Toward Patterns for the "Quality without a Name"

Adrienne H. Slaughter
Carol Strohecker

Workshop position paper, CHI'00, Human Factors in Computing Systems, Assoc. for Computing Machinery.
Originally appeared as Working Paper 2000-01, Mitsubishi Electric Research Laboratories.

Abstract

We offer an interaction design pattern for selecting instantiations of a given device, and suggest additional points for discussion at the CHI'2000 workshop on interaction design patterns. We emphasize the need for thoughtful documentation of the purpose of design patterns, and the importance of applying them in a participatory design framework. We recommend that case studies of practioners formulating and applying such patterns precede production of a report on the range of interaction design patterns.

Toward Patterns for the "Quality without a Name"

Adrienne H. Slaughter
Carol Strohecker
Mitsubishi Electric Research Laboratory (MERL)
201 Broadway
Cambridge, MA 02139 USA
+1 617 621 7594, +1 617 621 7517
slaughter@merl.com, stro@merl.com

ABSTRACT
We offer an interaction design pattern for selecting instantiations of a given device, and suggest additional points for discussion at the CHI'2000 workshop on interaction design patterns. We emphasize the need for thoughtful documentation of the purpose of design patterns, and the importance of applying them in a participatory design framework. We recommend that case studies of practioners formulating and applying such patterns precede production of a report on the range of interaction design patterns.

Keywords
Interaction design, design patterns, Java framework, construction kits

INTRODUCTION
During the past few years, with various colleagues, we have developed several software construction kits that differ in content and interaction design, yet bear similarities to one another so fundamental as to constitute a genre [1, 8, 14]. We are currently developing a Java framework that generalizes the key concepts and structures, in an attempt to facilitate implementation of future kits. This endeavor shares concerns with pattern-observing and pattern-making, in the realms of both interaction design and software architecture [3, 5].

We see patterns as starting points, no collection of them being sufficient to realize an entire implementation of a building, an urban setting, a software architecture, or an interaction design. Patterns are summaries of experience, time-honored combinations of observation and practice. Understanding how to apply such wisdom poses a steep learning curve and requires the support of thoughtful documentation.

Here we describe a pattern encountered in the course of interaction design for two of our kits. We also include some more general thoughts about patterns and pattern languages, in the hope that they will become part of the discussion at the CHI workshop.

AN INTERACTION DESIGN PATTERN
Choosing Among Different Instantiations

You have a certain type of selectable item, associated with different instantiations, only one of which can be active at once. You provide users with a palette or menu (with icons or words representing the different instantiations), which enables selection of the desired instantiation.

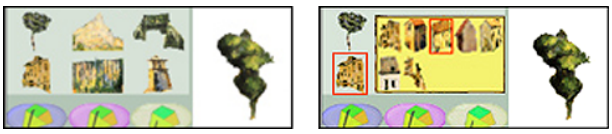
Problem Statement

In many applications, it is common for a device to have multiple possible instantiations. The user will select one instantiation for one purpose, and another instantiation for another, related purpose. The instantiations are mutually exclusive (only one can be active at a time). In order to decide which instantiation to employ, the user needs to know which instantiation is currently selected, and to see it simultaneously with a potential new selection, so that the two can be distinguished and compared.

Existing Examples

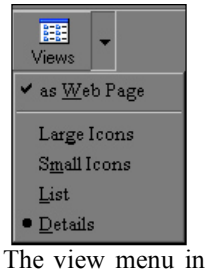
Examples from our construction kits

These images are from a prototype for a construction kit in which users compose cartoon-like dancers from six body parts: head, torso, arms, and legs. It is possible to choose from among several instantiations for each part. When the mouse is on the character's head and the user presses the mouse, a palette appears with smaller images of alternate instantiations. As the user drags the mouse within the palette, a highlight box moves to indicate which head will be selected when the mouse is released.



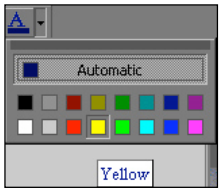
These images are from the WayMaker kit. Users arrange elements of urban landscapes into the form of a map, which the software then transforms into a series of street-level views along pathways through the mapped environment. The tree on the right is an example one of the elements, a landmark, currently instantiated as a tree. By clicking on it, the user can activate the palette displaying other possible options for this landmark, including a bridge, a tower, a house, etc. When the user presses on one of these options, another palette pops up, which allows the user to specify which particular bridge, tower, or house the landmark should become. This example is interesting because it embodies the pattern in two levels.

