The Anatomy of HCI Design Patterns

Christian Kruschitz
Department of Informatics-Systems
University of Klagenfurt
Klagenfurt am Wörthersee, Austria
chris@isys.uni-klu.ac.at

Martin Hitz
Department of Informatics-Systems
University of Klagenfurt
Klagenfurt am Wörthersee, Austria
hitz@isys.uni-klu.ac.at

***Abstract*—Human-Computer Interaction design patterns became an often-used tool in the HCI community to encapsulate and share design knowledge. A good design pattern consists of several different components. There are the *content elements*, which encapsulate the description of the problem, context and solution. *Relationships* between patterns are an essential part to best profit from the full reuse potential promised. *Categorization* of patterns helps to navigate more effectively within large pattern repositories, and *formalization* of HCI patterns provides the possibility to use them in software applications such as pattern management tools or user interface design tools. This work gives a brief overview of the above-mentioned components.**

Usability; HCI patterns: History, Organization, Relationships, Pattern Management Tools, Standardization

I. Introduction

Over the past 13 years, Human-Computer Interaction (HCI) design patterns have become an important and frequently used tool for reusing and exchanging design knowledge in the domain of HCI [13]. The cornerstone of design patterns as a tool of knowledge management was laid back in the late 1970s when the mathematician and architect Christopher Alexander published several books [3, 4, 5, 6] in which he proposed the concept of design patterns and pattern languages. Although this concept was originally meant to support reuse of architectural design knowledge, Ward Cunningham and Kent Beck have adopted this principle to object-oriented-programming (OOP) and user interface (UI) implementation in 1987 [8, 33]. They presented five patterns for designing window-based user interfaces in Smalltalk.

Design patterns made their breakthrough in the software engineering community when Gamma et al. published one of the bestselling books in software engineering, “Design Patterns: Elements of Reusable Object-Oriented Software” [19]. From this time on many design patterns and pattern languages in software engineering as well as user interface engineering have been published.

In Human-Computer Interaction, the start of the design pattern era was 1996 when Coram et al. [13] published the first design patterns of a pattern language for user-centered interface design. Their intention was to provide high level patterns with which user interface designers could build graphical user interfaces which are pleasurable and productive to use. In 1997 the first CHI workshop on pattern languages in user interface design was organized. The participants explored the use of a pattern language in user interface design to make HCI knowledge reusable in different applications [7]. At this time user interface toolkits have emerged to support user interface designer and software engineers in order to avoid “reinventing the wheel” again and again. However, the workshop participants stressed that a more general description of user interface design know-how, which is detached from a specific implementation platform, would be desirable and agreed that design patterns could be an appropriate tool. Design patterns reside on a higher level of abstraction than UI toolkits and are not bound to source-code for a specific implementation of the addressed problem. Furthermore, patterns are written in such a general way that they give pattern users the possibility to decide how specific widgets should be arranged to concretize the patterns’ solutions.

In the following years workshops were held [7, 9, 18, 22] and design patterns and pattern languages in the domain of HCI were published [10, 12, 13, 15, 17, 21, 34, 37, 39].

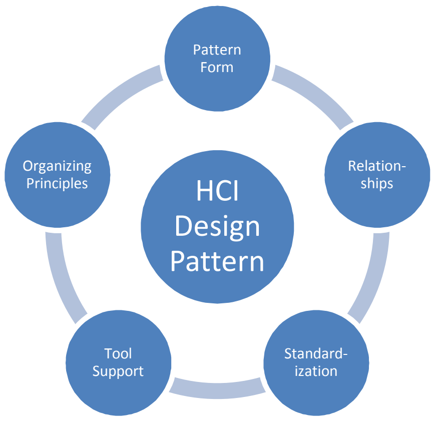

Figure 1: Components of HCI Design Patterns

In the following section, we define the terms *HCI design pattern*, *design pattern catalogue/collection*, and *pattern languages*. Furthermore, we describe the components (see Figure 1), which are part of HCI design patterns and describe the research and approaches, which were undertaken in the last years. However, we can only give a rough, non-exhaustive overview.

