ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/304358483

Usability Pattern Identification Through Heuristic Walkthroughs

Conference Paper · July 2016

DOI: 10.1007/978-3-319-40409-7_22

CITATIONS
0

READS
22

1 author:

Manuel Burghardt
University of Leipzig
79 PUBLICATIONS **95** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Computational Musicology View project

Computational Film Studies View project

All content following this page was uploaded by Manuel Burghardt on 20 March 2019.

The user has requested enhancement of the downloaded file.

Usability Pattern Identification through Heuristic Walkthroughs

Manuel Burghardt

Media Informatics Group
Institute for Information and Media, Language and Culture
University of Regensburg
manuel.burghardt@ur.de

Abstract. Patterns are a popular means to document design knowledge in different fields of application, including HCI. Accordingly, a large number of different HCI pattern formats have been suggested. However, relatively little is said about how to systematically identify such patterns. In order to make the process of identifying patterns more transparent and comprehensible, I apply a usability inspection method to generate data that can be used as input for the systematic creation of usability patterns. This article describes the basic idea of identifying usability patterns through a series of heuristic walkthroughs and illustrates the approach by means of a case study in the field of "linguistic annotation tools".[1]

1 Introduction

During the 1970s, Christopher Alexander observed that in architecture there seems to be a "timeless way of building" (Alexander, 1979) in which certain successful – yet rather implicit – design solutions for towns appear over and over again. Accordingly, Alexander suggests to capture such implicit design knowledge, which implements a certain "quality without a name", in the form of what he calls *patterns*

1.1 Usability Patterns

Patterns are a useful format to capture design knowledge, and thus have quickly found their way into the HCI community (Borchers, 2001). The main advantage of usability patterns is that they express "a relation between a certain context, a problem, and a solution" (Alexander, 1979: 247), i.e. they do not solely describe an isolated solution, but also document for what kind of problem and in what kind of context the pattern can be used successfully. The benefits of patterns as a means to document design knowledge are also underlined by a survey (Kruschitz & Hitz, 2010), indicating that the majority (approx. 71%) of those who are using patterns during the design process feel it improves the overall process.

[1] The work described in this paper is part of a PhD project finished in 2014 (cf. Burghardt, 2014).

1.2 Pattern identification

When it comes to the creation of design pattern, two aspects are to be considered: First, patterns have to be identified, and second, they have to be written down in a generic pattern format. While there exists a plethora of different pattern formats in the area of HCI, (cf. for instance Borchers, 2001; Obrist, 2010; Tidwell, 2011; Van Welie & Trætteberg, 2000)[2] as well as guides on how to write down patterns (Wellhausen & Fiesser, 2012), Martin et al. (2001) note that little is said about how to systematically identify patterns. Alexander (1979: 258ff.) describes three basic approaches for the identification of patterns, that may be characterized as being more inductive or more deductive, or a mix of both modes of reasoning. In software engineering and HCI, the inductive approach, which is oftentimes referred to as *pattern mining*, seems to be particularly popular, following the assumption that patterns are present in the artifacts that already exist (DeLano, 1998). In practice, however, such inductive pattern mining approaches remain rather vague. As a matter of fact, very little is known about a structured process for pattern creation in the field of HCI, as it mostly relies on the implicit knowledge of experienced designers (Krischkowsky et al., 2013). This notion seems to be conflicting with the fundamental idea that usability is not a vague criterion, but rather can be tested and engineered systematically, by applying a large set of available methods.

In order to close this gap, and to make the process of identifying patterns more transparent and comprehensible, I apply a usability inspection method to generate data that can be used as input for the systematic creation of usability patterns.

2 A Systematic Approach to the Identification of Usability Patterns

The basic idea of the approach is to apply a usability inspection method to evaluate several competing products in one common domain of application (e.g. linguistic annotation tools), assuming that the solution to usability problems identified for one product is possibly present in a competing product. These problems (weaknesses) and solutions (strengths) can then be used to generate usability patterns for that very domain of application.

2.1 Description of evaluation method

As it is crucial for the pattern identification process to analyze multiple competing products, a large-scale user study would be too laborious and costly; that is why I decided to use an expert-based inspection method. While inspection methods typically recommend the use of multiple evaluators, I suggest a single evaluator approach, which is more likely to produce consistent results that in

[2] For a short review of different pattern formats (not only for HCI) cf. Kruschitz & Hitz (2009).

turn can be used as appropriate input for generic design patterns. Furthermore, the involvement of multiple evaluators is more prone to produce rather heterogeneous results with regard to the form, description and severity of identified usability problems.

