Patterns as a link between HCI and architecture

Jan Borchers
Stanford University
http://www.hcipatterns.org/

January 24, 2003

Abstract

This position paper for the CHI 2003 patterns workshop outlines a few recent thoughts on the value of pattern languages for HCI as a bridge between our discipline and the field of architecture. The connection between architecture and HCI has been made repeatedly in the past, and the beginning adoption of the patterns concept in HCI may be an example of the closeness of these two fields. This suggests that maybe the much older field of architecture can give some guidance for how to further establish and develop the field of patterns in HCI.

1 Introduction

The idea of patterns originated from Christopher Alexander's work [1], [2] in urban architecture, although similar formats to document successful design and engineering solutions emerged among the renaissance "master builders" of the fifteenth century [9]. Software engineering adopted the format in the late eighties [4], leading to the general acceptance of software patterns as a very useful form of documenting successful software engineering solutions (see [8] for an early and prominent example).

However, this adoption did not include a few central points of the original pattern language idea: "Software design patterns are considered a useful language for communication among software designers, and a practical vehicle for introducing less experienced designers into the field. The idea of end users designing their own (software) architectures has not been taken over. On the one hand, this makes sense, because people do not live as intensely "in" their software applications as they live in their environments. On the other hand, though, a good chance to push the concepts of participatory design and work ethics forward by introducing patterns has not been taken advantage of." [5, p.25] I will look at possible reasons for this discrepancy in the next section.

2 HCI and architecture

In his influential *Software Design Manifesto* [10], Mitchell Kapor points out that, when creating an actual building, the architect is the professional responsible for the overall design, and the construction engineer, while being a professional peer, takes direction from the architect's design in order to build it. Moreover, the clients or future inhabitants will discuss their ideas and concerns with the architect, not the construction engineer, because the architect is the one responsible for most of the criteria that make the building work for them.

Now, there is an interesting parallelism between architecture and HCI: Architects design physical buildings that their inhabitants directly interact with and live in. User interface designers design virtual interfaces that their users directly interact with and, conceptually, often "live" in while at their task. This closeness to architecture is obvious for 3-D virtual reality environments, but because of the mental model that a user forms of any system while interacting with its interface, it also seems a valid claim for user interfaces in general.

To complete the analogy, the software engineer who deals with the internal structure and inner workings of the software product plays a similar role as the construction engineer in a real building project (see Figure 1).

Kapor claims that the interface designer, just like an architect, should be concerned with what the Roman architecture critic Vitruvius called *commodity* (suitability for the purpose or task) and *delight* (the aesthetic value, or pleasure of using the building or product), while the software engineer, like the construction engineer, will focus on Vitruvius' third criterion for a good building, "firmness" (the robustness and functional correctness of the product). The

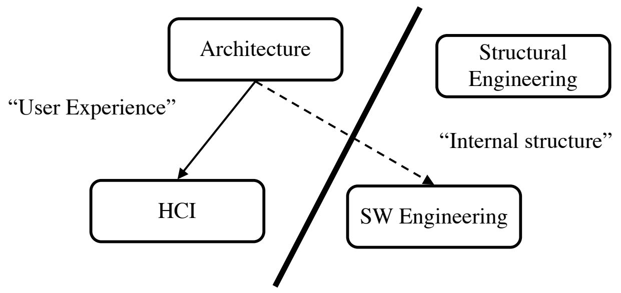

Figure 1: Patterns were adopted first by software engineering, the equivalent of construction engineering in the case of real building projects, but have even more potential in HCI, the equivalent of architecture in the case of actual buildings.

reason for hard-to-use, non-intuitive software, consequently, is that it is largely engineered instead of designed, with too little influence given to the profession of interface design.

3 What this means for patterns

Looking at this closeness of HCI and architecture, it seems almost surprising that software engineering, not HCI, adopted the patterns concept so quickly and widely (Figure 1). On the other hand, early references in key HCI texts such as [11],[12] indicate that there has long been an interest in this link, although it did not emerge widely until the late 90's [3].

However, if we look at how HCI is adopting patterns, the following points support a more fundamental link to architecture:

- Alexander primarily intended his language for inhabitants and laypeople, not architects, to allow them to voice their concerns during the design process of their environments and neighborhoods. This coincides with the interest in user-centered design that is so prominent in HCI. (This, of course, may not be a link to architecture in general, but rather just to Alexander's position, which was in fact not very highly regarded among his peers at the time.)

- HCI has been better than software engineering at pushing patterns to be readable by people outside their own discipline, including end users or clients. This is partly natural since HCI, just as architecture, is concerned with the immediate user experience of the designed artifact. The inner structure of a software "building" is not really of primary concern to those outside the engineering profession, just as is the case in structural engineering.

