

Discovering Computing: Perspectives of Web Designers

Brian Dorn
Departments of Computer Science and
Multimedia Web Design and Development
University of Hartford
200 Bloomfield Ave
West Hartford, CT 06117
bdorn@acm.org

Mark Guzdial
School of Interactive Computing
Georgia Institute of Technology
85 5th St NW
Atlanta, GA 30332-0760
guzdial@cc.gatech.edu

ABSTRACT

This paper presents findings of an exploratory, qualitative study of professional web and graphic designers who regularly write computer programs. These participants have a wide variety of educational backgrounds, including some who had a few classes in computer science. Our participants report having found a career in which they enjoy being both creative and technical. They want to learn more about computing, but do not find computing curricula as meeting their needs. We discuss their perceptions of the computing discipline, motivational aspects of their jobs, and their interest in learning more about computing. We also consider implications for computing curricula if we were to try to meet the needs of these professionals who are unlikely to appear in our current classes.

Categories and Subject Descriptors

K.3.2 [Computers and Education]: Computer and Information Science Education—*literacy, computer science education*; I.7.2 [Document and Text Processing]: Document Preparation—*hypertext/hypermedia, scripting languages*; K.7.1 [The Computing Profession]: Occupations

General Terms

Design, Human Factors

Keywords

Web development, informal education

1. INTRODUCTION

Computing is a critical part of the economies of the developed world. Many people use computation in a serious way, including writing programs, even without training as professional software developers. How do non-computing professionals come to discover computing? What do they want to know about computing? In this paper, we explore

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ICER'10, August 8–11, 2010, Aarhus, Denmark.

Copyright 2010 ACM 978-1-4503-0257-9/10/08 ...\$10.00.

these issues within one group of professionals. We found that participants in our study have discovered and do value computing—many explicitly express a desire to know more about programming techniques. Yet, at the same time, participants struggle with computer science as both an academic and professional discipline. The words of one participant set the stage nicely:

P10: I fly airplanes for fun. I wasn't happy just learning to fly airplanes. A lot of people can just fly airplanes. They just move the stick around, and they get [the plane] on the ground and everything's fine. But you put them in an unusual situation, and they don't understand it. If you have an understanding of the aerodynamics of the airplane and the wings and how they work, generally you're more likely to come out of [the situation].

I think programming's probably the same way. I've written classes and thought, wow I've just created a binary tree here. If I knew what I was doing when I was doing it, because I had the academic understanding, then I'd probably look for a base class that's already been optimized, and I wouldn't have to rewrite it. So, that was a really long way of saying yes, I think that an academic study would make me a better programmer, but not by a whole lot. And I don't think that all of the really great programmers out there had academic programming educations.

In his analogy, participant 10 conveys the power of a deep understanding of one's tools. Such an understanding allows pilots to cope with dangerous situations, and it allows programmers to recognize efficient ways to use the tools at their disposal when solving a problem. While he desires this additional knowledge about computer systems, he reminds us quite clearly that one need not be trained as a computer scientist to excel. After all, he is a self-taught programmer with a Psychology degree who builds web applications.

As a computing-related occupation outside the typical scope of computer science, professional web designers serve as a unique population with whom we can investigate informal learning and computational literacy. As we see in the above quote, at least some of these designers explicitly *reject* academic computer science. A study of these designers gives us insight about the breadth of computing careers beyond a typical occupation in software engineering.

Jeannette Wing’s vision of Computational Thinking is about more than just programming: it is thinking about systems at various levels of abstraction, and it is for everyone [16]. Her compelling argument for general literacy of foundational computational concepts has spurred high profile discussion regarding computing education in recent years (see e.g., [3, 8]). In a sense, web designers and developers have already made a professional commitment to computational thinking, but they differ from traditional computer scientists in that they have often developed their understandings outside the confines of CS classrooms.