From the broad spectrum of available inspection methods, I have chosen the popular *heuristic walkthrough* method (Sears, 1997). The heuristic walkthrough is a hybrid approach that borrows ideas from *heuristic evaluation*, *cognitive walkthrough*, and *usability walkthroughs*, to make up for the specific drawbacks of each single method (Sears, 1997). As I am interested not only in usability problems, the heuristic walkthrough was extended to also document good designs and potential solutions for previously identified problems. For both, usability problems and strengths, a short *ID* (for easy referencing later on), a *title*, a short *problem description* and the *heuristic* (Nielsen, 1994a) that – in case of a problem – was violated, or – in case of a strength – was obeyed, are documented.

2.2 Usability pattern structure

Van Welie (2001) describes a usability pattern structure that seems to integrate particularly well with data gathered from a heuristic walkthrough (cf. section 2.3). The following list shows the complete template with abbreviated descriptions taken from Van Welie (2001: 102ff.).

Name – short, catchy name that either relates to the problem or the solution
Problem – user problem the pattern is trying to solve, and the objectives it is trying to achieve
Usability principle – higher-level usability principle the solution is based on
Context – contextual information on when a certain pattern can be applied successfully
Forces – trade-offs that have to be made in order to find a solution
Solution – formulated in a rather abstract, generic way, so it can be applied and implemented for various different scenarios
Rationale – makes clear why a pattern works; this argumentation can be based on aspects of usability such as *performance speed*, *learnability*, *memorability*, *satisfaction*, *task completion* and *errors*
Example – concrete examples (typically a screenshot) help to understand the pattern and underline that the solution has been proven to work in existing applications
Counterexample – optional section with examples of applications that do not use the pattern
Known uses – further examples of applications that implement the pattern
Related patterns – other patterns that address related problems may be referenced

2.3 Integrating the heuristic walkthrough results with the pattern structure

As was mentioned before, usability problems and strengths were documented – together with one or more associated heuristics – during the heuristic evaluation.

These sources of input can be translated step-by-step into Van Welie's pattern structure.

Heuristics While heuristics on the one hand facilitate the systematic discovery of usability problems, they can also be used as positive usability principles that help to formulate a solution for the problem. For this reason, heuristics are used as an explicit reference in the *usability principle* section, but also as implicit input in the *solution* section of a design pattern.

Usability problems Usability problems that were identified during the heuristic walkthroughs are an essential input for several fields of the pattern: First of all, a problem can be used to find an adequate *name* for a pattern, as the name should always relate to the problem or the solution. Second, one or more problems will be paraphrased in the *problem* section. Additionally, all associated usability problems are listed alphabetically by means of their ID at the end of the problem section, to make the process more transparent. Third, usability problems can be used to describe the *context* in which the patterns can be used. They can also serve as input for the extended context section, the *forces*. As "many usability problems have fairly obvious fixes as soon as they have been identified" (Nielsen, 1994b: 31), they may also be used as input for the formulation of an adequate *solution*. Finally, usability problems can help to describe tools that qualify as a *counterexample* for the respective pattern.

Strengths Similar to the usability problems, the identified strengths can be used as input to formulate an adequate *name* for the pattern. Much like the usability problems, specific strengths can help to elaborate the context of application, which is documented in the *context* and *forces* sections. Furthermore, strengths of a product can be generalized in the *solution* section of a pattern. If appropriate, they can also be documented in the *example* (screenshot and short description) or *known uses* (mention of tool name) section.

2.4 Pattern identification process

The systematic identification of usability patterns starts by examining the results of the previously conducted series of heuristic walkthroughs for a number of competing products in one common domain of application. As was described before, the heuristic walkthrough method allows the evaluator to reveal usability problems as well as particularly good solutions for certain problems. Conducting an evaluation of several competing products, however, leads to a large number of identical or at least thematically similar problems and strengths. Accordingly, an important prerequisite for the pattern identification process is to create clusters of similar problems and strengths that recur for different evaluation objects. During this clustering phase, three basic scenarios may occur:

1. Existence of problems and appropriate solutions – Ideally, a solution to a recurring problem can be identified in one of the test objects, which may then be used to relate both, the problem categories and the good solutions in one common usability pattern.
2. Existence of solutions without concrete problems – Some patterns may also be created merely on the basis of good solutions, without explicitly documented usability problems. Note that with a single-evaluator approach, not all potential usability problems might be identified. It is, however, easy to derive the problem situation that is improved by those strengths retrospectively.
3. Existence of problems without concrete solution – In a number of cases, a group of similar problems could be identified without having a concrete solution in the heuristic walkthrough data. In these cases, new solutions were created based on the usability problems and the heuristics that are violated when the problem occurs.

