-based programming approach is the connection to realistic big data sources, the ability to embed the programming in an ebook form to better integrate learning materials (see Section 4), and automated, guiding feedback through program analysis.

2. COURSE DESIGN

2.1 Learning Objectives and Dispositions

Four learning objectives for computational thinking were defined at our university. Students are required to:

1. Formulate problems and find solutions using computational or quantitative thinking in their field of study.
2. Give examples of the application to, and discuss the significance of, computational thinking in at least two different knowledge domains.
3. Apply computational methods to model and analyze complex or large scale phenomena.
4. Evaluate the social and political impact of computing and information technologies

In meeting these learning objectives the course design was also influenced by the CSTA's "dispositions or attitudes" that a computational thinker should exhibit [3]:

- "Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open ended problems
- The ability to communicate and work with others to achieve a common goal or solution."

Though defined for K-12 education, these dispositions seem equally relevant to university-level students. The first four dispositions influenced our use of student-defined big data projects where complexity, difficulty, ambiguity and open-endedness are present. The last disposition influenced our use of multi-disciplinary cohorts.

2.2 Content

The structure of the course is shown in Table 1. Though shown as a separate component the social impacts topic is woven throughout the course. A more complete description of this aspect of the course is beyond the scope of this paper.

The computational modeling topic uses NetLogo [19], a multi-agent development and simulation environment. Typical student work involves a student selecting a model relevant to their major from the library of pre-defined models in diverse areas (e.g., art, biology, sciences, games, mathematics, networks, social science, and system dynamics). Each student reads the description of the model and performs computational simulations by varying the model's parameters and observing the model's visualizations. Students in a cohort demonstrate and explain their models to each other. Finally, the cohort collectively and each student individually identify the properties for a model's abstraction and the programming constructs that manipulate these properties (via calculation, decision, and iteration). Through this work students begin to see the role of abstraction, the programming elements that determine the model's behavior, and the relevance of these computational techniques to many areas of study.

The fundamentals of algorithms topic uses a custom version of Blockly, a block-based programming language. Typical student work initially involves assembling specialized blocks to perform visually interesting computations (e.g., guiding an avatar through a maze using decisions and iteration). Subsequent classwork (in cohorts) and homework (individually) progressively involves the full Blockly environment and our custom "big data" blocks which connect students to realistic data streams. The current example data streams are in meteorology (weather forecasts), economics (stock market prices), geosciences (earthquake reports), and sociology (crime statistics). Initial algorithms constructed in Blockly use decision and iteration to calculate properties (e.g., averages) of data in simple lists while later algorithms involve more complex logic to filter and transform lists and data with more complex structure (e.g., the equivalent of Python lists and dictionaries). Blockly allow students to gain confidence in their ability to construct algorithm before having to cope with the syntactic detail of a textual programming language. An important aspect of Blockly is that the Python code for a Blockly algorithm can be rendered at the student's request. This makes the transition from Blockly to Python a more progressive step for learners.

Table 1: Course Topics

Topic (Length)	Description
Computational Modeling (4 weeks)	Model-based investigation of how complex global behavior arises from the interaction of many "agents", each operating according to local rules. Students use case-based reasoning and encounter basic computation constructs in a highly supportive simulation environment.
Fundamentals of Algorithms (2 weeks)	Study of the basic constructs of programming logic (sequence, decisions, and iteration) and program organization (procedures). A block-based programming language is used to avoid syntactic details. Students can see how these constructs are expressed in Python.
Data-intensive Inquiry (7 weeks)	Project-based exploration of complex phenomena by algorithmically manipulating large-scale data from real-world sources. Students construct algorithms in Python using a supportive framework for accessing the data.
Social Impacts (2 weeks)	Explore and discuss contemporary societal issues involving computing and information technology.

The data intensive inquiry topic introduced students to a carefully selected subset of Python. Initial student work involves cutting and pasting the Python code generated for previous Blockly exercises into a standard Python programming environment (e.g., IDLE or Spyder). This environment offers an important, authentic programming experience to offset any perceived penalty in usefulness that students perceive in Blockly. Students are initially encouraged and progressively discouraged to refer to the Python code automatically generated for algorithms written in Blockly. Generation of basic visualizations (line graphs, scatter plots, histograms) via a Python library is incorporated into the student work. Finally, students propose, complete and present a multi-week project that takes advantage of a big data source related to their major. Big data has become pervasive in almost all disciplines, so learning to work with it is an authentic, relevant experience for students that can be customized for each student while maintaining a common context in the class.

