Invited Research Overview: End-User Programming

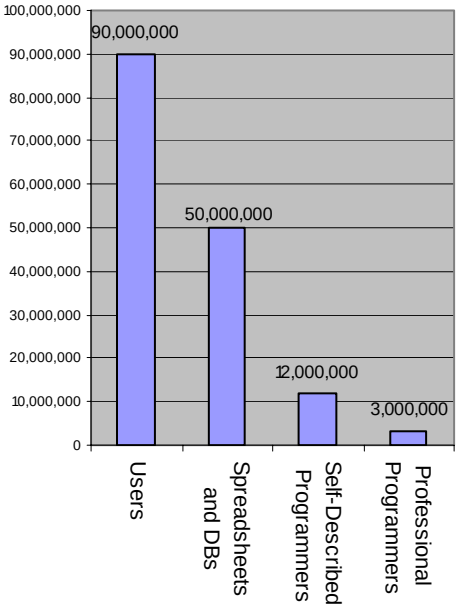
Brad A. Myers
Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
bam@cs.cmu.edu
http://www.cs.cmu.edu/~bam

Amy J. Ko
Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
ajko@cmu.edu
http://www.cs.cmu.edu/~ajko

Margaret M. Burnett
School of Elec. Engr. & Computer Science
Oregon State University
Corvallis, OR 97331
burnett@eecs.oregonstate.edu
http://web.engr.oregonstate.edu/~burnett/

CHI 2006, April 22–27, 2006, Montréal, Québec, Canada.
ACM 1-59593-298-4/06/0004.

Abstract
In the past few decades there has been considerable work on empowering end users to be able to write their own programs, and as a result, users are indeed doing so. In fact, we estimate that over 12 million people in American workplaces would say that they "do programming" at work, and almost 50 million people use spreadsheets or databases (and therefore may potentially program), compared to only 3 million professional programmers. The "programming" systems used by these end users include spreadsheet systems, web authoring tools, business process authoring tools such as Visual Basic, graphical languages for demonstrating the desired behavior of educational simulations, and even professional languages such as Java. The motivation for end-user programming is to have the computer be useful for each person's specific individual needs. While the empirical study of programming has been an HCI topic since the beginning the field, it is only recently that there has been a focus on the End-User Programmer as a separate class from novices who are assumed to be studying to be professional programmers. Another recent focus is on making end-user programming more reliable, using "End-User Software Engineering." This paper gives a brief summary of some current and past research in the area of End-User Programming.



Estimates for the number of people in the US in 2006 who use computers at work, who use spreadsheets at work, who describe themselves as programmers, and who say they are professional programmers [47].

Keywords

End-User Software Engineering, Natural Programming, Programming by Demonstration, Programming by Example, Visual Programming, Empirical Studies of Programmers (ESP), Psychology of Programming

ACM Classification Keywords

D.3.2 [Programming Languages]: Language Classifications; D.2.6 [Programming Environments]: Integrated environments; D.2.5. [Testing and Debugging]: Debugging aids; 1.3.6 [Computer Graphics]: Methodologies and Techniques-Languages.

End-User Programming: What and Why

One way to define "programming" is as the process of transforming a mental plan of desired actions for a computer into a representation that can be understood by the computer [16]. Expressed this way, it seems obvious that the study of humans and programming should be a topic of HCI. Indeed, this area of study has a long history, and has gone under many names, including "Software Psychology" [49], "Psychology of Programming" [7, 15] and the "Empirical Studies of Programming" (ESP), which is also the name of a series of eight workshops from 1986 through 1999.

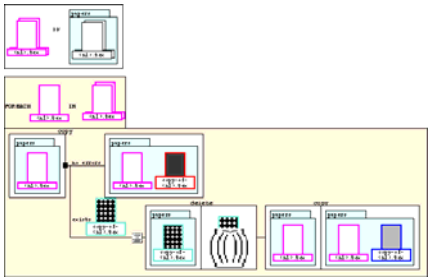
Most of the early work focused on studying professional programmers or novice programmers. A "professional" programmer might be defined as someone whose primary job function is to write or maintain software. A "novice" programmer might be defined as someone who is learning how to program.

In contrast, "end-user programmers" (EUP) are people who write programs, but *not* as their primary job function. Instead, they must write programs in support of achieving their main goal, which is something else, such as accounting, designing a web page, doing office work, scientific research, entertainment, etc. End-user programmers generally use special-purpose languages such as spreadsheet languages or web authoring scripts, but some EUPs, such as chemists or other scientists, may need to learn to use "regular" programming languages such as C or Java to achieve their programming goals.

