AI & Soc (2001) 15:377–385

AI & SOCIETY

Open Forum

Usability Pattern Language: Creating A Community[1]

Jan O. Borchers[1], Sally Fincher[2], Richard Griffiths,[3]
Lyn Pemberton[3] and Elke Siemon[4]
[1]*Computer Science Department, Stanford University, Stanford, California, USA*
[2]*Computing Laboratory, University of Kent, Canterbury, UK*
[3]*School of Information Management, University of Brighton, Brighton, UK*
[4]*Department of Computer Science, Technische Universität Darmstadt, Darmstadt, Germany*

1. Introduction

This report summarises the results of a workshop on HCI design pattern languages at the INTERACT'99 International Conference on Human–Computer Interaction in Edinburgh in August 1999. It includes descriptions of the participants' position papers and submitted patterns, a definition of HCI design pattern languages, several organisational principles for such patterns and an example of what an actual HCI design pattern might look like.

2. Workshop History

This workshop followed a series of earlier events dealing with human–computer interaction (HCI) design patterns in which many of the workshop participants had been involved. A first workshop at CHI'97 (Bayle et al., 1998) initiated the discussion about pattern languages for interaction design in the HCI community. A more concrete definition of HCI design pattern languages, and a taxonomy of HCI design patterns, were developed during a workshop on interaction pattern languages at ChiliPLoP'99:

[1] This article is an Appendix to J.O. Borchers: A Pattern Approach to Interaction Design.

An Interaction Pattern Language generates space/time interaction designs that create a system close to the user's mental model of the task at hand, to make the human computer interface as transparent as possible (Borchers, 2000).

The present workshop took this definition as a starting point. It turned out, however, that people from different disciplines still had very different ideas of what an HCI design pattern should be. The workshop aimed at bringing together those researchers, especially from interaction design and software engineering, to 'create a community' of HCI design pattern people with a shared understanding.

For an introduction to the concept of design patterns in general, please see the article to which this report is appended. In addition, we recommend the original work on patterns (Alexander, 1979; Alexander et al., 1977).

3. Workshop Topics

The initial list of topics to be discussed at the workshop included:

- defining HCI design patterns, and HCI design pattern languages, in the context of an interdisciplinary workshop;
- finding, documenting, maintaining, disseminating and teaching HCI design patterns;
- applying HCI design patterns, and evaluating patterns and the results of their application;
- comparing formal versus informal pattern representations, and single patterns versus pattern languages;
- using tasks to organise patterns;
- using HCI design patterns to bridge interdisciplinary gaps;
- examining existing case studies and examples of HCI design patterns, and comparing HCI design patterns to existing formats such as user interface design guidelines.

Not all of those issues were addressed in detail. The following sections give our central findings.

4. Position Papers

At the beginning of the workshop, the participants briefly outlined their positions in relation to HCI design patterns. These positions are summarised below to convey an idea of the scope of the workshop; the complete position papers can be found online (see 'Further Information').

Jan O. Borchers (Stanford University) pointed out that Alexander's idea of user participation has not carried over to software engineering, and that patterns are more suitable for HCI than for software engineering. He described how to use patterns in software engineering, HCI and the application domain to improve interdisciplinary communication in the design team (Borchers, 2001), and how HCI design patterns were used in teaching.

David England (Liverpool John Moores University) used patterns to express solutions for temporal aspects of a user interface. His goal was to bridge the gap between HCI and software engineering, especially for the design of virtual worlds.

Sally Fincher (University of Kent at Canterbury) identified Capture of Practice, Abstraction, Organising Principle and Value System as the four major elements that characterise a pattern language. Presentation is only a minor element in this context.

Richard Griffiths (University of Brighton) stated that the idea of HCI design patterns is much more important than any particular HCI design pattern language, as it acknowledges the fact that HCI design must resolve socio-technological forces with interlocking high- and low-level design issues. Therefore, and since architecture is a much older discipline than HCI, any particular HCI design patterns should still be considered tentative.