The View Menu in Microsoft Windows Explorer



The view menu in Windows Explorer allows the user to specify how the system should display folders and files. The options include different display styles and/or levels of detail about each folder or file, allowing the user to show the files in different ways. If a user clicks the Views button, the current view is switched to the next view in the list. If the arrow is pressed, the menu pops up and the user can select which view to use. The current view is indicated by a dot next to the name of that view.

The Text Color menu in Microsoft Word



The Text Color palette in Word allows the user to specify the color of text. It is self-evident that text can only be one color (at least in this context). The text color button has a colored bar underneath the A to show the last selected text color. If the user clicks the button, selected text is set to be that color. If the user presses the arrow, the menu pops up and allows one of many colors to be selected. The color that will be selected when the mouse is released is indicated by an inverted 3D rectangle, and the name of the color is shown in a tool tip.

The Marquis Tool in Adobe Photoshop (or many of the other tools in Photoshop)

The Marquis Tool button in Photoshop shows the currently selected tool. A small arrow on the button indicates the availability of alternate selections. If the user presses and holds the mouse on the button, a menu pops up, with the currently selected tool grayed out and unable to be selected (since there is no reason to re-select a tool that is already selected). The user makes a selection by dragging the mouse to its image and releasing the mouse button. The tool that will be selected when the mouse is released is indicated by a lighter, inverted 3D rectangle.

A General Solution

Provide an icon that reports the current state of the tool. Signify that a menu or palette pops up by an illustrative device such as a mini arrow. Clicking on the icon will activate the tool in its current instantiation; pressing the mouse on the icon will allow the current instantiation of the tool to be changed. In the menu/palette that displays instantiations and enables selections of the tool, signify which one is currently active to provide feedback.

```
currentIcon = currentToolInstanceIcon
On <mouseClick>
        activate currentToolInstance
On <mousePress & hold>
        show ToolInstancePalette (with currently selected
            tool signified)
        while <mouseDrags> showNextToolInstance (the
            tool that will be selected if the mouse is released)
On <mouseRelease>
        currentToolInstance = showNextToolInstance
        currentIcon = currentToolInstanceIcon
        activate currentToolInstance
```

Related Patterns

This pattern is related to patterns that can be described as Choice From a Small Set, Status Display (or Currently Selected Option), Small Groups of Related Things, Localized Object Actions [c.f., 15].

CONSIDERATIONS IN DEVELOPING PATTERNS

Anthologies of design pattern patterns are offered as compendia of accumulated wisdom, general descriptions of past experience that can inform similar work in the future. Unfortunately, however, such anthologies may appear as catalogs of solutions – pick and choose, combine willy-nilly, and *voila!* You have a good design. How can producers of design patterns ensure that their efforts will be well understood and sensibly applied?

Design patterns producers need to make clear that their effort is conceptual in nature. Illustrating patterns with pseudo-code is one way to help communicate the intended conceptual level, but does the language of reuse counter

that message? [5] Patterns are practical but not to be taken literally. Individual application contexts call for variations in particulars of the patterns. Furthermore, the patterns' usefulness depends on the skills, tastes, communication, and coordination of the designers who bring them to bear on a given problem [6, 4]. These non-patternable essentials have a profound effect on the outcome of the design process and contribute to the "quality without a name," the loss of which Richard P. Gabriel bemoans [4].

Even if the summarial, conceptual intent of the patterns is communicated well, would people tend to misinterpret such an anthology anyway? Consider that many people prefer following recipes to adapting them to their own current purposes. There is an obvious analogy to pattern use. "Recipe-following" may have to do with issues of trust and control running so deep within a personality that it may be pointless to chide against taking such an approach. It may be a matter of intellectual style [16]. Furthermore, depending on the situation, it can be quite effective.

We encounter a similar problem on the relatively surface level of skills. In our work we have been tempted to provide code starters ("seeds") and code generators ("wizards") to get people started who don't know Java well, but who do understand our conceptual approach and want to create kits within the genre. By attempting to support varying programming skills, we may be on the road toward a cookie-cutter approach that bears a resemblance to the unfortunate aspect of pattern use. We are looking for ways to temper this effect.

As the CHI community attempts to promote beneficial ways of thinking and talking about interaction design, we may need to separate the language of accumulated wisdom from the language of generating new interaction designs. Observations that can be described as patterns result from an analytic process; it is inevitably problematic to invert such a process to a synthetic one.

