Interacting with Computers 23 (2011) 189–201

Contents lists available at ScienceDirect

Interacting with Computers

journal homepage: www.elsevier.com/locate/intcom

The organization of interaction design pattern languages alongside the design process

Christian Hübscher *, Stefan L. Pauwels, Sandra P. Roth, Javier A. Bargas-Avila, Klaus Opwis

University of Basel, Department of Psychology, Center for Cognitive Psychology and Methodology, 4055 Basel, Switzerland

ARTICLE INFO

Article history:
Received 30 March 2010
Received in revised form 1 February 2011
Accepted 24 February 2011
Available online 4 March 2011

Keywords:
Design patterns
Pattern languages
Interaction design

ABSTRACT

This work explores the possibility of taking the structural characteristics of approaches to interaction design as a basis for the organization of interaction design patterns. The *Universal Model of the User Interface* (Baxley, 2003) is seen as well suited to this; however, in order to cover the full range of interaction design patterns the model had to be extended slightly. Four existing collections of interaction design patterns have been selected for an analysis in which the patterns have been mapped onto the extended model. The conclusion from this analysis is that the use of the model supports the process of building a pattern language, because it is predictive and helps to complete the language. If several pattern writers were to adopt the model, a new level of synergy could be attained among these pattern efforts. A concluding vision would be that patterns could be transferred freely between pattern collections to make them as complete as possible.

1. Introduction

In the project that was the trigger for this research (see Pauwels et al., 2009, 2010) one of the challenges was to build a library of interaction design patterns for an internal system. This library had to be designed to cover the whole design space of this application with patterns. Then it had to be positioned as an authoritative source of information about interaction design for the business analysts and developers in the company. This is the background of this work. The article now explores whether it is possible to take knowledge from approaches to interaction design as a basis for the organization of interaction design patterns. Publicly available pattern collections (Tidwell, 2006; van Duyne et al., 2007; Yahoo! Inc., 2009; van Welie, 2009) are used to illustrate the analysis.

1.1. The problem of pattern categorization

In recent years, many interaction design pattern collections have been published and more are appearing each year (for an overview, see: http://www.hcipatterns.org). With collections growing bigger, the question of pattern categorization becomes more important. According to Dearden and Finlay (2006), the organization of pattern languages is an important area of research in human–computer interaction (HCI).

There is currently a certain "duplication of effort" (Dearden and Finlay, 2006, p. 88) in the field of interaction design patterns and this indicates that in building pattern languages, the wheel has already been (re-)invented several times. Ironically, this is exactly what the concept of patterns intends to prevent designers from doing. Because pattern collections are all organized differently, it is very hard to compare them and to transfer individual patterns from one collection to the other. With the *pattern language markup language* (PLML) (Fincher, 2003), important steps have been made toward a standardization of the structure of individual patterns, but no such organized effort has yet been made to find a unified organization of pattern languages. To have such a standard in the *organization of languages* may bring synergies to the HCI field as a whole and make it easier for individual projects to build their own pattern language, based on the work of others.

1.2. Interaction design patterns and pattern languages

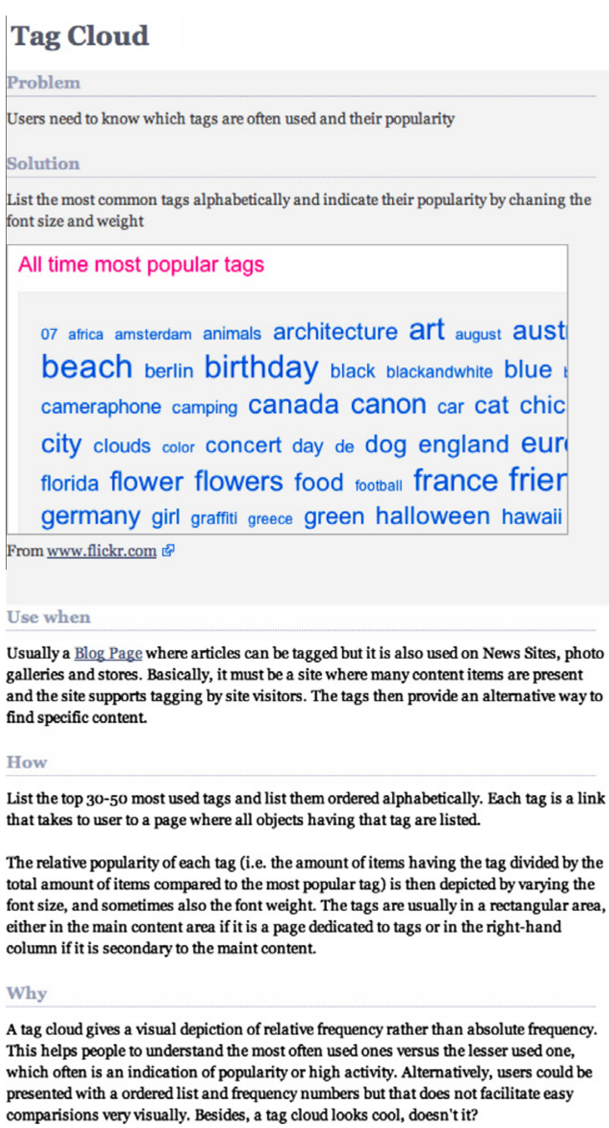
Dearden and Finlay (2006) define a pattern as "a structured description of an invariant solution to a recurrent problem within a context" and a pattern language as "a collection of such patterns organized in a meaningful way". The concept of design patterns was originally developed by Christopher Alexander (1964) in the field of architecture. Software engineers Gamma et al. (1995) then adopted the concept to describe software design patterns (see Gabriel (1996) for more about these efforts). Borchers (2001) later applied patterns to problems in the field of HCI. One emerging area of an application for design patterns, which is closely related to HCI, is e-learning (see Dimitriadis et al., 2009).

* Corresponding author. Tel.: +41 44 292 2127.
E-mail address: christian.huebscher@unibas.ch (C. Hübscher).

0953-5438/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.intcom.2011.02.009

This study focuses on *interaction design patterns*: "A problem is stated in the domain of human interaction issues, and the solution is stated in terms of suggested perceivable interaction behavior" (Dearden and Finlay, 2006, p. 52). The description of a pattern is structured with the topics *problem*, *context*, *solution*, and *forces* included in many cases; however, the structure and naming of these sections vary among pattern collections (see Fig. 1). Most include further sections such as examples.

Alexander's (1964, 1979) idea behind design patterns is (see Kohls and Uttecht, 2009) that they should support design as a problem-solving task to achieve fitness between form and context. A design problem occurs because competing "forces" have to be satisfied. Thus multiple design patterns can solve the same problem in different contexts (e.g. *checkboxes* and a *list builder* both solve the problem of selecting multiple items). In this case, they have identical problem attributes but the context attributes make clear when to choose one pattern over the other. Alexander uses the term *forces* to describe these context-dependent constraints that have an effect on how to solve the problem. The proper configuration of a group of patterns is in itself a pattern on a more abstract level. Alexander stresses the point that one needs a *pattern language* to achieve a coherent design and that "a bunch of good ideas" (Alexander, 1999, p. 75) are not enough to do the job.


Tag Cloud

Problem

Users need to know which tags are often used and their popularity

Solution

List the most common tags alphabetically and indicate their popularity by chaning the font size and weight

From www.flickr.com

Use when

Usually a Blog Page where articles can be tagged but it is also used on News Sites, photo galleries and stores. Basically, it must be a site where many content items are present and the site supports tagging by site visitors. The tags then provide an alternative way to find specific content.

How

List the top 30-50 most used tags and list them ordered alphabetically. Each tag is a link that takes to user to a page where all objects having that tag are listed.

The relative popularity of each tag (i.e. the amount of items having the tag divided by the total amount of items compared to the most popular tag) is then depicted by varying the font size, and sometimes also the font weight. The tags are usually in a rectangular area, either in the main content area if it is a page dedicated to tags or in the right-hand column if it is secondary to the maint content.

Why

A tag cloud gives a visual depiction of relative frequency rather than absolute frequency. This helps people to understand the most often used ones versus the lesser used one, which often is an indication of popularity or high activity. Alternatively, users could be presented with a ordered list and frequency numbers but that does not facilitate easy comparisions very visually. Besides, a tag cloud looks cool, doesn't it?