The organization of these three topics is progressive, meaning that each core concept is encountered in three different contexts. Students see the use of abstraction and algorithms (sequence, decision, iteration, functions) in the computational modeling topic by exploring the NetLogo programming of a model of their own choosing. They encounter these same elements again in the fundamental of algorithms topic where they modify and create algorithms in the context of a block-based programming language extended to access and manipulate big data. Finally, the same elements are seen a third time in the data-intensive inquiry topic where a major project is completed in a text-based programming language (Python).

Both Blockly and Python have been extended to use CORGIS (Collection of Real-time, Giant, Interesting, Situated) [20], a publicly available gallery of big data sources designed for educational use by novice students. The CORGIS project has many real-world datasets including geological sciences, history, psychology, social media, and many more. All of the CORGIS datasets are examples of big data – each having some aspect of high volume, high velocity, or high variation. The sense of what constitutes “big” must be interpreted, of course, in relation to the capability of the students involved. The CORGIS collection is an open-source project with tools to rapidly create new data sources for students.

3. PEDAGOGICAL APPROACH

The course work is organized to achieve a balance between context-based learning and learning across contexts. Context-based learning provides students with a motivating framework for their learning [21]. In particular, if computational thinking is embedded in the disciplinary material of a student’s major there is greater likelihood that the student will appreciate the relevance of computational thinking to their own needs and goals. In addition, the meaning of aspects of computational thinking may be more clearly learned in context because it is related to knowledge with which the student may already be familiar. However, there are equally important reasons to learn across contexts. The notion of transference refers to the ability to use in some context what has been learned in a different context. Transference is especially relevant to computational thinking because it is a generic skill that can be applied in many different situations. Learning across contexts enhances the ability of a student to recognize in new situations the applicability of computational thinking. Furthermore, by seeing computational thinking concepts in different contexts it is more likely that the student has gained a clear notion of these concepts. Also, the student may develop a deeper appreciation for different ways of knowing by appreciating how techniques relevant to the values and practices of their primary discipline are also relevant to the values and practices of other disciplines.

Context-based learning is achieved by allowing each student to specialize major aspects of their work to be relevant to their discipline or interests. As an individual, a student self-directs the selection of a computational model to explore and the selection, exploration, and completion of a project relevant to their major field of study. To support students’ self-direction we have used the CORGIS tools to quickly create new data sources for students who could not find a suitable library in the existing collection.

Learning across contexts is achieved by organizing students into interdisciplinary [22] cohorts that foster collaborative learning [23]. Each cohort contains students with 4 or 5 different majors. Students will perform all class room activities within these groups. Students also collaborate virtually using the course book, a custom-built, interactive web-based platform with embedded coding activities and real-time, shared text writing (similar to Google Docs). Collaborative learning also relates to each student’s role in their cohort [23]. As a member of the cohort, a student is responsible for:

Presentation: describing to the other cohort members the significance of the project they have selected.

Interaction: asking questions and providing feedback about the projects of other cohort members, thereby gaining insight into how computational techniques are used across disciplines.

Support: helping other members of the cohort with the mechanics of the tools and frameworks that are common across projects.

The assumption is that collaboratively learning computational thinking within interdisciplinary cohorts will foster “learning across contexts”. The expectation is that regular interaction with peers from different disciplines will provide students an opportunity to share and listen to others perspective. This will help students to form an understanding of how computational thinking applies in other disciplines.

4. TECHNOLOGICAL INNOVATION

A key technological element of this class is an open-source e-textbook platform named "Rhinestone", based on the popular Runestone project [24]. Rhinestone is a port of Runestone from the Web2py web framework to the Flask web framework, making it easier to extend Rhinestone with new features and directives to support the general education needs of our classroom.

Rhinestone’s biggest extension is automatic, continuous server-side storage of all student work as changes occur. This extension uses HTML5 local storage features to robustly backup data on the client in the case of disconnection and reducing the responsibility of students to manage their work. Another new feature is top-level support for collaboration. This extension allows members of a cohort to collaborate in the style of Google Docs. The underlying technology is Google MobWrite [25], a real-time communication library that provides differential synchronization between multiple users. Third, rather than being interleaved as in the Runestone model, the content of the book is divided into two dynamically linked sections – the readings (largely static content meant to be the definitive material of the course) and the class/home work (largely interactive problems and exercises that are completed by students).