Ger Koelman (Hollandse Signaalapparaten B.V.) stated that there are many striking analogies between Alexandrian patterns for architecture and issues typically discussed in user interface design. Examples are the hierarchical structure of patterns, and their abstraction level (from form-giving to beautification).

Diane Love (Lockheed Martin) emphasised that Alexander thought of patterns as the rules that indirectly control a design or growth process; some of his ideas anticipated fractal theory. People are excellent at naming and recognising patterns, and could navigate a design problem space especially well with patterns in hypertext form.

Fernando Lyardet, *Gustavo Rossi* (Universidad Nacional de La Plata) and *Daniel Schwabe* (PUC Rio de Janeiro) presented a set of user interface design patterns for navigation design in object-oriented hypermedia applications. Their goal is to separate navigation from user interface design, facilitating different interfaces for the same navigational structure.

Michael J. Mahemoff (University of Melbourne) introduced his work on domain-specific pattern languages for internationalised software and safety-critical systems. His patterns are often drawn from very similar systems, and may fit together in a more cohesive way than more broadly applicable patterns, coming closer to Alexander's idea of a language of tightly combined patterns.

Barbara Mirel (Lucent Technologies) presented her work on a pattern language for visual querying and analysis in retail category management and marketing. The patterns range from user task down to UI technology, with the goal of capturing design principles for software that supports work in this domain especially well.

Kimberly Perzel and *David Kane* (SRA International) submitted a position paper outlining their pattern collection for web applications. Their patterns address the fact that the web, with its increasingly interactive sites and applications, is not just a new technology but also creates new media and socio-economic dynamics.

Elke Siemon (Technische Universität Darmstadt) presented an example of a pattern covering both HCI and software engineering aspects. It showed the difficulties of writing patterns for an interdisciplinary team that still separate the design tasks successfully, as between architecture and civil engineering.

John C. Thomas (IBM), who also co-chaired the CHI'97 workshop, applied Christopher Alexander's 15 fundamental properties of good design to the field of HCI design. These properties are at the heart of Book One of Alexander's upcoming work, *The Nature of Order* (Alexander, 2002).

Janet Louise Wesson (University of Port Elizabeth) outlined her UNION methodology for user-centred software design, and explained how its user interface (UI) design heuristics could be replaced by a pattern language. UI designers would

use these patterns to map attributes in the UI model to suitable interaction objects in a given domain.

Peter Windsor (Usability Ltd) set forward a distinction between data- and task-oriented HCI design patterns, and suggested adding human and system costs and benefits to each pattern. He doubted that patterns would be appropriate to teach novice designers without first providing basic knowledge about how to deal with users and tasks.

5. A Definition of HCI Design Pattern Languages

This issue was revisited several times throughout the workshop. During the workshop, there was some agreement that pattern languages are certainly more than a value-free collection of guidelines for design. Alexander (1979: 283) gives an example of a MADHOUSE BALCONY pattern by one of his students that solves the conflict of mental patients who enjoy having a view from their rooms, but who also have 'a tendency to jump off buildings'. The suggested solution are balconies around the patient rooms with chest-high railings around them. This pattern has all the formal properties of a pattern, yet it is not a pattern because it does not improve the environment – according to a certain (in this case Alexander's) value system: such a balcony will not help mental patients to heal themselves. We finally decided to give a 'user-centred' definition that describes HCI design pattern languages by first defining what they can be used for:

> The goals of HCI design pattern language are to share successful HCI design solutions among HCI professionals, and to provide a common language for HCI design to anyone involved in the design, development, evaluation, or use of interactive systems.

Instead of going into more detail in the definition, the workshop later agreed on a 'typical' HCI design pattern as an example (see below).

6. Organising Principles

The goal of organising patterns is to support an iterative design process. Alexander's patterns, for example, are sorted by geographical scope, and designers can begin at large-scale issues and subsequently unfold their design, adding smaller-scale refinements to it. Several organisation principles were discussed during the workshop, reflecting viewpoints from different disciplines. The participants largely agreed upon two fundamental schemes.