This paper presents results from an interview study involving a group of web and graphic designers who regularly engage in programming activities. In conducting these interviews, we sought to explore two broad research questions:

1. What role does computing play in the career paths of these designers?
2. How do these designers learn new information about computing, and are there topics related to programming that they wish to know more about?

We use an interview study here because our goal was to form hypotheses for later exploration. The practices of web and graphic designers with regards to software development may be highly sophisticated and nuanced, necessitating an observation-based method appropriate for studying implicit learning [4]. However, our earlier work in this space [10, 11] suggests that their practices are more explicit, making the above research questions amenable to interview-based methods of inquiry.

The remainder of this paper outlines the method used in our study (Section 2), details relevant aspects of the participants’ backgrounds (Section 3), and presents four themes related to the field of computing and computing education derived from the interviews (Section 4). We conclude with a discussion of the results and their implications for computing educators.

2. METHOD

Participants for this study were recruited through email from local Meetup groups¹ for web and graphic designers. Those who volunteered for the study were pre-screened to ensure that they were either professionals or students working in the web and graphic design field and had firsthand experience with programming. After screening, face-to-face interviews were conducted with 12 participants who were each compensated with a \$15 gift card for their participation.

The data for the study was collected in two parts. Participants completed a survey that collected demographic information and data about their educational and professional background. We then conducted a semi-structured interview with each participant regarding their professional duties and interests, their use of software tools like Photoshop, and their use of programming languages.

Following collection, survey data was aggregated and audio recordings of the interviews were transcribed. We then employed a multi-step thematic analysis to analyze the interview data. Thematic analysis is a qualitative analytic method that aims to provide a rich and detailed account of

¹<http://www.meetup.com>

Table 1: Participant Job Titles

Title	Frequency
Web Designer/Developer	7
Programmer/Software Developer	2
Other: Animator	1
Other: Interactive Animator	1
Other: Designer/Developer	1

the data collected [6]. The end result of a thematic analysis is a collection of themes based on common patterns observed in the data (e.g., interview transcripts). It is important to note that while themes are necessarily repeated by various participants throughout the interview corpus, it is not the goal of such an analysis to convey the prevalence or relative importance of the themes.

To conduct our analysis, transcripts were coded initially in a hybrid fashion that included both a top-down approach with categories related to specific questions asked of all participants (e.g., job duties, use of Photoshop) and an inductive approach allowing for additional emergent categories in the data. These initial codes were then refined using a second phase of analysis that examined coded segments to generate new sub-codes, which permitted additional discrimination and serve as the basis for the themes presented here.

We have selected representative quotations to illustrate each of the identified themes. In preparing these transcript excerpts for presentation in this paper, we have edited quotations for anonymity and brevity as necessary.

3. PARTICIPANT BACKGROUND

Twelve people participated in the interview study—five women and seven men. All were actively involved in the web/graphic design profession. Ten participants reported working in the field (four of whom were self-employed), and the remaining two were students currently enrolled in degree programs in web development.

All but two participants held a bachelor’s degree, but their formal programs of study had wide variation. As one might expect, some of the degrees earned were directly related to the field: Visual Communications, Animation, and Internet Programming. However, only four of the participants, or one-third, had such degrees. One person held a traditional Computer Science degree, and the remaining five degree holders were trained in a variety of disciplines (e.g., humanities, theology, packaging engineering).

Despite the lack of common educational backgrounds, the participants identified very strongly with the same occupation. Table 1 lists the participants’ job titles. Most identified primarily as Web Developers, while only two chose the title Programmer. Others used a more descriptive title, like Interactive Animator or Designer/Developer, to convey that their jobs incorporated artistic design skills as well as technical skills.

Participants represented a broad cross-section of the field and were diverse in terms of their age and experience. Participants reported between 2 and 13 years of experience with Photoshop and between 2 months and 15 years of experience using scripting or programming languages. The average self rating of scripting or programming expertise on a scale from 1 (novice) to 5 (expert) was 3.21. All participants indicated experience

with more than one scripting/programming language, with the most common languages being JavaScript, ActionScript and PHP.

