Human Factors International

User Interface Patterns: Reusable solutions for design

White paper

Dean T. Barker, MS, CUA
Executive Director
Human Factors International, Inc.

July 12, 2007

Human Factors International
410 West Lowe
PO Box 2020
Fairfield, IA 52556

800–242–4480

hfi@humanfactors.com
www.humanfactors.com

Table of Contents

Dean T. Barker, MS, CUA
Executive Director
Human Factors International

About the Author

Dean T. Barker is an Executive Director for Human Factors International. He is a technical expert participating in ISO/IEC JTC1/SC7, the International Organization for Standardization (ISO) committee responsible for developing standards for Software and Systems Engineering. He is also co-author of Designing Effective Speech User Interfaces, published by John Wiley & Sons. He has fifteen years of experience in usability and software product development, including substantial experience in international software product development for various markets in North America, South America, Europe, and Asia. Mr. Barker has a B.A. in Business Management and Communication and a M.S. in Software Engineering from the University of Minnesota.

Dean has introduced a “pattern approach” to user interface design to several Fortune 500 companies and has written and researched User Interface Patterns extensively as part of his graduate thesis in Software Engineering.

All about patterns...

A Pattern is "a plan, diagram, or model to be followed in making things."[1]

The general field of design exhibits a rich history of patterns. Designers in many disciplines, from architecture to musical composition, have endeavored to categorize, create, and employ patterns in order to achieve a higher level of order and quality in their work.

In recent years, the discipline of software engineering has consciously moved toward the use of patterns as an aid in systems architecture, analysis, and programming. The aim of this movement is toward achieving not only the benefits of greater order and quality of product, but also toward producing reusable assets in order to create more efficient software engineering processes. Similarly, a patterns-based approach to user interface design can be a powerful tool and help an enterprise realize the benefits of reuse and quality.

The work that defined the commonly accepted concept of design patterns was Christopher Alexander's landmark book on architectural design entitled *The Timeless Way of Building*. Alexander followed *The Timeless Way of Building* with *A Pattern Language*, in which he and his co-authors published a thorough catalog of architectural patterns. Since the publication of these books, many others have adapted Alexander's concepts to their own design disciplines. Specifically, a number of people have contributed to and extended a pattern-based approach to software design. The book *Design Patterns: Elements of Reusable Object-Oriented Software* defined an approach to creating and using software programming objects that has become generally accepted. Subsequently, a handful of books have been published which discuss user interface design patterns and offer a number of pattern examples.

Like software engineering, a pattern-based approach to user interface design requires that practitioners have both a common vocabulary and a common repository of specific patterns to draw from. In abstract form, these elements together define a "pattern language" to be used by a practitioner community. In *The Timeless Way of Building*, Alexander defines pattern languages as "finite combinatory systems which allow us to create an infinite variety of unique combinations, appropriate to different circumstances, at will." In its concrete form a pattern language is realized by documenting individual patterns in a format called a pattern description and collecting them in a patterns catalog (also called a library or repository). Once a pattern language is defined and documented, designers can use the patterns to quickly assemble them in a modular manner, thus making the design process more efficient.

1. The American Heritage Dictionary of the English Language, Fourth Edition

The Rational Unified Process (RUP), an object-oriented development methodology, offers a practical definition of a pattern as "a solution template for a recurring problem that has proven useful in a given context. Good patterns successfully resolve the conflicting forces that define the problem, and one pattern is chosen over another based on the way it resolves those forces."[2] In addition, the RUP documentation notes "a software pattern is instantiated by binding values to its parameters. Patterns can exist at various scales and levels of abstraction, for example, as architectural patterns, analysis patterns, design patterns, test patterns and idioms or implementation patterns."

The analogy of a pattern as a template is useful. For example, consider using a word processor to write a memo. You could "start from scratch" and define all the design parameters of the memo, such as beginning with a blank document and adding a "To" field, "From" field, "Date" field, setting page margins, and specifying cosmetic attributes such as font size and style. Or, by using a template, you can simply write the memo using a pre-defined format for everything. A template

- allows you to focus on the content
- provides consistency to all memos you publish
- produces a product ostensibly following best practices.

