Pattern Languages in Interaction Design: Structure and Organization

Martijn van Welie, Gerrit C. van der Veer

Vrije Universiteit, Faculty of Sciences, Department of Computer Science,
Sub section "Human Computer Interaction, Multimedia & Culture",
de Boelelaan 1081a, Amsterdam, The Netherlands, {martijn,gerrit}@cs.vu.nl

Abstract: Now that individual patterns for Interaction Design have started to appear, the issue of structuring collections of patterns into Pattern Languages becomes relevant, both from a theoretical and a practical perspective. In this paper, we investigate how Pattern Languages in Interaction Design can be structured in a meaningful and practical way. A top-down approach is taken where patterns for Interaction Design are organized hierarchically, from high-level design problems to low-level design problems. In addition, the usefulness of additional views and classifications for practical use are discussed.

Keywords: Patterns, Pattern Languages, Interaction Design, Web design, Mental models

1 Introduction

The use of patterns in Interaction Design, or related fields such as web design and GUI design, is slowly gaining momentum in practice. After initial investigations of the applicability of patterns for Interaction Design (Borchers 2001), actual patterns collections are now publicly available in books (van Duyne, 2002, Graham 2003, Borchers 2001) or online (van Welie, 2000, Tidwell 1998). With a total count of more than 250 published patterns, the organization and classification of patterns is becoming a practical issue. Pattern organization is necessary to facilitate the selection of individual patterns but also to find patterns that are applicable in a broader context of any given design problem.

A pattern by itself is just a small piece of the entire design knowledge "puzzle". Each pattern describes a proven solution to a problem in a certain design context. When all the pieces of the puzzle are "put together", we can see how an entire body of design knowledge is unfolded. Understanding this puzzle is the long-term goal in pattern-research. It will show the paved roads of design, but it will also say when the road should be abandoned in search of new and innovative solutions.

2 The idea of a language

An individual pattern may already be very valuable for designers but when patterns are related to each other we can potentially reach a far more valuable thing. Such a set of connected patterns is called a *pattern language*. When Alexander wrote his book on architecture design patterns (Alexander et al 1977), it did not just contain patterns; the patterns formed a language. His language was hierarchical and started out on the level of cities, then neighbourhoods, houses until the level of windows or seats was reached. In Alexander's idea, the language actually "generated" the design by traversing from high level patterns to the lowest level of patterns. From the design of cities down to the design of window seats, a hierarchy of *scale*.

The question is now whether we can create a similar sort of pattern language for Interaction Design. One big difference with architecture is that user interfaces are not strictly hierarchical in a geometrical sense. There is certainly a 2D display involved but what is shown on it varies over time. Therefore, a strict hierarchy based on the usage of screen estate is not suitable for interaction design. However, the hierarchical nature of architectural patterns can also be interpreted as a hierarchy of

problems. The highest level problems are broken up in smaller problems for which solutions appear to exist. They just happen to map directly to a geometrical metaphor in architecture, working from large areas to small areas. The important thing to understand is that such a problem-hierarchy approach can be applied to other domains as well.

For Alexander the 'language' idea was the most important way to structure and relate patterns to each other. However, it is by no means the only possible organization. Other fields that have 'adopted' patterns such as Object Oriented Software Design, use a different organization, categorizing them in *creational*, *structural* and *behavioural* (Gamma et al 1995). That categorization is a pragmatic one that makes sense in that field, although it does not match Alexander's ideas. The question now arises whether we should try to form pattern languages in the Alexandrian sense or whether we should find more specific organizations that suit Interaction Design better. In Interaction Design several alternative organizations have been proposed. Mahemoff (1998) distinguishes patterns for *tasks*, *users*, *user-interface elements*, and *entire systems*. Fincher (2000) and Mullet (2002) also investigated possibilities for structuring pattern languages. However, most of these approaches were developed at a time when hardly any patterns had been written which made it difficult to come up with a sensible organization scheme. More recently, substantial bodies of patterns have been published, e.g. van Duyne et al (2002) which makes the discussion more relevant. Now that we have patterns and we are starting to find out how to write patterns that make sense, we can re-investigate the directions for pattern languages.

3 Connecting patterns

The basic assumption in the concept of a pattern language is that patterns are related to each other, forming a network of *connected* patterns. These relationships are at the heart of the pattern language because they create actual additional value over single patterns. That additional value is the kind of synergy we are looking for when building pattern languages. In the patterns that are publicly available, there are already patterns that 'link' to another pattern in several different ways. A closer look reveals that there are some fundamental relationships distinguishable, resembling the types of relationships known from Object Oriented

Modelling. To illustrate these relationships, we will look at web design patterns as an example.