One participant summarized his work, and the work typical for many of the participants, as follows:

P1: From 9 to 6 during normal work hours I'm a front-end developer. Um, that means I take comps² designed by designers done in Photoshop, slice them up, and create standard-based web interfaces that render correctly across most grade A browsers. In addition I do the JavaScript coding that adds a level of interactivity and enhanced user interface to the templates.

Several participants also added that a portion of their time is spent working directly with clients on the design aspect of web development.

4. THEMATIC ANALYSIS RESULTS

With this understanding of our participants' background and the nature of their work, we turn to the results of our thematic analysis. Though participants were not formally trained in a traditional computing discipline, like computer science, a number of themes related to computing emerged from the interview data:

1. Participants expressed their perceptions of traditional computing careers and those who choose such careers.
2. Participants commented on their personal experiences in computer science classrooms.
3. Participants shared their excitement about working with the Web and the power that programming affords.
4. Lastly, while reflecting on what they would like to know more about, several participants relayed a desire for knowledge of topics that are traditionally part of the CS curriculum.

The following four subsections present each of these themes in turn.

4.1 Perceptions of Traditional CS

The first theme relates to the participants' perceptions of traditional computer science. This includes both thoughts on the type of work one might be expected to do as well as views on the people engaged in that line of work.

4.1.1 *The Job*

Many participants described themselves as front-end developers involved with user interactivity and visual components of web systems. In contrast, they expressed a view that those who work as traditional programmers or software developers often spend their time developing low-level code. They explain that this sort of coding is "behind the scenes" and is often not connected to the interactive experiences of the user. Participant 2 hints at this disconnect from the user when reflecting on her interdisciplinary college degree:

²In graphic design a "comp," or comprehensive layout, is a prototype design created to illustrate major components of a page.

P2: I was able to take different samples from different places and instead of just being let's say an MIS major, or computer science major, you know it's—you're not going to be front-end anything with computer science. You're going to be back-end everything.

Participant 9 echoes this sentiment and provides additional explanation while relaying a story about the distinction between scripting and programming:

P9: I met a guy who programs ATM machines, and he busted me for calling myself a coder. He says, well no, you're a scripter. And the coders you know, they're down there in the dirty behind the scenes playing with the rendering buffers and you know, moving bits of memory back and forth.

These comments and others like them provide insight into how these web developers view traditional programming careers. Additionally, with an emphasis on user interaction and design, participants also clearly positioned their interests outside the scope of what they believe a traditional computer scientist might do:

P4: I think as a front-end developer, you focus more on the design and the usability, and you're focusing more on the audience. And then on the back-end I think you're focused on more, these are like the software developers. And they're programming something, and they don't really see what it's gonna look like; they're just making it work.

P7: Being a designer and developer, I'm a go-between in that regard. Understanding design concepts and where to place items. Whereas a developer would just go through and list things in order of functionality and not in terms of interactivity.

These quotes illustrate the value of a focus on the interaction between the computer and the human user for those engaged in web development. Because of this explicit emphasis, our participants felt separated from traditional software development, which they saw as focused on the "back-end." They also related stories of professional developers dismissing their work practices. As a result, our participants are unlikely to develop an identity within the professional software development community of practice [12].

4.1.2 *Personality Traits*

Not only did participants distance themselves from software development careers in terms of the job description, but they also described their personalities as different from those who choose such careers. Yardi and Bruckman [18] previously reported on teenagers' negative perceptions about computing careers, noting a number of remarks about the field being boring, asocial, and lacking creativity (e.g., "I don't think I want to be a programmer because it's too tedious and I don't think I could do that" [18, p.42]). Our participants expressed some similar negative stereotypes of traditional programmers and contrasted them with their own personalities.