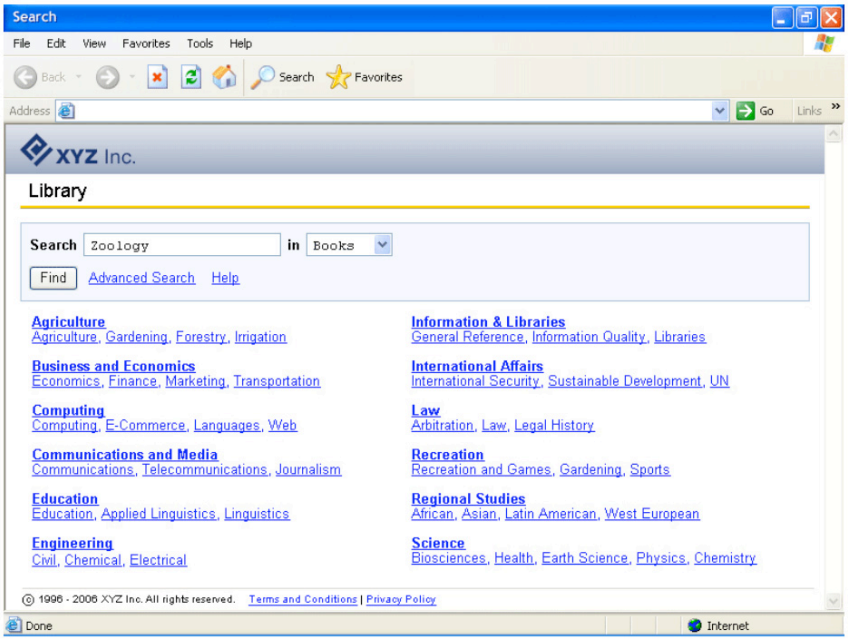
User interface design (UID) patterns can be considered as design templates, each being useful for solving a given design problem and none including specific values for its parameters. Because user interface design is an interdisciplinary function, a UID pattern catalog can capture solutions to design problems related to

- information architecture
- interaction design
- visual design.

Thus far, most of the literature on user interface patterns has focused primarily on interaction design. In part, this may be the best direction for our field due to the relationship between good interaction design and usability.

So, what are user interface design patterns? They are simply prefabricated designs which are intended to be reused to create other designs. For example, the Web page below is a template from HFI's Usability Central, a customizable repository of information and tools for user interface design. The Search page has been designed and documented specifically with the intention of reuse in design projects for Web sites and applications.

1. The American Heritage Dictionary of the English Language, Fourth Edition



A UID pattern can be used in many design scenarios. However, because they are "generic" design templates, the pattern documentation does not include specific content (although it may contain "mock" content). Those values would be documented in the design specification, not the pattern. This is not to say that UID patterns are without details. They should be documented at an appropriate level of detail to be used effectively by designers in a given organization.

User Interface Patterns are not guidelines or standards. A pattern is a reusable design module while standards and guidelines are rules for designing. Guidelines are recommendations for design styles or conventions intended to aid designers and developers in their detailed design. Guidelines are based on psychological principles of cognitive theory and human-computer interaction as well as best practices in the field of software design. Standards are compulsory specifications, typically for low-level design conventions and are usually intended to have some level of governance within an enterprise. User interface standards provide users of a system with a more usable interface by providing consistency in detailed design.

The relationship between patterns, guidelines, and standards can be somewhat complex. Certainly patterns can be implemented in a software product development organization as standards (i.e. compulsory for use with a given product or product line). Additionally, standards and guidelines can be included in the details of patterns (e.g. guidelines for widgets as part of component patterns). For example, in a case study called Implementing a Pattern Library in the Real World: A Yahoo!