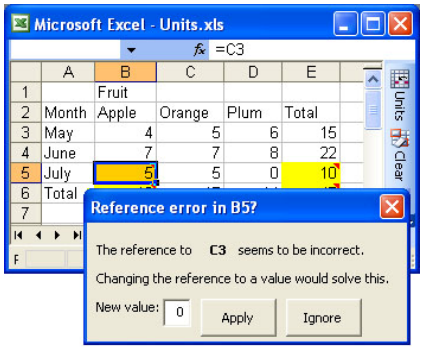
Why try to provide programming capabilities for everyone? The computer is a unique tool in its malleability—it can be programmed to perform almost any computation, but only by those who know how. Although prepackaged software applications come with increasingly complex functions, they still cannot do every task needed by every individual, and, in particular, cannot be customized to each individual's needs. Sometimes software designed or customized in a particular social context is so well situated in the community that uses it (called "situated software" [48]) that it provides formfit solutions for very particular needs, even though it might not have generally accepted notions of design quality or generality. Spreadsheets are a case in point: they have proven the enormous power of allowing individuals to create their own customized computations [35], and much EUP research aims to generalize spreadsheet's success to other domains.

Research in End-User Programming

Programming has always been recognized as a difficult task. This led to many research threads that tried to make programming more accessible by pushing on different aspects of computing technology.



The Pursuit system [32] uses PBD to create programs to manipulate files, which are then represented using a visual programming language.



Microsoft Excel spreadsheet augmented by the Ucheck system that tries to help the user find errors [1].

One such thread was motivated by a desire to harness the power of the human visual system. This work (starting as early as 1959! [13]) focused on using graphics to help make the programming easier, which is called "Visual Programming" (VP). Surveys of VP include [3, 6, 11, 33, 50]. Although at first, proponents of visual programming expected it to be something of a panacea, eventually, formal research (e.g., [12]) showed that every notation has advantages and disadvantages, and provided a vocabulary for comparing visual notations to textual alternatives.

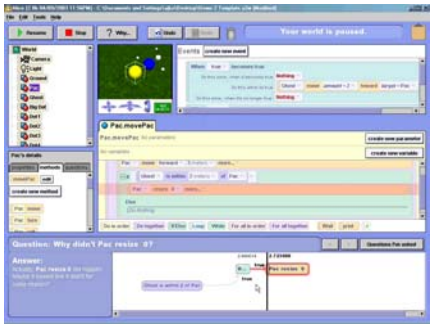
Another research thread has been to bring the advantages of direct manipulation to the programming task, by letting the user *demonstrate* the desired program by example by going through the steps, which is called "programming by example" (PBE), or "programming by demonstration" (PBD). Some of these systems use artificial intelligence techniques to try to automatically generalize the program from the user's examples. Although PBD systems have been shown to help people create programs from scratch, if people make errors or want to modify the resulting programs, there must be some static representation, for which some PBD systems have used visual languages. Surveys of PBE/PBD include [9, 26, 33], and notable research systems include [8, 14, 24, 25, 28, 30, 52]. PBD is also used by macro recorders in commercial spreadsheets and other "scriptable" applications.

Research has also focused on how to make the programming *environment* more supportive of programming, both to help with learning and to overcome various difficulties. For example, "syntax-directed editors" [29, 31, 41, 57] try to eliminate problems with syntax for textual languages. Surveys of programming environment research include [19, 36, 46]. Notable research in programming support environments for novice and end-user programmers include [41, 54]. Commercial systems include Apple's HyperCard, Microsoft's Visual Basic, and Adobe/Macromedia's products: Lingo for Director, Authorware, and Flash.

Another way to classify EUP systems is by the community or task for which they are aimed. For example Logo and its derivatives [39, 40] were designed for kids. Other kids' languages include [17, 36, 41, 44, 53] and the Lego Mindstorms commercial product. For scientists, there have been many "domain-specific languages" (DSL) [10] that have features specifically for particular areas of science. Other languages have been devised to enable specific tasks such as the authoring of software by teachers (e.g., Authorware and [58]), and end-user authoring of web pages [2, 4, 18, 27, 45].

A problem is that the programs from EUPs tend to be buggy and lack forethought in design. One thread of research considers this problem by focusing on fundamental issues that relate to people themselves, such as why programming is hard to learn and hard to perform [20, 22, 23, 42, 51, 55, 56], and how people think about programming concepts [37, 38, 47].

