

Community-Based Scaffolding to Promote End-User Learning

Brian Dorn

*College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
dorn@cc.gatech.edu*

Abstract

End-user programming provides a unique opportunity to study informal computer science education and knowledge acquisition in the real world. We seek to explore the use of community-based scaffolding to deliver computing content knowledge to an audience of graphic design professionals who engage in scripting. We propose an online community that leverages prepared, instructional case studies alongside tools that promote collaborative development and code review activities.

1. Introduction

Computer science education research has tended to favor the study of formal learning experiences in relatively traditional classroom settings. This research is an attempt to strike out in a different direction: exploration of informal CS learning “in the wild” among user populations with little to no structured computing education. Specifically, we are interested in how end-user programmers come to learn the computer science that they must know to create the artifacts (scripts, macros, etc.) that they do. Further, we seek to explore how we can scaffold these users to go beyond learning syntactic constructs to learning about additional computing concepts, like data structures and software testing, which might be of use to them. Natural questions emerge:

- What do end-user programmers currently know about computer science?
- What additional concepts could they learn that would have immediate, tangible benefits?
- How can these concepts be “taught” in a way that matches end user practices?

We view the end-user programmer not simply as a user of software tools, but simultaneously as a learner

of computing concepts. Our goal is then to find mechanisms to deliver computing content of use to these learners as it becomes needed. Addressing this issue is an important step in bridging the gap between those who can and those who can not harness the full computational power of their technological devices.

2. Work to Date

Our work is contextualized in the domain of digital media manipulation and graphic design. Users of tools like Photoshop, Illustrator, and GIMP have access to scripting languages within their tools to aid in automation. In order to get a sense for the breadth of scripting among graphic designers, we distributed an online survey through several discussion boards [2]. Though we had fewer responses than expected, we did learn that graphic designers are taking part in substantial programming activities. They write programs of significant length and level of sophistication, despite having little to no formal training in computer science. As in other domains, we learned graphic designers program to save time on repetitive tasks and to add new, non-standard features to their applications.

To learn what they do, they report relying heavily on code examples and documentation like FAQs. They reported a high degree of familiarity with computer science terminology, and given the scope of their reported activities and habits (particularly, their propensity to share code) we have reason to believe that they could benefit from knowing even more about more formal code development practices. Their reliance on example code and their sharing practices also suggest a strong potential for delivering computing knowledge through online case-based learning aids (e.g., STABLE [4]) and in other types of community-based support.

We conducted a follow-on study analyzing the content of code contained in a popular Photoshop script-

ing community [3]. Given that our previous work indicated these users seek out example code from which to learn, the characterization of code provided by corpus analysis is a useful step in understanding the computing constructs end-user programmers are likely to encounter when seeking help, as well as an indication of those concepts that may need additional attention if we wish users to engage with them.

Our initial analysis of the computing constructs appearing in the Adobe Exchange repository revealed some common patterns as well as some unexpected ones. For example, we found that projects were far more likely to contain relational and mathematical operators than logical operators. Research on construct complexity with novice students (e.g. [6]) seems to match some of these patterns. Yet, some findings were counterintuitive. We found that projects contained use of exception handling mechanisms at the same rate as definite (for) loops.

We believe the constructs that were sparsely used indicate a possible need for additional examples or other types of instructional materials in communities aiming to provide a comprehensive resource. These topics included logical operators, advanced control structures (e.g., indefinite loops, nested structures, recursion), and higher-order code modularization techniques (e.g., user-defined objects, reusable code modules). As our goal is to further enable end-users to learn about computer science, these topics form the base of content which we will attempt to build into a case-based learning aid. Additionally, we aim to augment these with content related to other advanced computing forms like abstract data types (e.g., linked-lists, trees).

3. Proposed Work

We propose the development of a new online community targeted at graphic design end-user programmers that explores the following research questions:

1. How can nascent end-user on-line communities foster development of normative CS knowledge?
2. How can collaboration technologies enhance the dialog necessary for learning?

This new community will consist of two primary components: a case-library of example projects and a collaborative development forum.

The library portion of the site will contain a collection of pre-built projects that aim to uncover more advanced computing concepts, like those mentioned earlier. We intend to provide cases written in a verbose and conversational style (à la *literate programming* [5]) so that they serve as minimalist instruction materials [1].

In addition to the initial set of cases, the community will enable users to contribute their own projects. All cases will be indexed and cross referenced by the project goals and the computing concepts contained therein.

The second component of the community will be a collaborative forum space where users can work on projects. Currently, most end-user programming communities rely on threaded forum technologies. The cumbersome nature of collaboration and iterative code development in threaded forums often makes it difficult for end-user programmers to help one another. By employing more sophisticated mechanisms, like the edit paradigm of a wiki, we aim to alleviate these problems.

We intend to employ a mixed approach to studying the community we develop. A longitudinal evaluation of the site will focus on interaction patterns of real end-users and the ways they engage with cases: What cases are most commonly used? What is the nature of user contributed cases? Smaller-scale controlled experiments will be used to investigate specific aspects of our design, such as: Does the wiki-style collaboration lead to noticeably different outcomes as compared to a typical forum? What information seeking behaviors emerge when users are engaged in a new development project?

4. Acknowledgments

This material is based upon work supported in part by the NSF under grant ITR-SoD 0613738.

References

- [1] J. M. Carroll and M. B. Rosson. Cases as minimalist information. In *Proceedings of the 38th Hawaii International Conference on System Sciences*, 2005.
- [2] B. Dorn and M. Guzdial. Graphic designers who program as informal computer science learners. In *ICER '06: Proceedings of the Second International Computing Education Research Workshop*, pages 127–134, 2006.
- [3] B. Dorn, A. E. Tew, and M. Guzdial. Introductory computing construct use in an end-user programming community. In *VL/HCC'07: Proceedings of the 2007 IEEE Symposium on Visual Languages and Human-Centric Computing*, to appear.
- [4] M. Guzdial and C. Kehoe. Apprenticeship-based learning environments: A principled approach to providing software-realized scaffolding through hypermedia. *Journal of Interactive Learning Research*, 9(3/4):289–336, 1998.
- [5] D. E. Knuth. *Literate Programming*. Center for the Study of Language and Information, Stanford University, 1992.
- [6] J. F. Pane, C. Ratanamahatana, and B. A. Myers. Studying the language and structure in non-programmers' solutions to programming problems. *International Journal of Human-Computer Studies*, 54:237–264, 2001.