Case Study,[3] the authors refer to "using the patterns library as a body of standards" and say that their goal was for the patterns to be understood as the "Yahoo! way" to approach certain designs.

Ultimately, it is incumbent upon each organization using a patterns catalog to define not only the user interface patterns, but user interface guidelines, standards, and the relationships between the three as well.

Documenting patterns

Each pattern in a catalog should be well-defined and documented in a Pattern Description document. Each pattern description must be uniform in order to be usable for the catalog's users. Thus, in order to implement a pattern approach, an industry or organization needs to agree upon a framework for pattern descriptions. Pattern frameworks vary from source to source. However, they commonly contain four basic elements:

1. Pattern Identification
2. Problem
3. Solution
4. Consequences

The *Pattern Identification* is the meta-data for a pattern. It provides for identification, classification, and use of patterns. Most importantly it should include a thoughtful Name for the pattern. The name should be succinct enough to facilitate conversation about the pattern, yet still provide enough information in order to convey the essence of the pattern to readers of the pattern description. Technical jargon may be used in pattern names depending on the audience for a catalog. However, in general, "popular" names are most useful. Pattern names can be difficult to construct, but having a good pattern name allows people to more easily discuss and collaborate.

The *Problem* is a description of the design problem that is solved by employing this pattern. It should provide context for the pattern as well as capture the motivation for use.

The *Solution* is the description of the design solution, including the individual elements of the pattern. For user interface design, a solution is described both in words and pictures. The textual description is secondary to the illustration of the pattern.

The *Consequences* of a pattern include the resulting state once the pattern is implemented as well as the trade-offs that exist.

In the pages that follow is a framework for documenting individual patterns. Within the framework, each attribute is defined.

3. Published for the 2005 ASIS&T IA Summit (www.leacock.com/patterns/)

Pattern Identification

Serial Number: The serial number is a unique sequential number tied to the name, which will identify each pattern in the catalog.

Name: The pattern name is its primary descriptor.

Aliases: Aliases are captured for other useful names for a pattern. Sometimes these are formed from use within an organization or as descriptors based on other archetypes in the industry. For example, the page series commonly known as a "shopping cart" is sometimes referred to instead as a "shopping bag."

Purpose (User Behavior): Each pattern is assigned a category to associate it with a design purpose. The categories are defined by the user interaction with the system (i.e. pattern) and structured as high-level tasks, such as Browsing, Searching, Buying, etc.

Scale (Structural Scale): Each pattern is classified in regards to a level of granularity regarding its concrete implementation as a user interface element. For example, in a Web user interface, classes that identify the scale of a pattern might be Page Series, Page, Subpage, and Widget.

Related Patterns: Related patterns are cross-referenced for ease of use in combining patterns as needed to address design problems.

Author(s): The author or authors are listed for each pattern.

Revision History: A revision history is kept in order to maintain version control.

Comments: This is a space that is reserved in each pattern description for miscellaneous author's comments.

Keywords: Keywords are assigned to the pattern for the purpose of searching the catalog for solutions to design problems.

Problem

Problem Description: This is a brief problem description that includes the context and motivation.

Design Constraints: This section documents any design constraints that impact the pattern.

Solution

Solution Description: This is a textual description of the pattern and is coordinated with the Structural Diagram as needed.

Design Rationale: Rationale is documented for each solution. This should include generally accepted heuristic principles and guidelines as well as the author's explanation of the design rationale.

Structural Diagram(s): This is an illustration of the pattern using a wire-frame approach. The illustration is a line drawing that is inserted in the document from a separately created drawing file.

Variations: If there are variations on a pattern, then a structural diagram is provided for each along with a separate solution description. In such cases, the Solution Description section will be a general description and details for each variation accompany its structural diagram.

Sample Code: If available and pertinent to the solution, then sample code may be included. For example, static HTML code might be included.