3.1 Aggregation
Consider the SHOPPING CART pattern. Using this pattern users can manage a list of items in a cart. The cart is actually a persistent list of items on which users can perform some operations such as delete, view, change quantity. This basic behaviour is covered by the LIST BUILDER pattern. Similarly, the checkout procedure is actually just a WIZARD with specific steps such as 'specify delivery address', 'payment selection', 'confirm' etc. The SHOPPING CART is a pattern that *aggregates* several other patterns. This is a form of a "*has-a*" relationship. The SHOPPING CART has a LIST BUILDER and also has a WIZARD.

3.2 Specialization
Patterns can also be *specializations* of other patterns. For example, the ADVANCED SEARCH pattern is basically a SIMPLE SEARCH but with extended options. It "inherits" the basic idea from the SIMPLE SEARCH pattern and extends it with advanced scoping, term matching and output options. We call this a "*is-a*" relationship, one pattern is a more specific version of an other pattern.

3.3 Association
When you are designing the "shopping" experience for a particular site, there are several patterns that may also be of use. For example, when you construct a PRODUCT COMPARISON you could offer the possibility to purchase the item directly from there, using the SHOPPING CART pattern. This is not a "has-a" or "is-a" relationship but simple a "*related-to*" relationship. A pattern may be *associated* to other patterns because they also often occur in the larger context of the design problem, or because the patterns are *alternatives* for the same kind of problems.

4 A pattern language for interaction design

If we try to apply Alexander's idea for a pattern language of scale, we must adopt the interpretation that 'scale' means scale of 'problems' rather than geometry. In Interaction Design there is certainly a 'scale hierarchy' of problems. We may not always be explicitly aware of it but it is the hierarchy we mean when talking of top-down design. Usually design is a top-down activity where we start with gaining understanding of the users and their tasks, the client's wishes, technical environment, business

context etc. Taking the example of web design again, design continues by laying out the foundations of the application in terms of the site concept, information architecture, and basic functionality. The concept outlines the basic characteristics of the site that will be filled in later on up to the point where individual screens and widgets are laid out. Such a top-down approach will 'generate' a design when patterns are available at all levels. This network of patterns uses all three kinds of connections between patterns. In the patterns themselves the type of connection is usually not made explicit but it is simply embedded in the pattern in a natural way.

When looking at such a networked set of patterns, we can also see layers of patterns emerging, when going from high level patterns to lower level patterns. These layers are rough delineations of the typical levels that are encountered in design. The levels we have identified so far are *posture*, *experience*, *task* and *action*.

4.1 Posture type patterns

Every site or application is there for a *purpose* or has a reason for existence, for commercial sites there are usually business goals to be achieved while other sites have more *personal* or *social* goals. Proper design has its foundations in understanding why the design project is started in the first place. These stated business goal feeds into the choice for a 'kind of site' that is adequate and effective.

From experience we know that many sites are actually quite similar in the sense that they serve the same goals and have a structurally similar site concept. This can be called the site's '*posture*' (Cooper95), 'genre' (van Duyne et al 2002) or '*type*'. For example, small corporate sites are often so similar that we can write patterns describing them. The same goes for news sites, community sites and so on. We can define several of such site postures that can be taken as a basis for new site design projects. Patterns that describe such typical site postures are therefore called posture patterns. Van Dyne et al (2002) also describe several of these site postures.

A posture pattern describes what the essentials of that posture are: what kind of site structure is usually used, which elements typically make up the homepage but also the main *experiences* that such a site is supposed to offer. It is like deciding whether you are going to design a 'sports car', a 'family saloon car', a '4x4' or a 'city car'. Each of these has specific characteristics and experiences that together form a type of car.

Many sites can be directly derived from the known postures but it is also common to design a site as a mix of postures. When a posture has been selected, several lower-level posture patterns will help to define concept level issues such as homepage design, promotion areas, navigation, templates etc. User research or contextual inquiry will help designers to decide which postures are most relevant.

