A Framework for Designing Contextualized Computing Curriculum

Ursula Wolz
RiverSound Solutions/Lang College
Montclair, NJ/New York, NY
ursula.wolz@gmail.com

Gail Carmichael
Shopify
Ottawa, Canada
gail.carmichael@shopify.com

Anne Marie Webber
Goffstown High School
Goffstown, NH
annemarie.webber@sau19.org

***Abstract*—Contextualized computing is a two-decades-old approach to providing foundations in computer science. The intent has always been to promote a diversity of approaches and to cast a wide net through which to bring under-represented groups into the field. Much of the curriculum design is ad hoc. This experience paper presents a disciplined model for curriculum design that provides a structure and methodology for integrating computing with another discipline. A framework for analysis is presented that abstracts the curriculum development process. It is grounded in 50 years of scholarship on instructional design. It extends established practice to concurrently develop goals, objectives, activities, and assessments across at least two domains. The methodology is applied in two non-traditional domains: an industry-based work-integrated learning program at a 'software as a service' company, and in an alternative high school mathematics class for students at risk. This approach is unique in its emphasis on promoting a principled framework for curriculum design. It extends traditional instructional design by applying methodologies of agile development to identify, execute, and assess instructional computing.**

***Keywords*—contextualized computing, interdisciplinary computing, computing curriculum design, high school computer science**

I. A Framework for Contextualized Computing

The broadening participation movement has fostered a variety of approaches to teaching computer science. Contextualized computing, described by Xu et. al in 2008 [1], is an approach that directly supports diversity.

Context may be brought into a computing-specific course, or computing may be integrated into another context altogether. An early example of contextualized computing was an effort to teach programming to digital artists [2]. Many other examples have been reported (such as [3], [4], [5], [6], [7]), typically with the intention of extending the reach of interdisciplinary computing [8], [9] or attracting a more diverse group of people to the field. Contextualized computing is a promising solution for the lack of quality computing instruction available [10]; indeed, as coding appears increasingly in high school graduation requirements, contextualized computing may be the only practical way to insert computer science into a curriculum without removing something else.

978-1-5386-5541-2/18/$31.00

A problem arises when courses are taught outside the well-established computer science community. This report describes a framework used in two contexts: high school alternative math, and industrial training. The framework provides a structure through which curriculum designers can attend to the intersections of the identified computing concepts and learning goals of the context.

II. A Concurrent Curriculum Design Framework

We present a framework to guide instructional design for contextualized computing curriculum. It can be used to integrate computing concepts with a targeted context to provide transparency in goals, expectations, and outcomes for both the contextual area and computing.

The framework is visualized via the diagram in Figure 1. The red circle (labeled A) encompasses computing concepts and skills in a single area of computer science, such as software engineering or data structures. The two greens circles represent skills and technologies from the contextual domain we want to integrate with. The larger green circle (labeled C) represents skills and technologies that are used more broadly over time. For example, in a work-integrated learning setting, it might capture all the skills that employees across a company use. In a high school setting it might address identified graduation requirements in the contextual discipline. The smaller dark green circle (labeled B) is a subset of the light green circle. It represents the skills that are required immediately in the contextual discipline. In the work-integrated learning setting, the dark green circle might be the skills that a student needs to have to succeed on a team they are joining imminently. In a high school course it is identified by state or national standards. The largest blue circle (labeled D) encompasses the broader skills, knowledge, and attitudes of computing. Not all of these are necessarily directly applicable to the specific context we are integrating with, but they contribute to breadth in the field of computing.

There are intersections between the area of computer science in the red circle and the context-specific skills (labelled 1), the context-specific skills used broadly over time (labelled 2), and the broader skills, knowledge, and attitudes of computing (labelled 3). The best opportunities to integrate computer science skills with the desired context exist in the first two intersections (labelled 1 and 2), while the third intersection

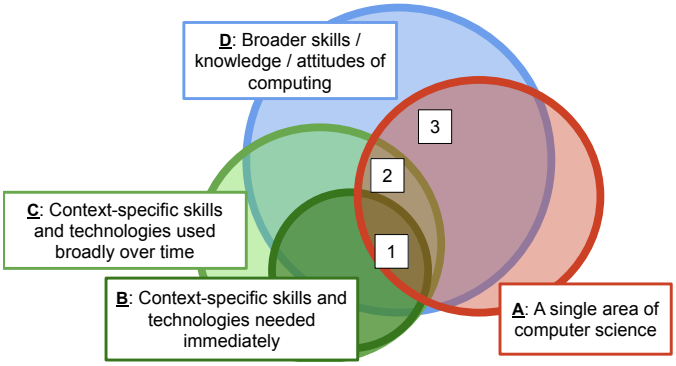

Fig. 1. Concurrent curriculum design framework.