The most natural organising principle for HCI design patterns is according to the same notion of scale, although the dynamics of user interfaces require incorporating time as well as space into such a classification. Building on an initial classification (Borchers, 2000), HCI design patterns should usually fall into one of the categories of the following scheme:

- society (beyond systems);
- multiple users;
- social position;
- system;

- application;
- UI structure (dialogue);
- components (containers, windows, layout);
- primitives (buttons and other simple widgets);
- physical properties.

An interesting observation is that the real world of technology changes fastest in the middle area of this scale, between system and components. Since patterns are by definition supposed to capture some timeless quality, it may be particularly difficult to write genuine patterns for this area. At the same time, many existing guidelines – and many of the submitted patterns – address exactly this middle level, which may render them obsolete rather quickly.

The second fundamental organising principle is to follow the HCI design process, leading from analysis- to structure-oriented patterns:

- The highest level is *culture and society*, followed by *environment* and *role* of the user.
- The next levels are *use* and *navigation* (incorporating affordances and, for example, issues of safety versus exploration).
- *Structural* levels follow those analysis-oriented ones.
- As an example, *tasks* can be further classified into retrieval, monitoring, proactive and reactive controlling, construction (writing a document), transactions, modifications (changing contents or structure), calculation, workflow and communication.

7. Submitted Patterns

We classified the submissions according to scale to show the validity of such organising principles, although this 'bucket-sorting' is of course by no means unequivocal.

An example of patterns at the *Multiple Users* level is Lyn Pemberton's LET PEOPLE OVERHEAR, which addresses issues of cooperative work.

Patterns at the *System* level include Ger Koelman's EXPLORABLE INTERFACE and Jan Borchers's INCREMENTAL REVEALING, which both deal with the overall impression that a system conveys.

Patterns that deal with how to design individual *Applications* include Lyn Pemberton's JUST THE USUAL about application vocabulary, and Peter Windsor's SITUATION DISPLAY and WORK QUEUE. They address the overall appearance for a certain type of applications.

Most of our patterns addressed the *Dialogue Level*, such as David England's AVATAR JOINING for temporal UI aspects of virtual worlds, Diane Love's selection patterns CHOICE OF SUBLIST FROM A LARGE LIST and CONFIGURING ORDERED SLOTS IN HARDWARE, and Fernando Lyardet et al.'s INFORMATION ON DEMAND, BEHAVIOUR ANTICIPATION, INFORMATION–INTERACTION DECOUPLING and BEHAVIOURAL GROUPING patterns for navigation. Richard Griffiths's GIVE A WARNING also falls into this category.

Patterns describing solutions at the *Components Level* include Barbara Mirel's INFORMATION FLOW FOR PRECISION for data visualisation, and Janet Louise Wesson's LIST SELECTION pattern.

The level of *Primitives* is covered by patterns such as Martin Hitz's SELECTION ITERATOR, which also contains implementation recommendations.

The *Physical Properties* level contains patterns such as John Thomas's SUPPORT THE HANDS WITH SPECIALISED TOOLS.

Fernando Lyardet et al.'s pattern INFORMATION ON DEMAND was also classified according to the *Design Process Categories* outlined above. As a retrieval pattern, it was put at the *Task Level* of this categorisation.