Fig. 1. Example of an interaction design pattern description (van Welie, 2009).

Pattern languages should be capable of *producing* coherent wholes; i.e. they should be *generative*.

Dearden and Finlay (2006) conclude that there are two evident forms of organization in Alexander et al. (1977): patterns come in sets according to *levels of physical scale* and build up a *network*, where patterns are referenced to other patterns. The linking of individual interaction design patterns is usually made in a "related patterns" section, where alternative solutions to similar contexts or patterns are placed. Van Welie and van der Veer (2003) distinguish between three fundamental relations:

- *Aggregation*: A design pattern can include others that complete it.
- *Specialization*: A design pattern can be derived and specialized from another design pattern.
- *Association*: Multiple design patterns can occur in the same context or solve similar problems.

An interesting question concerns the *completeness* of a pattern language. Alexander (1979) argues that a pattern language can be morphologically and functionally complete: It is *morphologically* complete when it can account for a complete design, without any missing parts, and *functionally* complete when it is self-consistent, i.e. it does not create forces that it cannot resolve. For interaction design patterns, van Welie and van der Veer (2003) say that a pattern language is complete when every good design that we find can be described using it.

1.3. Recent work on the organization of interaction design patterns

In architecture, Alexander (1979) defined *physical scale* as being the main organizational principle for patterns. The organization of interaction design patterns, on the other hand, is not so straightforward; therefore, we must put more effort into coming up with good organizational principles for patterns in the field of HCI. Different approaches on how to work out an organization of interaction design patterns have been suggested (for an overview, see Dearden and Finlay, 2006). Several authors argue that the best way to organize a pattern language is alongside a design process. Fincher and Windsor (2000) discuss different organizing principles. Their final solution brings their taxonomies in an order that could be associated with the phases of a design process. They distinguish: *analysis space* (context and values), *problem space* (structure: tasks; structure: information) and *solution space* (structure: scale). Van Welie and van der Veer (2003) argue that the organization should be based on a top-down design process and they distinguish the following levels: *business goals*, *posture level*, *experience level*, *task level*, and *action level*. These authors' organization is done according to a design process but they do not explicitly relate it to concrete user-centered or interaction design approaches. They only mention Cooper et al. (2007) as a basis for their choice of the category "posture type patterns". Borchers (2000) maps different types of patterns onto Nielsen's (1993) usability engineering lifecycle but he does this more to argue that we can use patterns across a whole project lifecycle than to discuss the organization of patterns. Dearden and Finlay (2006) give an overview of the different requirements for an organizing principle for pattern languages. According to Fincher and Windsor (2000), an organizing principle should *taxonomise*, *proximate*, be *evaluative* and *generative*; i.e. it should enable users to find (related) patterns, it should allow users to consider the problem from different viewpoints and to build new solutions. Fincher (2002) argues that it would also be desirable that an organization is *predictive*; i.e. it should actively support the process of identifying new patterns. Using the periodic table of the elements in chemistry as an example; she argues that such an organization would help to discover missing patterns. This is a very interesting idea, which suggests giving preference to a top-down categorization based on a certain model over a bottom-up approach.

2. Structural characteristics of interaction design approaches

As mentioned above, several authors suggest that the organization of interaction design patterns should be based on a design process (e.g., van Welie and van der Veer, 2003; Fincher and Windsor, 2000). There is a vast literature on the topic of how to proceed in designing user interfaces, and many design processes have been published so far. This study seeks to extract the relevant aspects from these published works and use them as a basis for structuring interaction design pattern languages. The focus is on the *design* aspect of such processes – even though patterns can also be used to support other phases of the process (see Borchers, 2000, or Granlund et al., 2001).

Different sources can be considered as *interaction design processes;* i.e. usability engineering lifecycles or user-centered design processes:

- Nielsen's Usability Engineering Lifecycle (Nielsen, 1993).
- Delta Method (Rantzer, 1996).
- Contextual Design (Beyer and Holtzblatt, 1998).
- OVID (Robert, 1998).
- Mayhew's Usability Engineering Lifecycle (Mayhew, 1999).
- Usage Centered Design (Constantine and Lockwood, 1999).
- The Elements of User Experience (Garrett, 2002).
- Universal Model of a User Interface (Baxley, 2002, 2003).
- Goal Directed Design (Cooper et al., 2007).

To define how to organize interaction design patterns alongside a design process, we extract the structural characteristics of these approaches in order to perform a mapping of different kinds of patterns onto them. Most approaches to interaction design foster a layered approach in which different levels of the user interface are designed one after another. Some approaches distinguish two phases; others have more levels of design.

Many design approaches make the distinction between *conceptual* and *concrete* design, or conceptual and physical design as Sharp et al. (2007) call it. Rantzer (1996), Beyer and Holtzblatt (1998), Robert (1998), Constantine and Lockwood (1999), and Cooper et al. (2007) make such a distinction in the processes that they describe. Mayhew (1999) distinguishes a 1st, 2nd, and 3rd level of design in her process, but in her levels 2 and 3 concrete design is carried out. In level 2, the central and recurring interactions are designed and in level 3 the rest of the user interface is specified. Here, her distinction is more a matter of scope than a matter of different aspects. One can argue that Mayhew's design process is also based on the distinction of *conceptual* and *concrete* design.

In *conceptual design,* the structural base of the user interfaces – the "user interface architecture" – is defined. The name for this design task varies: conceptual design (Rantzer, 1996), user environment design (Beyer and Holtzblatt, 1998), conceptual model design (Mayhew, 1999), content model (Constantine and Lockwood, 1999), and interaction framework (Cooper et al., 2007). In this phase, the designer works out relationships between user objects, organizational schemes, and workflows. The deliverables of these tasks are, for the most part, diagrams, storyboards, and sketches of user interfaces. Some of these deliverables do not really look like user interfaces.

In *concrete design,* the user interface – in the form of concrete user interface elements – is defined. This task is called prototyping (Beyer and Holtzblatt, 1998; Rantzer, 1996), detailed design (Cooper et al., 2007; Mayhew, 1999), or the implementation model (Constantine and Lockwood, 1999). The deliverables of concrete design are interactive prototypes or renderings of screens, which often look and behave very similarly to the real system.

Besides the distinguishing of two phases of design, there are several authors who describe an approach with three or more levels. These approaches, however, are not contradictory to the notion of *conceptual* and *concrete* design; they are more an extension of this thinking. The classical work of IBM (1992) explains the levels of designing a user interface with the metaphor of an iceberg, which has the three levels: *structure*, *behavior*, and *presentation*. The approaches by Garrett (2002) and Baxley (2003) build on these levels. Garrett (2002) has a model with the five layers: *strategy*, *scope*, *structure*, *skeleton*, and *surface*. The layers *strategy* and *scope*, however, are more to "set the stage" for doing the interaction design, although *structure*, *skeleton*, and *surface* are similar to the layers of the aforementioned "iceberg model". The tiers of *structure*, *behavior*, and *presentation* bring in a more sophisticated discrimination between different types of design tasks and hence patterns. Following this thinking, using website navigation as an example (see e.g., Leuthold et al., 2011), one can distinguish patterns that describe the *structure* of navigation (e.g., hierarchical vs. flat), the *behavior* of navigation (e.g., dynamic vs. static), and the *presentation* of navigation (e.g., left vs. horizontal placement in the layout). All these different aspects of navigation are influenced by their own forces and therefore it makes sense to distinguish between these aspects by using different interaction design patterns.

An elaborate model in this sense is Baxley's (2003) *Universal Model of the User Interface*. In his model, the same three main tiers exist as in the "iceberg model": *structure*, *behavior*, and *presentation*. However, these three tiers are further divided into three sub-layers each:

- Structure
  - *Conceptual model.*
  - *Task flow (formerly called structural model by* Baxley (2002)).
  - *Organizational model.*
- Behavior
  - *Viewing and navigation.*
  - *Editing and manipulation.*
  - *User assistance.*
- Presentation
  - *Layout.*
  - *Style.*
  - *Text.*

The models of Baxley (2003) and Garrett (2002) are similar, because they both describe a layered top-down design approach. In general, they can be seen as similarly well suited to organizing patterns into categories, but Baxley's (2003) model is much more fine-grained. It actually distinguishes nine layers that are relevant for interaction design patterns. For this reason, it is taken as a basis for this analysis. However, Garrett's (2002) model has a wider scope and therefore it will be taken as an extension of Baxley's model to cover the whole range of interaction design patterns (see Section 3.2).

