Any discussion of "computer science education" presumes, as a prerequisite, that the participants have a reasonable consensus as to what the subject entails. It's therefore worthwhile to note that, as a field deeply interwoven with metaphors and problems derived from technology, "computer science" (and by extension, "computer science education") has historically been a swiftly moving target. To some degree, other scientific or technical disciplines have this feature as well—the technology of (say) DNA sequencing or spectrometry has affected even introductory education in fields like biology and chemistry. Still, computer science seems especially and unavoidably fluid (or maybe fragile) in this respect. Four decades ago, a central concern was memory management and programming in Fortran; three decades ago, the design of screen interfaces for home computers and programming in Pascal; fifteen years ago, the design of Web interfaces and programming in Java; and today (arguably) designing apps for mobile phones and programming Arduino microprocessors. We seek enduring ideas in the form of "computational thinking", but even the typical examples put forward here are derived from a relatively narrow, and still rapidly evolving, set of core concepts in traditional programming courses (stacks, recursion, modularization, debugging, and so forth).

None of this is meant to argue that computer science education is a hopeless or vacuous exercise; but it compels us to step back and reflect upon what we really feel is essential and (relatively) constant in the field. The argument of this paper is that generally these discussions have been hobbled by an implicit assumption that, when we talk about computer science education, we are talking about narratives of skill acquisition, employability, national competitiveness, and efficiency. We are constantly reminded that "children need skills for the 21st century", or that "technology skills are imperative if America is to remain competitive", or that "many jobs are expected to open up in technical fields".

To put the matter bluntly, narratives of this type are trite, and a brittle foundation on which to place any discussion of the purposes of education in any field (computer science included). Broadly, they ignore the crucial lens through which to view any educational discussion—namely, that of biography, or purpose, or individual narrative. Rather than imagining that students are deeply concerned with national competitiveness, or even with mere employability (is that really enough to motivate a high school student’s interests or life decisions?), it is far more realistic and productive to think of them as in the process of forming a biography, and (as part of that task) an intellectual personality.

In other words, the central question of computer science education—the one from which decisions of hardware, projects, and specific domain content should flow—is
"what kind of person can one be, or become, through an interest in digital or computing technology?" This is, again, the question of purpose: or presenting students with a plausible, dignified, enjoyable, and admirable imagined biography. These are the student’s purposes, of course, not those (competitiveness, efficiency, moment-to-moment skill acquisition) derived from the language of their employers or political leaders.

With this alternative focus in mind, it is useful to think of computer science education as part of a broader technological "maker movement" that is as much about empowerment, or personality--about an approach toward the control of technology in one’s life--as it is about any one particular choice of content. The "maker movement" is something of a catch-all term for a recent blossoming of public interest in personal and expressive construction.1 "Making", in this recent context, is associated with a variety of powerful but accessible technologies, prominently including desktop fabrication (with 3D printers, laser cutters, computer-controlled milling machines, sewing machines, and the like), innovative materials (shape memory alloys, conductive paints and fibers, specialty textiles and papers, and many more), embedded computation (the Arduino microprocessor and its many cousins in the landscape of hobbyist computation, along with numerous connected sensors and actuators), and interface elements (such as nanoprojectors, gesture recognition devices, and wearable input devices). Besides these technological elements, the movement is loosely identified with a number of websites, enterprises, institutions, and publications--e.g., Make magazine (and its "Maker Faire" events and makezine.com site), the instructables.com "how-to" site, the Exploratorium museum in San Francisco, and SparkFun Electronics in Boulder, Colorado (just to name a few).