(labelled 3) provides an opportunity to strengthen abilities in computing and indirectly contribute to success.

A. Procedure for Curriculum Development

Contextualized computing requires a dual design process in that instructional goals for both computing and the contextual subject area must be developed. As early as 1949 Tyler [11] proposed a framework for developing large curriculum projects by addressing the following questions:

1) What is the educational purpose?
2) What experiences are available to meet that purpose?
3) In what manner can these experiences be organized?
4) How can it be determined that the purposes were attained?

Instructional Systems is the field of study that provides theory and practice for instructional design, typically of a single content area. One well-recognized design procedure is that of Gagne, Briggs and Wanger [12]. A process model developed by Dick & Caray [13] defines stages of analysis from instructional goals to student outcomes. Within the contextualized computing framework this requires addressing instructional goals, learning objectives, student activities, student assessment (such as testing), and instructional assessment (evaluation of the course).

B. An Iterative Concurrent Process

Starting with the traditional process model, our framework supports an agile iterative cycle. The initial task is to define a *desired context*. In Section III we report on three: natural language, development at a particular software company, and high school alternative mathematics. Defining the context is followed by identifying the *area of computer science* we wish to integrate, which may be (or correlate to) an academic course.

Integration informs the five steps enumerated above as they are applied to the diagram in Figure 1. Defining the instructional goals of a learning experience means deciding broadly what our learners should be able to do by the end of the experience. At this stage we can identify concepts from the area of computer science that relate to the desired context skills (intersections 1 and 2 in the diagram). We can also look at the broader skills in computer science that relate to the course and would benefit performance in the context indirectly (intersection 3). Then we can consider ways to connect the concepts from the area of computer science with the practice of the desired context, or vice versa. The same considerations apply when breaking the instructional goals down into more detailed, specific learning objectives.

When designing student activities, we have an opportunity to look even more closely at the overlaps between the area of computer science and the desired context. A simple approach might be to re-skin a traditional activity found in an academic course with surface features from the desired context. Or we might design an activity to support learners in making explicit connections between the skills and knowledge from the area of computer science and those used in the context. For even deeper integration, we can have our learners solve a problem in the context using the course's skills and knowledge. We also have an opportunity to have our learners perform similarly to how participants in the integrated context would; for example, we might ask learners to pair program in a learning experience the same way they would on a team in an industry setting.

III. Examples of Designing for Two Different Contexts

We now illustrate examples of applying the framework using two dramatically different contexts: an industry-based work-integrated learning course connecting systems programming to the workplace context, and a high school alternative math class in which the final exam was a collaborative programming project.

We emphasize that this is an experience report rather than a formal quantitative research study on the efficacy of our methodology. We are reporting this at RESPECT as a vehicle for inviting others into using our approach to further enhance and codify the framework.

A. Systems Programming in an Industrial, Work-Integrated Learning Context

Shopify, a leading global commerce company, partners with two local universities to offer a curriculum-aligned work-integrated learning program [14]. The program aims to close the gap between what is learned in a computing degree and the skills required of an industry software developer. Students participate in learning activities at Shopify and rotate through a number of team placements throughout their computer science degrees. One author led the design and launch of the first version of the program.

We ran an experimental course designed to connect a campus-based academic systems programming course with the context of software development practice at Shopify. In this example, the 'red circle' of Figure 1 represents the systems programming course, while the 'green circles' represent the particular skills used on teams in the company.

1) Instructional Goal : The instructional goals were to: (1) enhance the learning occurring at the partner university's systems programming course with relevant practical skills;

(2) illustrate how the concepts taught in the academic course applied to the work done at the company; (3) Provide a less structured learning experience where students could help each other to research more open-ended tasks that instructors don't generally have the 'answers' for.

The goals land in the intersections labeled 2 and 3 in Figure 1 in that we targeted skills and concepts from the academic course that were relevant to the tools and languages used for software development at the company, but not necessarily those that students would need immediately during their next team placements.

2) Learning Objectives: The course's learning objectives were as follows:

1) Use the same operating system as the on-campus course in a new, practical context.
2) Use a command-line interface in an applied setting.
3) Practice writing C code for input/output on a real device.
4) Work with hardware designed within the company.
5) Investigate how memory management works in a high-level programming language used often in the company.