Another research thread that has received recent attention is the development of ways to make EUP software more reliable using concepts from software engineering, such as explicit support for detecting the presence of errors, tracking down bugs, and reuse [1, 5, 21, 43]. Two recent large collaborative efforts, one in the U. S. (the EUSES Consortium http://eusesconsortium.org/), and one in Europe (the Network of Excellence on End-



Whyline [21] showing a user asking why an event did not happen, and the resulting visualization.

MIDTERM	FINAL	COURSE	LETTER
91	86	88.4	B
94	92	92.6	A
80	75	77.4	C
90	86	86.6	B
89	89	93.45	A
88.8	85.6	87.69	

Forms/3 annotates cells of a spreadsheet to help the user test and debug the formulas and values [5].

User Development, http://giove.cnuce.cnr.it/eudnet.htm, which resulted in a new book [59]) have produced a number of promising results in this area.

Future Directions

In spite of all this research, programming is still out of reach of most people. It is still too difficult, and involves concepts such as abstraction, iteration, conditions, and recursion, that are foreign to people. Is it possible to make what we have called a "gentle-slope system" [34], where everyone can start programming with little effort, and learn incrementally as needed? Can the barriers to learning EUP systems be low enough so that the power of customizing the computations can be accessible to everyone? How can systems help the end-user programmer be more productive and produce more reliable code? Can artificial intelligence technologies be effectively applied to customize systems to do what users want? These and many other questions are open for future research.

Acknowledgements

The authors are supported in part by the National Science Foundation as part of the EUSES Consortium under NSF grants ITR CCR-0324770 and CCR-0325273. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

References

[1] Abraham, R. and Erwig, M. "Header and unit inference for spreadsheets through spatial analyses," in *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. Sept. 27-30, 2004. pp. 165-172.

[2] Bolin, M., *et al.* "Automation and Customization of Rendered Web Pages," in *ACM Conference on User Interface Software and Technology (UIST)*. October 23–27, 2005. Seattle, WA: pp. 163-172.

[3] Burnett, M., "Visual Programming," in *Encyclopedia of Electrical and Electronics Engineering*, J.G. Webster, Editor 1999, John WIley & Sons Inc.

[4] Burnett, M., Chekka, S., and Pandey, R. "FAR: An End-User Language to Support Cottage E-Services," in *Proc. Human-Centric Computing Languages and Environments.* Sept. 5-7, 2001. Stresa, Italy: pp. 195-202.

[5] Burnett, M., Cook, C., and Rothermel, G., "End-User Software Engineering." *CACM*, Sept, 2004. **47**(9): pp. 53-58.

[6] Burnett, M., Goldberg, A., and Lewis, T., *Visual Object-Oriented Programming: Concepts and Environments.* 1995, Prentice-Hall/Manning Publications.

[7] Curtis, B., "Fifteen Years of Psychology in Software Engineering: Individual Differences and Cognitive Science," in *Proceedings of the 7th International Conference on Software Engineering*, 1984, IEEE Computer Society Press. pp. 97-106.

[8] Cypher, A. "EAGER: Programming Repetitive Tasks by Example," in *CHI*. April, 1991. New Orleans, LA: pp. 33-39. Proceedings SIGCHI'91.

[9] Cypher, A., ed. *Watch What I Do: Programming by Demonstration*. 1993, MIT Press: Camb., MA.

[10] Deursen, A.v., Klint, P., and Visser, J., *Domain-Specific Languages: An Annotated Bibliography.* 1998. http://homepages.cwi.nl/~arie/papers/dslbib/.

[11] Glinert, E.P., ed. *Visual Programming Environments: Paradigms and Systems and Visual Programming Environments: Applications and Issues*. 1990, IEEE Computer Society Press: Los Alamitos, CA.

[12] Green, T.R.G. and Petre, M., "Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework." *Journal of Visual Languages and Computing*, 1996. **7**(2): pp. 131-174.

[13] Haibt, L.M. "A Program to Draw Multi-Level Flow Charts," in *Proceedings of the Western Joint Computer Conference.* March 3-5, 1959. San Francisco, CA: **15**. pp. 131-137.

[14] Halbert, D.C., *Programming by Example.* Computer Science Division, Dept. of EE&CS, University of California, 1984, Berkeley, CA. 84. PhD thesis. Also: Xerox Office

Systems Division, Systems Development Department, TR OSD-T8402, December, 1984. PhD thesis. Also: Xerox Office Systems Division, Systems Development Department, TR OSD-T8402, December, 1984.

[15] Hoc, J.-M., *et al.*, eds. *Psychology of Programming*. 1990, Academic Press: London.

