A Strategy for Teaching "Real" Computer Science through Wearables

Michael Eisenberg\textsuperscript{1}, Swamy Ananthanarayan\textsuperscript{1}, and Katie Siek\textsuperscript{2}
\textsuperscript{1}Department of Computer Science, University of Colorado, Boulder CO USA
\textsuperscript{2}Department of Computer Science, Indiana University, Bloomington, IN USA

Abstract - The past two decades have seen a remarkable growth in the versatility, affordability, and aesthetic range of computationally-enhanced textiles and wearable devices. This burgeoning interest in wearables offers significant potential for computer science education: programming one's clothing and accessories might have obvious motivational appeal for students. On the other hand, there are intellectual obstacles to the project of founding computer science education on the use of wearable devices: the actual programming required for most projects is simply too restricted. What is needed, then, is a strategic framework for incorporating e-textiles and wearables into a more varied, challenging, and content-rich view of computer science education. This paper describes such a framework for fungible programming that envisions a hybrid of wearable and desktop/tangible computing as a basis for a "real" computer science curriculum. We illustrate this idea with examples from an ongoing educational project in which children employ programmable wearable devices to monitor their outdoor activity.

Keywords: Fungible programming, e-textiles, programmable wearables, computer science education

1 Introduction: The Problem of Wearables in CS Education

Over the past two decades, a remarkable subculture of computer science has emerged, focusing on the integration of accessible, programmable computational elements into clothing and other textile-based artifacts. Hobbyists, crafters, and students make use of such devices as the LilyPad Arduino \cite{2,3,5} and its offshoots (such as the Flora \cite{7}) to incorporate dynamic behavior into hats, jackets, T-shirts, and a myriad other wearable artifacts \cite{4,8,10}. A perusal of websites such as YouTube or Instructables will serve to demonstrate the variety of creative projects incorporating these devices, and will likewise illustrate the enthusiasm with which young people in particular have taken to the idea of "wearable programming".

The advent of this subculture has been enabled by several concurrent technological developments: the availability of lightweight embedded computing devices (along with their associated programming platforms), the wide variety of sensors and actuators compatible with these devices, and (not to be overlooked) the arrival of materials such as conductive threads, tapes, and inks that enable users to connect together these various elements in creative craft projects. One might (for instance) create a shirt that lights up in various colors in response to sound, or a glove that signals the wearer in the presence of infrared light, or a knitted scarf that emits tones in response to a change in temperature; indeed, such projects are just the sorts of things that appear in numerous incarnations on hobbyist websites.

Projects of this sort hold powerful motivational appeal for computer science students—and, by implication, for computer science educators. Rather than working on abstract, "make-work" problems—writing an algorithm that sorts lists of numbers in ascending order, or searches for a string in a body of text—one might imagine introducing computer science and programming through a task such as making an interactive Halloween costume, sports uniform, ballet dress, or cosplay outfit. Programming one's own clothing taps into a cultural tradition of fun, creative display that has been associated with youth culture for decades. While a student might regard typical computer science assignments as "schoolwork", the challenge of making a programmable theatrical costume might be perceived as having a personal, idiosyncratic, irreplaceable purpose.

From the standpoint of the computer science educator, then, we should be vigorously exploring opportunities for incorporating e-textiles and wearables into computer science education, if for no other reason than to encourage students to (willingly) do their homework. Yet at the same time, there are significant obstacles to realizing this educational potential. Most pointedly, the difficulty is that there are profound intellectual gaps between the sort of programming issues raised by e-textile programming. A computer science educator wishes to introduce students to strategies for programming at the large, or at least medium scale—ideas such as modules, problem decomposition, object-oriented programming, debugging, and so forth. These are strategies that surface when students attempt programming projects of at least moderate size.