3. A model for the organization of interaction design patterns

It is a goal of this research to find a model that is based on an interaction design process and that can be used to organize interaction design patterns. As indicated above, Baxley's model has the required properties, so its detailed structure is presented here as described by Baxley (2002, 2003). However, the model does not cover the full range of interaction design patterns as defined by Dearden and Finlay (2006). Therefore the authors have made an extension to the model with the introduction of the category "requirements patterns". Following this, the relationship between the model and technical platforms is discussed.

3.1. The original structure of Baxley's model

Baxley's "Universal Model of the User Interface" (2003) has similarities with the other models described above but it divides design tasks in a more sophisticated way. The model has nine layers divided between three tiers (see Fig. 2). Baxley divides these layers into further topics and sub-topics (Baxley, 2002), see Table 1.

The nine layers of Baxley's model distinguish different aspects of a user interface; for example, whether we are dealing with the *structure* of the user interface or with its *behavior* and whether the behavior is for the manipulation of data by users (i.e. *editing and manipulation*) or for helping them by doing so itself (i.e. *user assistance*). These nine layers can be seen as building on each other.

Baxley (2002) breaks down most of the layers in a topical manner (see the column "topics" in Table 1). These sub-divisions cannot be seen as clearly building on each other. Most of them are topical in nature and are often just different aspects of a layer. This finer structure is optimized for the design of web applications (which is the topic of Baxley's book; Baxley, 2002) and it has been created to provide an optimal structure for the "patterns" that Baxley (2002) discusses. Baxley mentions "interaction design patterns" for all the different levels of the user interface of web applications; however, he does not call them "patterns" but rather "conventions": "Unfortunately, the use of the word 'pattern' in this context, although definitely accurate, is a bit arcane." (Baxley, 2002, p. 14). Baxley seems to have developed the model to organize interaction design patterns – as well as for other purposes – but uses a different terminology.

3.2. An extension of Baxley's model

The definition of *interaction design patterns* is meant to distinguish these patterns from *user interface software design patterns* (Dearden and Finlay, 2006) in the area of patterns concerning the user interface (see Fig. 3). The former are concerned with the perceivable aspects of the user interface and the latter with the inner working of the system related to the user interface.

The perceivable aspects of the user interface, which can be documented as interaction design patterns, can be of two kinds: (a) *user requirements* i.e. a function to enable a user to achieve a certain goal and (b) the *conceptual implementation* of these requirements in the form of a user interface. Baxley's Model is very detailed but does not cover the whole scope of interaction design patterns. The conceptual implementation (*how* it is done) is the scope of Baxley's model, which further breaks down the different solutions into categories. The user requirements (*what* is implemented in the user interface) can be seen as beyond the scope to Baxley's model and suggest an extension of the model (see Fig. 4). Baxley describes a requirements phase in his book but does this within his larger scope design process (Baxley, 2002, p. 44):

(1) *Understanding* (user needs, competition, business opportunity, technical constraints).
(2) *Vision* (core design values, opportunity statement, persona profiles and goals).
(3) *Requirements* (functional, technical, business, usability).
(4) *Design* (structure, behavior, presentation).

This extended model bears a similarity to Garrett's (2002) model, which also contains a layer dealing with requirements (see Fig. 5). This *scope* layer deals with the question of whether a feature or function is part of a system's functional requirements or not, whereas the *strategy* layer does not deal with *solutions* to users' problems but rather with the definition of the *needs of users*

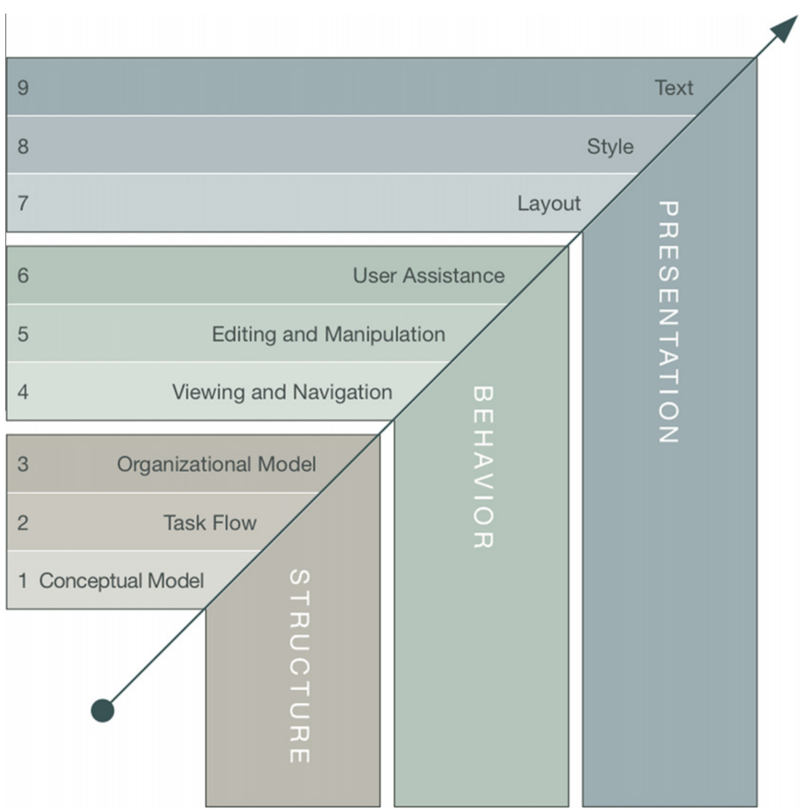


Fig. 2. Illustration of Baxley's "Universal Model of the User Interface" (Baxley, 2003).

Table 1
The layers broken down into topics (Baxley, 2002).

Tier	Layer	Topics	Sub-topics
Structure	*Conceptual Model*	"Examples"	Store; Catalog
	Structural Model	Pages	Views; Forms; View/Form Construct
	(later called *task flow* by Baxley, 2003)	Workflows	Hubs; Wizards; Guides (hub/wizard hybrid)
	Organizational Model	Classification schemes	Objective: Alphabetic; Numeric; Chronologic; Geographic Subjective: Topical; Functional; Audience-based; Metaphorical
		Models of association	Indexes; Hierarchies; Webs
Behavior	*Viewing and Navigation*	Navigation	High-level navigation; Low-level navigation; Utility navigation
		Selecting objects and issuing commands	Shared controls; Dedicated controls
		Viewing lists of data	Changing column sets; Paging; Sorting; Filtering; Searching
	Editing and Manipulation	Input controls	Check boxes; Radio buttons; List boxes; Menus; Text boxes; Buttons
		Common interaction problems and solutions	Selecting a single item; Selecting multiple items; Selecting a date
	User Assistance	Help	Conceptual help; Procedural help; Definitional help; Instructional help
		Alerts	Error alerts; Status alerts; Confirmation alerts
Presentation	*Layout*	Simplicity	Clarity; Reduction; Leverage
		Consistency	Web conventions; Templates and grids; Standards and guidelines
		Order	Grouping; Hierarchy; Alignment
	Style	Evaluating style	Individuality; Brand consistency; Appropriateness for the audience and function
		Preventing style from messing other things up	Working within the medium; Legibility (contrast, line length, typeface, type size, font styling, density/leading, balance the variables affecting legibility); Providing visual cues to behavior (visual cues for text-based hyperlinks, visual cues for clickable images)
	Text	Eliminate superfluous text	Eliminate superfluous text
		Text: what's it good for	Navigation; Titles; Labels; Instruction and help; Marketing messages
		Writing for the web	Be courteous, not patronizing; Leverage the context; Don't repeat yourself; Avoid multisyllabic words that obfuscate

or the *business objectives* and therefore is beyond the scope of interaction design patterns.

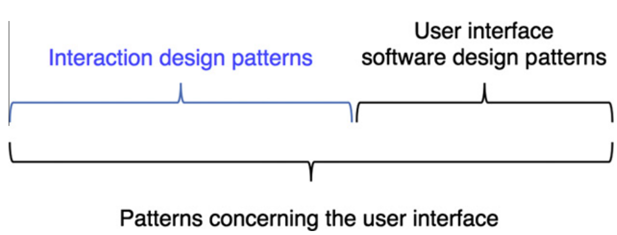


Fig. 3. Different patterns concerning the user interface.