[16] Hoc, J.-M. and Nguyen-Xuan, A., "Language Semantics, Mental Models and Analogy," in *Psychology of Programming*, J.-M. Hoc, *et al.*, Editors. 1990, Academic Press. London. pp. 139-156.

[17] Kahn, K., "ToonTalk -- An Animated Programming Environment for Children." *Journal of Visual Languages and Computing*, 1996. **7**(2): pp. 197-217.

[18] Kandogan, E., *et al.* "A1: end-user programming for web-based system administration," in *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology.* 2005. Seattle, WA: pp. 211-220.

[19] Kelleher, C. and Pausch, R., "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers." *ACM Comput. Surv.*, 2005. **37**(2): pp. 83-137.

[20] Ko, A.J. and Myers, B.A. "Development and Evaluation of a Model of Programming Errors," in *IEEE EUP/VL/HCC.* 2003. New Zealand: pp. 7-14.

[21] Ko, A.J. and Myers, B.A. "Designing the Whyline, A Debugging Interface for Asking Why and Why Not questions about Runtime Failures," in *CHI.* 2004. pp. 151-158.

[22] Ko, A.J., Myers, B.A., and Aung, H.H. "Six Learning Barriers in End-User Programming Systems," in *IEEE VL/HCC.* Sep 26-29, 2004. pp. 199-206.

[23] Lewis, C. and Olson, G. "Can principles of Cognition Lower the Barriers to Programming?" in *Empirical Studies of Programmers: Second Workshop.* 1987. Norwood, NJ: Ablex Publishing Corporation.

[24] Li, Y. and Landay, J.A. "Informal prototyping of continuous graphical interactions by demonstration," in *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology.* 2005. Seattle, WA: pp. 221-230.

[25] Lieberman, H. "Constructing Graphical User Interfaces by Example," in *Proceedings Graphics Interface.* May, 1982. Toronto, Ontario, Canada: pp. 295-302. GI'82.

[26] Lieberman, H., ed. *Your Wish is My Command.* 2001, Morgan Kaufmann: San Francisco.

[27] Lin, J., *et al.* "DENIM: finding a tighter fit between tools and practice for Web site design," in *CHI.* Apr 1-6, 2000. The Hague, The Netherlands: pp. 510-517. Proceedings CHI'2000.

[28] McDaniel, R.G. and Myers, B.A. "Getting More Out Of Programming-By-Demonstration," in *Proceedings CHI'99: Human Factors in Computing Systems.* May 15-20, 1999. Pittsburgh, PA: pp. 442-449.

[29] Miller, P., *et al.*, "Evolution of Novice Programming Environments: The Structure Editors of Carnegie Mellon University." *Interactive Learning Environments*, 1994. **4**(2): pp. 140-158.

[30] Miller, R.C. and Myers, B.A. "Interactive Simultaneous Editing of Multiple Text Regions," in *Proceedings of USENIX 2001 Annual Technical Conference.* June, 2001. Boston, MA: pp. 161-174.

[31] Minas, M. "Diagram Editing with Hypergraph Parser Support," in *1997 IEEE Symposium on Visual Languages (VL).* Sept. 23-26, 1997. Capri, Italy: pp. 226-233.

[32] Modugno, F. and Myers, B.A., "Visual Programming in a Visual Shell -- A Unified Approach." *Journal of Visual Languages and Computing*, 1997. **8**(5/6): pp. 276-308.

[33] Myers, B.A., "Taxonomies of Visual Programming and Program Visualization." *Journal of Visual Languages and Computing*, Mar, 1990. **1**(1): pp. 97-123.

[34] Myers, B.A., Smith, D.C., and Horn, B. "Report of the `End-User Programming' Working Group," in *Languages for Developing User Interfaces.* 1992. Boston, MA: Jones and Bartlett. pp. 343-366.

[35] Nardi, B.A., *A Small Matter of Programming: Perspectives on End User Computing.* 1993, Cambridge, MA: The MIT Press. 162.

[36] Pane, J.F. and Myers, B.A., *Usability Issues in the Design of Novice Programming Systems.* School of Computer Science Technical Report, Carnegie Mellon University, CMU-CS-96-132, August, 1996. Pittsburgh, PA. http://www.cs.cmu.edu/~pane/tr96/. Also appears as Carnegie Mellon University Human-Computer Interaction Institute Technical Report CMU-HCII-96-101.

[37] Pane, J.F. and Myers, B.A. "Tabular and Textual Methods for Selecting Objects from a Group," in *Proceedings of*

VL 2000: IEEE International Symposium on Visual Languages. September 10-13, 2000. Seattle, WA: IEEE Computer Society. pp. 157-164.