3) Student Activities: Students met for one hour per week at Shopify during the same semester they took the on-campus systems programming course. At Shopify, they were given open-ended problems centered on working with Raspberry Pis, interacting with hardware designed in-house at the industry host, and researching the Ruby interpreter source code. They were asked to acquire and write simple programs in C for peripherals for the Pis, participate in a workshop exploring how to work with the Bluetooth interface of the in-house hardware, and write a report on what they learned about memory management in the Ruby interpreter.

4) Student Assessment: Assessment for this course was informal: students were expected to attend each weekly hour, and to complete their report in a timely manner. They were excused for good cause. Unsatisfactory participate was noted in their next performance review.

5) Instructional Assessment: Instructional assessment was determined by biweekly student discussions with formally assigned program mentors as well as an anonymous feedback form at the end of the semester. We also received detailed and thoughtful suggestions directly from some students. This offering was one of the first in the program.

Results of the experimental course were mixed. Observations showed that students met all the desired objectives. However, they spent too long working with the Pis and peripherals, causing them to spend more time in the intersection labeled 3 in Figure 1 than was ideal. Spending more time on skills directly applicable to their next team placements (intersection 1) and connecting more directly to skills that other teams use (intersection 2) would have improved student satisfaction.

B. The Alternative Mathematics Classroom

We used our framework to design curriculum for an alternative mathematics class for at-risk high school students. These students will not qualify to enroll in computing courses, and our challenge was to integrate foundational programming skills into their mathematics experience. The school supported a radical projects experience in which students built a physical museum exhibit. The alternative math class contributed a physical quilt.

Turtlestitch (turtlestitch.org), a version of Snap! that emphasizes Turtle Geometry, provided the context for introducing computing. In our framework, the 'red circle' (labelled A) course content is coding in Snap!, the small context 'green circle' (labeled B) is geometry and algebra skills, the large context 'green circle' (labeled C) is common core mathematical practice that articulated directly with 'blue circle' (labeled D) big computing.

1) Instructional Goal : The instructional goal was to engage the alternative math student in mathematics and coding to reinforce that formal math was useful, and more to the point, that they could master it. The secondary goal was to empower them as coders and provide basic skills in information technology. The practical goal was to create a collaborative quilt. The completed quilt was their 'final exam.'

2) Learning Objectives: Learning objectives came from three sources: (1) Common Core mathematical practice, (2) selected Common Core high school standards in algebra and geometry, (3) CSTA CS Standards. Objectives included (1) Algebra HSA.Q.A.1 Reason Quantitatively and Use Units, (2) HS Geometry G-CO:12 Construct Geometric Figures, and (3) G-MG:1&3 Apply geometric concepts in modeling situations. Because this a math class, CSTA standards were mapped to the primary Common Core math practice standards.

3) Student Activities: The learning module was called 'Problem Solving Unit – Operation Quilt.' Students were shown basic turtle stitches (how to make a square). Algorithms were introduced as Snap! scripts. Discussions occurred that mixed geometry and coding, such as what the term 'algorithm' meant in computing and mathematics. Geometry definitions in relation to turning through a circle (e.g. core trigonometry) were introduced by having students use turtle movements to draw standard polygons, construct a circle (using a control structure), understand how variables can impact scaling, and how to construct re-usable components by defining their own blocks. Students were asked to discover on their own or as a group how to draw any polygon, and how to approximate a circle. They were challenged to identify the parameters for unique design. Once students had mastered essential geometry and coding skills, each designed and implemented at least two images for a 'seasons' themed quilt shown in Figure 2.

4) Student Assessment: Assessment was tied directly to the production of blocks on the quilt. They were also required to write short reflective essays on the experience. Students were hesitant at first, thinking they were bad at math, and since Turtlestitch was being taught in a math class they must be bad at it, too. When they learned that giving up wasn't an option, they persevered through all of the coding they were tasked with, and then some. Everyone passed.

Students rose to this task and accomplished it with excellence. They excelled at helping each other with their coding, troubleshooting, and embroidering. When setbacks occurred,

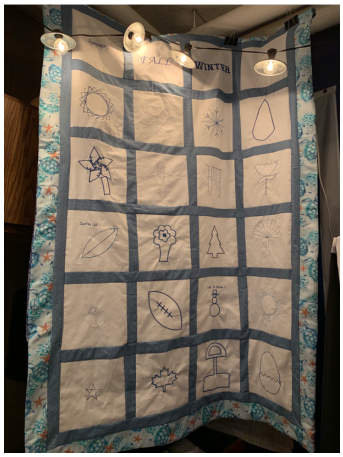

Fig. 2. Final exam quilt from the alternative math class

such as designs that were too large, they eagerly embraced and adopted advanced concepts such as the use of variables to support scaling.