Known Use: Visual examples (e.g. screenshot) and references (e.g. file location, URL, etc.) are included for known instances of this pattern, including both internal and external usage if available. Also, both "good examples" and "bad examples" may be included if appropriate. Good examples are shown in order to illustrate proper use of the pattern in a production environment. Bad examples are shown in order to illustrate potential pitfalls in use of the pattern. These are sometimes referred to as "antipatterns."

Consequences

Consequences Description: This section may include consequences of use, usage rules, implementation issues, and trade-offs.

Additional Resources: If available, the author should list additional resources for each pattern.

Internal: Internal resources include style guides, templates, etc.

External: External resources include published books, websites, etc. For example, for a "Wizard" pattern the Microsoft Online Guidelines might be referenced as well as user interface design books related to the topic.

A sample pattern description

This section includes a sample pattern whose purpose is to illustrate the framework previously defined. Note that the level of detail in the functional diagram will vary from pattern to pattern as needed.

Shopping Cart List

Serial Number:	023
Name:	Shopping Cart List
Aliases:	Cart
Purpose:	Purchasing
Scale:	Subpage
Related Patterns:	Buying Page Series, Shopping Cart Page, Shell, Graphical Button Widget

Author(s):	Dean Barker
Revision History:	Version 1.0 – Sample
Comments:	None
Keywords:	Buying, Shopping, List, Cart, Checkout

Problem

Problem description
As customers select items for purchase they need a view of the items they have gathered in order to see the complete collection prior to checkout (i.e. purchasing).

Design constraints
The Shopping Cart List is part of the Shopping Cart Page, which emphasizes the cross-sell space in order to encourage additional purchasing.

The initiation of the Checkout process from the Shopping Cart List should be obvious and intuitive.

The Shopping Cart List shall be Editable.

Gift wrapping options shall be made available via the Shopping Cart List.

Solution

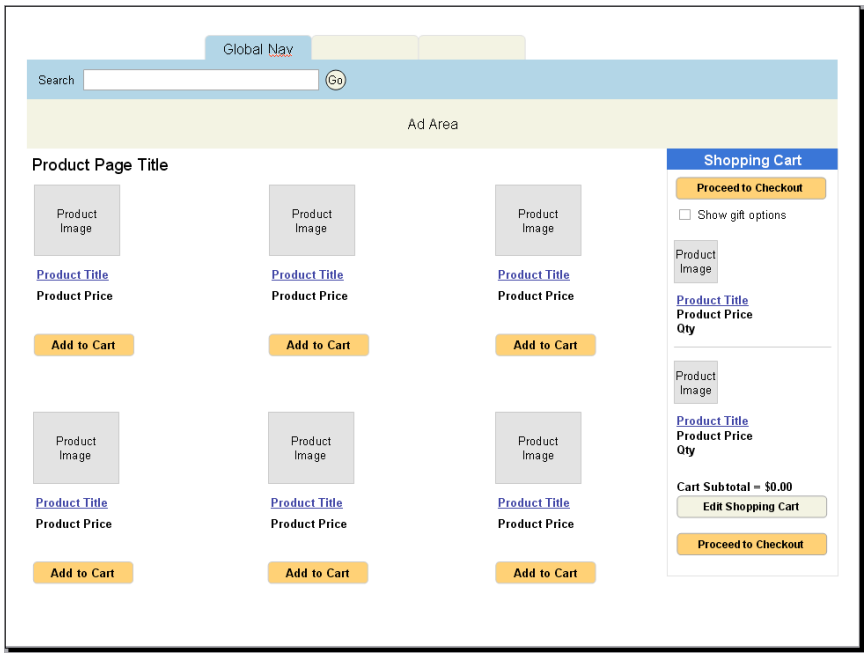
Solution description
The shopping cart is visually offset from the workspace (i.e. Cross-Sell Subpage Pattern) via a graphical treatment and explicit Subpage label (i.e. Your Shopping Cart).

It includes the most recently added item at the top of the list and item data appears with thumbnail where available. The product type is also denoted via an appropriate icon next to the list item.