- If we push the analogy of architecture and HCI a little further, we could imagine interface design firms competing with their design proposals for a new software product, aiming for awards on design qualities, and once a design firm has been selected, the construction specialists (software engineering firms) to be contracted in to turn the interface design into reality, i.e., a working systems architecture [13]. Patterns appear to be an excellent medium to capture the design values of a "school" or design firm, and their beginning success in HCI could be an indicator for the fact that HCI is increasingly becoming a discipline that cares about design as well as technical and cognitive issues. (The same trend is suggested by the stronger focus on design at recent CHI conferences.) It could, in fact, be a sign of a maturing discipline!

- There is a fundamental difference in the way students of architecture and computer science are taught; architecture begins by studying good (and bad) examples of existing buildings and styles, both ancient and recent, and students gradually discover the timeless qualities behind those very different schools. This is exactly what Alexander tried to capture in his language and approach, and if user interface designers are to be taught in a similar fashion, then HCI design patterns would be the natural vehicle to communicate the timeless qualities that user interface designs as varied as a command-line interpreter and a virtual reality environment might have in common (for example, using concepts and objects from the user's domain of expertise). What this requires, of course, is to strike the right balance between too technology-centric, short-lived patterns that essentially only describe what one particular (often graphical) user interface toolkit already implements, and "golden rules" that are always right, but never concrete and constructive enough to be of real value to the designer. I have used pattern approaches in two quite different university courses with some encouraging results [6], and the Stanford HCI curriculum (http://hci.stanford.edu/), in which I have taught several courses, is an example of a program that tries to move somewhat more towards course contents and structures as they are known from architecture, using curricular vehicles such as case studies and design studios.

- Finally, there are other voices that have considered links between architecture and HCI. Winograd [13, p. 10–16] gives a well-rounded overview of the potential – and dangers – of drawing analogies between the two fields, and points out that interaction styles (such as the 3270 terminal style or the Microsoft Windows application style) have evolved that suggest some similarity to the styles and fashions in the much older field of architecture. In the same volume, Rheinfrank and Shelley show that design languages are a common feature of architecture, HCI, and other areas, and that their emergence and explicit creation can move forward the quality of interaction and design in a field; and Denning and Dargan base their action-centered design theory directly on Alexander's pattern language concept. On a more theoretical level, Chalmers [7] proposes that in fact informatics (in particular HCI), linguistics, and architecture are all subfields of the same discipline, semiology.

4 What now?

I believe that the excitement that the patterns concept from architecture has created in at least some parts of HCI is in fact to a large extent a result of a deeper link between the two disciplines. I would like to find out if we can use this connection to get some guidance and precedence in the decisions we are facing in order to move the idea of HCI design patterns forward. (And, on a more practical side, I would like to extend the portal for more of the recent activities in the field of HCI Patterns at http://www.hcipatterns.org/.)

References

[1] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.

[2] Christopher Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.

[3] Elisabeth Bayle, Rachel Bellamy, George Casaday, Thomas Erickson, Sally Fincher, Beki Grinter, Ben Gross, Diane Lehder, HansMarmolin, Brian Moore, Colin Potts, Grant Skousen, and John Thomas. Putting it all together: Towards a pattern language for interaction design. *SIGCHI Bulletin*, 30(1):1723, January 1998.

[4] Kent Beck and Ward Cunningham. Using pattern languages for object-oriented programs. Technical Report CR-87-43, Tektronix, Inc., September 17, 1987. Presented at the OOPSLA87 workshop on Specification and Design for Object-Oriented Programming.

[5] Jan Borchers. *A Pattern Approach to Interaction Design*. Wiley, 2001.

[6] Jan Borchers. Teaching HCI Design Patterns: Experience From Two University Courses. *Patterns in Practice: A Workshop for UI Designers*, CHI 2002, online at http://www.hcipatterns.org/, 2002.

[7] Matthew Chalmers. Informatics, Architecture and Language. In A. Munro, K. Hook and D. Benyon (eds.): *Social Navigation in Information Space*, Springer, 1999.

[8] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object- Oriented Software.* Addison-Wesley, Reading, MA, 1995.

[9] Stephan de Haas. Softwarearchitektur?! Ein Vergleich mit dem Bauwesen. *OBJEKTspektrum*, 6:60–70, 1999.

[10] Mitchell Kapor. A software design manifesto: Time for a change. *Dr. Dobb's Journal* 172:62–68, January 1991.

[11] Donald A. Norman and Stephen W. Draper. *User-Centered System Design: New Perspectives on HumanComputer Interaction.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[12] Donald A.Norman. *The Psychology of Everyday Things*. Basic Books, New York, 1988.

[13] Terry Winograd. *Bringing Design to Software*. Addison-Wesley, 1996.