Bringing Educational Theory to End-User Programming

Shreenivasarao Prabhakararao
Department of Electrical Engineering and Computer Science
Oregon State University, Corvallis, OR, 97331
prabhash@cs.orst.edu

INTRODUCTION

It is estimated by 2005 that there will be 55 million end-user programmers compared to 2.75 million professional programmers. These end users are specifically disadvantaged compared to professional programmers with respect to their motivation, background, interests and programming experience. They view software applications as a tool to help them solve their problems and regard computers "as a means to an end rather than objects of intrinsic interest " [13].

Recent years have seen the explosive growth of end-user programming. End-user programmers are writing an unprecedented number of programs, due in large part to the significant effort put forth to bring programming power to end users. Unfortunately, this effort has not been supplemented by a comparable effort to help them increase the correctness of these often-faulty programs. One of the most widely used real-world end-user programming paradigms is the spreadsheet. Despite its perceived simplicity, evidence from this paradigm reveals that end-user programs often contain an alarming number of faults [9].

To address this problem we plan to work towards providing the end user with *just enough fine grained information* they need to help them create correct and reliable spreadsheets. One means to facilitate such help is to develop an on-demand explanatory and advisory system that would encourage exploratory learning by the user and thus *effective use* of the system's features.

The focus of this research is to bring the principles of educational theory to end user programming to help end users build their skills that enhance their information seeking power and thus succeed in creating correct spreadsheets. We think this research would also benefit the educationally disadvantaged populations equally because the disadvantages (one specific disadvantage being lack of programming experience/background) faced by both these groups seem to overlap.

This work is part of a vision we call "End user software engineering". Our goal is to bring the benefits of software engineering research to end users without requiring them to learn the underlying software engineering theory and techniques. End-user software engineering is a highly integrated and an incremental concept of software engineering support for end users. Hence its components are not separate individual tools, but rather a blend of knowledge sources that come together seamlessly. A continually evolving prototype of end-user software engineering concepts exists for Forms/3 [1], a member of the spreadsheet paradigm. It is in this setting that we are planning to implement the concept of explanatory and advisory system that would draw heavily from the educational theory principles.

RELATED WORK

There is considerable research done in the area of interactive explanatory and advisory systems. Jussien and his colleagues have devised explanations to help find the causes of a program failing in a constraint-programming paradigm [2]. Our approach would differ from theirs based on the fact that our explanations while relevant to debugging has a primary goal of encouraging users to use the debugging features provided by our system.

Lieberman described interfaces that give and take advice [3]. Our approach would also help the end users by offering advice to them and suggesting actions to take, but we do not plan to make our system learn from the users' actions.

Carroll described about intelligent advisory interfaces that helps in training the users [4]. Unlike their approach our approach is to provide the users with just enough information to help them succeed in their task. Further more we draw guidance from the Blackwell's theory of Attention Investment [5]; hence we plan our system to be an *on-demand* explanatory and advisory system, which would offer explanation or advice when demanded but would not offer explanation or advice for unasked questions.

We draw heavily on HCI research. In addition to Blackwell's attention investment theory [5], Green et al.'s research on cognitive dimensions [6] and research related to psychology of curiosity [7] have been the major research that influences our work.

For this part of the work we plan to draw from the Carroll's minimalist theory of learning [8], which proposes a principle of encouraging users to get into action on meaningful tasks immediately in order to learn.

SURPRISE-EXPLAIN- REWARD

An underpinning of end-user software engineering is a strategy, which we refer to as surprise-explain-reward. The essence of this strategy is to arouse the user's curiosity through an element of surprise and then encourage them, through explanations and rewards to get the user into action [10].

The surprise-explain-reward strategy is a curiosity-centered approach. Research in curiosity indicates that surprising by violating user's assumptions can trigger a search for explanation [11]. The surprise element reveals to the users the presence of something they do not understand. This *information-gap* [7] motivates the user to explore in order to close the information gap.

EMPHASIS ON "EXPLANATION"

The implementation of the surprise element in our system is grounded in the research findings about the psychology of curiosity. In our strategy a feature that surprises the user must also inform the user. This is made possible by a low cost explanation system via tool tips. This present implementation of the explanation system is limited.

The focus of this work is to bring the principles of educational theory to end user programming. Faced with incomplete and conflicting information, users make assumptions and take actions on as-if basis [4]. Users form sub-goals using clues in the environment [12]. Guiding the actions of users with a built in explanatory and advisory system is intended to ameliorate their problems of maintaining correctness of their programs. Our goal is to span our explanation system over all the end-user software engineering devices; so as to further guide the user actions increase the correctness of their programs.

FUTURE DIRECTIONS

Presently, we are collecting data about what educational theories and what principles of these theories could be brought into end-user programming. We are also investigating the behaviors of population with varied backgrounds and varied levels of knowledge and experience working with end-user applications.

We plan to conduct some formative studies to direct our efforts in bringing educational theory to end-user programming. We also plan to conduct some experiments in future, to investigate the effectiveness of such an explanation and advisory system. It is our hope that this research will contribute to the integration of principles of educational theory into end-user programming.

REFERENCES

[1] M. Burnett, J. Atwood, R. Djang, H. Gottfried, J. Reichwein, and S. Yang, "Forms/3: A First-order Visual Language to Explore the Boundaries of the Spreadsheet Paradigm", J. Func. Prog. 11, 2, Mar. 2001, 155-206.

[2] Jussien, N. and Ouis, S., "User-friendly explanations for constraint programming", Proc. ICLP'01 11th Wkshp.Logic Programming Environments (Paphos ,Cyprus,Dec 2001).

[3] Lieberman, H. "Interfaces that give and take advice", Human-Computer Interaction for the New Millenium, J.Carroll, ed., ACM Press/Addison-Wesley, 475-485,2001.

[4] Carroll, J.M and Aaronson, A.P., "Learning by doing with simulated intelligent help." Communications of the ACM 31.9(1988): 1064-1079.

[5] Blackwell, A. and Green, T. R. G., "Investment of attention as an analytic approach to cognitive dimensions". In T. Green, R. Abdullah & P. Brna (Eds.) Collected Papers Wkshp.Psych. Of ProgrammingInterest Grp, 1999, 24-35

[6] Green, T.R.G and Petre,M., "Usability analysis of visual programming environments: a 'cognitive dimensions' framework", J. Visual Languages and computing 7, 2(June 1996) 131-174.

[7] Lowenstein, G., "The psychology of curiosity", Psychological Bulletin 116, 1 (1994), 75-98.

[8] John M. Carroll, "The Nurnberg Funnel: Designing minimalist instruction for practical computer skill", Cambridge, MA: MIT Press. Carroll, J.M. 1991.

[9] R. Panko, "What We Know about Spreadsheet Errors", J. End User Computing, spring 1998.

[10] Wilson, A., M. Burnett, L. Beckwith, O. Granatir, L. Casburn, C. Cook, M. Durham, G. Rothermel, "Harnessing Curiosity to Increase Correctness in End-User Programming," in Proc. CHI '03 (Ft. Lauderdale, FL, April 5-10 2003)

[11] Hastie, R., "Causes and effects of causal attribution", J.Personality and Social Psychology, 46, (1984), 44-56.

[12] Reimann, P. and Neubert, C., "The role of self explanation in learning to use a spreadsheet through examples", J. Computer Assisted Learning 16, (Dec.2000), 316-325.

[13] Nardi, B. and J. Miller, "Twinkling Lights and Nested Loops: Distributed Problem Solving and Spreadsheet Development," Int. J. Man-Machine Studies, 34 (1991), pp.161-184.