A trigger for Gift wrapping options is provided in proximity to the control which initiates the Checkout procedure. Also, a subtotal for all items selected for purchase is included as part of the list.

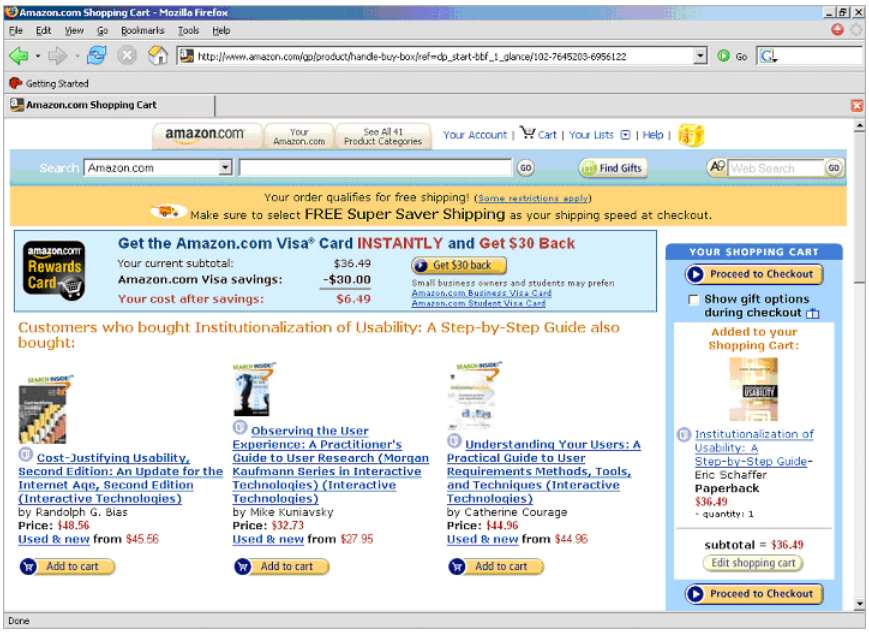
Design rationale
The Shopping Cart List is intended to be obvious and easily understood, in the context of the Shopping Cart Page, but because of the business requirement of cross-selling on that page, the Cart takes up a minimal amount of real estate (25% of the workspace).

Structural diagram(s)



Shown in context of Shopping Cart Page

Known Use



www.amazon.com/gp/product/handle-buy-box/ref=dp_start-bbf_1_glance/102-7645203-6956122

Consequences

Consequences description
Shopping Cart List needs to provide for editing of cart, gift wrap options, and proceeding to Checkout. If Shopping Cart List is not obvious from Shopping Cart Page, then that functionality will be obfuscated.

The Shopping Cart List is juxtaposed to the cross-selling area and must provide for cross-selling and additional shopping, but still be sufficiently intuitive and obvious as to allow users immediate access to functions discussed above.

If list has numerous items, then vertical scrolling will be required and subtotal will be below the fold.

In usability testing, the "Checkout" button was not obvious, so redundant buttons are encouraged.