II. Definition

A. HCI Design Pattern

An HCI design pattern describes a recurring problem together with a proven solution. An HCI design pattern, in the following referred to as "pattern" or "design pattern", has a well-defined form, which is dependent on the individual author's preferences. A pattern form should be used consistently across a pattern language or pattern collection. This makes it easier for pattern users to understand the problem, context, and solution of a pattern throughout a pattern collection/language. The pattern itself, when it is part of a collection or a pattern language, may have references to other patterns.

B. Design Pattern Catalogue/Collection

Patterns are stored in design pattern catalogues or collections. The patterns in such a catalogue are categorized to support faster navigation within the repository. In this case, patterns show almost no relationships among each other and thus do not form a fully interconnected system. There are several patterns which stand alone and have no connections to predecessor or successor patterns. Furthermore, such a collection usually does not completely cover a specific application domain.

C. Pattern Language

In contrast to a pattern catalogue / collection, a pattern language is a complete collection of patterns for a given family of design problems in a given domain. A pattern language describes problems by means of high-level design patterns, which are solved by low-level design patterns. The design patterns are connected through relationships, so that they constitute a network.

In a pattern language, the "words" are the patterns, while the connections between patterns represent the "rules of grammar" which are situated in the pattern itself. When words and rules of grammar are combined, a "sentence" is generated. Sentences can be built in many different forms when the rules of grammar are followed. So there is not only one path through a pattern language, it offers several possibilities to solve a design problem. A good example is "The Design of Sites" by van Duyne et al., a pattern language that allows designers to articulate an infinite variety of Web designs [15]. Figure 2 visualizes a part of a pattern language with focus on online shopping [36].

III. Pattern Form

Patterns are written by researchers and UI designers in a well-defined format, the pattern form. This form is dependent on the author's preferences but several basic forms have been established in the history of design patterns. Those are described below in more detail.

A. Alexandrian Form

Christopher Alexander has invented the concept of design patterns as a problem/solution pair and presented them in a common format [3], which consists of:

- **Picture:** Shows an archetypal example of the pattern in use.
- **Introductory Paragraph:** This sets the pattern in the context of other, larger scale patterns.
- **Headline:** A short description of the problem.
- **Body:** Detailed description of the problem.
- **Solution:** The solution of the pattern which is written as a design instruction.
- **Diagram:** Shows the solution in the form of a diagram.
- **Closing Paragraph:** It gives references to other patterns and describes how this pattern fits with other, smaller patterns.

This pattern form is used with minor changes by Todd Coram and Jim Lee [13], Jan Borchers [10], Ian Graham [21], Mark Irons [24], Douglas van Duyne et al. [15], and Eric Chung et al. [12].

B. UI Pattern Form

This pattern form was developed at the INTERACT patterns workshop in 1999 [22]. It comprises seven content elements:

- **Name**: Shortly describes the pattern's intent.
- **Sensitizing Example**: This component should sensitize the reader to the application of the pattern. It is usually a screenshot or drawing of the pattern's solution.
- **Problem Statement**: Describes the conflicts (trade-offs) between "forces" guiding the design approach.
- **Body:** Textual description of the pattern's intent.
- **Solution Statement**: Tells what to do (and not how to do it).
- **Technical Representation:** This example is more detailed and intended to inform HCI experts about the pattern's solution.
- **Related Patterns:** References to "successor patterns" which enhance or are similar to the pattern.

C. Tidwell Form

Jenifer Tidwell is using a very minimalistic form, which is used throughout her book "Designing Interfaces" [34].

- **Name:** Describes the pattern's intention and defines a unique reference number.
- **Sensitizing Image:** This image sensitizes the reader to the pattern's solution.
- **What:** Short problem statement.
- **Use When:** Describes the context in which this pattern can be used.

- **Why:** Describes the design rationale.
- **How:** Represents the solution part of the pattern.
- **Examples:** Screenshots of instantiated pattern with a short description.