P2: I went to a meeting for some kind of programmers, something or other. And they were OLD, and they were nerdy, and they were boring! And I'm like, this is not my personality. Like I can't work with people like that. And they worked at like IBM, or places like that. They've been doing, they were working with Pascal. And I didn't—I couldn't see myself in that lifestyle for that long.

P5: I don't know a whole ton of programmers, but the ones I know, they enjoy seeing them type up all these numbers and stuff and what it makes things do. Um, whereas I just do it, to get it done and to get paid. To be honest. The design aspect is what really interests me a lot more.

For these participants, the desire to be creative, coupled with an interest in aesthetics and design, is an important differentiating characteristic. For example, participant 6 states simply, "I don't know how programmers can stay interested in just the code, because I'm a very visual kind of guy." These perceived social differences further distance web developers from traditional software development communities of practice. While affiliation with such communities is obviously not essential, a closer relationship could lead to the introduction of a range of practices beyond basic programming that are directly relevant to web developers' careers.

4.2 CS Curriculum

Several participants had first-hand experience taking one or more programming classes. Most often, participants had an introductory programming course in either college or high school. Only one participant held a computer science degree. Two others started their post-secondary studies as computer science majors, but changed majors soon after arriving on campus. In the quote below, participant 7 discusses his reasons for changing his major:

P7: I started out in computer science, but didn't like it at all. The fact that I wasn't learning anything new. I took an intro to programming course, and then I talked to some other people in the program and it was all repetition and I guess there wasn't any really new—'cause at the time there was only like Java was just starting, and C was like the graduate language, but you still had to learn on Pascal I think. So you weren't really learning any concepts. You were learning the languages, and I didn't like that at all. So that's why I left.

Other participants echoed the sentiment that introductory computing courses were all about learning the syntax of another language, rather than learning concepts. As many of the participants had experience with various programming languages prior to enrolling in college, the syntax-driven curriculum of most introductory CS courses was frustratingly basic. Even more problematic for participant 7 was the perception among CS students that the entire curriculum was just about learning several different languages.

The lack of a design component in introductory CS coursework was also seen as troublesome. Participant 4, who was encouraged to major in CS after expressing an interest in

being a web developer, lamented the absence of design in her CS classes:

P4: I started out as a computer science major with programming and even though I told them I wanted to be a web developer, they recommended that I do computer science. But once I got into it, I'm like programming is not really what I wanna do. I think I'm more of a front-end designer. I like to program to make it work, but I didn't see that I was getting any coursework where I would design anything. And I found out that another department had a multimedia component, and I mean I was taking classes in Flash, video editing. And that was, I felt like that was a much better fit. And the classes were a lot smaller. The professors seemed to be way more involved. And it was much more project based than reading the book and completing different things. I think that was a much better way to learn it.

It is interesting to note that several of our participants had tried the introductory courses that our discipline has to offer. While we know little about the design or progressiveness of these courses, computing educators can probably recognize aspects of a traditional introductory course from the participants' descriptions. Simply put, these web developers did not find what they were looking for as students in our classrooms. These perceptions about what was missing can guide computing educators in evaluating future introductory experiences.

4.3 The Appeal of Web Programming

Despite the somewhat negative tone to the views expressed in the previous two themes, the study participants had a genuine excitement for their careers in computing and their use of programming as a tool in their line of work. In particular, the Web as a medium was powerfully motivating. Participant 7 summarizes his excitement by saying, "I was into art before, but on the Web you could see your code alive."

But what unique features of programming for the Web make it appealing? Participant 11, a non-traditional student seeking a second career as a web designer, talks about her decision to go back to school and the attraction of web work:

P11: I really kind of looked at a lot of different options, and I thought well [web design] would be a way for me to tie in all my technical, you know, expertise and then also throwing in the creative piece. I guess cause it's kind of unusual I think to have, to like technical and creative. So when you find something that pulls both of those together, it's really exciting. Cause I've kind of been without it for so long. It's like oooh this is it!