Garrett's (2002) *structure*, *skeleton*, *surface* and Baxley's (2003) *structure*, *behavior*, *presentation* both cover the conceptual implementation of the user interface but they have certain differences in the mapping of patterns onto the model; for example, Baxley (2003) puts "layout" in the *presentation* tier whereas Garrett (2002) sees it as part of his *skeleton* layer (which otherwise would correspond more to *behavior*). For the sake of this analysis, however, it is not relevant to analyze these differences to a further extent.

These "requirements patterns" and "patterns for the conceptual implementation of a user interface" are described below. The definitions are meant to distinguish between these two types of

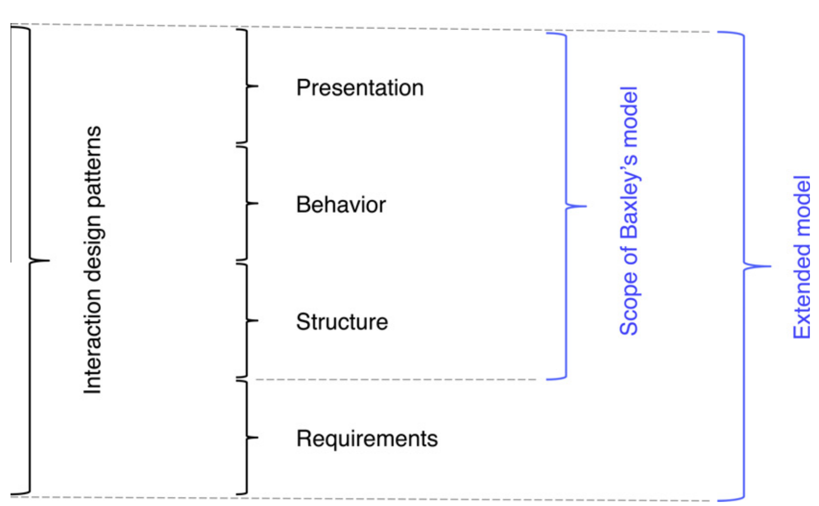


Fig. 4. The different interaction design patterns.

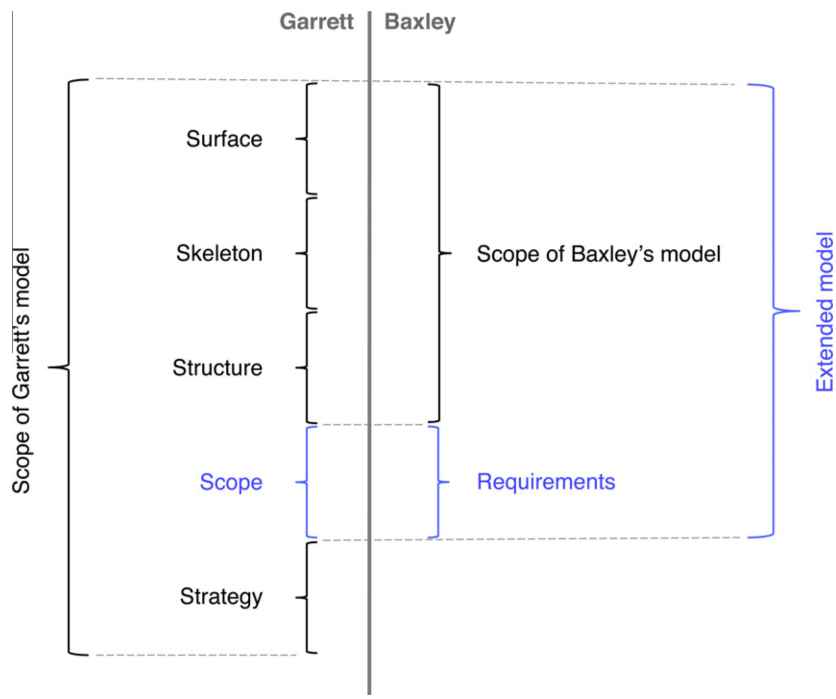


Fig. 5. Comparison of the models of Garrett (2002) and Baxley (2003).

patterns. It is important so separate them accurately otherwise this extended model will not work as an organizational model.

3.2.1. Requirements patterns

Patterns describing abstract features that allow the user to achieve a certain goal are what we call "requirements patterns". In the example of a fictitious "publish to Twitter" function, the pattern would focus on the goal of the user (let friends know of discoveries made while surfing). This pattern would focus on why a user needs a "publish to Twitter" function as opposed, say, to a "subscribe to RSS feed" function. These patterns focus on the forces that distinguish the different goals a user could have, but they do not describe the way that the user achieves these goals in the form of an interaction. This would be described independently of the conceptual and technical implementation. The forces are described on the level of a goal that a user wants to achieve as opposed to another goal. The patterns aim at the optimization of *utility* (Grudin, 1992). If such patterns are described independently of their implementation, the patterns are valid under different circumstances and on different technical platforms. If, on the other hand, the feature is described as a pattern "Twitter icon", the user requirement is mixed up with the conceptual implementation and the solution is no longer the best if circumstances change. In a situation in which multiple such functions (for Twitter, Facebook, etc.) have to be provided for a certain object, the function would no longer be implemented as an icon but would maybe rather be part of a menu. Thus described in an implementation-agnostic way, the requirement pattern "publish to Twitter" would be "stable" under various circumstances.

The patterns discussed above are functional requirements but there are also patterns that describe non-functional requirements. The pattern "site accessibility" (van Duyne et al., 2007) is such an example. It is a very high-level pattern, which might also contain several more detailed patterns (e.g. "hidden jump to navigation link" for users with screen readers, good contrast, the use of certain HTML tags). So the category of "requirements patterns" is meant to include both functional and non-functional requirements that have an impact on perceived aspects of the user interface.

3.2.2. Patterns for the conceptual implementation of a user interface

The patterns for the conceptual implementation of a user interface describe ways to realize user requirements on a conceptual level. These patterns are the different parts used for the implementation of a user requirement. Which of these parts does this best depends on the specific circumstances. These patterns focus on the forces, which are influenced by different configurations of such patterns. If a function is the only function to be used in a list of objects, this function can be conceptually implemented as an icon. If this function is one among many others it might be implemented as a menu. These patterns do not focus on "specific end goals" of users but more on "generic sub-goals". The patterns aim at the optimization of one aspect or more of *usability* – the effectiveness, efficiency, or satisfaction of the user (Hornbaek, 2006) – but not on the *utility* (Grudin, 1992). With the help of Baxley's model, the different solutions for the conceptual implementation can be broken down into nine categories.

3.2.3. The scope of platform applicability of Baxley's model

Because Baxley (2002) introduced his model to explain "how to make the web work", one might ask whether this model is also valid for other platforms. Baxley later started to call it the "Universal Model of the User Interface" and discussed ATMs, DVD menu systems, Amazon.com and Microsoft Word to support this point (Baxley, 2003). The fact that IBM (1992) used the same layers of design in a pre-web area shows that at least the main tiers *structure*, *behavior*, and *presentation* are relevant beyond the web. But one might ask, where do the patterns for mobile or rich web interaction design belong? To explain this, it is better to look at the relationship between the layers of Baxley's model and the technical platform.