D. Other Pattern Forms

In our view, the above mentioned pattern forms have the greatest impact in the domain of HCI design patterns. However, there are other influential approaches stemming from the software engineering domain, which are briefly mentioned below:

- The Gang of Four Form [19]
- The Portland Form [31]
- The Coplien Form [1]
- The POSA Form [11]

More pattern forms, which were recently used, can be found at Sally Fincher's portal [17].

IV. ORGANIZING PRINCIPLES

Alexander has organized his pattern language into levels of physical scale. He starts with high-level patterns which describe the size and distribution of towns and proceeds to low-level patterns which describe individual rooms [3].

In analogy, an organizing principle for HCI patterns, as Fincher and Windsor mentioned [18], should allow users to find patterns they need within a large repository. It would be advantageous to support users to consider the problem from different points of view and allow for building new solutions, which have not been previously considered. Fincher and Windsor have adapted the Alexandrian structure of scale to UI design, starting with a high-level category *Society*, and descending via *System*, *Application*, *UI Structure* and *Component* to the low-level categories *Primitive* and *Physical Detail*. However, they do not consider this categorization sufficient for UI designers to find a pattern for their problem. So they suggested a second and a third structure. The second is based on the design-by-type-of-task, where they have defined tasks based on the information flow, which includes categorizations such as *Task-Retrieval*, *-Monitoring*, *-Controlling*, *-Construction* and others. The last categorization structure comprises categories such as *Volume*, *Complexity*, *Structure* and *Dynamics*.

Another approach has been put forward by Mahemoff and Johnston [29]: UI patterns can be assigned to four different categories. First, the *Task* category comprises all patterns addressing actions users might perform. Second, the *User Profile* category gathers patterns focusing on user groups. Third, *User-Interface Elements* helps designers and programmers to understand when to use a specific interface element or widget. Finally, *Entire System* patterns capture the issues of specific kinds of systems.

Van Welie and van der Veer [36] are organizing their patterns by means of "scaling the problem". As design is considered a top-down activity, their categorization is top-down as well. Problems are scaled from high-level problems like *Business Goals* to more detailed problems like *Task Level* and *Action Level* as shown in Figure 2. Other possibilities to scale or group design patterns suggested by van Welie and van der Veer are to organize them according