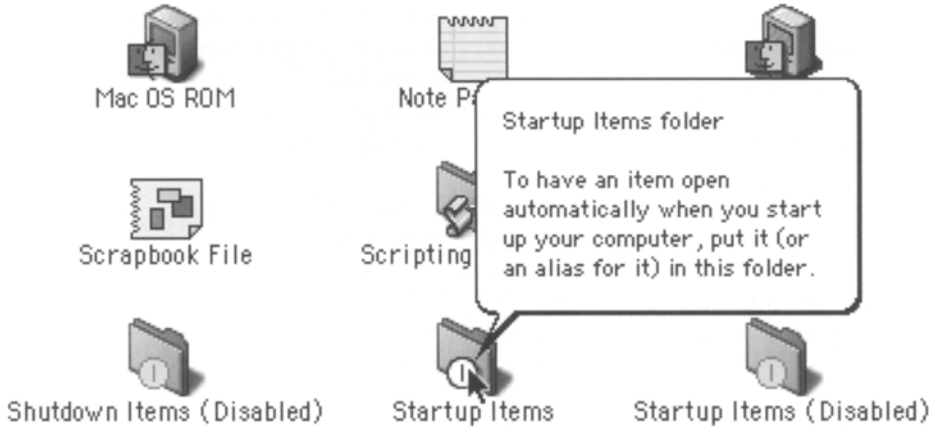
Some patterns do not address design issues of interactive systems, but rather aspects of the software development process such systems require, and therefore do not fall easily into the above categorisation. Examples include Michael Mahemoff's ONLINE REPOSITORY for developing internationalised software.

8. A Sample HCI Design Pattern

To give an idea of what HCI patterns could look like, the participants agreed on the form and contents of a 'typical' HCI design pattern, DESCRIPTION AT YOUR FINGERTIPS.

It captures the idea of adding temporary information to objects in the user interface, delivering short explanations without permanently cluttering the interface space. (Pattern constituents are labelled in square brackets.)

[Pattern Title:] DESCRIPTION AT YOUR FINGERTIPS

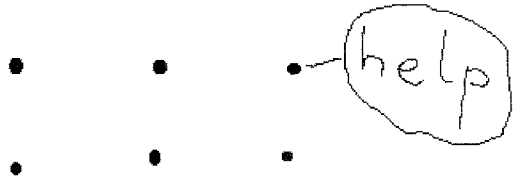


Sensitising example.

[Problem Statement:] You are putting interactive objects on a dynamic medium such as a screen and you want to provide various levels of context-sensitive help supporting uninterruptable tasks. Extensive explanations tend to clutter the interface, 'Problem Statement', but users may need such help. They do not want to leave the context of their current task, and experts may not want to see the help at all.

[Existing Examples:] In the Mac OS, a small balloon of textual help appears when the user turns on this feature and moves the mouse over an object. In Windows Tool Tips the same happens if the mouse hovers over an object. In Netscape, the URL of a link is displayed in a fixed position at the bottom of the screen if the cursor is moved over it. In a voice mailbox, options are explained if the user waits for a while.

[Formation of a General Solution:] Therefore, provide a short description of the object either close to it or in a fixed position, and let users turn it on and off or only provide it on some explicit user action (e.g., hovering).



Schematic.

Alternative representation:

On <start trigger>

give <description>

at <location>

On <end trigger>

extinguish <description>

[Reference to Constituent/Related Patterns:] You can use THREE-STATE BUTTONS to implement descriptions like this. Longer explanations can go into ON-LINE HELP, possibly delivered via an INTELLIGENT AGENT, or in the MANUAL.

The overall structure of this pattern follows the Alexandrian format quite closely. It is amended by an alternative, pseudo-code representation of the concept which is more suitable for representing the pattern dynamics over time. A similar pattern has been described by Tidwell (1998) under the name SHORT DESCRIPTION.

9. Other Workshop Activities

Part of the workshop was spent in Writers' Workshops (Borchers, 2000) on some of the submitted patterns. This turned out to be a very rewarding activity, for the reviewers as well as for the authors.