In the logic of the model, different platforms can be seen as orthogonal to the tiers *structure*, *behavior*, and *presentation* (see Fig. 6). Tidwell's (2006) patterns are more or less platform-agnostic and can be used on several platforms. Her "one-window drill-down" pattern works on different platforms and she uses examples from the iPod, Mac OS X, and a character-based e-mail application to explain the pattern. The platform-independent description of

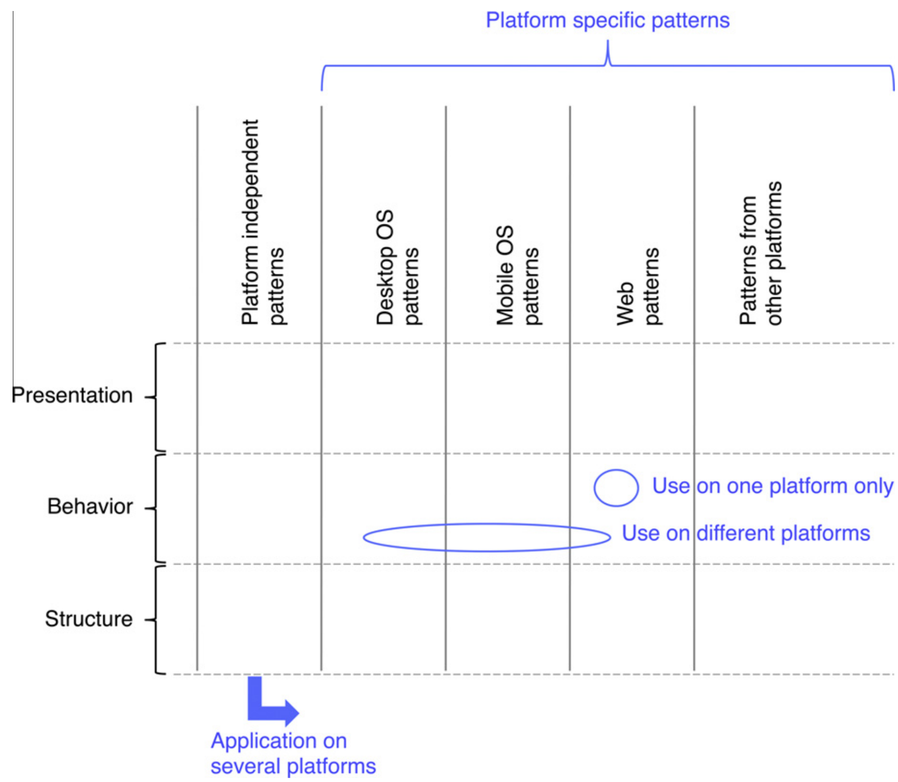


Fig. 6. Pattern collections for different platforms.

the pattern and also the implementations on the different platforms unambiguously belong to the *structure* tier. Although her pattern "movable panels" is not so platform-independent it nevertheless belongs to the *behavior* tier. The pattern "expanding screen width" (van Duyne et al., 2007) is rather web-specific but it applies to both "classic" and "rich" websites and belongs to the *presentation* tier. On the other hand, the pattern "self-healing transition" (Yahoo! Inc., 2009) is a pattern from the *behavior* tier (*user assistance*, because it helps the user to better understand the effects of a manipulation) and is targeted at rich interaction web interfaces. It cannot be used in "classic" web applications, because of technical limitations, even though this distinction will fade away more and more with the establishment of new web standards. However, the "self-healing transition" could also be used on a desktop OS or a modern mobile OS.

These examples show that there are whole pattern collections that are rather platform-independent (e.g. Tidwell, 2006). However, in collections written for special platforms there are also patterns that are more or less platform-independent whereas others make no sense in another environment. Aside from the question of platform, all the patterns concerned with the conceptual implementation of the user interface can be categorized into Baxley's model in a stable way.

4. The analysis of four pattern collections

Following the presentation of Baxley's model, we discuss some interaction design pattern collections in relation to the following eleven categories (10 for interaction design patterns):

- *Requirements* (with an impact on perceived aspects of the user interface).
- *Conceptual Model.*
- *Task Flow.*
- *Organizational Model.*
- *Viewing and Navigation.*
- *Editing and Manipulation.*
- *User Assistance.*
- *Layout.*
- *Style.*
- *Text.*
- *Software Design* (user interface software design patterns).

Because there are dozens of interaction design pattern collections (some published as books but most of them available in the World Wide Web), the analysis focuses on a small sub-set:

- Book: *Designing Interfaces* by Jenifer Tidwell (2006).
- Book: *The Design of Sites* by van Duyne et al. (2007).
- Website: "Welie.com" by Martijn van Welie (2009).
- Website: "Yahoo! Design Pattern Library" by Yahoo! Inc. (2009).

These four interaction design pattern collections have been chosen for this analysis because on the one hand, they contain a certain number of patterns similar in scope but on the other hand, they are well established; i.e., it is likely that they will still be around in the future. In 1997, Tidwell started with an online pattern language called "Common Ground" (Tidwell, 2009) and then, based on that work, published the book *Designing Interfaces* (Tidwell, 2006), making it the pattern book with the longest traceable history. The book *The Design of Sites* (van Duyne et al., 2007) has already been published in its second edition. The first edition dates back to 2002 (van Duyne et al., 2002), which makes it the first fully-fledged collection of interaction design patterns available in the form of a book. The roots of the website "Welie.com" by van Welie (2009) date back at least to 2000 (see van Welie and Trætteberg, 2000) thus it can be said that it is one of the most established online collections of interaction design patterns. The Yahoo! Design Pattern Library (Yahoo! Inc., 2009) is the most recent collection in this analysis, and contains the least number of patterns. However, the fact that it is the only corporate collection of interaction design patterns that is at least partially public makes it an interesting candidate for this analysis. There is also a case study available for the Yahoo! Pattern Library (Leacock et al., 2005).

4.1. The process of categorization

In the analysis of these four collections, all the patterns were put into the proposed categories in order to explore whether:

- All patterns can be classified into the categories.
- There are layers that do not have any patterns in them.
- The distribution of patterns across the layers is even or not.
- There is any ambiguity in classifying patterns in such a way.

The first author conducted the analysis of the four pattern collections. It was identified, for each pattern, on which layers its forces operate. Because there are certain patterns with forces on different levels, the analysis distinguishes a first and a lower priority of mapping. However, for all the patterns it was possible to decide on which level the forces mainly operate.

To control for interrater effects, the second author performed an independent categorization for the first priority mapping of 20% of the patterns from each pattern collection (74 patterns). An interrater reliability analysis using the Kappa statistics was performed to determine consistency among raters. The interrater reliability was found to be high, with Kappa = 0.766 ($p < 0.000$), 95% CI (0.662, 0.870).

The analysis of the mappings has shown that indeed most of the patterns could be categorized into these layers (see Section 4.2) and a comparison of these mappings over all the collections shows interesting differences (see Section 4.3). There were also several patterns that could be mapped onto multiple layers (see Section 4.4). The detailed results of the analysis can be found on http://goo.gl/OWQI2.

4.2. The analysis of the pattern collections

The first priority mapping of patterns from Tidwell (2006), van Duyne et al. (2007), van Welie (2009), and Yahoo! Inc. (2009) is shown in the following tables (Table 2–5).

4.2.1. Book "Designing Interfaces" by Jenifer Tidwell

The scope of Tidwell's (2006) patterns is the design of desktop applications, websites, web applications, and mobile devices. The patterns are rather platform-independent and they are illustrated with examples from several platforms. Her book *Designing Interfaces* uses the following organization scheme for a total of 82 patterns:

- Organizing the Content.
- Getting Around.
- Organizing the Page.
- Doing Things.
- Showing Complex Data.
- Getting Input From Users.
- Builders and Editors.
- Making It Look Good.

All of these patterns could be classified into the layers of Baxley's model; there are no requirements and no software design patterns (see Table 2). There are layers of Baxley's model that remain empty. There were no patterns for the *organizational model,* despite Tidwell discussing models of organization in the introduction to chapter 2. There were no patterns for *text* and only one for *conceptual model*. The other layers contain patterns, but most patterns are situated in the *behavior* tier. The *structure* tier contains the fewest patterns.

4.2.2. Book "The Design of Sites" by van Duyne et al.

The patterns of van Duyne et al. (2007) are written especially for the design of pre-Web 2.0 websites. The 107 patterns in their book, *The Design of Sites,* are organized in the following way:

- Site Genres.
- Creating a Navigation Framework.
- Creating a Powerful Homepage.
- Writing and Managing Content.
- Building Trust and Credibility.
- Basic E-Commerce.
- Advanced E-Commerce.
- Helping Customers Complete Tasks.
- Designing Effective Page Layouts.
- Making Site Search Fast and Relevant.
- Making Navigation Easy.
- Speeding Up Your Site.
- The Mobile Web.

Most of these patterns could be fitted into Baxley's model (see Table 3) but 10 of them are software design patterns (e.g.,

Table 2
Tidwell's categories (Tidwell, 2006) mapped onto the layers.