4.2 Experience patterns

From the basic posture and from user research, designers will have to determine what are the *main user goals and tasks* that need to be supported and to what extent. We will call this the '*experience*'. The user experience is not just about tasks and goals but also about how the users reach their goals using a site concept, how they perceive the site and whether it gives them the appropriate satisfaction. Experiences should therefore be understood as a broader goal for which we are designing. The experience-level patterns describe common experiences and which lower level patterns can be used to create that experience. Typical experiences are activities such as "shopping", "playing", "browsing", "information gathering", "problem solving" or "sharing thoughts". When describing for example 'shopping', it is necessary to specify what it is without taking into account the technology we are using. When we understand how shopping works we can then add references to lower level patterns that can be used to create the experience. See Figure 1 for an excerpt of our shopping pattern where we have tried to summarize the most important aspects of shopping and have listed what lower-level patterns can be used to implement them.

Experiences are the high level goals for which the users come to a site. When applying it to car design, experiences can include 'sporty driving behaviour' or 'luxury feeling'.

Shopping Experience

From www.bn.com

Problem	Users want to look for products of interest and potentially purchase them
Use when	You are building a web site where you sell products, typically an e-Commerce site but it can also be a site with paid content. The sort of products that you are trying to sell may vary a lot, ranging from books, electronics, to holiday and clothes. Some products can be delivered directly by downloading it and others will have to be delivered 'later' by some logistical process. No matter what product you are trying to sell, there are well known aspects to shopping that apply to all products and to all ways of shopping.
Solution	**Create an online shopping experience that matches off-line shopping experiences**

Shopping involves several fundamental activities that apply to both online and offline shopping activities. These activities needs to be supported for each type of product and domain. How to do that best is largely domain dependent, but some basic ideas can be defined:

- **Discovering**. People need to know what they can buy in the store, as far as they don't already know it. Even if they have been in the store before they need to be informed of new products that are for sale. Even if there are no new products to sell, there may be products that should be brought under the users attention because of other reasons e.g. because they are discounted, very popular etc. Use Hotlists

- **Browsing**. Most people like to browse through the store for seeing what they have and whether something attracts their attention. Browsing is made easier when products are categorized in ways that customers expect them to be. The categories allow them to browse in a specific manner that is a bit more directed than no structure at all. Use structured navigation such as a Double Tab with Breadcrumbs so that people are fully aware of where they are and where they can go to.

- **Comparing**. Often people do not know exactly which product they want. They may have several options that they want to compare using a Product Comparison or Product Configurator.

- **Trying**. When people try a product they want to make sure it is the right product for them. Trying is all about 'seeing' certain aspects of the product. In many cases it is even possible to 'interact' with the product by 'virtually touching it', seeing close-ups, table of contents or a preview of a part of the object. Sometimes it may also be possible to try the real thing with some limitations on the use of it. In other words, create a Virtual Product Display

- **Asking Opinions**. Many shops have shop assistents that help customers to find the right product for them. Online this is difficult to achieve but one could create Product Advisors or collect recommendations/ratings/comments of other people that bought the product.

- **Choosing**. Choosing is not the same as buying. Customers may choose several products and before they actually start buying, discard several of them at the last minute. Give them a place to keep products they may want to buy such as a Shopping cart or wish list