One of our kits has brought this dilemma particularly into focus. WayMaker users create maps from representations of Kevin Lynch's "elements of the city image," and then see street-level scenes along pathways within the mapped domain [7, 10, 11, 13]. Lynch's process of identifying the structural elements was analytic, yet practitioners inverted that process by using the elements as a basis for generating urban designs. As with Alexander's work, the outcomes were often unsatisfying. Lynch studied a range of reasons for this problem, re-explaining that the elements are general and include only structural, not personal nor aesthetic, aspects of urban experience. However he focused most strongly on the problem that, in spite of his example of having worked personally with urban dwellers in identifying the elements, his colleagues neglected city inhabitants when applying the elements in a design process. In Lynch's ideal world, non-professionals would contribute to creating the environments they inhabit. Lynch emphasized the need for a participatory process in the practice of design, a caution we take seriously in developing WayMaker and other kits, and in identifying contexts for their use [8, 10].

CONCLUSION

In summary, we advocate for participatory design in identifying interaction design patterns. Case studies of practitioners working with users to formulate and apply patterns could help in ascertaining whether they serve equally well the needs of the designers and the end-users.

We look forward to a sustained conversation about these issues, and ultimately to an anthology treating a range of interaction design patterns.

ACKNOWLEDGMENTS

Several other people have also contributed to design, development, and use of the kits informing our framework: AARCO medical illustrators, William Abernathy, Edith Ackermann, Aseem Agarwala, Noah Appleton, Maribeth Back, Barbara Barros, Dan Gilman, Mike Horvath, John Shiple, Doug Smith, students at Harvard University's Graduate School of Design, colleagues at MERL, and anonymous friends. Discussions with Mario Bourgoin, Stephanie Houde, Sarah Kuhn, Warren Sack, Tim Shea, and with Fred Martin, Seymour Papert, Brian Silverman, and other members of the Epistemology & Learning Group at the MIT Media Lab, have inspired and/or improved various kits as well as the more general formulations. We thank John Evans, Aradhana Goel, Tim Gorton, and Milena Vegnaduzzo for participating in trials of the Kit4Kits. MERL supports the research.

REFERENCES

1. Ackermann, E., and Strohecker, C. Build, launch, convene: Sketches for constructive-dialogic play kits. MERL TR99-30, Mitsubishi Electric Research Laboratory, Cambridge, MA, 1999.
2. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., and Angel, S. A Pattern Language: Towns, Buildings, Construction, Oxford University Press, New York, 1977.
3. Erikson, T., and Thomas, J. Putting it all together: Pattern languages for interaction design. Proceedings of CHI'97, 226. See also the summary report of the workshop at http://www.pliant.org/personal /Tom_Erickson/Patterns. WrkShpRep.html.
4. Gabriel, R. P. The failure of pattern languages. Journal of Object-Oriented Programming 84-88, 1994.
5. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, 1977.
6. Goldberg, A. What should we teach?

7. Lynch, K. The Image of the City. MIT Press, Cambridge, MA, 1960.
8. Lynch, K. Reconsidering the image of the city, in Rodwin, L. and Hollister, R, M. (eds.). Cities of the Mind: Images and Themes of the City in the Social Sciences. Plenum Press, New York, 151-161, 1984.
9. Strohecker, C. Construction kits as learning environments. Proceedings of IEEE International Conference on Multimedia Computing and Systems 2, 1030-1031, 1999.
10. Strohecker, C. Toward a developmental image of the city: Design through visual, spatial, and mathematical reasoning. Proceedings of Visual and Spatial Reasoning in Design (University of Sydney and Massachusetts Institute of Technology), 33-50, 1999.
11. Strohecker, C., and Barros, B. A prototype design tool for participants in graphical multiuser environments. CHI'97 Extended Abstracts, 246-247, 1997.
12. Strohecker, C., and Barros, B. Make way for WayMaker. Presence: Teleoperators and Virtual Environments 9:1, 97-107, 2000.
13. Strohecker, C., Barros, B., and Slaughter, A. Mapping psychological and virtual spaces, International Journal of Design Computing, University of Sydney, 1998.
14. Strohecker, C., and Slaughter, A. Kits for learning and a kit for kitmaking. Submitted to CHI'00. Also available as MERL TR2000-02, Mitsubishi Electric Research Laboratory, Cambridge, MA, 2000.
15. Tidwell, J. Common ground: a pattern language for human-computer interface design. http://www.mit.edu /~jtidwell/ui_patterns_essay.html, 1999.
16. Turkle, S., and Papert, S. Epistemological pluralism: Styles and voices within the computer culture. Signs 16:1, 1990.