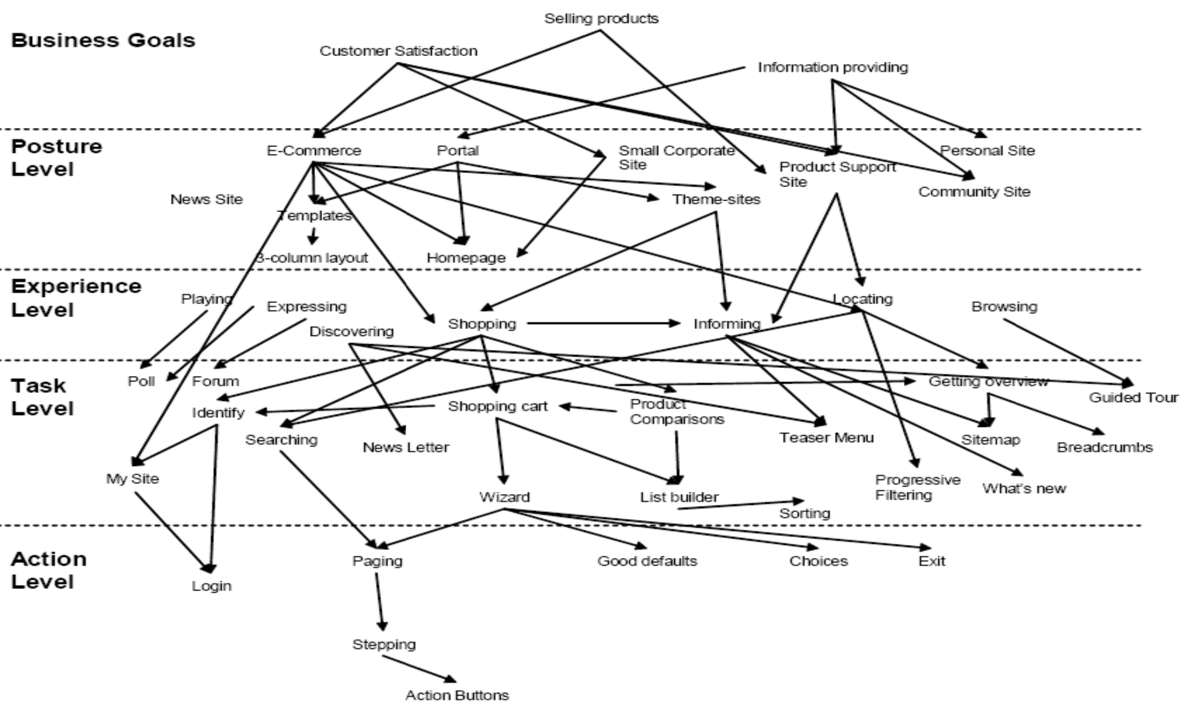

Figure 2: A part of a pattern language for Web design with focus on shopping [36]

to their *Function* or to *Problem Similarity*, where *Function* can be subdivided into *Navigation*, *Searching*, *Product*, *Display*, *Layout*, and other sub-categories.

V. RELATIONSHIPS

Relationships between design patterns are a key concept to gain the full reuse potential of individual design patterns. In HCI patterns, relationships are typically described very briefly, only specifying the connections to other patterns which may be applicable to a particular design problem. However, proper consideration of relationships promises even more powerful search and navigation opportunities. In the past, software engineering researchers have proposed possible categorizations approaches for relationships in the domain of OOP patterns.

Primarily, relationships help to understand the complex interdependencies among design patterns. Pattern users can use relationship information in addition to the aforementioned pattern classifications to identify patterns which are applicable to specific design problems. Furthermore, relationships can be exploited for browsing large re-use repositories [23]. To improve and quicken the finding process, the browsing paradigm can be combined with the search paradigm as well. First the user searches for a specific design problem and based on the query results it is possible to browse through the repository to identify the best matching solution.

Noble's [30] proposal consists of three primary relationships (a pattern *uses* another pattern, a patterns *refines* another pattern, a pattern *conflicts* with another pattern) and a number of secondary relationships such as *used by*, *refined by*, *variant*, *similar*, *combines*, and others.

Zimmer's [40] approach deals with the classification of relationships in Gamma's [19] design patterns collection. He classifies relationships into three categories: a pattern *uses* another pattern in its solution, a pattern *is similar* to another pattern, and a pattern can be *combined* with another pattern. Beside this categorization, it is possible to modify existing relationships to use their altered version between different patterns. The application of categorized relationships allows to structure patterns in different layers. Zimmer has identified three semantically different layers: *basic design patterns and techniques*, *design pattern for typical software problems*, and *design patterns specific to an application domain*.

VI. TOOL SUPPORT

There are various tools which are exploiting the reuse potential of HCI design patterns. These tools can be categorized into online libraries/catalogs, pattern management tools, and pattern-based UI design tools. Due to space limitation we describe, in our mind, the most important ones.

A. Pattern Libraries/Collections

Pattern libraries or collections are focusing on the categorization and dissemination of patterns via the Internet. Sometimes there are basic mechanisms provided to create and submit patterns to a repository. Such an online pattern library is the *Yahoo! Design Pattern Library* [39, 27], a part of the Yahoo! Developer Network. The founder's intention of the Yahoo! Library was to provide a tool to increase the consistency and usability across Yahoo! and the productivity of the UI design team. Today the design pattern library is an often-used tool for UI designers and researchers. Currently, there are 47 patterns in six categories available. Each pattern undergoes an extensive review process within Yahoo!. They are reviewed, revised, and rated. After review, the patterns get published and made available to the public. All patterns in this library are under the Creative Commons Attribution 2.5 License (June 2009). Main features of the library are:

- A *blogging tool* for discussing patterns in the library.
- A *history function* which helps users to see which changes were made over time.