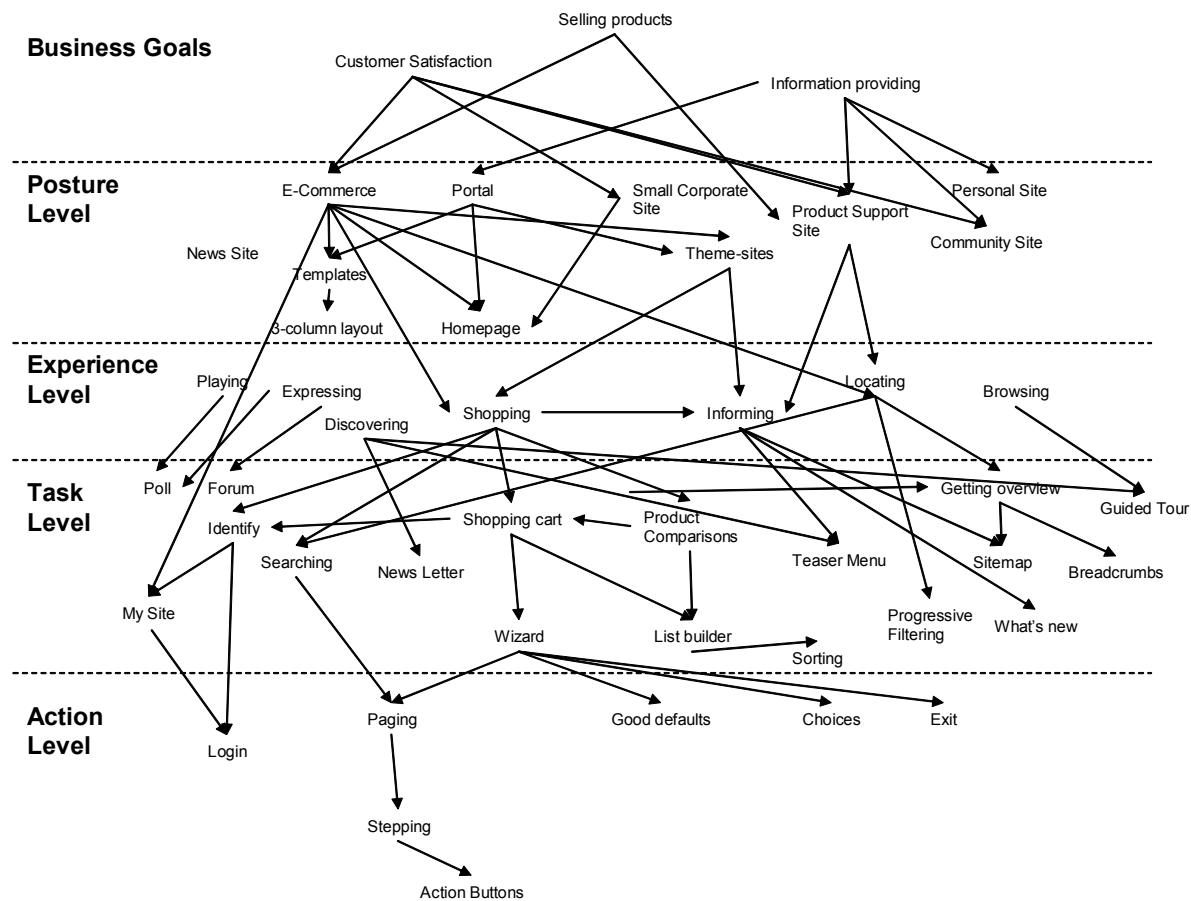**Figure 1:** An excerpt of the "Shopping" experience pattern

Figure 2: A partial pattern language for web design (centred around "shopping")

Every site type has a primary experience that it wants to offer. For example, an e-commerce site is primarily for a *shopping* experience. However, secondary experiences may include *community-building* (between buyers) or information gathering (about the products). Interaction designers need to balance these experiences and create a consistent user experiences for the entire site.

In practice, this will mean that an e-commerce site will use some elements from secondary experiences. In a similar way, a news site may use elements from a shopping experience for dealing with premium paid for content.

4.3 Task patterns
The task level is the level where we start to see most concrete and well-known patterns such as SHOPPING CART or PRODUCT COMPARISON. These will point to lower-level task patterns such as WIZARD or LIST BUILDER that are needed in high level task patterns. Task patterns are describing solutions to small user problems that are part of a higher level "experience". Typically a task pattern describes a series of interactions on one or more objects for solving a problem. Such a series corresponds to a task sequence needed to achieve a task goal. Task patterns are relatively domain independent. The posture and experience patterns set the context specifics and the task patterns are used to fill in the blanks. Task patterns can often be 'drawn' using flow diagrams and sketches.

4.4 Action patterns
Action level patterns are not really related to a clearly defined user goal. A PUSHBUTTON or CLEAR EXITS are actions that are only meaningful in real tasks such as "order", "go the next step" etc. We call these "action patterns" and they are often similar to widgets. They occur is almost all task patterns and are the lowest level of building blocks we still want to call a pattern. The solutions described in them are usually specific uses of well known widgets or describe custom-made widgets.

The different levels and associated patterns can be shown in a graph of connected patterns, see Figure 2. In the graph all types of inter-pattern relationships

are shown using a directed arc. Actually, the relationships are also contained in the patterns themselves, every time a reference is made to other patterns as part of a context or solution statement. Figure 2 only shows a partial graph centred on the shopping experience. Because of the complexity of the domain, a complete graph could have more than 250 connected patterns leading to a, perhaps not very comprehensible, diagram.

5 Pattern languages as mental models

Although patterns describe proven solutions seen in every day products, recognizing them as patterns and structuring them takes substantial experience. Entire languages therefore capture the knowledge of the designers that wrote the patterns and make that knowledge accessible to others. Novice designers have a very limited pattern language in their knowledge repertoire and as designers become more experienced the scale and complexity of the pattern language they use increase. Pattern languages are definitely 'living' things.