Also note, that not all of the identified problems and strengths will qualify for the derivation of a specific usability pattern. Typically this is the case whenever the identified problems are not domain-specific, but rather general usability problems (e.g. bad menu structuring, bad use of typography, etc.).

3 Case study with step-by-step guide

This section describes all the steps that are necessary to translate results from a series of heuristic walkthroughs into the usability pattern format. The approach is illustrated by means of a case study, in which 11 tools from the application domain "linguistic annotation" were evaluated to discover a total of 207 problems and 84 strengths. From this data, 82 problems and 62 strengths were used to derive a total of 26 usability patterns, which are publicly available in a pattern wiki (www.annotation-usability.net). The systematic creation of one of these patterns is described in the following step-by-step guide.

3.1 Clustering according to similar content (Step 1)

Analyzing the problems according to their description content, one cluster of problems emerged that was concerned with the issue that the "annotation scheme" cannot be created inside the annotation tool, but rather has to be created in an external code editor, by means of XML markup. At the same time, there are several tools that implement an integrated annotation scheme editor, which seems to be a good solution for that problem. After the clustering, seven usability problems (cf. Table 1 for an example) were identified that are concerned with this issue, but also six strengths (cf. Table 2 for an example) could be interpreted as partial solutions for the issue. The cluster of problems and strengths that all address the issue of "annotation scheme creation" is then used as input for the different sections of the pattern structure.

Table 1. Example usability problem for the issue "annotation scheme creation".

Tool	Brat
Problem	Annotation scheme cannot be created / edited inside the tool (ID: BRA10)
Description	The annotation scheme needs to be created outside of the tool, in a code editor. If the user changes the scheme outside of the tool, the browser needs to be refreshed to apply the changes.
Heuristics	Error prevention, flexibility and efficiency of use

Table 2. Example strength for the issue "annotation scheme creation".

Tool	CATMA
Strength	Annotation scheme creation and modification via GUI (ID: CAT03)
Description	The annotation scheme can be created and edited directly in the tool, by means of graphical elements and a simple form window.
Heuristics	Flexibility and efficiency of use

3.2 Pattern name (Step 2)

In this case, the name of the pattern is closely related to the solution of the problem, which, in short, is to provide an integrated annotation scheme editor. Accordingly, the pattern is named as follows:

> **Name**: P4.1 - Integrated annotation scheme editor

Note: The pattern ID "P4.1" indicates that this is the first pattern in the fourth category ("Annotation scheme"), as the 26 patterns identified for linguistic annotation tools were eventually categorized according to six main groups of patterns. This grouping of patterns is optional, but recommended for larger pattern collections, as it facilitates the navigation in the collection.

3.3 Problem description (Step 3)

In the problem description section, the seven previously identified problems are paraphrased and summarized. In addition, all related problems are listed by means of their ID (alphabetic order). The respective *problem description* section for pattern P4.1 looks like this:

> **Problem description**: The creation of an annotation scheme that defines different levels of annotation as well as concrete annotation items on each level is a crucial task in any annotation project. Typically, annotation schemes are defined by means of document grammars known from markup languages like XML or SGML. Users without technical knowledge about markup languages will have difficulties in creating a scheme in XML syntax. At the same time, many tools require to define an annotation scheme outside the annotation tool, which makes the task

even more challenging for markup novices.

Related problems: BRA10, [...]

3.4 Usability principle (Step 4)

Nielsen's (1994a) heuristics cannot only be used to discover usability problems, but also to derive fixes for identified problems. As the heuristics were already used during the heuristic walkthrough, it suggests itself to use the same set of heuristics to describe the usability principles the solution of the pattern is based on. If the heuristics documented for the problems and strengths of the respective cluster are very heterogeneous, only the most important, predominant heuristics are mentioned in the pattern section. In this example, the two predominant heuristics that are violated by the identified usability problems, and that in turn can be used to fix these problems, are "error prevention" and "flexibility and efficiency of use": Markup novices, who have to create and modify an annotation scheme in an external XML editor are likely to produce errors during this process. At the same time, having to switch between the actual annotation tool and an external editor slows down the user and thus reduces his flexibility and efficiency.