Further online pattern libraries are Welie.com [37], which provides 130 UI design patterns with the possibility to export them to PLML [16]. Furthermore, website users can comment and discuss certain patterns.

As an addition to her *Designing Interfaces* book [34], Tidwell provides a pattern library with selected patterns where she updates and publishes new patterns. This library is available at [35].

B. Pattern Management Tools

Pattern management tools are focusing on manipulating patterns, navigating through pattern libraries, and providing mechanisms to add relationships between patterns to create a pattern language. They are easy to access and pattern users can communicate with others via the pattern repository.

MOUDIL (Montreal Online Usability Patterns Digital Library) [20] is a comprehensive framework for capturing and disseminating patterns. It provides features and tools like:

- Submission of patterns in *different formats*.
- International *review and validation* of submitted patterns.
- A *pattern editor* for adding semantic information to the patterns.
- A *pattern navigator* which allows navigating in different ways through the pattern library.
- A *pattern viewer* which provides different views of the pattern.

Unfortunately, the prototype of this pattern library is not longer available online.

Currently under development is another online pattern management tool which employs XPLML [26], an improved version of PLML. XPLML provides a set of common content elements, and it is possible to add semantic information to design patterns. The tool will offer features such as:

- A *pattern editor* with functions to support pattern authors in writing and updating design patterns.
- A *design pattern language visualization tool* for presenting relationships between patterns in a pattern language.
- The *pattern form transformation* allows pattern users to change the presentation form of a design pattern. For example, if a user prefers the Alexandrian form, the tool provides mechanisms to change the pattern form from e.g. Tidwell's to the preferred (Alexandrian) form in order to maximize user acceptance.
- A *wiki functionality* which should involve all interested users in developing new and improving existing patterns.

C. *Pattern-based UI design tools*

The last category describes pattern-based UI design tools. They provide functions for using design patterns in UI design activities. Patterns are used for generating user interfaces in a semi-automatic way. These tools usually provide a defined set of UI patterns, which can be used within the tool as building blocks to create the UI system.

PIM (Patterns in Modeling) [32], a model-based UI development tool, aims to support UI designers in composing the UI models through pattern application. With *PIM* it is possible to develop user interfaces on a more abstract and conceptual way. This helps designers to handle very complex systems more easily. Users can put their attention on conceptual properties rather than being distracted by technical and implementation details.

A further tool, developed by Ahmed and Ashraf, is called *Task Pattern Wizard* [2]. It is based on XUL (XML User Interface Language) [38] to describe the patterns and models. UI design patterns are used as modules for establishing task, dialog, presentation, and layout models. The tool guides the UI designer through the pattern adaption and integration process and it provides functions for using, selecting, adapting, and applying patterns within the proposed framework PD-MBUI (Pattern-Driven and Model-Based User Interface). The framework tries to unify the pattern-driven and model-based approaches, two methods for UI and software engineering. A more detailed description of the framework is given in [2].

DAMASK, developed by Lin and Landay [28], is a prototyping tool to produce Web UI's across different devices with the support of design patterns. The tool relies on two components. The *layer component* specifies which parts of the UI can be used across all devices and which can only be used on a single device. The second component is the *pattern component*: In *DAMASK,* an HCI design pattern consists of pre-defined UI elements that are optimized for each device. The pattern repository of *DAMASK* has 90 patterns from "The Design of Sites" [15] which can be extended by the UI designer. The UI designer sketches out a UI for one device and *DAMASK* constructs an abstract model from which it generates the UI's for the other devices. Once the first layout is established, the UI designer can refine this layout and *DAMASK* changes the UI's for the other devices accordingly. Furthermore, the tool provides a function for testing the established UI's.