Most important, the rhetoric associated with "maker culture" is one of adopting a stance of personal empowerment in the (often daunting) face of technological complexity. Makers are commonly encouraged to "open up" their technology and look inside, to "void the warranty", to build with screws (which can be undone) rather than nails... and to be willing to make not only a final product, but also a mess. Consistent with this style, maker projects—the type seen at Maker Faires—are often assumed to be idiosyncratic, artistic, and technologically innovative. Especially interesting is the notion that makers have a personal stake in their work, and that their projects serve to define them: in this light, to be a "maker" is (at least potentially) to adopt a bit of the persona of the artist, or inventor, or independent scholar, or outsider. A "maker" can be a little bit subversive, challenging the technical powers-that-be; or heroic, using his or her constructions to effect change; or even persuasive, designing projects that in turn empower other makers.

Consider, in reflecting upon this too-brief (and somewhat caricatured) description of a culture, how little of this is visible in the culture of computer science education.

1 See Crawford [2009], Frauenfelder [2010], and Anderson [2012] for several recent popular books emerging from the maker movement.
Is there anything analogous, in the computer culture of the young, to a "Computer Science Manifesto"? Is there anything analogous to free-wheeling, widely-attended popular events like the Maker Faires? Is there a sense that computer science education is providing students not merely with skills (which, without purpose, are singularly empty things to acquire), but with a mission, a movement, or a community?

 Viewing computer science education as an element of maker culture need not be seen as some sort of radical break with tradition. Indeed, maker culture is itself part of an indelible historical trend in American technological history. The "makers", as just described, are the natural descendants of the homespun builders described in de Tocqueville’s [1840] account of early nineteenth century America [see chapter XIX, vol. 2], or the "Arts and Crafts" movement of the late nineteenth century, or the radio hobbyists of the early twentieth century, or the Homebrew Computer Club (and its variants) of the 1970’s. There is nothing unprecedented, then, about the current maker movement; it is just the latest manifestation of a recurring element of American culture. Computer science education, viewed in this context, can tap into an enduring and compelling narrative for young people—a narrative that evolves (along with technology, the sciences, and the arts) from generation to generation, but that seems consistent in its ability to convey a sense of personal and technological optimism.

What are the particular features that we would expect to find in a re-imagined computer science education along these lines? The following are themes that would likely shape a "maker-inspired" foundation for CS education:

Computers as part of a technological ecosystem. The "maker" approach is, as its name would suggest, centered on the creation and completion of artifacts. In computer science, such artifacts might be "pure software", without any tangible component; under this interpretation, a "maker-based CS education" would tend toward the creation of full working programs or systems rather than small-scale exercises with (e.g.) language constructs. More generally, though, the maker approach suggests that computer science be viewed as one (highly important and powerful) element of engineering and construction. In this view, computer science is not exclusively about the production of software or "virtual worlds", but rather about extending dynamic, complex, or programmable behavior into constructions of all sorts. Thus, a maker-oriented computer science would likely exhibit a disproportionate intellectual focus on embedded computing (working with microprocessors) and distributed systems (creating systems with multiple communicating elements). It would also see computers as elements in a creative technological landscape that includes 3D printers and scanners, a growing selection of sensors and actuators, and "computer-friendly" materials such as conductive thread or ink. A version of CS education that focuses on construction is much more likely to weave the student’s computer science expertise into a familiarity with a workshop-full of supporting and related devices; it is also likely to introduce students to (for example) elements of 3D geometry and design, and basic electronics. In this way, a maker orientation in CS education is likely to extend the subject beyond the usual confines of "computational thinking", with its relentless emphasis on software patterns.
Pluralism and an autodidactic approach to hardware, software, and materials. Discussions of computer science education are all too often hampered by tribal affinities centering on issues such as choice of language, device, or (most depressing of all) textbook. In the mid-1980's such debates took the form of "should we teach BASIC or Logo to elementary school children?", and the arguments have not gained appreciably in content ever since. In maker culture, there is a relaxed and ecumenical quality to such questions: Maker Faires are populated with display booths in which people use all sorts of microprocessors, desktop computers, programmable handheld devices, and more. The spirit of the event is "whatever works": the point is to make a beautiful or expressive construction, and to gain expertise in the tools needed for that purpose. Note that the essential difference, once again, is one of purpose: if the idea is that we have to teach students "the skills they need to compete for jobs", then our first priority is to decide what those skills might be and ensure that they are consistently taught. Makers, by contrast, don't especially worry about skills as elements of a resume, or about competing with China, but rather about the quality and meaning of the things that they construct. This ecumenical approach extends beyond hardware and software to style (maker projects might have a homespun look, or a steampunk aesthetic, or a sleek Art Deco design) and materials (wool, wood, acrylic, fiber optics, and myriad other types of "stuff" show up in these projects). Makers are likewise opportunistic about sources of instruction: one learns from websites, online communities, colleagues in "hacker spaces", and (where appropriate) formal instruction. A CS education fashioned along these lines would be far less concerned with standardizing a choice of curriculum, and far more concerned with strengthening students' readiness to master those resources ready-to-hand.

