More Natural and Open User Interface Tools

Proposal to attend the ACM CHI 2005 Workshop on the Future of User Interface Design Tools

Brad A. Myers and Andy Ko
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{bam, ajko}@cs.cmu.edu
http://www.bam.hcii.cmu.edu/

ABSTRACT

Our research is highlighting some potential directions for the future of user interface design tools. One approach is to make the tools and their SDKs more usable, effective and understandable by making them more natural. Another is to take advantage of an "open data model" to more easily integrate new components. In addition, programming-by-demonstration techniques and model-based automatic generation still hold much promise.

INTRODUCTION

We would like to attend the CHI 2005 Workshop on The Future of User Interface Design Tools. Members of our group have created many user interface tools, including Garnet, Amulet and their many components, as well as a variety of handheld and model-based systems [7]. We have also created many tools that used programming-by-demonstration (PBD) [3], and participated in past "futures" discussions (e.g., [5, 6]).

In this paper, however, we would like to focus on two other aspects of our work with which the community may not be as familiar: Natural Programming and the preliminary work on an Open Data Model.

NATURAL PROGRAMMING

The goal of the Natural Programming research project [8] is to make it easier to write programs by taking HCI principles into account in the design of programming languages, programming environments, and software development kits (SDKs). Since programmers are people, it makes sense to utilize all of the available HCI techniques to improve the tools that programmers use. The first assignment in one of the first author's courses is for students to apply Nielsen's 10 heuristics to evaluate the UI and/or API for a UI tool. Students always generate long lists of problems that could be corrected. Of course, our own toolkits try to follow these HCI guidelines in their user interfaces (e.g., all the names in Amulet use a consistent naming scheme).

We go much further in the Natural Programming project, where we begin all new research with extensive user studies about how people *naturally* perform tasks, and then try to embody these new findings in our designs of new programming tools.

Our early Natural Programming research was focused on the design of a new language for children [11]. Since the goal of the environment was to make it particularly easy to create animated interactive software, the results are relevant to future tools in this area. As part of this research, we discovered that people often drew pictures to show the graphical parts of an interface, but preferred to use language to describe the dynamic behaviors. People generally used an event-based phrasing to discuss responses to actions (e.g., "when PacMan loses all his lives, it's game over") but a constraint-based phrasing was also sometimes used ("PacMan cannot go through a wall"). Aggregate operators (acting on multiple objects at once) were used much more often than iterating through a set and acting on each object individually (e.g., "Move everyone below the 5th place down by one"). Participants rarely used Boolean expressions, and were likely to make errors when they did (i.e., their expressions were not correct if interpreted according to the rules of Boolean logic). We invented a new format for entering Boolean operations that appeared to be more successful [10]. Based on all of these findings, we designed a new programming language, called HANDS, which a study showed could be used by children with no experience to create small programs [8]. This research can provides guidance on language design and environment structures for future user interface design tools so it will be easier for programmers to create graphical interactive programs.

Our current work in the Natural Programming project is focusing on the programming environment itself. We studied novices writing interactive programs in Visual Basic, and found that entering code took a small fraction of the time compared to debugging This was due to errors made while coding and poor debugging tools [2]. We are building new debugging and code construction tools to help address both of these problems. The WhyLine [1] helps with debugging by allowing programmers to ask *why* events did or did not happen in their program. For construction, the environment will help keep track of dependencies, necessary transformations, and other to-do items.

Another branch of research is studying how people learn new SDKs [12]. Internet resources, particularly Google, have emerged as new and effective tools that provide quick access to a large collection of tutorials and sample code.

However, we observed common difficulties in integrating example code and using search engines to find different ways to accomplish similar tasks. Preliminary findings suggest that help systems, documentation and tools can be improved to make the learning and use of SDKs substantially easier, which could influence future UI tools.

