

# User-Centered Design Practices Management and Communication

*Ashraf Gaffar, Naouel Moha, Ahmed Seffah*  
Human-Centered Software Engineering Group  
Concordia University, Montreal Canada  
Email: {gaffar@cse, n\_moha@encs, seffah@cse}.concordia.ca

## Abstract

Poor design of interactive systems has been attributed to insufficient communication between different people involved in the design process. The software engineering community has popularized patterns as one possible solution to this problem. Interaction design patterns, also referred to as Human Computer Interaction (HCI)- or User Interface (UI) patterns, have been proposed as one step towards building more reusable components for user interface applications. By recording the indications and remedies of typical problems, patterns can improve communication among developers and support design reuse. The idea is to capture information about frequently encountered problems and how they are solved and present them to designers in a readily usable format. Our research aims to develop a pattern-assisted framework emphasizing feedback from end-users as well as designers' best practices and prototyping tips. UPADE is a small-scale pilot project to test the possibility of using patterns as design building blocks. The feedback from this project -collected from actual users and lab testing- is used in building more comprehensive pattern dissemination and assimilation environment for developers.

## 1 Introduction

In 1977, Christopher Alexander introduced his revolutionary book "The Timeless Way of Building" (Alexander, 1979). His idea was that one could achieve excellence in architecture by learning and using a carefully-defined set of design rules, or patterns; and though the quality of a well-designed building is hard to put into words, the patterns themselves that make up that building are remarkably simple and easy to understand (Tidwell, 1979). Pattern Languages (Alexander, Ishikawa, & Silverstein, 1977; Mahemoff & Johnston, 1998) represent and accumulate knowledge of good design by grouping patterns into collections. When related patterns are woven together they form a "language" that provides a structured process for the orderly resolution of software development problems. Pattern languages are not formal languages, but rather a set of interrelated patterns. Nonetheless they do provide a vocabulary for talking about a particular problem. They may include a method for connecting patterns into whole "architectures" for the domain. The idea was originally applied in town planning and architecture, but has been recently taken up by software designers as well.

## 2 Patterns as a Design Approach

An important factor in improving design quality for interactive systems is the effectiveness in capturing and communicating the essential elements of good designs. There have been many partially successful approaches to this task: study of exemplars, practice under the instruction of a master, design principles to capture the master's implicit knowledge, design rationale for organizing application of principles to cases, design guidelines making principles specific, and software toolkits embodying guidelines (Casaday, 1997). As a new approach, patterns are used to document the essentials of design solutions in a form that can be understood, learned, and applied to situations that are similar to existing cases.

Patterns make it possible to talk about design at a higher level of abstraction. Instead of thinking in terms of individual classes and their behavior, we can start to think in terms of collaborating classes, their relations and responsibilities. In early design it is enough to know that we are using a specific pattern at some point. Details about how this pattern is actually implemented are only the subject of later stages. Patterns are medium grained building blocks from which to compose systems. Given a broad collection of patterns on different scales (e.g. architectural patterns, design patterns and language-idioms) it should be possible to combine them together into a design that can be mapped to different programming languages. Moreover, the ability to learn and understand existing user interface systems from source code is greatly enhanced by visualizing the software systems as patterns, rather than seeing them as complex collections of classes and method implementations. Patterns provide this medium-grained

abstraction and can be an effective tool for understanding user interface systems. Pattern solutions are defined and represented by a general structure of the classes constituting the pattern, and an assignment of responsibilities to the participating classes. These pattern solutions can be customized for various needs and situations. The use of design and implementation patterns is either consciously built into a system design, or pattern-like solutions get created as a result of problem solving and implementation by first principles.

## 2.1 Automating Pattern-Oriented Design

The fundamental research question of this paper can be stated as follows: “*How can we automate HCI pattern creation and composition in order to facilitate the development of user interfaces*”. To improve the quality of user interfaces we have to stop developing them from scratch for every new application and make use of reusable design components. Patterns assure design reuse benefits early in the development lifecycle and address the general problem of how to design complex interactive software. The following are the principle objectives:

*-Follow a systematic design methodology* to join design components together; a methodology that help control pattern creation, composition, and code generation. In this regard, we are applying the Seven C’s methodology outlined in (Gaffar, 2005).

