Position paper for the CHI 2003 Workshop
"Perspectives on HCI Patterns: Concepts and Tools"

Sherman R. Alpert
IBM T.J. Watson Research Center
PO Box 704
Yorktown Heights, NY 10598 USA
salpert@us.ibm.com

Getting Organized: Some outstanding questions and issues regarding interaction design patterns

The potential usefulness and power of interaction design patterns is great, but *many* questions and issues must be resolved before HCI and/or UI patterns can become widespread and useful tools. The following is my "starter set"; I would like to discuss and work toward resolutions to the following at least.

First, interaction design patterns exist at multiple levels of abstraction and granularity. This issue also arises in software design patterns. Buschmann et al. (1996) assert that patterns can range from *idioms* to *design patterns* to *architectural patterns*. Micropatterns might be concerned with method and variable naming conventions and macropatterns might describe reusable frameworks such as Model-View-Controller (Alpert, Brown, & Woolf, 1998). In the domain of interaction patterns, we should address questions such as: Are HCI patterns the same as UI patterns? I believe, no they are not. One possible view is that HCI design patterns deal with high level concerns and perhaps guidelines involving user psychology—e.g., in general, *Exploit recognition over recall*, which can be manifested in providing a selectable list of operations (or menu of operations, or a set of radio buttons or push-buttons, one per operation) rather than having a user type an operational command. HCI patterns may also include architectural patterns, such as *Use the client-server model for highly interactive Web-based applications*. UI patterns, on the other hand, might be viewed as more specific, on the order of *When an application has this problem/situation: different information is relevant in different contexts, i.e., orthogonal information must be shown, using different screen layouts, depending on mode or context or task and there is not enough screen real estate to display simultaneously all the information relevant to all contexts; Solution: the "Sharing Screen Space" pattern (Alpert, 2000)*. The upshot: **what constitutes an HCI pattern?; what precisely defines UI patterns? How are they the same, different? Also: what chunk of knowledge—at what grain size—is appropriate to "qualify" as an independent pattern?**

Next, **What are the scenarios of use—how do we envision HCI and UI design patterns being used?** When would UI designers consult a *PatternsBase* (a database of design patterns)? Why? How should DPs be organized in the PatternsBase? If patterns are accessible/editable by a computational tool, how are DPs indexed in the PatternsBase? How would a user search for/find appropriate patterns for an application's situation? The rationale behind deriving scenarios of use is that **a reasonable set of use scenarios is necessary to drive the design of design patterns tools.**

Another (continuing) question is, **What information or knowledge necessarily must be included in the pattern?** This may also be (partially) driven by tool use scenarios. As one example, if one use scenario is, "I have this application *X* that has problem/context/situation *P*; what design pattern can be applied to solve/address *P*?" then interaction patterns ought to include an *Applicability* section as do the Gang of Four software design patterns. I've seen many interaction patterns that do not include such a section. An *Applicability* section specifies in specific terms when a pattern should be considered as a solution. For example, the *Applicability* section of a software design pattern might say,

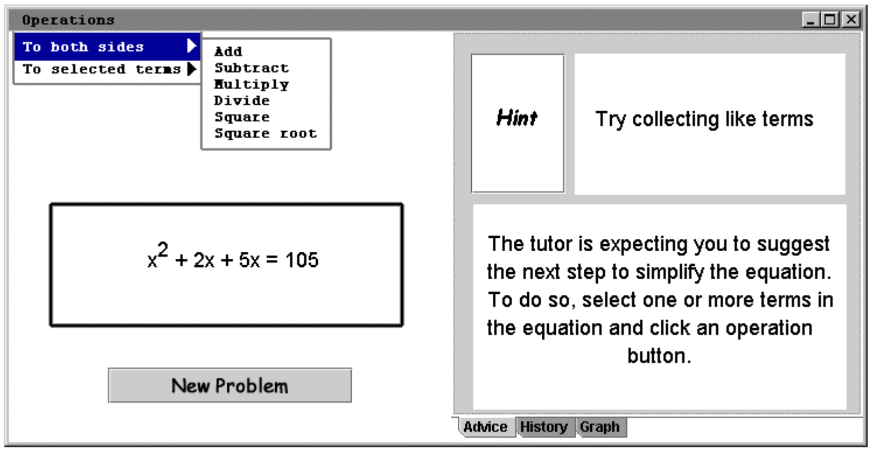
> Use the Adapter pattern when
>
> - you want to use an existing class, and its interface does not match the one you need
> - you want to create a reusable class the cooperates with unrelated or unforeseen classes, that is, classes that don't necessarily have compatible interfaces… (Gamma et al., 1995, p. 140).