The ability to be technical through writing code while simultaneously creating something that is inherently visual was a property of the Web that appealed to these developers. Furthermore they were motivated by the notion that their creative artifacts could be immediately consumed by a public audience. As another participant puts it:

P6: I was taking these just letters and numbers and symbols and then all of a sudden it was like this beautiful floral thing. I was manipulating things that the world could see.

For many participants the appeal of web work was closely related to a recognition that programming is a fundamentally powerful tool. When another participant was asked what it was about web design that interests her, she exclaimed:

P12: The coding! I don't like to code. But the things that the code can do is amazing, like you can come up with this and voilà, you know, it's there. JavaScript for one. The plug-ins and stuff. I think that's very interesting, intriguing and stuff. Because I mean like the code is just, there's so much you can do with code and stuff. It's just like wow.

The flexibility and sheer reach of toolkits currently available online was only one significant feature of programming mentioned. Over the course of the interviews, participants praised programming as a means to transform raw data into information, as a way to provide user interaction in designed artifacts, and as a significant time saver that helps automate tedious tasks. Additionally, some participants described programming as a creative component that permits new types of expression to which an artist would not otherwise have access. For example, participant 9, formally trained at an art college as an animator, elaborates as follows:

P9: My most common I think goal in scripting is to try to express artwork in a way that only, with code, you can. Hence the term interactive animator. My first ventures into it I was using it as a way to—I had made a movie, and basically what I had done is I had made it so the code could randomize what different characters in the movie looked like. Or would randomize different paths, and the movie had like three or four different endings that would be randomly determined while you're watching it.

Our participants articulated a distinct passion and joy for programming. As teachers, we hope to hear our students express this kind of excitement for what they can do with code [13]. However, there is an important message beyond excitement conveyed in these participants' comments. They may not necessarily enjoy the act of programming itself, but they write code in order to do something that they love.

4.4 Need for More Computing Knowledge

When asked what they were interested in learning next, most participants had a desire to know more about the tools and languages they already use or wanted to learn new languages, like Ruby or Python. However, other more advanced computing topics also came up. Among these topics were a better understanding of relational databases, techniques for memory management, knowledge of "best practices" in software development (e.g., software design patterns), a deeper understanding of algorithmic complexity, and as P1 puts it "... trying to think more like computer scientist. You know,

being able to code more algorithmically." The final theme we explore here is the role of additional knowledge about computer science recognized by our study participants.

In the opening quotation to this paper, participant 10 expresses his desire to know more about fundamental principles of computer science. His interest is driven by a thirst to know more about the inner workings of the systems and languages he uses. He was not content to just understand the syntax of a language, and he was not alone in this interest. While P10's interests are driven primarily by curiosity, participant 1 shares a desire for foundational CS knowledge for the practical reason that it, unlike many aspects of programming on the Web, does not change as frequently:

P1: So I mean technology changes. So what I am ideally looking to focus on are like the foundation. The things that change less, you know what I'm saying? Like computer science um, theory, you know I'm saying I mean? That kind of like, it's applicable to what I do, and it's not so constantly shifting.

Participants recognized that additional conceptual knowledge would help them cope with inevitable changes in the tools and languages they use, while perhaps also giving them access to more effective and efficient ways of programming. Participant 10 illustrates this point while reflecting on his discovery of design patterns.

P10: I was the kind of programmer that could make stuff work. But I didn't really have solid understandings. At one point I picked up a book on design patterns and I looked at it, and I was like that's really, that's really interesting. And some of them I use, and some I do, you know, anti-patterns. And now I've made a correction. This academic look at what I do for a living changed the way that I wrote software. So I was like well I wanna keep doing that because it made me a better programmer. And it was more fun to program, and it was more thought provoking.