*-Abate disparity between textual pattern description and pragmatic user interface implementation.* Being written in text format makes patterns hard to integrate with other design tools (Gaffar, Sinnig, Seffah & Forbrig, 2004). They become less appealing to designers and developers who often need to accomplish their tasks within a reasonable amount of time. It is rare to find designers who don’t use tools in their work. Patterns can be more useful if built and treated as software components that can interact with other design tools.

*-A tool to leverage the use of patterns.* In order to demonstrate how tools can communicate with software pattern components and how new designs can be automated, we built UPADE. It is a tool that interacts with pattern components and assists developers to use patterns at three different levels: the pattern level, the design level and the code level, as detailed in section 3.1.

## 3 UPADE Architecture and Basic Features

UPADE provides a unified interface to support the development of user interface designs and improves software production. It is a prototype written in Java that aims to support HCI pattern writers and user interface developers. By leveraging the openness and flexibility of Java, UPADE enables developers to easily and effectively describe, search, exchange and extend their own patterns as well as those created by others. At the same time, UPADE develops a systematic approach to combine patterns, support their integration at the high design level and automate their composition.

### 3.1 UPADE Architecture

As a tool for automating the development of HCI designs, UPADE embodies three key functionalities. First, it helps both pattern writers and developers to use existing relationship between patterns to define new patterns or create a design by combining existing patterns. Second, in order to facilitate patterns combination, the tool supports different hierarchical, traceable design levels. In our case study, three levels are possible: Pattern Level, Design Level, and Code Level. At the pattern level, developer can see description of patterns, search a specific pattern, create a new pattern and save it into the database. At the design level, developers can develop a systematic approach to combine patterns, support the integration of patterns at different design stages, replace one pattern occurrence by another and automate pattern composition. Finally at code level, developers can see the structure of the design in terms of classes, methods, associations and inheritance relationships in a particular programming language. Third, the UPADE provides a mechanism to check and control how patterns are created or modified. By using the database information, UPADE automatically examines the patterns and offers a related feedback to the designer.

### 3.2 Feature Description

Each pattern describes a collaboration of elements that provide a solution to a problem in a given context. The main user interface includes the following components:

-“**Browse**” provides a description of existing patterns, some illustrated diagrams and several practical examples. When software developers run the UPADE application, it will open in ‘Browse’ panel by default. In this mode, UPADE produces and delivers patterns information. The information is presented using the incorporated format showing related design process(es), pattern category, name, description and examples. Categories are presented as a browse tree (figure 1) for navigation. By default, UPADE will allow browsing patterns with their associated process name. However, software developers can switch to browse by category.

-“**Search**” improves efficiency of using UPADE by accelerating the searching of Patterns. The Search window presented to the developer offers two kinds of search. Users can have a simple search for patterns by keywords. They can also select from several advanced search criteria in the ‘Criteria Combo Box’ and apply it.

-“**Edit**” helps developers create their own patterns or modify existing ones. Since patterns are reusable components, a well-developed pattern should be saved for reuse in other designs (Florijn, Meijers, & Winsen, 1997). Using the “Edit” tab in UPADE, developers can create their own patterns and associate implementation rules -or constraints- with them. Therefore, “Edit” will allow end users full control over their design standards, and the implementation of patterns in source code that meet those standards. The use of constraints allows developer to decide how certain patterns can or can not be combined with other patterns.