For a UI design pattern, an *Applicability* section might appear as:

> This pattern applies when …:
>
> 1. A single application has multiple contexts, states, tasks, or user informational needs.
> 2. Each of these contexts requires the user to interact with UI elements that are relevant to *that* context alone, but:
> 3. We don't want to overwhelm or confuse the user with a window that simultaneously displays all UI elements relevant to *all* contexts,
>
> … (Alpert, 2000, p. 3).

With regard to tools, the information in the *Applicability* section would be useful when attempting to find applicable/appropriate patterns for a particular context/situation/problem.

We must also ask the question, "**Can the use of DPs stifle creativity?**" This may also be somewhat dependent on usage scenarios, for example just how and when UI design patterns will be used by designers. Once an appropriate pattern ("solution" to a design problem) is found, will it be used as is or improvised upon? Here's a concrete example: in an intelligent tutoring system for elementary algebra (Alpert, Singley, Fairweather, 1999), we needed a way to show an equation, a way to select a term or terms in that equation, a way to select an algebraic operation to work on the selected term(s), a "give me a hint" button, etc. If we had simply consulted a pattern language or PatternsBase of UI patterns and strictly followed its advice/solution, the UI might have looked like the following [menus for operations, tabbed pages to share screen space, text fields for the equation and hints]:

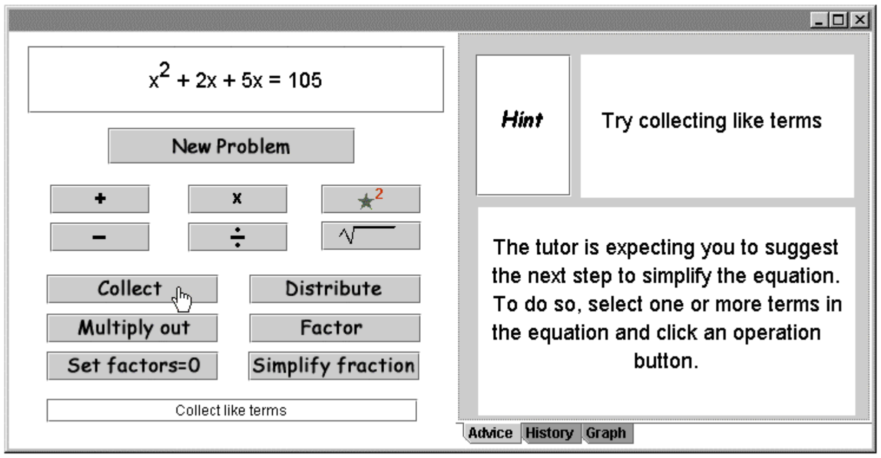


$x^2 + 2x + 5x = 105$

Try collecting like terms

The tutor is expecting you to suggest the next step to simplify the equation. To do so, select one or more terms in the equation and click an operation button.

or the following [sets of normal buttons for operations, tabbed pages to share screen space, text fields,]:



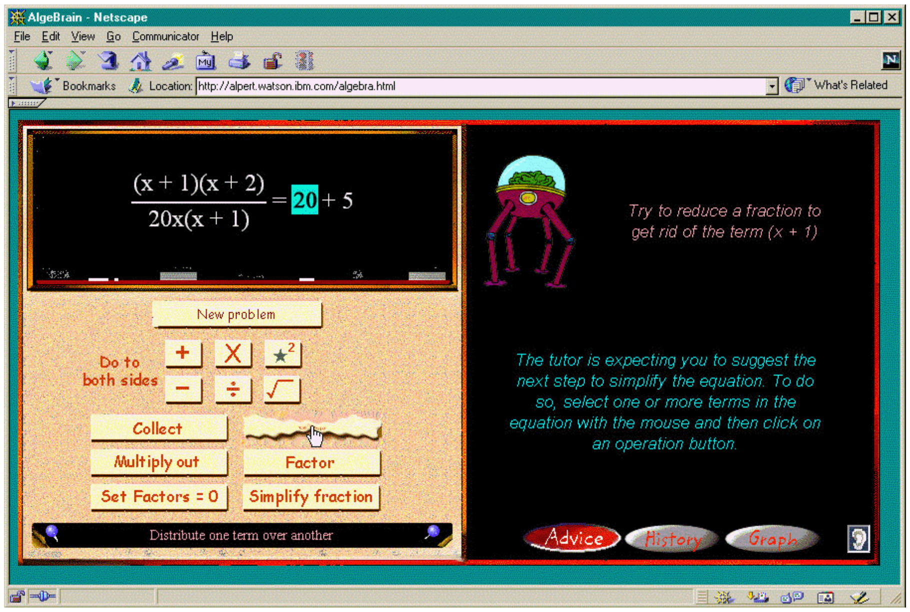
$x^2 + 2x + 5x = 105$

Try collecting like terms

Collect like terms

The tutor is expecting you to suggest the next step to simplify the equation. To do so, select one or more terms in the equation and click an operation button.

The above UIs are certainly usable, but are also rather drab. This application was intended to be used by middle school students who might require something more motivating, stimulating. The actual implementation looked like the following; the "paper" buttons on the corkboard crunch (and make a crunching sound) when clicked, the animated interface agent dances, applauds, etc. and can be clicked on to ask for hints. This UI is more interesting and aesthetically pleasing. But would design patterns "lead" me to such a solution?



$$\frac{(x+1)(x+2)}{20x(x+1)} = \boxed{20} + 5$$

Try to reduce a fraction to get rid of the term (x + 1)

The tutor is expecting you to suggest the next step to simplify the equation. To do so, select one or more terms in the equation with the mouse and then click on an operation button.

Another issue: **Are HCI/UI design patterns different for different targets (platform, operating system)**? In *The Smalltalk Companion* (Alpert, Brown, & Woolf, 1998), for example, we point out how and why many of the Gang of Four's patterns change significantly when the target language is Smalltalk as opposed to the C++ implementations portrayed in the original *Design Patterns* (Gamma et al., 1995). Martijn van Welie has nicely organized interaction patterns in three categories based on target environment: GUI, Web, and mobile (see http://www.welie.com/patterns/) – are these categories sufficient? How about 3270 green screens? Should patterns be categorized by target?

Lastly, the notion of **anti-patterns** may be very useful for the HCI community. Anti-patterns discuss what *not* to do, how *not* to solve a particular problem. We've already seen this genre of work in the UI community in the form of "bloopers" publications (e.g., Johnson, 2000). The sort of "don'ts" presented in such publications can be expressed in pattern form, just as "do's" (solutions) can be.

References

Alpert, S.R. (2000). Sharing Screen Space Among Multiple Application Contexts, or "Real Estate is Expensive," In *Proceedings of CHI 2000 Workshop on Pattern Languages for Interaction Design*, http://www.it.bton.ac.uk/staff/rng/CHI2K_PLworkshop/.

Alpert, S.R, Brown, K. & Woolf, B. (1998). *The Design Patterns Smalltalk Companion*. Addison-Wesley (Software Patterns Series).

Alpert, S.R., Singley, M.K., & Fairweather, P.G. (1999). Deploying Intelligent Tutors on the Web: An Architecture and an Example, *International Journal of Artificial Intelligence in Education*, *10*(2), 183-197.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns*. Wiley.

Gamma, E., Helm. R., Johnson, R., & Vlissides, J. (1995). *Design Patterns*. Addison-Wesley.

Johnson, J. (2000). GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers. Morgan Kauffmann.