Pattern Languages in Human-Computer Interaction (Suite Overview)

Jan O. Borchers
Telecooperation Research Group
Linz University
4040 Linz, Austria
+43 732 2468 9888
jan@tk.uni-linz.ac.at

ABSTRACT

The concept of pattern languages has found its way from urban architecture into software engineering quite successfully. HCI, however, has only started to explore the possibilities of this approach in recent years.

This paper briefly introduces pattern languages in general, and gives an overview of the state of the art of pattern languages in HCI. Due to its well-balanced mixture of formality and human-centered clarity, the pattern language approach promises a very suitable tool for communication within HCI, and between HCI and other disciplines.

Keywords

Design patterns, pattern languages, interaction design, HCI education

INTRODUCTION

CHI as a research discipline has come a long way. Other disciplines have begun to take its increasingly coherent body of rigorously formulated methodologies, amended by empirically proven design guidelines, very seriously. Methodologies like User-Centered Design begin to find their way into commercial software development, and standardization activities have begun, like the ISO standard 14915 on multimedia user interfaces.

The problem: Communication

This rich body of knowledge, however, often proves worthless when practitioners try to apply it to real-world projects, because it is not formulated in a way that helps the designer, or users involved in the design, to learn about the ideas behind the guidelines and checklists.

On the other hand, every user interface designer has experienced the problem of relating design guidelines and similar knowledge to developers: Most guidelines appear to them to be either self-evident, or too unspecific to follow, or of too limited applicability. Computer scientists are used to a formal, algorithmic approach with exact rules that design often simply cannot offer due to the imprecise nature and complexity of the factors involved. Both problems are essentially problems of language.

The solution: Pattern Languages

Using the idea of pattern languages can help to form a framework in which HCI knowledge can be expressed in a form that helps to communicate it to others in the HCI community, and even to relate it more easily to other members of an interdisciplinary team. Let us therefore take a brief look at the history of pattern languages.

PATTERN LANGUAGES IN ARCHITECTURE

The idea of pattern languages was originally described by architect Christopher Alexander in his books about urban architecture, mainly [1] and [2]. To create architectures that better fit and adapt to the needs of their users, Alexander suggests creating a language of *patterns* that capture the essence of successful architectural solutions to recurring problems which are described as systems of competing forces: "As an element in the world, each pattern is a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves. As an element of language, a pattern is an instruction, which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces, wherever the context makes it relevant." [2, p. 247].

A tpyical example of the more than 250 patterns Alexander describes in [1] is the *Street Cafe:* To solve the problem of creating an *Identifiable Neighborhood* with *Activity Nodes* and *Public Squares* (all these are names of other patterns defined in his text), the *Street Cafe* pattern can be applied: Several rooms of a cafe create a *Gradient of Privacy,* from a terrace with tables near the busy path, to more quiet rooms in the back. Newspapers in one corner make it an inviting place to pass some time, etc. The pattern solves a number of competing forces, e.g., of privacy and the desire to observe passers-by. It can be implemented using a number of smaller patterns, like *A Place to Wait, Sitting Wall,* and *Opening to the Street.*

Pattern languages in software design

The software engineering community was introduced to the Pattern concept in 1987, and it was in fact an HCI design case. Beck and Cunningham [6] built on Alexander's ideas to create a small pattern language capturing user interface design rules for Smalltalk applications, and presented their findings at an OOPSLA'87 conference workshop. Using their language of just five patterns, several application domain experts were able to design their own user interfaces based on Smalltalk mechanisms without any prior knowledge of the language.

These essentially didactic possibilities may have been one of the major reasons why the software engineering

community took up the idea of design patterns so eagerly, especially after Gamma, Helm, Johnson and Vlissides (frequently referred to as the *Gang of Four)* published their seminal book describing a comprehensive pattern language for object-oriented software design [9]: It offered software engineering a new and practical way to communicate software design experience.

Since then, a software design pattern is generally considered to be a proven solution of a recurring software engineering problem that balances the competing design constraints optimally for a certain type of situation.