VII. Standardization Approaches

To our knowledge, the only serious standardization approach was developed at a workshop at a Human-Computer Interaction conference in 2003. Due to the vast amount of design pattern forms, Fincher et al. [16] proposed a standard pattern form for HCI patterns called PLML (pronounced "pell mell"). The goal was to provide a standard pattern form where common elements should help pattern authors and users to use design patterns across different collections. PLML is specified in XML and comprises 16 content elements on which the workshop participants agreed. Research showed that only van Welie's pattern collection [37] makes use of PLML. He provides an export function to transform patterns from his collection to PLML. However, this approach suffers from certain technical limitations as Kamthan points out [25]. He mentions that the design principles behind the PLML DTD are not specified and that elements are not strictly enough defined, because of the broad use of the XML `ANY` element in the specification. Kamthan also points out that PLML does not describe semantic relationships between patterns, which are necessary when using PLML in a pattern language.

Since the publication of PLML, researchers tried to improve it. PLML v. 1.2. developed by Deng et al. [14], is an augmented PLML with some additional elements but does not solve serious shortcomings such as the lack of formalized relationships among patterns.

VIII. Conlusion and Future Work

The concept of HCI design patterns is widely accepted tool to represent design knowledge in a reusable format in order to avoid "reinventing the wheel" again and again. However, in the last years many concepts concerning the components of HCI patterns were proposed, such as pattern forms, organizing principles, and standardization approaches. This diversity leads to blurred conceptualization and may confuse especially novice users. To exploit the full reuse potential of patterns, a unification of the above discussed components should be established, which does not constrain pattern authors in their work but supports pattern users by easing understanding and instantiation of patterns to specific design problems [26].

REFERENCES

[1] M. Adams, J. Coplien, R. Gamoke, R. Hanmer, F. Keeve, and K. Nicodemus, "Fault-Tolerant Telecommunications System Patterns", in Pattern Languages of Program Design 2, pp. 549-562, addison-Wesley, 1996

[2] S. Ahmed and G. Ashraf, "Model-based User Interface Engineering with Design Patterns", in Journal of Systems and Software, vol. 80, pp. 1408-1422, 2007

[3] C. Alexander, S. Ishikawa, M. Silverstein, "A Pattern Language", Oxford University Press, 1977

[4] C. Alexander, "Notes on the Synthesis of Form", Harvard University Press, 1970

[5] C. Alexander, "The Oregon Experiment", Oxford University Press, 1975

[6] C. Alexander, "The Timeless Way of Building", Oxford University Press, 1979

[7] E. Bayle, "Putting it all together: Towards a Pattern Language for Interaction Design: A CHI workshop", in SIGCHI Bulletin, vol. 30, pp. 17 – 23, 1998

[8] K. Beck, W. Cunningham, "Using Pattern Languages for Object-Oriented Programs", OOPSLA'87:Workshop on the Specification and Design for Object-Oriented Programming, 1987

[9] J. Borchers, "CHI Meets PLoP: An Interaction Patterns Workshop", SIGCHI Bulletin, vol. 32, 2000

[10] J. Borchers, "A Pattern Approach to Interactive Design", Wiley, 2001

[11] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stahl, „Pattern-Oriented Software Architecture: A System of Patterns", vol. 1, Wiley, 1996

[12] E. Chung, J. Hong, J. Lin, M. Prabaker, J. Landay, A. Liu, "Development and Evaluation of Emerging Design Patterns for Ubiquitous Copmuting", in Proc. Of Designing Interactive Systems, 2004

[13] T. Coram, J. Lee, "Experiences – A Pattern Language for User Interface Design, 1996, Available at: http://www.maplefish.com/todd/papers/experiences.html, Accessed on: June 16, 2009

[14] J. Deng, E. Kemp, G. Todd, "Focussing on a Standard Pattern Form: The Demelopment and Evaluation of MUIP", in Proc. of the Seventh ACM SIGCHI New Zealand Chapter's International Conference on Computer-Human Interaction, pp. 83-90, ACM Press, 2006

[15] D. van Duyne, J. Landay, J. Hong, "The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience", Addison-Wesley, 2003

[16] S. Fincher, "Perspectives on HCI Patterns: Concepts and Tools (introducing PLML)", Interfaces, vol. 56, pp.26-28, 2003

[17] S. Fincher, "The Pattern Gallery", Available at: http://www.cs.kent.ac.uk/people/staff/saf/patterns/gallery.html, Accessed on June 18, 2009