Few e-textile projects achieve this sort of size or complexity. It is fair to say that the vast majority of such projects have a relatively simple structure—something along the lines of "wait for a trigger sensor value, then turn on an
The e-textile projects mentioned earlier all have this basic structure: one might wait for a temperature value above a certain threshold value, at which point the program sends a signal to turn on an LED light. From the CS educator's standpoint, a project of this sort might have value as an introduction to language syntax (e.g., conditionals), but little else.

What is needed, then, is a strategy for exploiting the motivational and aesthetic potential of e-textile and wearable computing while, at the same time, leading toward projects of greater complexity and challenge. This paper describes such a strategy, which we call fungible programming, and illustrates it with (still relatively early) examples derived from an ongoing project centering on children's fitness and health. The basic idea behind fungible programming is to create "hybrid projects" in which wearables are treated not, or not only, as reactive display items, but as mobile sensors and actuators loosely linked to larger projects in desktop or Web programming. The term "fungible" here is used to highlight the idea that the programs designed for wearable devices create and employ data that may be freely exchanged, for a variety of purposes, with other programs and devices. Rather than viewing wearable devices as the sites of (tiny) standalone programs, as is true of most projects, we can view them as the mobile "outer skin" of complex software systems, as suggested by the diagram in Figure 1.

The remainder of this paper is organized as follows: in the second section, we introduce several working examples that suggest (though they hardly exhaust) the ideas behind fungible programming. We also explore the application of these ideas to educational programming by describing recent work in combining wearables and a group project conducted in a local after-school computer clubhouse. The third section goes beyond these initial examples by outlining (perhaps mildly futuristic) scenarios for a fuller computer science curriculum based on e-textiles and fungible programming. Here, we address conceptual issues that effectively extend the traditional portrait of the by now "classic" Von Neumann architecture for computers. In the final section, we discuss some of the opportunities for evolving computer science education in a more craft-based direction, and some of the obstacles that must be overcome in this evolution.

2 Fungible Programming: Some Initial Prototypes

In this section we concretize the framework described in the previous section by showing several working prototypes of the notion of fungible programming. For our examples, the wearable component consists of three modules: a microcontroller base, a sensor module for ultraviolet light, and a button that the user may employ for manual signals. The base is a custom printed circuit board of our own design, with our own surface mount components, designed to be lightweight and to lay flat against various (presumably textile) surfaces; the microcontroller may be programmed through the (by now fairly "standard") Arduino programming environment, which is both freely available and well-supported by a large community of users.

Figure 1: A basic framework for "fungible programming" of wearable devices. In typical projects with wearables, the "main" desktop computer endows the wearable devices ("W") with programs that work autonomously. In fungible computing, the wearable devices interchange data with the desktop computer or "main" project. The result is that the wearables are "extensions" into the world of the larger project.

Figure 2: Clockwise from upper left: wearable base with coin cell battery and RF wireless module; UV sensor; 3D printed case to house the device; wearable button.

1 A description of an earlier version of this wearable device can be found in [1].
The basic idea behind this particular wearable design is grounded in our own work in allowing youngsters to track their outdoor activity. The UV sensor is, in effect, a means of keeping track of the time that the wearer spends outdoors. The microcontroller base records the passage of time by using a watch crystal and the microcontroller's real time counter; when connected to the UV sensor it periodically timestamps events and stores the timing data in memory. In this way, the device may be regarded as a "monitor for time spent in sunlight." Because of this particular device's low weight and light power requirements, it may be run on a single coin cell battery for weeks. Figure 2 shows the various components of the system, including a plastic case that we designed and printed on a 3D printer.

It should be noted that the design choices for our own wearable system are motivated by a specific, health-related project; but the larger discussion of fungible programming in this paper does not depend on any one design for wearables. Rather, as we will discuss later, the idea of fungible programming can be tested and explored with an endless variety of specific choices and designs for wearable components. Our own initial experiments, however, are based on the wearable system shown in Figure 2.

