End Users Creating More Effective Software

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

Presented at:
Dagstuhl Seminar on End-User Software Engineering
http://www.dagstuhl.de/07081/
February 19, 2007

Abstract:

End-User Software is created using a variety of different techniques and paradigms. The "creating" part is defined as the process of representing the desired program in a computer-understandable form, and entering that representation into the computer. Programs can be represented using textual languages, visual (also called graphical) languages, spreadsheets (which are often included as a type of visual language), programming by example, or simple menu-based specifications. Textual languages range from general purpose languages such as Java, to special purpose languages such as StarLogo and MatLab for simulations and the HANDS language for kids, to "programming" using English as in Metafor. There is also research on how tools can make it easier to enter the text, such as the Alice syntax-directed editor. Visual Programming languages use 2D as part of the meaning of the language, and systems from members of the seminar include AgentSheets, Pursuit, Stagecast, StarLogo, and others. Programming by Example systems watch as the users perform tasks, and use AI (in particular, Machine Learning) to infer the program from the examples. Systems from members of the seminar include Eager, Robofox, Mondrian, and Gamut. Finally, there are some systems that allow limited customization of behaviors using menus and dialog boxes, such as simple parameter tweaking, specifying the behaviors of the characters in the Sims game, and drag-and-drop creation of dynamic web pages in CLICK. An orthogonal issue for software creation is the programming paradigm, such as imperative, event-based such as in Visual Basic and HANDS, functional such as in spreadsheets, and constraint programming as in CoDraw. One goal for EUP is to create what we call "gentle slope systems", where it is easy to get started ("low threshold"), sophisticated things can be done ("high ceilings"), and there are no barriers or walls where users must stop and learn many new things before making progress.

Dagstuhl Seminar Proceedings 07081
End-User Software Engineering
http://drops.dagstuhl.de/opus/volltexte/2007/1093