Layer	Tidwell's categories (number of patterns)								**Total**
	Organizing the Content	Getting Around	Organizing the Page	Doing Things	Showing Complex Data	Getting Input From Users	Builders and Editors	Making It Look Good	
Requirements									**0**
Conceptual Model	1								**1**
Task Flow	5	5	1	2					**13**
Organizational Model									**0**
Viewing and Navigation	1	4	2		7				**14**
Editing and Manipulation			1			2	7		**10**
User Assistance	1	1	2	6	4	8	2		**24**
Layout			6	2	2				**10**
Style		1			1	1		7	**10**
Text									**0**
Software Design									**0**
Total	**8**	**11**	**12**	**10**	**14**	**11**	**9**	**7**	**82**

Table 3
Categories of van Duyne et al. (2007) mapped onto the layers.

Layer	Categories of van Duyne et al. (number of patterns)													Total
	Site Genres	Creating a Navigation Framework	Creating a Powerful Homepage	Writing and Managing Content	Building Trust and Credibility	Basic E-Commerce	Advanced E-Commerce	Helping Customers Complete Tasks	Designing Effective Page Layouts	Making Site Search Fast and Relevant	Making Navigation Easy	Speeding Up Your Site	The Mobile Web	
Requirements		1		3	8	4	5	1			1	1	1	**25**
Conceptual Model	12													**12**
Task Flow		1	1			4	2	5			2			**15**
Organizational Model		6												**6**
Viewing and Navigation		1								2	6		1	**10**
Editing and Manipulation								2			1			**3**
User Assistance								4			3			**7**
Layout			1	2		1			6		1		1	**12**
Style					1						1			**2**
Text				3							2			**5**
Software Design				3				1		1		5		**10**
Total	**12**	**9**	**2**	**11**	**9**	**9**	**7**	**13**	**6**	**3**	**17**	**6**	**3**	**107**

Table 4
Van Welie's categories (van Welie, 2009) mapped onto the layers.

Layer	Van Welie's categories (number of patterns)															Total
	User needs (83)									Application needs (12)			Context of design (35)			
	Navigating around	Basic interactions	Searching	Dealing with data	Personalizing	Shopping	Making choices	Giving input	Miscellaneous	Drawing attention	Feedback	Simplifying interaction	Site types	Experiences	Page types	
Requirements						1	1	1	4	1				8	2	**18**
Conceptual Model													14			**14**
Task Flow		1	3	3	2	8	1	1							9	**28**
Organizational Model																**0**
Viewing and Navigation	25	4	5	8			2									**44**
Editing and Manipulation		1		2	1											**4**
User Assistance		1	4				1	1			2	2			2	**13**
Layout			1	1					1	3						**6**
Style										4						**4**
Text																**0**
Software Design																**0**
Total	**25**	**7**	**13**	**14**	**3**	**9**	**5**	**3**	**5**	**8**	**2**	**2**	**14**	**8**	**13**	**131**

Table 5
Yahoo!'s categories (Yahoo! Inc., 2009) mapped onto the layers.

Layer	Yahoo!'s categories (number of patterns)						Total
	Search	Navigation	Browsing	Selection	Rich interaction	Social	
Requirements						12	**12**
Conceptual Model							**0**
Task Flow						1	**1**
Organizational Model							**0**
Viewing and Navigation	(1)	4	2	1			**7**
Editing and Manipulation					1		**1**
User Assistance				2	15		**17**
Layout			1				**1**
Style							**0**
Text							**0**
Software Design							**0**
Total	**(1)**	**4**	**3**	**3**	**16**	**13**	**39**

fast-loading images) and 25 describe requirements patterns (e.g., *e-mail notifications*).

With the other patterns, all nine layers of Baxley's model are covered. The patterns are more or less evenly distributed across all the three tiers. It is interesting to see that there are some categories that fit directly into one tier of Baxley's model, whereas others show up in two tiers, and still others are spread across all three tiers; for example, the patterns under "site genres" all fit into the *conceptual model* layer, whereas the patterns from the chapter "the mobile web" are spread across all three tiers.

4.2.3. Website "Welie.com" by Martijn van Welie

Earlier versions of van Welie's website distinguished the patterns in *web design patterns*, *GUI patterns*, and *mobile UI design patterns* but today, he just lists patterns in the categories below and the examples all seem to be from websites and web applications (van Welie, 2009).

The actual online catalogue *Welie.com* (May 2009) contains 131 patterns and has the following structure:

- User needs
  - Navigating around.
  - Basic interactions.
  - Searching.
  - Dealing with data.
  - Personalizing.
  - Shopping.
  - Making choices.
  - Giving input.
  - Miscellaneous.
- Application needs
  - Drawing attention.
  - Feedback.
  - Simplifying interaction.
- Context of design
  - Site types.
  - Experiences.
  - Page types.

All but 18 of van Welie's patterns can be classified into Baxley's model. They are all requirements patterns and include all "experiences (context of design)" patterns and some of the "user needs" type patterns (e.g., the pattern *testimonials*). There are no software design patterns in this collection. There are patterns distributed across most of the layers of Baxley's model but the *text* and the *organizational model* layers have no patterns. The other patterns are distributed more or less evenly across *structure* and *behavior* but with a few in the *presentation* tier. All but one of the "shopping" patterns fit into the *task flow* layer.

4.2.4. Website "Yahoo! Design Pattern Library" by Yahoo! Inc.

The public patterns of the *Yahoo! Design Pattern Library* (Yahoo! Inc., 2009) cover different web design issues and many of them are Web 2.0 specific. The actual online catalogue contains 39 patterns (May 2009) and has the following structure:

- Search.
- Navigation.
- Browsing.
- Selection.
- Rich interaction.
- Social.

All but the "social" patterns, which can be seen mostly as requirements, could be classified into Baxley's model (see Table 5; Yahoo! categorized the pattern *search pagination* under "search" and "browsing", so by category the count of *viewing and navigation* patterns is 8 and the total count of patterns is 40). There are no software design patterns in Yahoo!'s pattern library. In the Yahoo! library, only one pattern could be found that fitted into the *structure* tier (*sign-in continuity*) and the *presentation* tier (*page grids*), respectively; the other patterns all fitted into the *behavior* tier. What is interesting about Yahoo!'s organization of the patterns is that there are some higher-level patterns (e.g., *pagination*) that contain sub-patterns (*item pagination* and *search pagination* in this case) as different solutions to these higher-level patterns. Table 5 only counts the "leaf patterns"; i.e. the higher-level patterns (e.g., *pagination*) have not been counted.

4.3. The pattern collections in comparison

Table 6 shows an overview of the mappings of the different pattern collections onto our categories. It shows that some of the four collections cover all (van Duyne et al., 2007) or most of the nine layers (van Welie, 2009) of Baxley's model. Other collections focus on behavioral issues, with only a few patterns related to structure or presentation (Tidwell, 2006; Yahoo! Inc., 2009). Van Duyne et al.'s (2007) collection is the only one that also contains software design patterns (the others all focus on interaction design patterns) whereas Tidwell's (2006) is the only one that also contains no *requirements* patterns but is fully focused on the conceptual implementation of the user interface. In this overview, there are clearly identifiable gaps in the pattern collections. It is also apparent what other collections one could consider to fill the gaps. An overview of the original categories of the pattern collections shows that such a comparison is not possible there because the libraries only have categories where they have patterns and the different categories are very hard to reconcile (see Table 7).

Table 6
Comparison of the four pattern collections (Tidwell, 2006; van Duyne et al., 2007; van Welie, 2009; Yahoo! Inc., 2009) by percentage (and by number of patterns).

Layer	*Tidwell*	*Van Duyne et al.*	*Van Welie*	*Yahoo!*	**Total**
Requirements		23.4% (25)	13.7% (18)	30.8% (12)	**15.3% (55)**
Conceptual Model	1.2% (1)	11.2% (12)	10.7% (14)		**7.5% (27)**
Task Flow	15.9% (13)	14.0% (15)	21.4% (28)	2.6% (1)	**15.9% (57)**
Organizational Model		5.6% (6)			**1.7% (6)**
Viewing and Navigation	17.1% (14)	9.3% (10)	33.6% (44)	17.9% (7)	**20.9% (75)**
Editing and Manipulation	12.2% (10)	2.8% (3)	3.1% (4)	2.6% (1)	**5.0% (18)**
User Assistance	29.3% (24)	6.5% (7)	9.9% (13)	43.6% (17)	**17.0% (61)**
Layout	12.2% (10)	11.2% (12)	4.6% (6)	2.6% (1)	**8.1% (29)**
Style	12.2% (10)	1.9% (2)	3.1% (4)		**4.5% (16)**
Text		4.7% (5)			**1.4% (5)**
Software Design		9.3% (10)			**2.8% (10)**
Total	**100% (82)**	**100% (107)**	**100% (131)**	**100% (39)**	**100% (359)**

Table 7
Original category names of the four pattern collections (Tidwell, 2006; van Duyne et al., 2007; van Welie, 2009; Yahoo! Inc., 2009).