Figure 4: Several working examples of the "non-wearable", stationary elements of a hybrid project approach. At top, a windmill kinetic artwork in which the rate of the turning mills can be controlled from data provided by the wearable. At center, a cherry blossom painting in which the number of blossoms that light up with LEDs is correlated with data from the wearable device. At bottom, a terra cotta fountain whose flow rate is controlled by data from the wearable.
The wearable device shown in Figure 2 may now be incorporated into clothing or accessories of the sort shown in Figure 3; a headband or messenger bag could now be used to record the amount of time spent outdoors by the user. Indeed, the sorts of artifacts in Figure 3 represent the typical sum total of most published wearable projects; in a standard scenario, the wearable device might also be connected to an LED light that illuminates when (e.g.) the UV level, or time spent outside, exceeds a given threshold.

The fungible programming approach now uses artifacts such as those shown in Figure 3 as the data-gathering "input devices" for more elaborate projects of the sort shown in Figure 4. For these projects, the basic idea is that the amount of time spent outdoors can be employed as a sort of "virtual currency" for powering computationally-enhanced craft items created at home. Once the user returns home, she sends the data from the wearable device via a wireless radio frequency signal (RF) to a microcontroller (or multiple controllers) embedded in the craft object itself; the home object may then respond in specific ways to the data conveyed by the wearable device. In the examples shown in Figure 4, a windmill portrait can turn faster in response to more time spent outdoors; a cherry blossom painting will respond with red LED lights (and a chirping bird) to data from the wearable; and a terra cotta fountain will increase its flow rate in response to more time spent outdoors.

A detailed discussion of the implementation of the Figure 4 objects is tangential to the main point of these examples (a thorough discussion of the cherry blossom painting may be found in [1]). For the purposes of this discussion, the larger issue is simply that these projects illustrate a hybrid of typical wearable projects (illustrated by Figure 3) and more complex computational projects for the home (illustrated by Figure 4). There is an obvious combinatorial advantage to this sort of approach: for example, either of the two wearables in Figure 3 could be combined with any of the craft projects in Figure 4, so that even in these very initial illustrations we have effectively shown six potential projects. More broadly, while our own laboratory’s interests focus on craft projects (of the form shown in Figure 4) as the representative of "complex, stationary" systems, there is no reason not to expand the scope of this vision to (e.g.) a project run on a desktop computer, or on a Web server, or (more generally) as a cloud-based system. In section 3, we will provide potential examples of this broader vision of fungible programming.

2.1 Student-Built Prototypes

A natural objection (or at least, source of skepticism) to the vision sketched above is that the fungible programming approach might prove too complex, or too demanding of attention span, to justify its use as the basis of a computer science curriculum. While this is an empirical question to be explored in practice, our own preliminary experiences with middle-school-age students gives us optimism that in fact the notion of fungible programming could prove feasible as a foundation for computer science education, even for relatively young learners.

During the past year, we worked with nine students between the ages of 11 and 14 over a period of several months at a local after-school enrichment program (focused on arts and technology). While a full description of our study is beyond the scope of this paper (and is the focus of another paper, in preparation, largely concerning the results of the students’ work), the purpose of mentioning this work here is merely to illustrate the plausibility of fungible programming for younger students. In our after-school group, students created their own wearable artifacts incorporating the (newer) system shown in Figure 2 earlier. Figure 5 shows two representative examples of this phase of the students’ work: a pencil pouch with a flower design, and a sports-themed book cover.
Figure 6: Electronic mural created by middle school children for combination with the wearable modules in Figure 5. Each individual segment of the mural reacts to data from its particular owner’s wearable device. (and sent via wireless RF signal). To take several examples: on the left of the mural, the galaxy spins using an embedded motor, and the boy seated on the beach plays a tune using a speaker. On the right, the mask vibrates and the octopus figure changes color.

project, however, the wearable elements were intended to gather data for use with a larger, in-class project—the mural created by the children and shown in Figure 6. Each portion of the mural was created with a specific child's wearable device in mind; for example, the boy sitting on the beach at the bottom left of the mural plays a musical tune in response to the data sent from its creator's wearable device.