Rhinestone also fully-integrates support for Blockly [26], a block-based visual programming language by Google that is based loosely on Scratch. The block-based nature of Blockly empowers students to focus on the semantics of their algorithm rather than its syntax. Moreover, Blockly blocks compile directly to JavaScript, making it runnable from the browser. These blocks can also be directly rendered as Python source code, which helps students transition from the high-level block-based programming to more authentic text-based coding. Our implementation of Blockly is more than just a coding environment - static program analysis and unit testing is used to deliver just-in-time, guided feedback that gives students contextualized assistance. An instructor identifies constraints for a question that are then matched with a hint – e.g., if students neglected to use an iteration block when processing a list of data, the environment links to the iteration chapter in the book.

With the support of CORGIS client libraries, the CORGIS blocks access rich data streams with apportioned complexity. For instance, particularly massive datasets can be sampled down for development purposes, switching to "full" mode on demand. Similarly, data sources that rely on an external, ever changing real-time source (such as weather, social media, or earthquake data) can be cached locally for access in an “offline mode”, avoiding problems with students’ internet connection and ensuring reproducibility during development and testing. When students have finished their development, the “online” access mode is used instead. Because students are encouraged to find their own dataset, they have a more personalized, engaged experience with their programming in comparison to a single set of instructor-provided data, even if that instructor-provided data is realistic.

5. PRELIMINARY RESULTS

5.1 Methods

The preliminary results from the first offering of the class in the Fall 2014 semester include an analysis of student motivation and the use of cohorts. Survey results are from 20 students in the class, 30% female and 70% male, representing an 80% respondent rate for the class as a whole. There was very little overlap between majors, with students in psychology, mechanical engineering, mathematics, theatre, university studies, and other disciplines. To assess the motivational impact of our pedagogical approaches and technological innovations, we surveyed students with the MUSIC Model of Academic Motivation Inventory (MMAMI). MMAMI is a validated instrument for measuring students’ beliefs related to the five key components of the MUSIC model [27]. The version used in our course consists of 26 statements that students responded to on a 6-point Likert scale (ranging from “Strong Disagreement” to “Strong Agreement”). The responses are then averaged into subscales relating to each of the components of the MUSIC model – eMpowerment, Usefulness, Success, Interest, and Caring. Examples of the statements include:

“The knowledge I gain in this course is important for my future.”

“I enjoy completing the coursework.”

Students were also surveyed on the different learning resources used in this course using a 4-point Likert scale. In particular, they were asked how helpful the resources were to their learning, their expectation of the usefulness of the resource to their long-term goals, and the interestingness of the resource. These questions asked about students’ experiences listening to lecture, reading the online textbook, getting help from the instructors in class, and working with their cohorts, NetLogo, Blockly, Python, and real-world data.

To better understand the quantitative data, qualitative data about the class was also collected by observing students working in cohorts during class time and by interviewing 9 students of the class at the end of semester. The following sections describe the findings of both quantitative and qualitative data of the study.

5.2 Analysis of Quantitative Data

As a baseline measure of success, the results from the MUSIC inventory suggest that students were overall motivated in this course. Students reported high average scores in all five areas of the MUSIC model, with no strong standard deviation. The results, shown in Figure 1, indicate that students “Agreed” in the belief that they were empowered, able to succeed, cared for, and that the course was interesting and useful. Our interpretation of this data is that, at a minimum, this course was successful in engaging students.

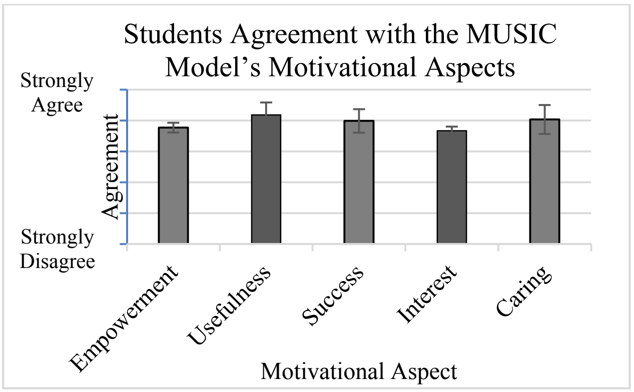

Figure 1: MUSIC Model Results

The follow-up surveys were useful in divining the sources of this engagement. The survey data is presented for helpfulness (Figure 2), usefulness (Figure 3), and interestingness (Figure 4).

The cohort model was a significant factor in motivation with students citing it as helping their learning (Figure 2) while also being both interesting (Figure 3) and useful (Figure 4) to their long-term goals. In fact, students’ cohorts were considered almost as useful as the assistance from instructors. Critically, no students thought that the cohort model was valueless, making the value of the collaborative learning experience very clear. Similarly, the negative results from the textbook and lectures, compared to the positive results from the cohort and instructors, reinforce the expanding literature on the value of active learning techniques compared to traditional lecture models.

