A Pattern Language for User Assistance

Mike Hughes | IBM Internet Security Systems | mhughes2@us.ibm.com

USER ASSISTANCE HAS COME A LONG WAY from mere stand-alone help files. We now find abundant instances of its being consistently woven into applications and, thus, the user's experience; it is no longer a document separated from the user's task. Accordingly, user-assistance designers need a design language that deals with user context and system interactions more directly—more in line with the way user-interface designers have come to work. An approach that has gained popularity with user-interface designers is that of *pattern language* as a way to derive and describe design solutions. This article introduces pattern language concepts and shows how they can be applied to user assistance to provide a common language for discussing design strategies that integrate user assistance and user experience.

PATTERN LANGUAGE. Architect Christopher Alexander introduced the concept of pattern language as a guide to design [1]. He describes a pattern language as a collection of patterns, each of which describes a relationship between (a) a certain context, (b) forces that recur within that context, and (c) spatial configurations that allow these forces to resolve themselves. Eventually, software designers and user-interaction designers were drawn to patterns as ways of capturing best design practices [3, 4, 5]. A pattern language for user assistance enables an application or family of applications to feel the same to the user in terms of "when users have this kind of problem, the system will make this type of user assistance available."

The following is an example of a pattern language for user assistance.

Trigger	The trigger statement should describe the condition (a point of pain or moment of opportunity) that leads the user to need assistance
Context	A description of the kinds of conditions or scenarios that could cause the trigger
Conflicting Forces	Statements contrasting user goals, needs, or states with opposing constraints, for example, "Users want to take action before they fully understand the what, how, and why"
Resolution	A description of the recommended approach
Example	Screen shots or wireframes when appropriate
Claims Analysis	Description of positive and negative usability consequences for the pattern being described
When to Apply	Examples or additional guidelines for the appropriate contexts within which to use the pattern
When Not to Apply	Guidelines or contexts when other patterns might be more appropriate

Figure 1: A Pattern Language for User Assistance

PATTERN: FIELD-LEVEL PUSHED HELP. The following is an example of a pattern using the pattern language presented in Figure 1.

Trigger: User is in a task and is uncertain about the purpose or rules around a specific field or interaction device, e.g., radio button group.

Context: Form-based transactions are usually intuitive in regards to what fields to fill in and what command buttons to click. What can still be unclear at times is what a particular field means, which business rules dictate allowable entries, or what the user should consider in deciding which value to enter or selection to make.

Conflicting Forces: The user might not have enough information to properly interpret or interact with a screen component, but...

- Text used to explain a component can take up valuable real estate.
- The user's focus is drawn to the component (field, drop-down, or button), and the user does not see UI text or labels that explain the component.
- The user might not be aware that he or she does not understand the concept and might not think to ask for information.

Resolution: The application determines when a particular field or screen element is the point of focus and provides critical information on how to interact with that component or interpret the information.

Field-level Pushed Help should:

- Attract the user's attention
- Apply directly to a core information need
- Be clearly associated with its component
- Must not occupy needed real estate of the focused transaction

Example: Figure 2 shows an example of a field definition for a functional argument in MS Excel being pushed to the user.

- The message could be triggered by the presence of the cursor in the date field.
- The message could appear as a slow-open so that the motion would signal its appearance to the user.
- The message could close automatically after a few seconds or be dismissed when cursor focus moves on.

CLAIMS ANALYSIS:

POSITIVE

- Puts information in front of user at likely point of pain or moment of opportunity
- Doesn't take up screen real estate when not needed

NEGATIVE

- Could be distracting to experienced users who no longer need to be told what is in the help message

When to Apply

- When interactions are not intuitive
- When the meaning or impact of a field is not obvious or well-known to the user

When Not to Apply

- Do not push help for a component when its meaning or use is likely to be clear.
- If the explanation cannot be delivered in a concise statement, consider using the Field-level Linked Help pattern instead.

BENEFITS OF PATTERN LANGUAGE

Design Benefits. Because patterns describe triggers and contexts, they keep designers focused on the user experience. The conflicting-forces component helps the designer understand the constraints within which the solution must be effective. The claims analy-

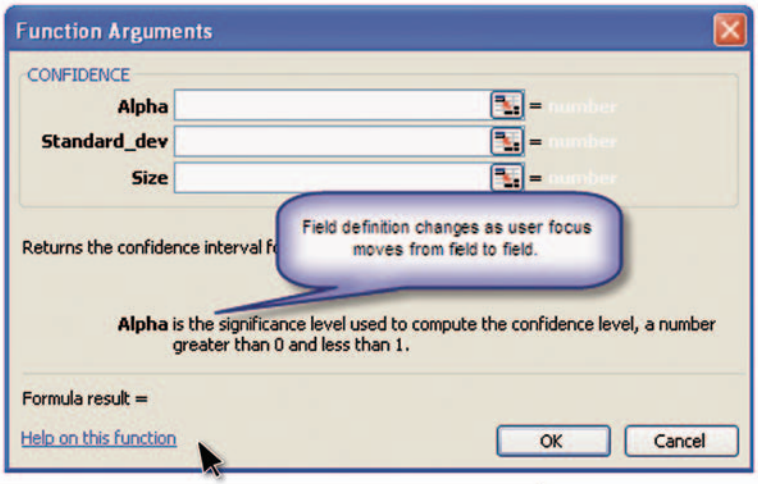

Figure 2: Field Definition Pushed to User. (Microsoft product screen shot reprinted with permission from Microsoft Corporation.)

sis helps designers anticipate potential problems and either design to mitigate those problems or examine other patterns that might not have those problems [2].

Communication Benefits. A problem many companies face is that they have diverse product teams that work on different products or parts of products separately. This diversity often results in inconsistencies in treatment and the kinds of user assistance offered. Pattern language is a way of communicating design solutions across an enterprise while still allowing designers the freedom to shape their solution to the specific style and needs of their application.

Now that user assistance has matured to being a fully recognized component of the user experience, it must evolve beyond the tools that manage static documentation. Pattern language allows consistent best practices to develop and be communicated across the design community within an enterprise.

ABOUT THE AUTHOR *Michael Hughes has a PhD in Instructional Technology from the University of Georgia and a Masters in Technical and Professional Communication from Southern Polytechnic State University. His professional focus is designing user interfaces that accommodate the "user as learner." Dr. Hughes works for IBM Internet Security Systems as a user experience architect identifying tools, methods, and standards to integrate the content and delivery of user assistance, including documentation, help, e-learning, and training.*

REFERENCES **1.** Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Schlomo, A. (1977). *A pattern language: Towns, buildings, construction*. Boston: Addison-Wesley. **2.** Carroll J M, Rosson M B (1992) Getting around the task-artifact cycle: how to make claims and design by scenario. *ACM Transactions on information systems*. Vol 10 No 2 April 1992 181-212 **3.** Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Boston: Addison-Wesley. **4.** Tidwell, J. (2005) *Designing interfaces: Patterns for effective interaction design*. Cambridge, MA: O'Reilly **5.** Van Duyne, D., Landay, J., & Hong, J. (2003) *The design of sites: Patterns, principles, and processes for crafting customer-centered web experience*. Boston: Addison-Wesley