[38] Pane, J.F., Ratanamahatana, C.A., and Myers, B.A., "Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems." *International Journal of Human-Computer Studies*, February, 2001. **54**(2): pp. 237-264. http://www.cs.cmu.edu/~pane/IJHCS.html.

[39] Papert, S., *Teaching Children Thinking.* MIT, AI Memo No. 247 and Logo Memo No. 2, 1971. Cambridge, MA.

[40] Papert, S., *Mindstorms: Children, Computers, and Powerful Ideas.* 1980, New York: Basic Books. 230.

[41] Pausch, R., *et al.*, "Alice: A Rapid Prototyping System for 3D Graphics." *IEEE Computer Graphics and Applications*, 1995. **15**(3): pp. 8-11. May.

[42] Pea, R.D. and Kurland, D.M., "On the Cognitive Effects of Learning Computer Programming," in *Mirrors of Minds: Patterns of Experience in Educational Computing*, R.D. Pea and K. Sheingold, Editors. 1986, Ablex Publishing Corp. Norwood, NJ. pp. 147-177.

[43] Raz, O., Koopman, P., and Shaw, M. "Semantic Anomaly Detection in Online Data Sources," in *24th International Conference on Software Engineering (ICSE).* May 19-25, 2002. Orlando, FL: pp. 302-312.

[44] Resnick, M. and Silverman, B. "Some Reflections on Designing Construction Kits for Kids," in *Proceedings of Interaction Design and Children conference.* 2005. Boulder, CO:

[45] Rode, J. and Rosson, M.B. "Programming at Runtime: Requirements and paradigms for nonprogrammer web application development," in *IEEE Symposium on Human-Centric Computing Languages and Environments.* 2003.

[46] Rosson, M.B., Carroll, J.M., and Bellamy, R.K.E., "Smalltalk Scaffolding: A Case Study of Minimalist Instruction," in *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*, 1990, pp. 423-429.

[47] Scaffidi, C., Shaw, M., and Myers, B. "Estimating the Numbers of End Users and End User Programmers," in *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05).* 20-24 September, 2005. Dallas, Texas: pp. 207-214.

[48] Shirky, C., *Economics & Culture, Media & Community, Open Source.* March 30, 2004. http://www.shirky.com/writings/situated_software.html. (First published on the "Networks, Economics, and Culture" mailing list).

[49] Shneiderman, B., *Software Psychology: Human Factors in Computer and Information Systems.* 1980, Cambridge, MA: Winthrop Publishers.

[50] Shu, N.C., *Visual Programming.* 1988, New York: Van Nostrand Reinhold Company.

[51] Sime, M.E., Green, T.R.G., and Guest, D.J., "Scope Marking in Computer Conditionals: A Psychological Evaluation." *International Journal of Man-Machine Studies*, 1977. **9**: pp. 107-118.

[52] Smith, D.C., *Pygmalion: A Computer Program to Model and Stimulate Creative Thought.* 1977, Basel, Stuttgart: Birkhauser Verlag. PhD Thesis, Stanford University Computer Science Department, 1975.

[53] Smith, D.C., Cypher, A., and Spohrer, J., "KidSim: Programming Agents Without a Programming Language." *CACM*, Jul, 1994, 1994. **37**(7): pp. 54-67.

[54] Soloway, E., Guzdial, M., and Hay, K.E., "Learner-Centered Design: The Challenge for HCI in The 21st Century." *interactions*, 1994. **1**(2): pp. 36-48.

[55] Soloway, E., *et al.* "Learning Theory in Practice: Case Studies of Learner-Centered Design," in *Proceedings CHI'96: Human Factors in Computing Systems.* April 14-18, 1996. Vancouver, BC, Canada: pp. 189-196.

[56] Soloway, E. and Spohrer, J.C., eds. *Studying the Novice Programmer*. 1989, Lawrence Erlbaum Associates: Hillsdale, NJ.

[57] Teitelbaum, T. and Reps, T., "The Cornell Program Synthesizer: A Syntax-Directed Programming Environment." *CACM*, 1981. **24**(9): pp. 563-573.

[58] Wiedenbeck, S. "Facilitators and inhibitors of end-user development by teachers in a school environment," in *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).* Sept. 20-24, 2005. Dallas, TX: pp. 215-222.

[59] Wulf, V., Paterno, F., and Lieberman, H., eds. *End User Development*. 2006, Kluwer Academic Publishers.