[18] S. Fincher, P. Windsor, "Why Patterns are not enough: Some Suggestions Concerning an Organising Principle for Patterns of UI Design", CHI'2000 Workshop on Pattern Languages for Interaction Design: Building Momentum, 2000

[19] E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley Reading, 1994

[20] A. Gaffar, H.Javahery, A. Seffah, D. Sinning, "MOUDIL: A Comprehensive Framework for Disseminating and Sharing HCI Patterns", CHI'03 workshop: Perspectives on HCI patterns: Concepts and Tools, 2003

[21] I. Graham, "A Pattern Language of Web Usability", Addison-Wesley, 2003

[22] R. Griffiths, L. Pemberton, J. Borchers, "Usability Pattern Language Workshop", at INTERACT'99, Available at: http://www.it.bton.ac.uk/staff/rng/UPLworkshop99/PositionPapers.html, Accessed on: June 17, 2009

[23] M. Hitz, H. Werthner, A Graph Oriented Approach to Enhance Reusability in *-bases, WISR'92: 5th Annual Workshop on Software Reusability, 1992

[24] M. Irons, "Patterns for Personal Web Sites", Available at: http://www.rdrop.com/~half/Creations/Writings/Web.patterns/index.html, Accessed on: June 19, 2009

[25] P. Kamthan, "A Critique of Pattern Language Markup Language", Interfaces, vol. 68, pp. 14-15, 2006

[26] C. Kruschitz, "XPLML: A HCI Pattern Formalizing and Unifying Approach", in Proc. of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems, pp. 4117 – 4122, ACM, 2009

[27] M. Leacock, E. Malone, C. Wheeler, "Implementing a Pattern Library in the Real World: A Yahoo! Case Study", ASIS&T IA Summit, 2005

[28] J. Lin, J. Landay, "Employing Patterns and Layers for Early-Stage Design and Prototyping of Cross-Device User Interfaces", in Proc. 26th International Conference on Human Factors in Computing Systems, pp. 1313-1322, ACM, 2008

[29] M. Mahemoff, L. Johnston, "Pattern Language for Usability: An Investigation of Alternative Approaches", in Proc. Third Asian-Pacific Conference in Computer Human Interaction, pp. 25-31, IEEE Coomputer Society, 1998

[30] J. Noble, "Classifying Relationships Between Object-Oriented Design Patterns", in Proc. of the Australian Software Engineering Conference, 1998, IEEE Computer Society

[31] "Portland Pattern Repository", Available at: http://c2.com/ppr/, Accessed on June 18, 2009

[32] F. Radeke, P. Fobrig, a. Seffah, D. Sining, „PIM Tool: Support for Pattern-Driven and Model-Based UI Development", 5th International Workshop: Task Models and Diagrams for User Interface Design, LNCS: Programming and Software Engineering, vol 4385, pp. 82-96, 2007

[33] R. Smith, "Panel in Design Methodology", OOPSLA'87:Addendum to the Proceedings on Object-Oriented Programming Systems, Languages and Applications, pp. 91-95, ACM, 1987+

[34] J. Tidwell, "Designing Interfaces", O'Reilly, 2005

[35] "Designing Interfaces – Online", Available at: http://www.designinginterfaces.com, Accessed on July 03, 2009

[36] M. van Welie, G. van der Veer, "Pattern Languages in Interaction Design: Structure and Organization", Human-Computer Interaction – INTERACT'03, pp.527-534, IOS Press, 2003

[37] "Welie.com – Patterns in Interaction Design", Available at: http://www.welie.com, Accessed on June 18, 2009

[38] "XML User Interface Language", Available at: https://developer.mozilla.org/En/XUL, Accessed on June 23, 2009

[39] "Yahoo! Design Pattern Library", Available at: http://developer.yahoo.com/ypatterns/index.php, Accessed on June 22, 2009

[40] W. Zimmer, "Relationships between Design Patterns", Pattern Language of Program Design, Addison-Wesley, 1994