Usability principle: Error prevention, flexibility and efficiency of use

3.5 Context (Step 5)

The context of a pattern describes in which scenarios and under which conditions it can be used. The usability problems, as well as concrete strengths, help to derive a more specific context of use. The context for this example is to provide users who are markup novices with an intuitive and efficient way to formulate annotation schemes.

Context: This pattern can be used to facilitate the creation of annotation schemes for users without technical knowledge about markup languages and document grammars.

3.6 Forces (Step 6)

This section of the pattern structure can be seen as an extension of the previous context section, as it describes some conflicting forces that may require trade-offs when the pattern is applied. The sources of input are the same as in the context section, but they are interpreted in a way that reveals potential conflicts and helps to balance existing contradictions. In this example, the main conflict is that on the technical side, annotation schemes are typically realized by means of a markup language, which is the ideal instrument for this kind of task, as it can be easily processed by machines. On the side of the user, who may not be familiar with the concept of markup languages, this may be a rather irritating and cumbersome way to create an annotation scheme.

Forces

- Annotation schemes are typically defined by means of document grammars (DTDs or XML Schemas).
- Annotation schemes are typically defined outside of the annotation tool, in a text editor that facilitates the creation of document grammars.
- For editing existing schemes during the annotation process, it is impractical to switch between the annotation tool and an external text editor.

3.7 Solution (Step 7)

This is an integral part of the pattern, as it describes a solution to the previously introduced problem. In this exemplary pattern it is possible to derive a solution from many positive examples (strengths) of other annotation tools. Essentially, the solution is to provide an integrated annotation scheme editor that allows the user to create schemes without having to use XML or some other markup syntax.

> **Solution**: The tool integrates a scheme editor that allows the user to define and edit annotation schemes inside the annotation tool. By providing a graphical user interface for the scheme editor it is also possible to hide technical details of the storage format from the user. Such an interface should utilize well-known metaphors for the creation of hierarchical structures, such as file-trees or ordered lists. It can also make use of established input elements, such as forms and input fields. It must be made clear via the interface which annotation items belong to which level of annotation, i.e. typically the annotation levels are at the highest hierarchical level of the scheme, while concrete annotation items can be subordinate to those different levels.

3.8 Rationale (Step 8)

The rationale section does not use input from the heuristic walkthrough data, but rather relies on a set of usability aspects that help to explain why and how the pattern improves the usability of an annotation tool. The five usability aspects that are used in this pattern collection were introduced by Nielsen (1993: 26ff.): *learnability*, *efficiency*, *memorability*, *error rate*, and *satisfaction*. In this example the suggested solution improves the usability of an annotation tool as it helps to increase the overall performance speed (*efficiency*), but also helps to quicker learn (*learnability*) the tool and to avoid errors (*error rate*) when using it productively.

> **Rationale**: As ad hoc modifications of the annotation scheme are part of the typical annotation process, an integrated editor for annotation schemes speeds up the overall annotation process (*efficiency of use*). At

the same time, the availability of a GUI for the creation and modification of annotation schemes increases the *learnability* of the annotation tool and decreases the number of potential *errors* that may occur when novices are forced to translate linguistic annotation schemes into formal markup languages.

3.9 Examples and prototypes (Step 9)

This section of the pattern contains screenshots that illustrate the solution as a concrete example in an existing tool. It is important to note that these examples do not present the pattern's solution in an isolated way, but rather as part of the overall tool interface. Therefore, an example that illustrates a good solution may, nevertheless, be accompanied by usability problems that are related to other interface aspects.

In the few cases where there is no positive example available in the tested tools, an interactive prototype that implements the proposed solution and that may serve as an example will be provided. For this example pattern, a number of tools can be identified that implement the suggested solution of an integrated, non-XML annotation scheme editor (cf. Figure 1)

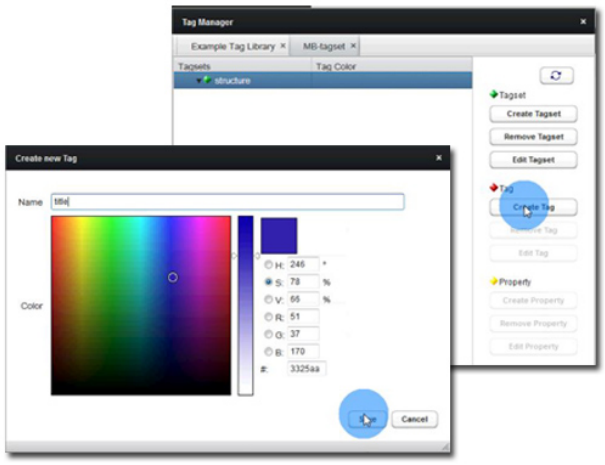

Fig. 1. CATMA – Integrated annotation scheme editor.