Again, there are many more observations to be made about the children's work and its relation to their conceptions of fitness and health; but for the purposes of this discussion the crucial point is that a "hybrid" approach to children's technology, in which (relatively) complex classroom projects are linked to wearables, is communicable to students. It would be precipitate to claim on this basis that a fuller computer science education could be founded on this strategy; the following section presents a more thorough (if speculative) argument to that effect.

3 Designing a CS Curriculum Based on Fungible Programming

The previous section presented a variety of working projects that suggest the plausibility of a “true” computer science curriculum based on fungible programming. This section, more frankly speculative, is intended to outline the sorts of activities that such a curriculum might entail.

The essence of the "fungibility" notion is that a variety of devices, independently programmed, can communicate results to one another in a natural, interchangeable way for a variety of projects beyond the scope of any one device. There is, it should be noted, nothing conceptually startling about this idea: one could view JPEG and STL formats, for instance, as multi-platform data files for use across many projects in just this sense. The purpose of emphasizing the notion here is that the burgeoning world of hobbyist devices—prominently including wearable devices, embedded microcontrollers, RF tags and readers, and the like—have not been incorporated into an educational framework for computer science in the way that they could. For the most part it is fair to say that the standard model for working with these devices is that they are first programmed from a desktop machine, and then placed into situations (like a "light-up jacket") in which they do their work autonomously. Wonderful as many of these examples are, they constrain the possibilities for education, since (as already noted) the programs associated with the vast majority of projects are limited in complexity.

A true CS curriculum for devices of this sort (and here we focus on wearables) should instead regard the hobbyist items as small-scale computers in their own right, capable of communicating back and forth with one another and with desktop devices. In a sense, then, rather than conceive of wearables as creating "dynamic clothing", we can view them as extending the capacity and range of the classical input and output "boxes" in the standard Von Neumann architecture. Wearables can be viewed as a type of "intelligent skin" for more complex programs created in classroom settings.

A few suggestions for how this idea might be implemented follow; and the reader is encouraged to brainstorm similar possibilities for him- or herself.

• One possibility might be to employ "tuned" wearables as the basis of a computationally-enriched scavenger hunt.
(not unlike the basic structure of "geocaching" activities). For instance, a sound sensor might first be programmed to seek out different frequencies (perhaps out of the range of normal human hearing), and the students' jobs would be to gather instances of these frequencies from their community. Once gathered, the job of the students would be to design a display program that (e.g.) would show the locations or distributions of the gathered frequencies on a desktop screen. The purpose of such a project would be to combine the strengths of wearables with a more challenging problem in data display. Similar ideas might be implemented with (e.g.) light or chemical sensors rather than sound sensors as a foundation. For instance, students might "gather" colors from outside that could then be added to a graphical palette.

• A machine learning project could be founded on programming simple three-layer neural networks into wearable devices so that students could now go out into the world and train the devices to respond to particular words, colors, or voices. For instance, a wearable might be equipped with a student accessible button (in fact, our system in Figure 2 employs just such a device); when the wearable correctly recognizes a voice (by, say, lighting a green LED) the student reinforces the network, and when it fails to do so the student induces a revision of the network weights. Over time, then, the student's wearable should become a completely individual device that recognizes only (say) the voices of selected people, or selected code words. Here, an introduction to machine learning can be naturally combined with the playful individuality of a wearable device.

• Students might design wearables that can communicate with each other over short distances, forming themselves into larger multi-person "display screens". For instance, a set of students positioned in a line might program their wearables to behave like individual cells in a one-dimensional cellular automaton. By standing nearby one another, the line of students could create interesting graphical patterns. Such a project could be accommodated to relatively small or large groups of students, and perhaps to more complex patterns of communication (e.g., two-dimensional arrays). (For an early, pioneering effort along these lines, see Colella's [6] work in participatory simulations.)