10. Next Steps

The next major event in this field was the workshop *Pattern Languages for Interaction Design: Building Momentum*, organised by Richard Griffiths, Lyn Pemberton, Adam Stork and Jan Borchers, held on 2 and 3 April 2000, during CHI 2000 in The Hague, The Netherlands. Subsequently, Alistair Sutcliffe, Adam Stork and Phil Gray organised a workshop on *Patterns in Human–Computer Interaction* for the British Computer Society HCI Group and IFIP Working Group 13.2., held in London on 16 November 2000, where an IFIP Task Group on HCI Patterns was formed. Jan Borchers and John Thomas organised a panel, *Patterns–What's In It For HCI?*, at CHI 2001 (Seattle, 31 March–5 April 2001), while a panel on *Patterns in Human-Computer Interaction Design* was arranged by Richard Griffiths and Lyn Pemberton for the IHM/HCI2001 Conference in Lille in September 2001. Reports are in preparation.

11. Further Information

See http://www.hcipatterns.org/, the HCI Patterns Pages, for information on HCI Patterns in general, as well as publications, ongoing activities of the IFIP HCI Patterns Task Group, and upcoming events in this field.

12. The Workshop Organisers

Dr Jan Borchers is acting assistant professor of computer science at Stanford University. He works on user interfaces for new media, and has designed a series of computer-based interactive exhibits. He developed an interdisciplinary pattern-based approach to interactive systems design that helps UI designers, software engineers and application domain experts talk to each other, which became the first book on HCI patterns (Borchers, 2001).

Richard Griffiths is a senior lecturer at the University of Brighton, School of Computing and Mathematical Sciences, where he teaches undergraduate and masters-level courses in HCI design. He has industrial experience in both information systems and system software design and implementation, has previously and is currently supervising technology transfer programmes in the multimedia industry, and undertakes consultancy in HCI design.

Dr Lyn Pemberton is a principal lecturer at the University of Brighton, School of Information Management, where she teaches courses in user-centred system design, creative design and computer-mediated communication. She has carried out requirements analysis, interaction design and evaluation on a range of projects funded by British Telecom, the DTI, the European Commission and the US Office of Naval Research.

13. Workshop Participants

- Jan O. Borchers, Stanford University
- David England, Liverpool John Moores University
- Sally Fincher, University of Kent at Canterbury
- Richard Griffiths, University of Brighton
- Martin Hitz, Universität Klagenfurt
- Ger Koelman, Hollandse Signaalapparaten B.V.
- Diane Love, Lockheed Martin
- Fernando Lyardet, Universidad Nacional de La Plata
- Michael J. Mahemoff, University of Melbourne
- Barbara Mirel, Scient
- Lyn Pemberton, University of Brighton
- Elke Siemon, Technische Universität Darmstadt
- Adam Stork, University College London (2nd day)
- Alistair Sutcliffe, UMIST (2nd day)
- John C. Thomas, IBM
- Janet L. Wesson, University of Port Elizabeth
- Peter Windsor, Usability Ltd

References

Alexander, C. (1979). The Timeless Way of Building. Oxford University Press, Oxford.
Alexander, C. (2002). The Nature Of Order. Oxford University Press, Oxford (in press).
Alexander, C., Ishikawa, S., Silverstein, M., et al. (1977). A Pattern Language: Towns, Buildings, Construction. Oxford University Press, Oxford.
Bayle, E., Bellamy, R., Casaday, G. et al. (1998). Putting it all Together: Towards a Pattern Language for Inter-action Design, *SIGCHI Bulletin*. **30**(1). 17–23.
Borchers, J. (2000). CHI Meets PLoP: An Interaction Patterns Workshop, *SIGCHI Bulletin*. **32**(1). 9–12. Workshop at ChiliPLoP'99 Conference on Pattern Languages of Programming, 16–19 March, Wickenburg, AZ.
Borchers, J. (2001). A Pattern Approach to Interaction Design. Wiley, New York.
Tidwell, J. (1998). Interaction Design Patterns. In *PLoP'98 Conference on Pattern Languages of Programming*, Illinois, extended version at http://www.mit.edu/~jtidwell/interaction_patterns.html

Correspondence and offprint requests to: Jan O. Borchers, Computer Science Department, Stanford University, Gates Building, Room 201, 353 Serra Street, Stanford, CA 94305-9020, USA. Email: borchers@cs.stanford.edu