In software design, as in architecture, patterns need to have certain qualities to distuingish them from other, simpler ways of describing problem solutions: They must

- Carry a *name* that is as self-explanatory as possible,
- clearly define the *context* in which they can be applied,
- describe a *proven* solution to a recurring problem,
- give *examples* where they have been applied,
- be flexible and adaptive enough to *generate* solutions in varying contexts,
- and *reference* other patterns to use synergistically, in order to create a solution as a whole of high quality.

That last item is especially important: Only when a comprehensive collection of patterns has been structured and interlinked in this way, it can become a *pattern language.*

PATTERNS IN HCI

At first sight, it seems that the pattern idea has been picked up by the HCI community only recently, especially since the CHI'97 workshop [5]. Surprisingly, however, Alexander's ideas have been referenced by major HCI works much earlier. In 1988, Donald Norman, in his 'POET' book which is a standard text in many HCI courses [10], cited Alexander as a designer whose work had particularly influenced him. Another HCI 'classic', Apple's *Human Interface Guidelines* [3], quotes Alexander's *A Pattern Language* as a seminal book about environmental design.

A more detailed adaption of Alexander's ideas was done by by the Utrecht School of Arts. In an overview of their interaction design curriculum, they outlined how they had adapted Alexander's approach to interaction design, "…using patterns to phrase guidelines in a consistent format that leaves room for subtleties." [4].

The CHI'97 workshop on Pattern Languages for Interaction Design [5] was an important event for the issue of pattern languages in HCI. It collected quite different opinions on how to carry the patterns idea over to HCI, for example from using patterns to describe observed user activities without judgment, to using them to capture HCI design practice. But in particular, the participants agreed that they "… felt we were at the very beginning of the enterprise of understanding the role and utility of pattern languages for interaction design." [5].

Within the last two years, the field has gained in momentum, and a number of proposals for pattern languages of HCI design issues have been published. A prominent example is J. Tidwell's pattern language for interaction design [11] that, in its present form, covers a substantial area of user interface design, but others have been presented, especially at the various PloP Conferences on Pattern Languages of Programming. A useful set of resources about interaction design patterns can be found at the Interaction Design Patterns Home Page, maintained by T. Erickson [7], who recently presented his ideas of pattern languages as a *lingua franca* for interaction design [8].

ABOUT THE PAPER SUITE

We have tried in this collection to capture the diversity of applications that the idea of pattern languages has produced in the HCI domain since then. Since patterns themselves by definition are nothing "original", but rather capture solutions that have proven themselves useful, and are therefore not strictly new findings, our focus is more on the question how the pattern approach can be applied to HCI research, practice, and teaching. The themes reach from fundamental questions about the genre of patterns, to their use in teaching, workplace environment design, and communication in interdisciplinary design teams.

Our common opinion is that pattern languages can give HCI a tool with the right degree of formality to capture experience, guidelines, methods, and values in human-computer interaction design. This collaboration has already established a fruitful exchange on this subject, and we hope to continue and expand it to a broader circle.

REFERENCES

1. Alexander, C. *A Pattern Language: Towns, Buildings, Construction*. Oxford Univ. Press, Oxford, UK, 1977.
2. Alexander, C. *The Timeless Way of Building.* Oxford University Press, Oxford, UK, 1979.
3. Apple Computer. *Macintosh Human Interface Guidelines.* Apple Computer, Inc., 1992.
4. Barfield, L. et al: Interaction Design at the Utrecht School of the Arts. *SIGCHI Bull.* **26**(3), 1994, 49-79.
5. Bayle, E. et al. Toward a Pattern Language for Interaction Design. *SIGCHI Bull.***30**(1), 1998, 17-23.
6. Beck, K., and Cunningham, W. Using pattern languages for object-oriented programs. *Technical Report CR-87-43,* Tektronix, Inc., September 17, 1987.
7. Erickson, T. *Interaction Design Patterns Home Page.* Established February 1998. http://www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html
8. Erickson, T. Interaction Pattern Languages: A *Lingua Franca* for Interaction Design? *Usability Professionals' Assoc. Conf.*, Washington DC, June 1998. See [7].
9. Gamma, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software.* Add.-Wesley 1995
10. Norman, D. *The Psychology of Everyday Things.* Basic Books, New York, 1988, 229;238.
11. Tidwell, J. Interaction Design Patterns. *PloP'98,* Illinois, Aug.'98. http://jerry.cs.uiuc.edu/~plop/plop98