A pattern language can be seen as a *mental model* (van der Veer & Puerta Melguizo, 2002) that a designer has. Writing down design knowledge using a pattern language is an activity of making a structured explicit representation of ones mental model. One can wonder to what extent experts have different mental models and therefore would write different patterns and pattern languages. Already today we see that patterns with the same name, but written by different designers, differ in terms on actual content. In addition, patterns may differ in the scope they take and the priorities on specific issues as described in the patterns.

Although there are differences in the mental models of designers, the process of making them explicit is likely to lead to convergence of their mental models. When designers will have 'access' to the externalized mental models of others through these pattern languages, they will learn from each other and adjust their own mental model of the field. Therefore, a certain amount of convergence is expected to follow though there will always be space for more personal views. The implications of these observations are that we should be aware that pattern writers *will* write different patterns and that it will not be easy to converge on a single pattern language for Interaction Design in the near future.

6 Towards 'complete' languages

Now that we have reached the point where many patterns are available, one may wonder how many patterns will need to be added in order to make a language 'complete'. On the one hand, it is not likely that we can state an objective criterion for when languages are complete since they only describe knowledge from a select group of writers. On the other hand, we can expect some convergence and need to find a way to discover missing patterns and acknowledge patterns that have already been written.

Therefore, one criterion could be that completeness is reached when the available patterns can account for all different qualitatively good designs one can find. In other words, when every 'usable' website out there can be described using a set of patterns, the language is complete. We say 'usable websites' because we are only interested in describing 'good' design. Alexander (1977) states that a pattern language is good when it is 'morphological complete'. However, whenever new good sites start appearing the language may turn out incomplete again. Nonetheless, it gives us a practical method for mining patterns and perfecting pattern languages.

7 Organizing patterns for practical use

Connecting all patterns into a pattern language is one way of organizing them. A language can be depicted as a graph showing all pattern names and connections, see Figure 2 for a partial example of such a graph. However, in practice when designers or engineers need to search for patterns for a particular problem, the graph may not be the best representation. The graph shows the fundamental relationships but there are many other practical ways in which patterns from a collection can be classified.

One other organizing principle, is by *function* or problem *similarity*. The idea here is that we group patterns according to their functional aspects. Certain groups of patterns may all deal with a common problem and therefore group together. Designers often need to make a decision about a functional aspect and may be best served by a set of patterns that can be classified as belonging to that functional aspect. Functional aspects may include *navigation*, *searching*, *product display*, *layout* and so on. Most existing pattern collections use such an organization. See Table 1 for an example that shows

the organization we currently use in our collection of patterns, see www.welie.com/patterns.

Another organization principle is based on usability *defect*. For example, when there is a problem in certain task sequence a designer needs alternatives for making the task execution time decrease. In that case, designer may want to filter on patterns that may increase entry speed or have an impact on the error rate.

Clustering by *user task* and *user type* might also be a relevant organization principle. We could have patterns that deal with selecting things, finding things, sorting, creating things for novice users, intermediate users or expert users.

In practice designers often work on a particular site posture and may be interested only in patterns that apply to such sites. For example, when working on an e-Commerce site, the collection can be filtered to show only the patterns that are 'connected' from the e-commerce site pattern. In this paper we have used web design as the domain for constructing a pattern language but it is also possible to create a similar language for GUI design or for designing interface for mobile devices.

For practical use several kinds of pattern classifications may turn out to be useful. These are merely different 'views' on a language while the fundamental pattern relationships are still being respected since those are embedded in the patterns themselves. The possible views are largely built using certain 'attributes' of patterns or the pattern fields themselves.

8   Tools for pattern languages

Since it is clear that there are several useful ways to organize patterns, designers should not be forced to choose one particular *view*. A logical consequence is that there is a need for tools to make patterns accessible in more than one way. Tools can generate different views or offer dedicated search functionality for selecting appropriate patterns. A web-based tool environment is probably best suited for the task since a pattern language itself already consist of hyperlinked patterns that allow users to go from one pattern to another.

In addition, tools can assist in developing a pattern language. Pattern writers should be able to contribute patterns or comment on existing patterns.

Pattern 'users' could form a community that evaluates patterns and helps others in discovering new patterns or other examples of pattern usages.