Tidwell	Van Duyne et al.	Van Welie	Yahoo!
• Organizing the Content • Getting Around • Organizing the Page • Doing Things • Showing Complex Data • Getting Input From Users • Builders and Editors • Making It Look Good	• Site Genres • Creating a Navigation Framework • Creating a Powerful Homepage • Writing and Managing Content • Building Trust and Credibility • Basic E-Commerce • Advanced E-Commerce • Helping Customers Complete Tasks • Designing Effective Page Layouts • Making Site Search Fast and Relevant • Making Navigation Easy • Speeding Up Your Site • The Mobile Web	*User needs:* • Navigating around • Basic interactions • Searching • Dealing with data • Personalizing • Shopping • Making choices • Giving input • Miscellaneous *Application needs:* • Drawing attention • Feedback • Simplifying interaction *Context of design:* • Site types • Experiences • Page types	• Search • Navigation • Browsing • Selection • Rich interaction • Social

Going back to the comparison using the unified categories (Table 6), one could see for example where to look to make Tidwell's (2006) collection more complete. It contains only one *conceptual model* and no *organizational model* patterns but the collection of van Duyne et al. (2007) has some of them. The missing *text* patterns could also be found there. Van Welie (2009) has more than three times the number of *viewing and navigation* patterns, so there might also be some potential here.

4.4. The patterns categorized into multiple layers

As mentioned above, there are several patterns that could be mapped onto multiple layers. This means that these patterns are not written with a clear focus in relation to these layers but describe aspects, which sometimes contain aspects of multiple layers in one pattern. An example from van Duyne et al. (2007) is the pattern "category pages", which covers aspects of *structure*, *behavior*, and *presentation*. The pattern describes how users should be able to navigate across different parts of a large site and that these parts should be organized into categories (*organizational model*). These categories should have a consistent navigation (*viewing and navigation*) placed in the layout in a consistent way (*layout*). The use of color coding (*style*) should support discrimination between these categories. Van Welie's (2009) pattern "home link" has aspects of the structure of navigational pathways (*task flow*) and of navigational elements (*viewing and navigation*).

5. Conclusions

We conclude with a discussion of the characteristics of the different interaction design approaches first. We then discuss Baxley's (2003) model and its extension, together with the analysis of the four pattern collections. In a final step, we present our concluding vision and suggest further steps that might be taken in order to achieve it.

5.1. The characteristics of different interaction design approaches

This study looks at several approaches to interaction design, to find out how the categorization of interaction design pattern languages can be enhanced by basing it on the structural characteristics of these approaches. Analysis of design approaches shows that most of them do not go into much detail in the stage where interaction design patterns come into play; i.e., when the user interface is designed. This raises an interesting question: why do most approaches see the task of designing the user interface rather like a black box? One explanation for this could be that experienced designers and researchers like Schön (1984) consider the process of design as unpredictable to some extent. Schön sees design as an ongoing *reflective conversation* with the entity to be designed. Thus the path to the final design cannot be predicted; it is developed *de novo* in every project. This is why a linear process is not a realistic model for design activities. On the other hand, it can be argued that the performance of a *reflective practitioner* has certain distinguishable stages anyhow. Just as an artist would usually first use a pencil to sketch the overall form of a picture and later progress to oil-based paint to finish it, the work of the interaction designer moves through certain phases, building on each other. The different layered design processes can be seen as a task analysis in this sense. So even if the real process is not linear in nature, one can say that the organization of interaction design patterns, according to such a model, is "alongside the design process".

5.2. Baxley's model

We have identified Baxley's (2003) "Universal Model of the User Interface" as an interaction design approach with a very detailed description of the design phase. This model is based on common principles of design processes, such as a layered procedure, but its distinction of design sub-tasks is much more fine-grained than that of the other design processes. So it fulfills two important requirements of models for the organization of a pattern language: it is based on a design process and it provides enough "slots" to perform an effective organization. We have learned that the model does not cover all the interaction design patterns as defined by Dearden and Finlay (2006) but can be extended slightly to achieve this. However, 80% of all the analyzed patterns fall into the categories covered by Baxley's model. The model has been extended with a new category called "requirements patterns" as a working title: this is a category for the interaction design patterns that are "above" Baxley's model. For our analysis, this is detailed enough but for other pattern collections it may be valuable to sub-divide this category even further.

Even though Baxley's model has been developed in the context of web design, it seems to be robust enough to work on different platforms. The platforms can be seen as orthogonal to the model, and the "requirements patterns", as defined here, are ideally platform-independent. It can be said that the highest level of categorization with the tiers *structure*, *behavior*, and *presentation* is much more robust than the nine levels or even the sub-categories used by Baxley (2002). A voice user interface also has certain *structure*, *behavior*, and *presentation* aspects to it. But if one looks closer at the nine layers of Baxley's model, it can be seen that most notably the *presentation* tier is more focused on visual than on voice user interfaces. On the other hand, the speech of a voice user interface also has certain "*layout*" characteristics (e.g. order), a "*style*" (e.g. consistency to brand), and is made up of "*text*" (e.g. superfluous text should be avoided).

There are possible aspects to a user interface, such as audio and video, that are missing from the model. It would be necessary to add these categories to the model to make it more complete for other collections of patterns.

5.3. The analysis of the four pattern collections

To answer the question whether it is possible to organize interaction design patterns according to an extended version of Baxley's (2003) model, an analysis of four pattern collections was conducted. It cannot be guaranteed that this categorization is correct in every aspect, but we are convinced that this approach can make important aspects of pattern categorization visible. The aim of this work is *not* to discuss how individual patterns are categorized but to show the benefits of taking such a "unified" top-down approach to pattern categorization for the construction of pattern languages.

One result of the analysis is that all of the patterns focusing on interaction design could be fitted into these categories. The analysis of the four pattern collections has also shown that some of them cover most or all of the categories. However, some collections focus on behavioral issues with only few patterns related to structure or presentation issues. From this, it can be concluded that some collections are more complete than others, in the sense that they span the whole problem space of user interface design in a systematic and detailed way. The extended model provides a stable framework of "problem slots".

These conclusions indicate that the use of such a model helps to build a pattern language that is not only *complete*, but also actively supports the process of identifying new patterns; i.e., it is *predictive*. In striving for a complete pattern language, one is forced to describe the full range of "user interface problems" and has to find all the common solutions to a given level of the model. For the individual patterns, the implication is that they cannot only describe solutions on a merely morphological or behavioral level (i.e. how they look or work) but must also describe them on a semantic level (i.e. what they are used for in the user interface). A certain user interface mechanism can be used for different purposes; for example the component of a drop-down list box can be used for filtering a table (*viewing and navigation*) or for the selection of different given values for data entry (*editing and manipulation*). Following this thinking, one is forced to describe not the user interface mechanisms per se but the solution to the user's problems.

5.4. A common model for the organization of interaction design patterns

As Gabriel (1996) reports from the area of software engineering, patterns are not used enough in projects in this field and we think that the same can be said for the field of HCI. From our point of view, one of the biggest problems is that existing pattern collections are far from complete and it is not easy for the individual designer to know where to look for missing patterns. This searching for patterns is time-consuming. The vision of this work is for pattern collections to be written on the same basis – for example, based on the extended model presented here – enabling individual patterns to move freely between collections to make them more complete.

Even with the extended model as defined above, there are still some problems to be solved before the vision can be fulfilled. There are patterns that do not fit unambiguously into the model; they have mixed forces according to the levels of the model. So if one were to write patterns with forces precisely formulated in relation to the model, the particular patterns would therefore be accurately separated; this would lead to clearer identification of the levels on which gaps in a certain pattern collection exist.