OPEN DATA MODEL

At the end of the Amulet project, it became apparent that one of the advantages of the Amulet toolkit programming model was that there was a well-defined and "open" description of all application data, which was available for other applications to inspect and modify. While this might seem to fly in the face of information hiding, we argued that there were substantial advantages to this "open data model" including: support for increased automation, extensive end-user customization capabilities, external agents and tutors, sophisticated search and replace, scripting and macros, alternative interfaces without re-implementing the application, plug-ins that operate in the same space, and significantly higher re-use of common code [4]. Now we are starting to see applications such as most of Microsoft Office do a reasonable job of exposing their internal data structures through COM interfaces. Similarly, with XML and the "semantic web," there is an increased interest in the concept of providing a standard, well-documented description of an application's internal data. Note that this is not the same as having the UI generated from an XML description, as in a number of projects, such as XIML, XAML, MXML, XUL, XUIML, PUC [9], etc. However, an application that supported both might also provide for external modification and control of the user interface elements, such as adding new visualizations to the Outlook calendar.

One place we have recently used a related concept is in the Citrine smart clipboard tool [13]. Citrine attempts to parse any data that is copied to the clipboard and places a standard XML description of what it recognizes onto the clipboard, thereby providing a structured interchange format between applications. This permits more intelligent copy-and-paste among applications.

OTHER WORK

The tremendous success of Interface Builders (also called resource editors) shows that at least part of a user interface is best created graphically. Our PBD research [3] showed that it is possible—but difficult—to expand what can be created to also support the *dynamic* behaviors of interfaces. We still think that PBD has potential, at least for the graphical parts of interfaces that are created dynamically.

Also promising is our work on model-based tools, where the user interface is automatically generated from a high-level description (model) [9].

CONCLUSION

Although new UI tools are clearly needed for new user interface styles and features (recognition-based input, multi-user, ubiquitous computing, etc.), the research discussed here focuses on fundamental issues about the design of the tools and toolkits themselves, which transcend the target of the tool. We expect research in all these areas to be important for improving future tools.

ACKNOWLEDGMENTS

Thanks to my many students who have contributed to these ideas. Support for this research has come from DARPA, NSF, and many companies.

REFERENCES

1. 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.
2. Ko, A.J., Myers, B.A., and Aung, H.H. "Six Learning Barriers in End-User Programming Systems," in *IEEE VL/HCC*. 2004. pp. 199-206.
3. Myers, B., McDaniel, R., and Wolber, D., "Programming by example: Intelligence in Demonstrational Interfaces." *Communications of the ACM*, 2000. **43**(3): pp. 82-89.
4. Myers, B.A., *The Case for an Open Data Model*. Carnegie Mellon University, School of Computer Science Technical Report, CMU-CS-98-153, August, 1998. http://reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-153.pdf.
5. Myers, B.A., Hollan, J., and Cruz, I., "Strategic Directions in Human Computer Interaction." *Computing Surveys*, 1996. **28**(4): pp. 794-809.
6. Myers, B.A., Hudson, S.E., and Pausch, R., "Past, Present and Future of User Interface Software Tools." *ACM Transactions on Computer Human Interaction*, 2000. **7**(1): pp. 3-28.
7. Myers, B.A., *et al*., "Taking Handheld Devices to the Next Level." *IEEE Computer*, 2004. **37**(12): pp. 36-43.
8. Myers, B.A., Pane, J.F., and Ko, A., "Natural Programming Languages and Environments." *Communications of the ACM*, 2004. **47**(9): pp. 47-52.
9. Nichols, J. and Faulring, A., "Automatic Interface Generation and Future User Interface Tools." *Submitted for Publication*, 2005.
10. 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*. 2000. Seattle, WA: IEEE Computer Society. pp. 157-164.
11. 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*, 2001. **54**(2): pp. 237-264.
12. Stylos, J. and Myers, B.A., "How Programmers Use Internet Resources to Aid Programming." *Submitted for Publication*, 2005.
13. Stylos, J., Myers, B.A., and Faulring, A. "Citrine: Providing Intelligent Copy-and-Paste," in *ACM Symposium on User Interface Software and Technology, UIST'04*. 2004. Santa Fe, NM: pp. 185-188.