Site types	User Experiences
My Site Portal Commerce Site Community Site Branded Promo Site Corporate Site News Site Brochureware Site	Shopping Community building Learning Document retrieval Entertainment
Navigation	**E-commerce**
Bread crumbs Double tab Meta Navigation Split Navigation Repeated Menu Progressive Filtering Teaser Menu Combined Menu Fly-out Menu Directory Trail Menu Scrolling Menu Shortcut Box Image Menu Guided Tour	Shopping cart Identify Registering Product Comparison Product Configurator Product Advisor Premium Content Lock FAQ Newsletter
Page Elements	**Searching**
News box Home Language Selector Hotlist Customization Window Favourites Poll Footer Bar Outgoing Links	Simple Search Advanced Search Search Area Sitemap Topic Pages Search Tips Search Index
Basic Interactions	
List builder Tabbing Paging Wizard Parts Selector Sorting Enlarged Clickarea	

Table 1: An example of a patterns classification

The potential users of such tools can be quite diverse, ranging from software engineers, interaction designers, visual designers, project managers to evaluators and clients. Each of these will have their own requirements for tool support, either in the views that are supported or concerning the pattern development functionality. Several projects are already underway that investigate tool support for patterns. For example, the UPADE tool (Javahery & Seffah 2002) is a tool where designer can create designs directly using patterns using drag-and-drop like functionality.

9 Conclusions

Creating pattern languages rather than pattern collections offers significant added value. We have described a way to apply the concept of a pattern language in Interaction Design using Web Design as an example. Our approach follows a top-down design methodology where high-level design problems are gradually decomposed into smaller design problems. Besides the proposal for a language for Interaction Design, we also argued that the structure of the pattern language must be seen separate from the different views we can have of the collection of patterns. Such views can be of use in different design contexts and should therefore be supported by tools. Tools should facilitate the use of patterns in practice by a variety of target users, not just designers but also engineers and other stakeholders in the design process.

We predict that the concept of a pattern language with proper support will be one of the most effective design knowledge management tools available. In order to substantiate this claim we need to write well-structured pattern languages with high quality patterns that can be accessed through tools with multiple ways to find and select patterns. Only an evaluation of such a system can truly support claims concerning the effectiveness of patterns in Interaction Design.

References

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I. & Angel, S. (1977), A Pattern Language, Oxford University Press,New York.

Borchers (2001) A Pattern Approach to Interaction Design, John Wiley & Sons; ISBN: 0471498289

Cooper, A. (1995) About Face: The Essentials of Windows Interface Design, John Wiley & Sons Inc; ISBN: 1568843224

Dearden, a., Finlay, J, Allgar, E. & McManus, B. (2002) Using Pattern Languages in Participatory Design. In Binder, T., Gregory, J. & Wagner, I (Eds.) PDC 2002, Proceedings of the Participatory Design Conference. CPSR, Palo Alto, CA.,2002. ISBN 0 9667818-2-1. pp. 104 - 113

van Duyne, D.K., Landay, J.A, Hong, J. 2002, The design of sites, Addison-Wesley, Boston, US.

Fincher, S., Windsor, P. (2000), Why patterns are not enough: some suggestions concerning an organising principle for patterns of UI design, position paper for CHI2002 workshop Pattern Languages for Interaction Design: Building Momentum

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995), Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Reading, Mass.

Graham, I. (2003), A pattern language for Web Usability, Addison-Wesley, Boston, US.

Javahery, H., Seffah, A. (2002) A Model for Usability Pattern-Oriented Designs, proceedings of TAMODIA 2002, July 18-19 2002, Bucharest, Romania.

Mahemoff, M. J. and Johnston, L. J. (1998). Pattern Languages for Usability: An Investigation of Alternative Approaches. In Tanaka, J. (Ed.), Asia-Pacific Conference on Human Computer Interaction (APCHI) 98 Proceedings , 25-31. Los Alamitos, CA: IEEE Computer Society.

Mullet, K. (2002), Structuring Pattern Languages to Facilitate Design, position paper for CHI2002 workshop "Patterns in Practice".

Tidwell, J. (1998), Interaction Design Patterns, in 'Proceedings of the Pattern Languages of Programming PLoP'98'.

Van der Veer, G.C. & Puerta Melguizo, M.C. (2002). Mental models. In: J.A. Jacko & A. Sears (Eds.) The Human-Computer Interaction Handbook: Fundamentals, evolving Technologies and emerging apllications. Lawrence Erlbaum & Associates. ISBN: 080583834. pp. 52-80

van Welie, M., van der Veer, G. & Eli¨ens, A. (2000), Patterns as Tools for User Interface Design, in 'International Workshop on Tools for Working with Guidelines', Biarritz, France, pp. 313–324.