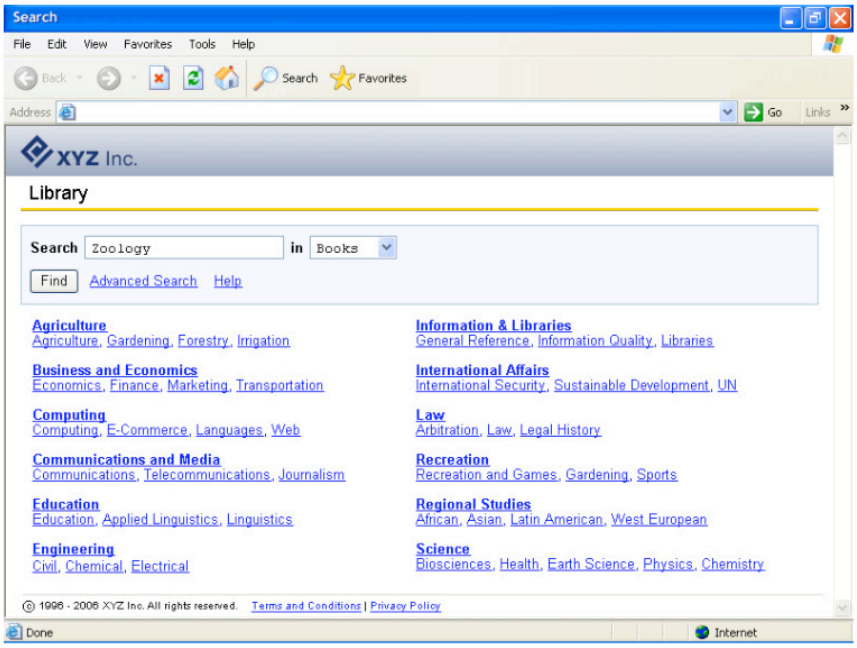
Additional resources
The Design of Sites, Van Duyne et al (pp 369), Published by Addison Wesley

Reusability, usability, and the business case

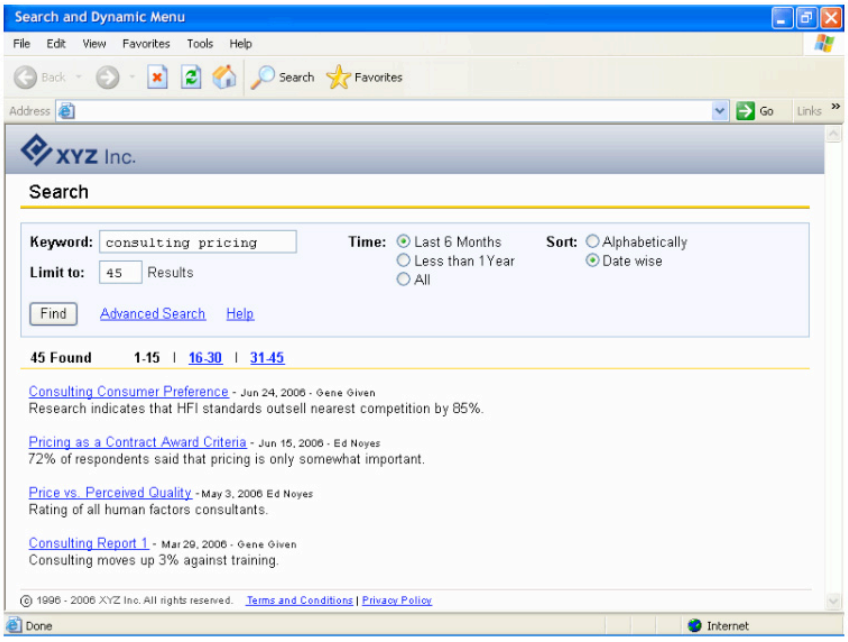
The purpose of a pattern library is to provide reusable design solutions. To accomplish design assembly and a high degree of reuse, a pattern library must be modular. The patterns should be defined at whatever level of granularity is appropriate for a given design environment. Typically, a given pattern is related to smaller scale patterns of which it is comprised, as well as larger scale patterns in which it is a component. This is what Christopher Alexander refers to as "Range of Scale," an important attribute of pattern languages. For example, looking at the example below, consider four implied patterns:

- Header
- Search Area
- Directory (i.e. the hyperlink menu)
- Footer

Those modules are assembled to produce the Search Page, which would be considered a fifth pattern at a higher level of granularity (i.e. a different scale).



In the next illustration, see how the Header, Footer, and a variation of the Search Area are used in combination with another other module (i.e. Search Results) to assemble a different Page Type.



This illustrates a modular approach to design which is found in other disciplines and provides for tremendous benefit to business.