For others, acquiring knowledge of more advanced topics was a requirement for taking on some new project. In particular, as the influence of the Web extends beyond the confines of traditional desktop computing, those involved in web development become naturally interested in creating interfaces and applications for mobile and ubiquitous devices, like the iPhone. One participant pointed out, however, that doing so requires considerable knowledge of topics that web developers may have never encountered before (e.g., memory management, the Model-View-Controller design pattern).

Unfortunately, just having an interest in more advanced computing concepts is not enough. When attempting to learn something new, participants expressed a preference for online resources like FAQs, tutorials, and code examples, while also using reference books. These results echo those of other studies of web and graphic designers [10, 14]. However, for many of these more complex topics, the typical learning strategies may not be enough because the available information is perceived to be either too basic or too advanced. For example:

P7: I'm looking for more concepts instead of examples I guess. I think the, my problem with

books was the same thing. They're teaching more of a language than the concepts, and so I just want a place where I can learn the concepts and that's it. And I really can't find that, you know? It's either you learn a language and you hope to find out about the concepts, but you never really do. Like the Model-View-Controller for example. Um you know they tell you what it is, but they don't—There's good uses for it, but just to do that all the time I don't think is good programming practice.

In addition to struggling with locating appropriate resources that target more abstract concepts, participant 7 also expresses his difficulty in determining how and when to employ new concepts in his own programs. It is certainly not unusual for novices to wrestle with the applicability of new knowledge [5], but this is particularly problematic on the Web, where many resources were not originally intended to serve educational purposes and therefore lack instructional guidance about the underlying rationale. Such explanations are critical in developing expertise [5].

5. DISCUSSION

The web designers in this study came from a wide variety of educational backgrounds. Whether the result of training at an art school or a post-college career change, these participants have discovered a career that simultaneously satisfies their interest in technical and creative pursuits. They have not followed the traditional academic computing education path, and they distance themselves from those employed as software developers. Nonetheless, they perceive computation as a powerful tool, and there are indications that additional conceptual knowledge and enhanced computational thinking skills would be directly applicable to their professional endeavors.

If, as educators, we subscribe to the goal of universal computational thinking skills, these themes have implications for our instructional offerings. There are lessons for our current curricula, for new degree programs, and for educational opportunities beyond the walls of our traditional classrooms.

5.1 Current Curricula

The negative perceptions of traditional computing occupations expressed by our participants further reinforce calls for reforming the message we send to students and the greater public about what an education in computer science really means (see e.g., [7, 9, 18]). Even though these web developers chose a computing-related career, they avoided computer science because they perceived it to be about learning languages and writing code far removed from the user experience. However, their interests in interactivity and design *are* closely aligned with established areas of computer science like Human-Computer Interaction and Ubiquitous Computing. Unfortunately, all too often these aspects of computing are delayed until late in degree programs. If we are to appeal to those with similar interests, we must promulgate the true breadth of our field and find ways to incorporate these activities early and often in our curricula.

5.2 New Degree Programs

Our results suggest a need to build additional degree programs in digital media and design-related computation that

explicitly incorporate a core in computer science. Our typical computing classes are aimed at creating software development professionals [1]. Participants in this study were not directly interested in that end goal, but they have learning needs that our computing classes and expertise could address, nonetheless.

Among our participants who had obtained a degree related to web design, their course of study was completed in an art school, a liberal arts department, or a specialized technical college program. As computing educators, we have an opportunity to communicate the importance of a strong foundation in computer science through establishing interdisciplinary programs with allied departments like art and communications. Some universities have been successful in creating such programs in recent years [2, 15, 17], but they are still far from commonplace. Further, in developing such programs we must carefully consider which components of our computer science curriculum are appropriate and relevant. As a community, computing educators have a responsibility to take a leadership role in building innovative programs across campus which have computational thinking as a cornerstone.