• Much as the students in our own clubhouse designed a "communal mural," one could imagine a classroom of computer science students creating a highly ornate or complex "communal wearable" to which all students contribute over time. Wearing this garment could be thought of as a sort of "prize" (think, for an analogy, of the coveted "yellow jersey" worn by the leader of the Tour de France bicycle race). The CS teacher could thus design an activity in which (e.g.) for every practice problem completed successfully, a personal LED is added to the classroom garment; and from time to time, individual students (as a sort of playful reward) are given a chance to wear the garment. This, again, might be a strategy for integrating the joy of personal display with the (arguably more abstract or mundane) tasks of the day-to-day computer science classroom.

Activities of this sort all seem to us implementable, and only experience will determine their educational value. Nonetheless, it should be noted that the design of projects of this sort are compatible with many (if not all) the standard elements of an introductory CS curriculum. Activities such as these could be blended with an introduction to various syntactic forms (conditionals, iteration); or with programming of graphical or musical data; or with ideas of graph search, machine learning, cryptography, pattern matching, emergence, and so forth. Perhaps most interesting is that the aforementioned notion of wearables as an "intelligent, computationally-enriched skin" or sensory membrane surrounding a core project could bring students to challenging concepts that do not always make their way into introductory CS courses—notions such as parallel or multiprocessor programming (the "cellular automaton" example mentioned above might involve simple concepts along these lines, as individual wearables would have to be synchronized to enable the project to work).

4 Related Work, and an Affective Rationale for Wearables in the CS Curriculum

The excitement of wearable computing has already become apparent in hobbyist culture, and seems to hold particular appeal for the young. Numerous recent books such as [4], [5], [8], and [10] speak to the creative ferment around these devices. Moreover, from the standpoint of demographics, accessible wearables seem to attract new populations to computer science, as noted by Buechley and Mako Hill [3] in an early study of the use of the LilyPad Arduino. In particular, these devices seem to hold strong (though not exclusive) appeal to young women and girls, who are traditionally underrepresented in computer science courses. While some computer science educators have sought motivational appeal in topics such as game design [11], robotics [9], and computer graphics [12], adding wearables to this constellation of "core motivating activities" could attract still different groups of students to a lifelong interest in computing.

More generally, incorporating wearables into computer science education suggests something of a different "lens" through which to view many of the debates that historically characterize this field. Rather than argue over (say) choice of language, or textbook, or the core notions of "computational thinking" (this last seems to be an endless source of argument in our own community), we could begin rather with a narrative or biographical view of the student's purposes in going into computer science. That is: who in fact does the student wish to be, or become, through learning computing?
Typically, educators answer this question in the least interesting way possible: by assuring students that a (presumably corporate) job awaits them at the end of all this effort. Not surprisingly, the question of acquiring a job might motivate some students (especially older ones) but it provides at most a thin, potentially grim-sounding future for many others. Merely telling students that they can spend a lifetime earning a salary (or worse, "competing with China") is, in effect, attempting to graft the motivations of employers or executives onto the lives of creative students.

In general, youngsters are attracted to computer science (when they in fact are attracted) by a vision of who they can be–of what type of person, what type of life, is associated with this study. A computer science education that includes, or focuses on, wearables not only has intellectual benefits (introducing embedded computing, distributed processing, and the like), but it has affective resonance for many students. A youngster can wear and display his or her own interests among their peers, in theatrical productions, in slam poetry contests, in sporting competitions, at dances. Computing means something to many students in these contexts. A CS curriculum that incorporates these devices–and that employs the strategy of fungible programming to bring wearables into the world of advanced computer science–stands a fair chance to make a difference in many students’ lives, and (we can hope) in the larger practice of CS education.

5 Acknowledgments

The work described in this paper was partially supported by the National Science Foundation under award IIS-1231645.

6 References