In terms of the languages, Python was an unsurprisingly popular component of the course, with high positive results for most students. This matches recent literature on Python’s suitability for introductory programming experiences. A more interesting result is for Blockly – students perceived it as being useful to their learning (and moderately interesting), but recognized that it had little long-term usefulness. This matches with the use of Blockly as “training wheels” for Python, meant to be gracefully discarded as the students gain familiarity with algorithmic concepts and are prepared to cope with Python’s syntax.

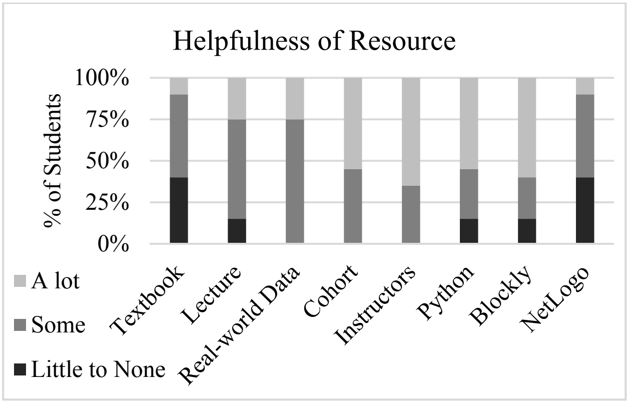

Figure 2: Survey Results on Helpfulness

The results related to NetLogo are more ambiguous – although few students found it very uninteresting, few reported it as very interesting or very useful to their long-term goals. There are positives and negatives to the use of NetLogo within this course that we are still exploring.

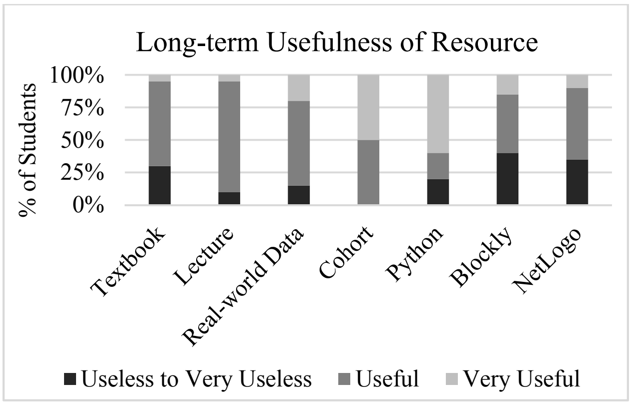

Figure 3: Survey Results on Usefulness

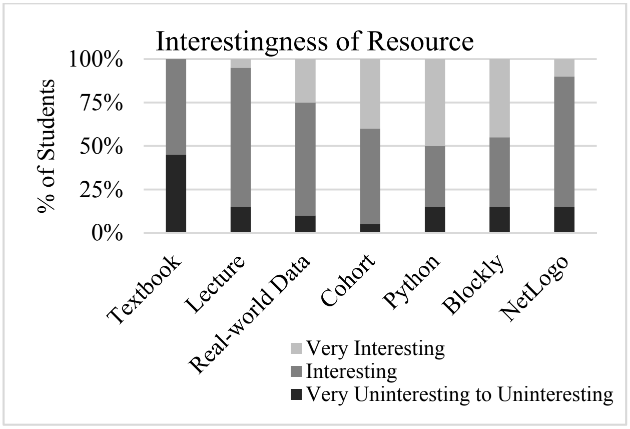

Figure 4: Survey Results on Interestingness

5.2 Analysis of Qualitative Data

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

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

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

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

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

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

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

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

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

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

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

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

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

6. CONCLUSIONS

This paper has outlined the design of a general education university course in computational thinking. The quantitative and qualitative assessments provide early evidence supporting key course design decisions and the achievement of important learning objectives. First, the course engendered a high level of student motivation and engagement. We see this as an especially critical finding for a general education course. Second, the multidisciplinary cohorts were seen as helpful to learning and useful to student's long-term goals. Importantly, the qualitative data indicates that the cohorts fostered learning across disciplines, a key learning objective. Third, the preliminary analysis of the qualitative data also indicates that students made gains on another key learning objective - a deeper appreciation for the use of computation in their own disciplines. Fourth, the use of Blockly and Python were largely supported by the assessment while the use of NetLogo is more ambiguous and requires further study. We had expected the assessment to more prominently support the use of real-world data. While helpful overall the response was weaker than for the multidisciplinary cohorts and instructors. We believe that the small negative view of the real-world data on measures of interestingness and usefulness may be due to the repeated use of overly simplified big data on exercises and assignments. We have plans to introduce more varied and more successively realistic examples of big data in the next offering of the course and to observe the effect of this change.