3.10 Optional counterexamples (Step 10)

The optional section counterexamples was left out in most patterns, as the information is largely redundant with the alphabetic list of concrete usability problems that is provided in the problem description section: All tools that are

documented with a usability problem for a specific pattern may as well serve as a counterexample.

3.11 Known uses (Step 11)

This section can be seen as an extension of the *examples* category, as it lists all tools that implement the solution of the pattern, but have not been mentioned in the examples section for means of reduced redundancy. In some cases – though not in this example pattern – the *known uses* section may also contain examples for tools that have not been part of the heuristic walkthrough evaluation, but that are known from previous studies or from hints in related literature.

Known uses: *GATE*, *Knowtator*

3.12 Related patterns (Step 12)

One of the strengths of the pattern format is the possibility to relate different patterns to each other, and thus create an interconnected repository for design knowledge. Relations may either point to a pattern from an external collection of HCI patterns, or to other patterns in the same collection.

In the domain of HCI, there are numerous pattern collections[3] that describe good design on various levels of abstraction, e.g. concrete interaction elements, general user behaviors, etc. As many of the patterns described in these collections are either redundant, or have a strong focus on web-based interfaces, I only relate to two other pattern collections that are quite popular throughout the HCI community (Van Welie & Trætteberg 2000, Tidwell 2011).

According to Conte (2002), pattern relations in general can be of the following four types: P_A *uses* P_B, P_A *refines* P_B, P_A *requires* P_B, P_A is an *alternative* for P_B. All relations in the pattern collection will be created according to these basic types.

For the example pattern P4.1, the following relations to internal as well as external patterns can be identified:

Related patterns
- Internal relations: This pattern uses P4.2
- External relations: This pattern uses the "Structured format" pattern (Tidwell 2011) and the "Input hints" pattern (Tidwell 2011)

4 Discussion

While patterns provide a detailed description of a problem and its solution in a generic form that can be understood by tool developers as well as tool users, it must be noted that the systematic identification of such patterns is far more

[3] On his website, Van Welie gives an overview of many other collections of HCI patterns (cf. http://www.welie.com/index.php).

time-consuming than the creation of a traditional usability report. It has also shown that not all problems and strengths are appropriate as input for a usability pattern. De facto, many of the observed problems are either too specific or too general to be suitable for the derivation of a pattern.

In a follow up-study, the pattern identification process was applied to the domain of "graph visualization tools", with a smaller number of evaluation objects: The evaluation of 3 different graph visualization tools (*Tulip*, *Cytoscape* and *Gephi*) by means of a heuristic walkthrough led to a total of 79 usability problems and 24 strengths. 43 problems and 14 strengths qualified as input for a total of 14 patterns. The results of this study indicate, that a decent amount of patterns can also be identified with a smaller number of evaluation objects.

Another point of discussion is the validity and applicability of the patterns. Van Welie & Van der Veer (2003) point out that pattern languages are always subjective to a certain degree, as they reflect the mental model of the designer who created it. For a discussion of the validity and applicability of a pattern collection, it is important to comprehend the creation of design patterns as an iterative process. This process starts with a first (subjective) suggestion of a "pattern candidate" (Ratzka, 2008) and is successively refined afterwards. In the software engineering and HCI communities, the successive refinement has been formalized as the so called shepherding process, which is essentially a reviewing process where the shepherds (experienced pattern authors) help to improve the pattern candidates of novice patterns authors (Harrison, 1999). The approach suggested in this article is well suited for the identification of pattern candidates that can be further refined in a subsequent shepherding process.

References

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

Borchers, J. O. (2001). A Pattern Approach to Interaction Design. Chichester: Wiley.

Burghardt, M. (2014). Engineering Annotation Usability - Toward Usability Patterns for Linguistic Annotation Tools. