What else should an HCI Pattern Language include ?

Yongmei Wu
Darmstadt University of Technology
Department of Computer Science
Programming Languages and Compilers
Tel: +49 (0)6151 164811 Fax: +49 (0)6151 166648
wu@pu.informatik.tu-darmstadt.de

ABSTRACT
Most present HCI design patterns and pattern languages are based on the traditional user interfaces, i.e., the graphical user interfaces. However, with growing popularity of computers, user interfaces for HCI are being integrated in more and more application domains. Today, large amount of user interfaces go beyond the desktop metaphor, which results in complexity and diversity in user interface design. To aid designing these user interfaces, I am involved in using and developing the related HCI design patterns and pattern languages. In this paper, through the practice of developing user interfaces for control a model robot, I conclude that application domain patterns are also critical to an HCI pattern language.

INTRODUCTION

The theory of design patterns and pattern languages has been proven being a successful vehicle for constructing "living buildings" (e.g., [1]) and guiding software design (e.g., [4]). User interface designers recently started to explore the utility of this theory in the field of HCI [3]. Several HCI design patterns and pattern languages were discovered (e.g., [10]). However, most of the existing HCI design patterns and pattern languages are based on the desktop user interfaces.
Today, with growing popularity of computers, user interfaces for HCI are being integrated in more and more application domains. Lots of user interfaces, e.g., user interfaces in applications of automation, control, and communication systems, go beyond the desktop metaphor. Developing these kinds of user interfaces is much more complex.
One of our HCI research work areas in the Chair Programming Languages and Compilers, Darmstadt University of Technology, is to explore using the pattern theory in our design practice, aims to develop design patterns and pattern languages for building user interfaces in the applications of automation, control and communication systems. The work presented here is just one of our examples.
In order to better explore the user interfaces in control system, I use a model robot in my research. In the development of user interfaces for controlling the robot, I have developed and used design patterns in three catalogues: robot control design patterns, interaction patterns, and software design patterns. I drew great inspiration from the experience: an HCI pattern language should organically integrate these three catalogues of design patterns into a structural network so as to help interface designers to build "positive" and "nurturing" user interfaces for the users.
In the following of this paper, section 2 introduces the design patterns developed and used in the development of user interfaces for controlling a robot. Section 3 discusses what else should be included in an HCI pattern language. Conclusion and future work is described in section 4.

DESIGN PATTERNS DEVELOPED AND USED IN DEVELOPMENT OF USER INTERFACES FOR CONTROLLING A ROBOT

To develop a user interface, we are usually confronted with the following forces:

1. In which application domain will the user interface be applied in?
2. By which interaction style will the users be supported interacting with the computer system?
3. What kind of interaction devices can be used to support the interaction style?
4. What kinds of expertise are needed for the development? How to allocate the design tasks to the experts with different technical background?
5. Which software architecture is efficient for support of the development? Which "fine-grain" object-oriented design patterns can be used to embellish the architecture?
6. How to achieve a quality user interface?

...

Both in architecture and software design, a design pattern describes a problem and the solution to release diverse forces. To conquer these forces, I struggled to use the pattern theory in the design. I have developed and used three catalogues of design patterns: robot control design patterns, interaction design patterns, and software design patterns.

Robot Control Design Pattern

An application domain pattern describes a recurring problem and the solution specific to an application domain. Robot control design pattern belongs to this kind.

The model robot is actually a robot arm attached with a gripper. We can move its arm forward, open and close its gripper for carrying an object from place to place by driving its motors with electrical signals. We also know the 3-D coordinate of its gripper by counting the impulses of its pulse switches. Although this model robot is relatively simple, it can be used to simulate robot arm control scenarios in general.

To interactively control the robot, "Select Me, Convey Me, Settle Me" (Appendix A) is used.

Interaction Design Pattern

To support "Select Me, Convey Me, Settle Me", I built a "Direct-manipulation user interface" LLDemo [9]. To provide dialogue possibility and interaction information, "Step-by-Step Instruction", "Status Display", "Control Panel", "Giving a Warning", "Interaction Feedback" [5][10] have been used.

With LLDemo, a user can control the robot with the mouse. Now I am exploring to use "Two-handed Manipulation" (Appendix B) for support of HCI.

Software Design Pattern

"Model-View-Controller" (MVC) [5] is a well known design pattern for building software architecture for user interface design. However, I observe that MVC has some pitfalls for developing user interfaces that go beyond the desktop metaphor, e.g., it doesn't deal with how to acquire continuous signals and drive an execution mechanism to fulfill the feedback in details. Instead of using MVC, I use "Acquisition-Computation-Execution-Expression" (ACEE) [11] design pattern to establish the software architecture. ACEE can also help to assign tasks to diverse experts. I integrate other "fine-grain" design patterns into ACEE in order to make it more concrete for building user interfaces for control applications. For example, use "Observer" [4] to build communication mechanisms within ACEE, use "Proactor" [7] for allocating complete asynchronous operations which includes mechanism for sampling data from the robot, use "Acceptor and Connector" [8] to accomplish establishing connection with the robot.

WHAT ELSE SHOULD A HCI PATTERN LANGUAGE INCLUDE?

Developing user interfaces, especially those going beyond the desktop metaphor, "Know the users" [6] is extremely critical. Our experience proves the importance of this point of view once more. To build an interface "affordance", "friendly", and "conviviality" to the users, we need deep knowledge in the related application domain. Only when we know the premises, e.g., in where and for what a user interface is used, can we decide what interaction styles are really necessary and how to implement the user interface to meet the demands.

Two major intentions of Alexander publication of his works (e.g., [1]) are to "make the environment nurturing to human beings" and to help inhabitants build their house, their street, and their town highly adapted to its particularities [2]. User interfaces, of course, should be both nurturing to the users and harmony to the environment. Therefore, in addition to interaction patterns and software patterns, we should also take efforts to discover application domain patterns. Only by doing so can we dig down and discover what interaction design patterns and software design patterns should be involved in an HCI pattern language and how to organically integrate them into a structural network.

CONCLUSIONS AND FUTURE WORK

Although we have some experience in using and developing HCI design patterns, we are still short of deep knowledge in the related application domains like robot control. We need further cooperation with domain experts to refine, expand and prove our design patterns. We are just at the start of the way to explore the utility of the pattern theory in HCI design. What we know just likes a drop of water in the ocean. To form a complete HCI pattern language, we have a very long way to go.

ACKNOWLEDGEMENT

I am grateful to my supervisor, Prof. Dr. Hans-Jürgen Hoffmann, for his support of my research work and many concrete guidance. All the ideas presented here are of the author's, permission is only granted to the CHI2000 Workshop on Pattern Languages for Interaction Design.

REFERENCES

1. C. Alexander et al., "A Pattern Language", Oxford University Press, 1997.
2. C. Alexander, J. O. Coplien, "The Origins of Pattern Theory", IEEE Software, Sept./Oct. 1999, pp 71-82.
3. E. Bayle et al., "Putting It All Together: Towards a Pattern Language for Interaction Design", SIGCHI Bulletin 30(1), pp 17-23, 1998, New York: ACM.
4. E. Gamma et al., "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley,1995.
5. http://www.it.bton.ac.uk/cil/usability/patterns: The Brighton Usability Pattern Collection.
6. Jakob Nielsen, "Usability Engineering", AP professional, Boston, MA, 1994:
7. I. Pyaral et al., "Proactor", PLoP'97, September 2-5, Illinois, USA.
8. D. Schmidt, "Acceptor and Connector", Pattern Languages of Program Design 3, Addison-Wesley Longman, Inc., 1998.
9. E. Siemon, Y. Wu: "On Techniques for Visual Task-oriented End User Programming", Technical Report, Darmstadt University of Technology, 1999.
10. J. Tidwell, "Interaction Design Patterns", http://www.mit.deu/~jtidwell/common-ground.html.
11. Y. Wu, "Acquisition-Computation-Execution-Expression (ACEE): A Software Architecture Pattern for Computer-supported Automation and Control Systems", PLoP'99, August 15-18, Illinois, USA.

Appendix A

Select Me, Convey Me, Settle Me

"Select Me, Convey Me, Settle Me" is a robotics domain pattern. It describes how to support a user to interactively control a robot arm to carry an object from place to place.

Name: Select Me, Convey Me, Settle Me

Example: In robotics domain, robot arms are often used. Usually, each arm is attached with a gripper. A user can move the arm forward, open and close its gripper for carrying an object from place to place, as shown in the following diagram, for example.

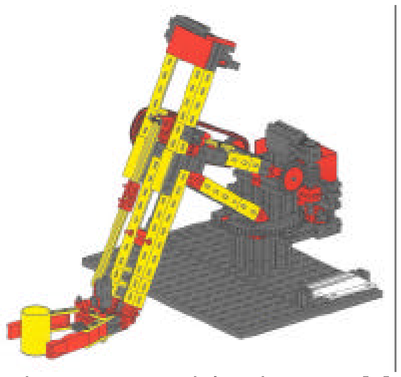

Diagram: A Model Robot Arm [2]

Forces: Using real world Teach-In methodology is accurate but a waste of time and short of flexibility. Although Off-Line Programming methodology improves these characteristics, it still lacks flexibility and intuitivity for control a robot.

Context: How to control a robot arm with a gripper to carry an object from place to place?

Solution: Create a virtual workplace, use "Visualize Application Movement Area" [4], for example. Reflect the real world objects, i.e., the arm, the gripper, the objects to be/being carried, and the surroundings of a robot, in the virtual workplace. Use the direct-manipulation devices like the mouse to Select an object, Convey it to the target position, then Settle it in the position, all in virtual workplace to achieve controlling a robot arm to carry an object from place to place.

Consequences: "Select Me, Convey Me, Settle Me" brings effectivity, flexibility and intuitivity for fulfilling a control task, it brings difficulty in construction of virtual workplace at the same time.

Known Uses: MARCO [1], LLDemo [3]

References:

1. B. Brunner, K. Landzettel, G. Schreiber, B.M. Steinmetz, G. Hirzinger: "A Universal Task-level Ground Control and Programming System for Space Robot Applications", i-SAIRAS 5th International Symposium On Artificial Intelligence, Robotics and Automation in Space, 1-3 June 1999, ESTEC, Noordwijk, The Netherlands.
2. "Experimentierbuch, Profi COMPUTING", *fischertechnik*, fischerwerke Artur Fischer GmbH & Co. KG.
3. E. Siemon, Y. Wu: "On Techniques for Visual Task-oriented End User Programming", Technical Report, Darmstadt University of Technology, 1999.
4. E. Siemon: "Layout Patterns for Application Specific User Interfaces", submitted to the CHI2000 Workshop on Pattern Languages for Interaction Design.

Yongmei Wu, MSc, Darmstadt University of Technology, Department of Computer Science, Programming Languages and Compilers
Tel: +49 (0)6151 16481, Fax: +49 (0)6151 166648, wu@pu.informatik.tu-darmstadt.de

Appendix B

Two-handed Manipulation

"Two-handed Manipulation" is an interaction pattern describes how to involve the non-dominant hand of a user in HCI.

Name: Two-handed Manipulation

Example: LLDemo [5] is a "Direct-manipulation User Interface", as shown in the following diagram, for controlling a model robot [2]. A user can move the robot arm forward by using the dominant hand, "press" the left button of the mouse on the *Gripper* and "drag" it in the *Movable Area*. During the manipulation of the dominant hand, the user can also use the non-dominant hand to "click" a two-switches input device to open and close the gripper.

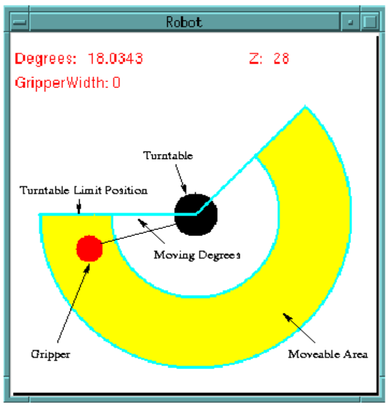

Diagram: LLDemo

Forces: Most user interfaces like Microsoft Windows only utilize the dominant hand of a user to manipulate the objects. However, people are used to work with two hands.

Context: How to involve the non-dominant hand of a user in HCI?

Solution: Let the non-dominant hand participate in manipulation. According to KC model [4],

- Allocate coarser action to the non-dominant hand while the dominant hand fulfills finer action.
- Assign the non-dominant hand to set the frame of the reference for the action of the dominant hand.
- Let the dominant hand involve in interaction at first. The non-dominant hand follows the dominant hand to carry out a task.

Consequences: Two-handed manipulation is more natural to human beings, it can also save time for fulfilling a task since two hands can manipulate the objects simultaneously. However, if the subtasks for the two hands are not appropriately allocated, two-handed manipulation is worse than one-handed since it may increase the cognitive load of a user.

Known Uses: Toolglass [1], "Two-handed spatial interface tools for neurosurgical planning"[3], LLDemo.

References:
1. E. A. Bier, M. C. Stone, K. Pier, W. Buxton, T. D. DeRose: "Toolglass and magic lenses: The see-through interface", Proceedings of SIGGRAPH'93, pp 73-80, ACM, New York.
2. "Experimentierbuch, Profi COMPUTING", ***fischertechnik***, fischerwerke Artur Fischer GmbH & Co. KG.
3. J. Goble, K. Hinckley, J. Snell, R. Pausch, N. Kassell: "Two-handed spatial interface tools for neurosurgical planning", IEEE Computer, Vol. 28, No. 7, July 1995, pp 20-26.
4. Y. Guiard: "Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model", Journal of Motor Behavior, Vol. 19, No. 4, 1987, pp 486-517.
5. E. Siemon, Y. Wu: "On Techniques for Visual Task-oriented End User Programming", Technical Report, Darmstadt University of Technology, 1999.

Yongmei Wu, MSc, Darmstadt University of Technology, Department of Computer Science, Programming Languages and Compilers
Tel: +49 (0)6151 16481, Fax: +49 (0)6151 166648, wu@pu.informatik.tu-darmstadt.de