5.3 Informal Learning

Lastly, and perhaps most difficult, is addressing the lack of educational resources for learners who are not in our classes and likely will never be there. Even if we are successful in making computational thinking fundamental at the elementary, secondary, and post-secondary levels, we fail to provide access for the vast population of people who learn about computing informally. Participants in this study noted difficulty in locating relevant information and in finding resources at the appropriate level for their understanding. As computing educators and computing education researchers, we must further examine the role that informal learning about technology plays in our society. How are people teaching themselves, and what computational knowledge might be beneficial? How do we design educational resources that convey deep conceptual information about computer science and also acknowledge people's interests and talents? In answering these questions we truly strive towards a vision of computational literacy that is for everyone, everywhere.

6. ACKNOWLEDGMENTS

The authors sincerely thank our 12 volunteers for participating in this study and sharing their stories with us.

This material is based upon work supported by the National Science Foundation under Grant Nos. 0613738 and 0618674. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

- [1] Computing curricula 2001. *Journal on Educational Resources in Computing*, 1(3es):1–240, 2001.
- [2] Computational media. Georgia Institute of Technology, 2008. Available at <http://lcc.gatech.edu/computedia/>, Accessed June 22, 2010.
- [3] O. Astrachan, S. Hambrusch, J. Peckham, and A. Settle. The present and future of computational

- thinking. In *SIGCSE '09: Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, pages 549–550, 2009.
- [4] J. Bransford, N. Vye, R. Stevens, P. Kuhl, D. Schwartz, P. Bell, A. Meltzoff, B. Barron, R. Pea, B. Reeves, J. Roschelle, and N. Sabelli. Learning theories and education: Toward a decade of synergy. In P. Alexander and P. Winne, editors, *Handbook of educational psychology, 2nd edition*. Erlbaum, Mahwah, NJ, 2005.
- [5] J. D. Bransford, A. L. Brown, and R. R. Cocking. *How People Learn: Brain, Mind, Experience, and School*. National Academy Press, Washington, D.C., expanded edition, 2000.
- [6] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006.
- [7] L. B. Cassel, A. McGettrick, M. Guzdial, and E. Roberts. The current crisis in computing: What are the real issues? In *SIGCSE '07: Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, pages 329–330, 2007.
- [8] P. J. Denning. The profession of IT beyond computational thinking. *Communications of the ACM*, 52(6):28–30, 2009.
- [9] P. J. Denning and A. McGettrick. Recentering computer science. *Communications of the ACM*, 48(11):15–19, 2005.
- [10] B. Dorn and M. Guzdial. Graphic designers who program as informal computer science learners. In *ICER '06: Proceedings of the 2nd International Workshop on Computing Education Research*, pages 127–134, 2006.
- [11] B. Dorn and M. Guzdial. Learning on the job: Characterizing the programming knowledge and learning strategies of web designers. In *CHI '2010: Proceedings of the 28th International Conference on Human Factors in Computing Systems*, pages 703–712, 2010.
- [12] J. Lave and E. Wenger. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, New York, NY, 1991.
- [13] A. McGettrick, E. Roberts, D. D. Garcia, and C. Stevenson. Rediscovering the passion, beauty, joy and awe: Making computing fun again. In *SIGCSE '08: Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, pages 217–218, 2008.
- [14] M. B. Rosson, J. Ballin, and J. Rode. Who, what, and how: A survey of informal and professional web developers. In *VL/HCC '05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 199–206, 2005.
- [15] C. Spradling, J. Strauch, and C. Warner. An interdisciplinary major emphasizing multimedia. In *SIGCSE '08: Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, pages 388–391, 2008.
- [16] J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.
- [17] Y.-L. Wong, J. Burg, and V. Strokhanova. Digital media in computer science curricula. In *SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, pages 427–431, 2004.
- [18] S. Yardi and A. Bruckman. What is computing?: Bridging the gap between teenagers’ perceptions and graduate students’ experiences. In *ICER '07: Proceedings of the Third International Workshop on Computing Education Research*, pages 39–50, 2007.