Finally, it should perhaps also be noted that, in the process of doing precisely what matters to them, makers tend also to develop skills (working with 3D printers, writing embedded programs, wiring computers into homebuilt robots) that make them highly employable, should that be their wish. The point here is not that computer science students shouldn't have jobs—rather that they should have biographies first, within which meaningful jobs are woven.

Projects: interdisciplinary, public, associated with supportive widespread communities. A central focus of maker culture is not merely the creation, but the display of constructions. Websites, Faires, and public spaces become the settings in which one's creations can be shown; the constructions themselves thus become the background elements for conversation within supportive communities. Public display (in a wide variety of potential media) is thus an element of intellectual growth. In a similar vein, a maker-inspired CS education would place a far larger emphasis on the display and sharing of programmed artifacts.

Attention to infrastructural issues: re-use, communication/dissemination, means of accreditation. Makers tend to think in terms of supporting the larger community of the maker movement itself: they work on novel fabrication tools (or improvements to old ones), on improved techniques for working with unusual or new materials, on sharing their ideas with colleagues. In this sense, the engineering style of the maker movement is one that stresses open or reusable design (again, "screws, not nails") and dissemination (via tutorials, instructional videos, websites, and the like). A CS education incorporating this philosophy would stress the idea that a program, or programmed artifact, should be designed for open use and re-use, for readability, for interest; and
that a truly effective computer scientist is one who knows how to explain his or her work in a blog post, article, or video. A natural corollary to this ethos is that a good computer scientist is also a good critic, or audience: someone ready to understand, critique, support, and respond to the work of others. Accreditation and assessment in the maker community thus come not from standardized tests (who could imagine such a thing?) but from the active participation of one’s colleagues—much as in the subcultures of (e.g.) music, fine arts, theater, and athletics. Computer science education can profitably move toward this modus operandi.

The argument as presented here—for refashioning computer science education through the example of maker culture—has been brief of necessity. There is much more to be said about how such a rethinking could be effected. The central point, however, is that specific decisions about education—choices of tools, techniques, means of assessment—follow from a decision about purpose. Why should a child, or teenager, or young adult, long to commit his or her intellectual life to the cramped vision of a remote executive? Why should his or her deepest goals be co-opted by the sterile language of "national competitiveness"? Education of this sort, inexplicable education, is doomed to fail, regardless of subject matter. By thinking instead in terms of the student’s purposes, we can arrive—as the maker movement, for all its complexity and flaws, seems to have done—at a more substantial and compelling purpose for education in general, and computing education in particular.

Acknowledgments. The National Science Foundation has supported the work on which this essay is based under grants DRL1114388, IIS1231645, IIS0856003, and CNS0940484; thanks also to SparkFun Inc. for their generous fellowship support. Conversations with Gerhard Fischer, Clayton Lewis, Roy Pea, Nathan Seidle, Diane Sieber, and Scot Douglass have been especially helpful. Thanks also to members of the Craft Technology Lab past and present: Ann Eisenberg, Ben Leduc-Mills, Swamy Ananthanarayan, Jeeun Kim, Leah Buechley, Nwanua Elumeze, Yingdan Huang, Glenn Blauvelt, and Sue Hendrix.

References