However, in order to provide for reusability within an enterprise, an organization must create a custom pattern library. User interface pattern books and web resources can provide great models for pattern libraries and are useful in regards to giving designers of custom libraries ideas for best practices. But, a custom library which gives an organization a cohesive system of patterns for modular design has the greatest return on investment. Given such a library, designers and developers can assemble user interface designs very rapidly. This approach creates a radically more efficient design and development process. Specific efficiencies include:

- Rapid prototyping
- Early requirements validation
- Reuse in design
- Reuse in development
- Extensible components for future design and development
- Design changes earlier in product cycle
- Lower design and development costs for future projects
- Streamlined migration projects
- Decreased time to market
- Production of corporate knowledge assets

This can be a substantial competitive advantage and is a way of working that can be adopted regardless of the software development method, whether it is agile or waterfall.

If thoughtfully created and with usability in mind as a goal, additional benefits of a patterns approach can be gained by optimizing the user experience. For example:

- Improved overall product usability and quality
- Improved design consistency within and between products
- Decreased customer errors, less time with user assistance
- Increased user productivity and customer satisfaction
- Decreased training costs via improved usability and consistency
- Decreased customer support costs via improved usability and consistency

The definition of what constitutes a pattern and the role of patterns in the discipline of user interface design is debatable. This is a relatively new and immature area in our field. However, it is also a very promising area and an approach that will mature and evolve into tools and

techniques that make design more effective and efficient for practitioners. Some may argue that the rationale for patterns is to provide best practice and improve usability. While that certainly is part of a pattern approach to user interface design, software engineering literature has made the business case for development patterns fundamentally on the basis of reuse. In general, that is indeed also the most compelling argument for a pattern approach to user interface design. Patterns provide for reusability. Usable patterns provide for usability. A mature design and development organization can and should be able to realize the promise of both.

Resources

Books about patterns or related concepts

Alexander, Christopher, et al. (1977). *A Pattern Language*. Oxford University Press, New York, NY. ISBN 0195019199.

Alexander, Christopher. (1979). *The Timeless Way of Building*. Oxford University Press, New York, NY. ISBN 0195024028.

Bateson, Gregory. (1980). *Mind and Nature, A Necessary Unity*. Bantam Books.

Borchers, Jan. (2001). *A Pattern Approach to Interaction Design*. John Wiley & Sons, Chichester, West Sussex, England. ISBN 0471498289.

Fowler, Martin. (1996). *Analysis Patterns: Reusable Object Models*. Addison-Wesley, Reading, MA. ISBN 0201895420.

Gamma, Erich, et al. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA. ISBN 0201633612.

Goldfedder, Brandon. (2002). *The Joy of Patterns: Using Patterns for Enterprise Development*. Addison-Wesley/Pearson Education, Boston, MA. ISBN 0201657597.

Graham, Ian. (2003). *A Pattern Language for Web Usability*. Pearson Education. ISBN: 0201788888.

Mullet, Kevin and Darrell Sano. (1995). *Designing Visual Interfaces*. SunSoft Press/Prentice Hall, Mountain View, CA. ISBN 0133033899.

Tidwell, Jennifer. (2005). *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Media, Incorporated. ISBN 0596008031.

Van Duyne, Douglas et al. (2007) *Design of Sites: Patterns for Creating Winning Websites*. Prentice Hall Professional Technical Reference. ISBN: 0131345559

Wisse, Pieter. (2000). *Metapattern: Context and Time in Information Models*. Addison-Wesley, Reading, MA. ISBN 0201704579.

Selected Web resources

www.mit.edu/~jtidwell/common_ground_onefile.html

www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html

www.cs.vu.nl/~martijn/gta/docs/TWG2000.pdf

hillside.net/patterns/

www.welie.com/

www.enteract.com/~bradapp/docs/patterns-intro.html

www.stanford.edu/~borchers/hcipatterns/chi99/griffiths.pdf

www.boxesandarrows.com/view/implementing_a_pattern_library_in_the_real_world_a_yahoo_case_study

developer.yahoo.com/ypatterns/

developer.yahoo.com/yui/

www.leacock.com/patterns/