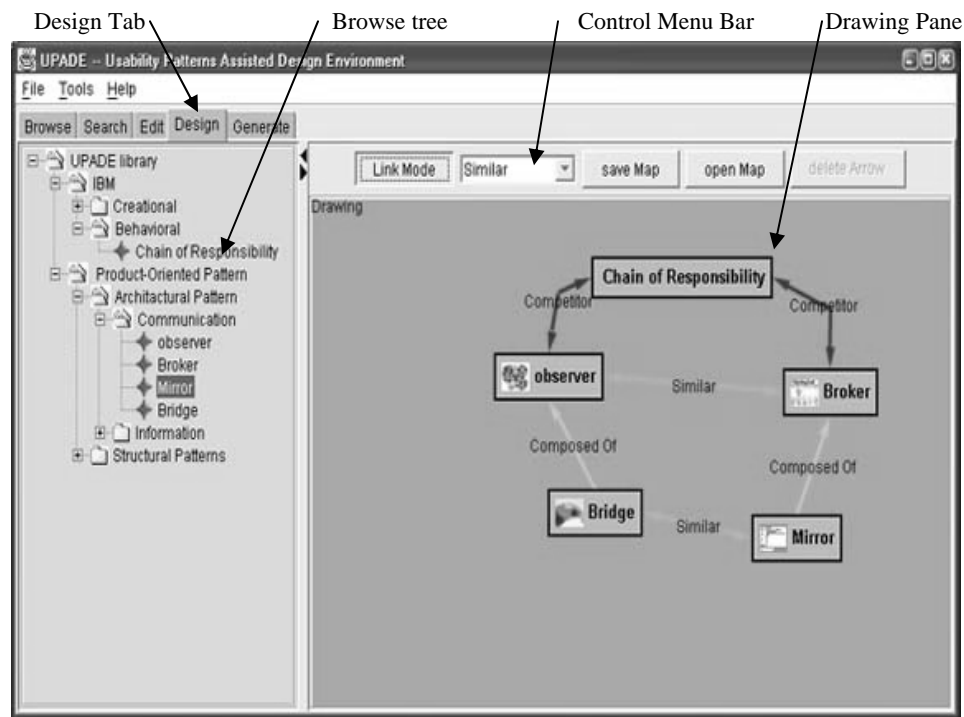


Figure 1: Navigating and Selecting Patterns

-“**Design**” develops structured steps to combine patterns, support integrating them at several phases of design, and automate their composition. As we know, a pattern is a generic description of a solution provided to address one- or a common set of problems. Although a pattern describes a solution, it does not impose how this solution must be realized. A pattern may, however, describe how it relates to other patterns and how it can be composed of other patterns. In this way, the abstract nature of patterns is preserved while the realization of solutions and idioms is reserved for different implementation strategies. For this reason, UPADE gives an environment to compose patterns together. When selecting “Design” tab, UPADE opens a new design workspace (figure 1). The main interface of “Design” workspace is separated into three areas:

- “**Browse Tree Pane**” is the section where developers can browse the entire pattern collections in the **UPADE** database. They can expand any collection to see the available patterns in it.

- **“Control Menu Bar”** is the menu for developers to control the drawing process and create their custom design.
- **“Drawing Pane”** is the area where developers can compose patterns by simply using drag and drop from the browse tree pane.

### 3.3 Practicing Pattern-Oriented Design

In “Design” mode, UPADE can support the design activities from more general to more specific details. It also supports combination and organization of existing patterns. For example, The Software developer can embed “Page Managers” patterns into “Information Architectural” patterns, and both “Navigation Support” patterns and “Information Containers” patterns into “Information Architectural” patterns. Moreover, the designer has the freedom to organize “Navigation Support” patterns and “Information Containers” patterns inside the layout; they can move, combine or delete them altogether. These activities aim to explore how to organize and combine the existing patterns to customize and format the new ones. Finally, “Design” editor provides a mechanism to check the validity of combining patterns as compared to a set of rules. It automatically examines the compatibility of certain patterns and gives the related instruction to the designer.