Another issue is that of platform. As it has been shown above, there are patterns that are platform-independent and others that are not. We think that it makes no sense to demand patterns to be fully platform-independent in every case. This would unreasonably limit the range of issues that can be described as interaction design patterns. However, it would be a requirement to describe individual patterns as platform-independent as possible to make them available on a wide range of technical platforms. Then, in order to know which patterns can be moved from one collection to another, the patterns need to be tagged by platform-dependence to filter them accordingly.

5.5. Outlook

This work shows why Baxley's model may be a good foundation for the organization of interaction design patterns to achieve pattern collections that allow a free transfer of patterns. But there are several other milestones to be met before this vision can be achieved. The first is that although the logic of the model seems to work there are several gaps in it (audio, video, etc.) that have to be filled to make it universally applicable. To handle large numbers of patterns, the category of "requirements patterns" on the one hand and the nine layers of Baxley's model on the other should be made more fine-grained in a robust way. Also, new trends in the technology have to be taken into account. It is not the aim of this work to prove that Baxley's model is the only one capable of performing such a task, but rather to show the advantages of such a "unified model". For the moment, no better model could be found.

Because in this study we have focused more on the *construction* of complete interaction design pattern languages, in the next stage it would be important to focus on the *usage* of these languages by interaction designers. It would be very interesting to investigate whether interaction designers engaged in a design activity could specify – when interrupted and asked – to which level of the model this activity belonged.

There is another issue that must be mentioned: This work focuses on the structural aspects of pattern organization. As mentioned in the introduction, there are two main aspects to pattern organization: the organization according to several *levels of design* and the *network* of patterns defined by reference. There is already some research about the linking of design patterns (van Welie and van der Veer, 2003), but it would be interesting to explore this question to a further extent, also taking into consideration the findings presented here.

Acknowledgement

The authors would like to thank Zürcher Kantonalbank (ZKB) in Zurich, Switzerland for the support and funding of this research as part of the UCD ZKBconnect project.

References

Alexander, C., 1964. Notes on the Synthesis of Form. Harvard University Press, Cambridge.

Alexander, C., 1979. The Timeless Way of Building. Oxford University Press, New York.

Alexander, C., 1999. The origins of pattern theory: the future of the theory, and the generation of a living world. IEEE Software 16 (5), 71–82.

Alexander, C., Ishikawa, S., Silverstein, M., 1977. A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York.

Baxley, B., 2002. Making the Web Work: Defining Effective Web Applications. New Riders.

Baxley, B., 2003. Universal model of a user interface. In: Proceedings of the 2003 Conference on Designing for User Experiences. ACM, New York, NY, USA, p. 1–14.

Beyer, H., Holtzblatt, K., 1998. Contextual Design: Defining Customer-centered Systems. Morgan Kaufmann, San Francisco, CA.

Borchers, J., 2000. Interaction design patterns: twelve theses. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, vol. 2, p. 3.

Borchers, J., 2001. A Pattern Approach to Interaction Design. Wiley, Chichester, England.

Constantine, L.L., Lockwood, L.A.D., 1999. Software for Use: A Practical Guide to the Models and Methods of Usage-centered Design. Addison Wesley, Reading, Mass.

Cooper, A., Reimann, R., Cronin, D., 2007. About Face 3: The Essentials of Interaction Design. Wiley Pub., Indianapolis, IN.

Dearden, A., Finlay, J., 2006. Pattern languages in HCI: a critical review. Human–Computer Interaction 21 (1), 49–102.

Dimitriadis, Y., Goodyear, P., Retalis, S., 2009. Using e-learning design patterns to augment learners' experiences. Computers in Human Behavior 25 (5), 997–998.

Fincher, S., 2002. Patterns for HCI and cognitive dimensions: two halves of the same story. In: Kuljis, J., Baldwin, L., Scoble, R. (Eds.), Proceedings of the Fourteenth Annual Workshop of the Psychology of Programming Interest Group, pp. 156–172.

Fincher, S., 2003. PLML: pattern language markup language report of workshop held at CHI September 2003. Interfaces 56, 26–28.

Fincher, S., Windsor, P., 2000. Why patterns are not enough: some suggestions concerning an organising principle for patterns of UI design. In: CHI'2000 Workshop on Pattern Languages for Interaction Design: Building Momentum.

Gabriel, R., 1996. Patterns of Software: Tales from the Software Community. Oxford University Press, New York.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-oriented Software. Addison-Wesley Longman Publishing Co. Inc., Boston.

Garrett, J.J., 2002. The Elements of User Experience. User-centered Design for the Web. New Riders, Indianapolis, IN.

Granlund, Å., Lafrenière, D., Carr, D.A., 2001. A pattern-supported approach to the user interface design process. In: Proceedings of 9th International Conference on Human–Computer Interaction HCI International.

Grudin, J., 1992. Utility and usability: research issues and development contexts. Interacting with Computers 4 (2), 209–217.

Hornbaek, K., 2006. Current practice in measuring usability: challenges to usability studies and research. International Journal of Human–Computer Studies 64 (2), 79–102.

IBM Corporation, 1992. Object-oriented Interface Design: IBM Common User Access Guidelines. QUE, New York.

Kohls, C., Uttecht, J., 2009. Lessons learnt in mining and writing design patterns for educational interactive graphics. Computers in Human Behavior 25 (5), 1040–1055.

Leacock, M., Malone, E., Wheeler, C., 2005. Implementing a pattern library in the real world: a Yahoo! case study. In: American Society for Information Science and Technology Information Architecture Summit, Montréal, Québec, Canada.

Leuthold, S., Schmutz, P., Bargas-Avila, J.A., Tuch, A.N., Opwis, K., 2011. Vertical versus dynamic menus on the world wide web: eye tracking study measuring the influence of menu design and task complexity on user performance and subjective preference. Computers in Human Behavior 27 (1), 459–472.

Mayhew, D.J., 1999. The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design. Morgan Kaufmann Publishers, San Francisco, CA.

Nielsen, J., 1993. Usability Engineering. Academic Press, Boston.

Pauwels, S.L., Hübscher, C., Leuthold, S., Bargas-Avila, J.A., Opwis, K., 2009. Error prevention in online forms: use color instead of asterisks to mark required fields. Interacting with Computers 21 (4), 257–262.

Pauwels, S., Hübscher, C., Bargas-Avila, J., Opwis, K., 2010. Building an interaction design pattern language: a case study. Computers in Human Behavior 26 (3), 452–463.

Rantzer, M., 1996. Field Methods Casebook for Software Design: The Delta Method – A Way to Introduce Usability. John Wiley & Sons, Inc., New York, NY. pp. 91–112.

Robert, D., 1998. Designing for the User with OVID: Bridging User Interface Design and Software Engineering. Software Engineering Series. Macmillan Technical Pub., Indianapolis, IN.

Schön, D., 1984. The Reflective Practitioner: How Professionals Think in Action. Basic Books.

Sharp, H., Rogers, Y., Preece, J., 2007. Interaction Design, second ed. Wiley, New York.

Tidwell, J., 2006. Designing Interfaces. O'Reilly, Beijing.

Tidwell, J., 2009. Common Ground: A Pattern Language for Human–Computer Interface Design. <http://www.mit.edu/~jtidwell/common_ground.html> (accessed 07.05.09).

van Duyne, D.K., Landay, J.A., Hong, J.I., 2002. The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-centered Web Experience. Wesley Professional, Addison.

van Duyne, D.K., Landay, J.A., Hong, J.I., 2007. The Design of Sites: Patterns for Creating Winning Web Sites, second ed. Prentice Hall, Upper Saddle River, NJ.

van Welie, M., 2009. Patterns in Interaction Design. <http://www.welie.com/patterns/> (accessed 07.05.09).

van Welie, M., Trætteberg, H., 2000. Interaction patterns in user interfaces. In: Proc. Seventh Pattern Languages of Programs Conference: PLoP, pp. 13–16.

van Welie, M., van der Veer, G., 2003. Pattern languages in interaction design: structure and organization. In: Proceedings of Interact, vol. 3, pp. 1–5.

Yahoo! Inc., 2009. Design Pattern Library. http://developer.yahoo.com/ypatterns/ (accessed 29.05.09).