7. ACKNOWLEDGMENTS

We thank Jason Riddle and Omar Saleem for preparing big data resources and Shelli Fowler for use of the TLOS Learning Studio as our classroom. This work was supported in part by NSF Grant TUES-1444094.

8. REFERENCES

[1] National Research Council, Report of a Workshop on the Scope and Nature of Computational Thinking: National Academy Press, 2010.

[2] J. M. Wing, "Computational thinking," *Communications of the ACM*, vol. 49, pp. 33-35, 2006.

[3] International Society for Technology Education. 2011, Computational Thinking Teacher Resources (Second Edition ed.). Available: http://www.iste.org/docs/ct-documents/ct-teacher-resources_2ed-pdf.pdf?sfvrsn=2

[4] L. Perkovik, et al., "A framework for computational thinking across the curriculum," Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education, Bilkent, Ankara, Turkey, 2010.

[5] N. Senske, "A Curriculum for Integrating Computational Thinking," presented at the ACADIA Regional Conference 2011 Lincoln, Nebraska, 2011.

[6] C. Kuster, et al., "Developing Computational Thinking Skills across the Undergraduate Curriculum," presented at the 44th Annual Midwest Instruction and Computing Symposium (MICS'11), Duluth, MN, 2011.

[7] S. Hambrusch, et al., "A multidisciplinary approach towards computational thinking for science majors," Proceedings of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, TN, USA, 2009.

[8] D. Kafura and D. Tatar, "Initial experience with a computational thinking course for computer science students," Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, Dallas, TX, USA, 2011.

[9] A. Ritz, "Evolution of a Computational Thinking Course," Presentation to CS6604 Class, Ed., ed, 2013.

[10] H. Qin, "Teaching computational thinking through bioinformatics to biology students," Proceedings of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, TN, USA, 2009.

[11] W. Booth, et al. (2013, Computational Thinking: Building a Model Curriculum. 11pp. Available: https://ciiwiki.ecs.baylor.edu/index.php/Computational_Thinking:_Building_a_Model_Curriculum.

[12] T. J. Cortina, "An introduction to computer science for non-majors using principles of computation," Proceedings of the 38th ACM Technical Symposium on Computer Science Education, Covington, Kentucky, USA, 2007.

[13] T. Li and T. Wang, "A Unified Approach to Teach Computational Thinking for First Year Non–CS Majors in an Introductory Course," IERI Procedia, vol. 2, pp. 498-503, 2012.

[14] L. Snyder, Fluency with Information Technology, 6th ed., Addison-Wesley, 2014.

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

[16] M. Guzdial and A. E. Tew, "Imagineering inauthentic legitimate peripheral participation: an instructional design approach for motivating computing education," Proceedings of the Second International Workshop on Computing Education Research, Canterbury, United Kingdom, 2006.

[17] R. E. Anderson, et al., "Introductory programming meets the real world: using real problems and data in CS1," Proceedings of the 45th ACM Technical Symposium on Computer Science Education, Atlanta, Georgia, USA, 2014.

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

[19] U. Wilensky, "Modeling Nature's Emergent Patterns with Multi-Agent Languages," in Center for Connected Learning and Computer-Based Modeling, ed. Northwestern University: Available at: http://ccl.northwestern.edu/papers/2013/mnep9.pdf, 2013.

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

[21] D. I. Cordova and M. R. Lepper, "Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice," Journal of Educational Psychology, vol. 88, pp. 715-730, 1996.

[22] L. R. Lattuca, et al., "Does interdisciplinarity promote learning? Theoretical support and researchable questions," The Review of Higher Education, vol. 28, pp. 23-48, 2004.

[23] P. Dillenbourg, "What do you mean by collaborative learning?," Collaborative-learning: Cognitive and Computational Approaches., pp. 1-19, 1999.

[24] B. Miller and D. Ranum, "Runestone interactive: tools for creating interactive course materials," Proceedings of the First ACM Conference on Learning @ Scale, Atlanta, Georgia, USA, 2014.

[25] N. Fraser, "Differential synchronization," Proceedings of the 9th ACM Symposium on Document Engineering, Munich, Germany, 2009.

[26] Blockly Website. (2014. Available: https://developers.google.com/blockly/

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