5) Instructional Assessment: The careful consideration for the intersection of math and coding skills allowed us to meet both the formal objectives of the required standards as well as the subtext of producing math- and tech-confident individuals. Given the opportunity to reflect on the experience, they articulated that problem solving is hard, and often frustrating, but that persevering to a tangible outcome is satisfying. The school administration was satisfied as well: the students all met their math requirement for graduation.

IV. Recommendations

Based on our experience, we recommend the following steps when applying our framework, with the most important considerations appearing first:

1) Identify the computing domain and context.
2) Integrate the specific skills students need now in the desired context (intersection 1 in the diagram) first.
3) Integrate with broader skills within the specific context but that students don't necessarily need immediately (intersection 2) next.
4) Integrate with the even broader set of skills, knowledge, and attitudes that contribute to being a stronger computer scientist (intersection 3) last.

Ensure the course and the context do not feel like separate, disconnected tracks. When possible, allow students to experience the practical implications of theory within the context before learning about the theory. Reduce the amount of time between experiencing the practice in the context and learning about the theory behind the practice. Apply the following rules iteratively:

- If a requirement in a course can be fulfilled by an experience in the context, avoid making students cover the requirement twice, instead focusing on connections to the theory or giving credit toward the course.
- Support students in actively reflecting on connections between the academic computing material and the context.
- Consider which 'direction' would work best: are you building a computer science course that integrates context, or are you integrating computing into an academic or industrial learning context?
- Consider whether you are designing a new course, adapting an existing course (such as Alternative Math), or providing a dual learning experience (such as the industrial experience).

V. Summary and Future Work

A framework for contextualized computer science curriculum is dependent upon addressing the natural intersections between the computing learning goals and the contextual learning goals. The framework provided here, based on sound principles of instructional design provides a framework for transparency in course development. We invite you to join us in this enterprise by visiting the (redacted) website of one of our authors.

References

[1] D. Xu, D. Blank, and D. Kumar, "Games, robots and robot games: Complementary contexts for introductory computing education," in *Proceedings of Third International Conference on Game Development in Computer Science Education (GDCSE'08)*, 2008.

[2] J. Maeda, *Design by numbers*. MIT Press, 1999.

[3] J. D. Bayliss and S. Strout, "Games as a "flavor" of cs1," in *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '06, (New York, NY, USA), pp. 500–504, ACM, 2006.

[4] R. E. Beck, J. Burg, J. M. Heines, and B. Manaris, "Computing and music: A spectrum of sound," in *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, (New York, NY, USA), pp. 7–8, ACM, 2011.

[5] M. Guzdial, *Introduction to computing and programming with Python: A Multimedia Approach*. Prentice-Hall, 2004.

[6] U. Wolz, C. Ault, and T. M. Nakra, "Teaching game design through cross-disciplinary content and individualized student deliverables," *Journal of Game Development*, vol. 2, no. 2, 2007.

[7] D. Xu, U. Wolz, D. Kumar, and I. Greenburg, "Updating introductory computer science with creative computation," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, (New York, NY, USA), pp. 167–172, ACM, 2018.

[8] L. N. Cassel and U. Wolz, "Interdisciplinary computing, successes and challenges (abstract only)," in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, (New York, NY, USA), pp. 738–738, ACM, 2013.

[9] U. Wolz and L. B. Cassel, "The role of interdisciplinary computing in higher education, research and industry," in *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, (New York, NY, USA), pp. 7–8, ACM, 2012.

[10] E. S. Roberts, "Meeting the challenges of rising enrollments," *ACM Inroads*, vol. 2, pp. 4–6, Aug. 2011.

[11] R. Tyler, *Basic Principles of Curriculum and Instruction*. University of Chicago Press, 1949.

[12] R. M. Gagne, L. J. Briggs, and W. W. Wager, *Principles of Instructional Design*. Harcourt Brace College, 1992.

[13] W. Dick and L. Carey, *The Systematic Design of Instruction*. Scott Foresman & Co, 1990.

[14] G. Carmichael, C. Jordan, A. Ross, and A. Evans Adnani, "Curriculum-aligned work-integrated learning: A new kind of industry-academic degree partnership," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, (New York, NY, USA), pp. 586–591, ACM, 2018.