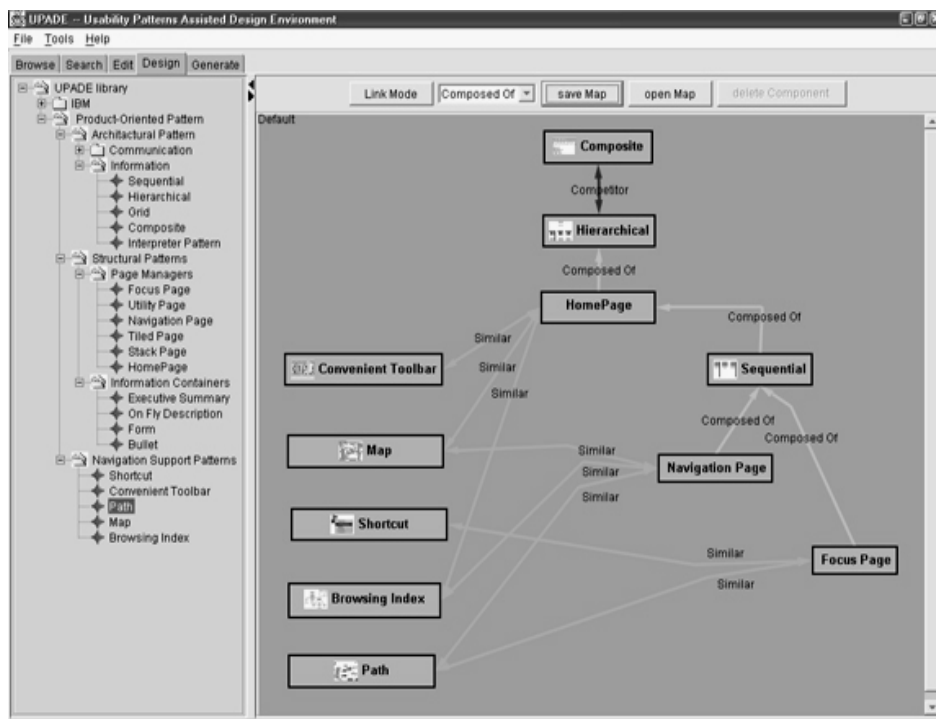


Figure 2: Combining Patterns in a New Design

Once the “Design” Tab is selected (figure 2), UPADE can help start a new pattern-oriented design as follows:

- Pattern developers need to browse the tree in the **“Browse Tree Pane”** to view available patterns.
- Then, they need to select one pattern name and drag and drop it into the **“Drawing Pane”** area.
- They repeat these two steps until all desired patterns are collected into the drawing area.
- Next, by selecting **“Link Mode”** button from the control menu bar, developers are guided to connect each pattern to the others by choosing from different connector types that are available in the combo box of the control menu. While developers can choose the way they want to connect patterns to generate their own design, UPADE will check the validity of all connection attempts selected by users and allow only valid ones. For example, some “similar” patterns are allowed to coexist together in the same design, while others are not.
- At the end, the developer can save the new pattern composition map into XML format for use by other XML-compatible tools.

One of the key differences between UPADE and other user interface development tools is that developers have visual control with drag and drop mechanism. Developers are also able to define their patterns, modify them and use them to create the code based on pattern characteristics. UPADE is designed to be customized and extended, with the realization in mind that most people have achieved a local set of conventions for style and structure, and only need a tool to assist them in creating new design more quickly that honors those conventions.

## 4 Conclusion

Like several other user interface development tools, UPADE introduces new benefits. We were able to determine their effectiveness by testing it in our lab setting as well as real design environments:

- It facilitates the process of browsing and searching patterns.
- It facilitates the process of gluing patterns in a controlled and validated way.
- It supports pattern modification and creation.
- It maintains a pattern view of the design

As explained above, UPADE is serving as a pilot project to test these functionalities and the improvements attained by using patterns as design building blocks. We are also conducting a survey on how patterns are currently used in the industry. The feedback from this project and other empirical studies as well as the survey are used in building a more comprehensive pattern environment for developers.

## 5 Acknowledgement

This work is supported by the National Science and Engineering Research Council (NSERC) Canada, le Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT) Canada, and Daimler Chrysler Software Technology Center (Web Engineering project, Star Phase I), Ulm, Germany.

## 6 References

Alexander, C. (1979). *The timeless way of building*. New York: Oxford University Press.

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language: towns, buildings, constructions*. New York: Oxford University Press.

Casaday, G. (1997). Notes on a pattern language for interactive usability. In CHI 97 Electronic publications, ACM. Retrieved mai 2, 2003 from <http://www.acm.org/sigchi/chi97/proceedings/short-talk/gca.htm>

Florijn, G., Meijers, M., & Winsen, P. van, (1997). Tool support for object-oriented patterns. ECOOP'97, Utrecht University, pp.472.

Gaffar, A. (2005) The 7C's: an iterative process for generating pattern components, HCI international, the 11<sup>th</sup> international conference on human-computer interaction, Las Vegas, Nevada, USA.

Gaffar, A., Sinnig, D., Seffah, A. & Forbrig, P. (2004) *Modeling Patterns for Task Models*, proceedings of TAMODIA, 3rd International Workshop on Task Models and DIAGrams for user interface design, Prague, Czech Republic.

Mahemoff, M. J, & Johnston, L. J., (1998). Pattern languages for usability: an investigation of alternative approaches", Asia-Pacific Conference on Human Computer Interaction (APCHI).

Tidwell, J. (1997). *Common ground: a pattern language for human-computer interface design*. Retrieved January 28, 2003 from [http://www.mit.edu/~jtidwell/common\\_ground.html](http://www.mit.edu/~jtidwell/common_ground.html)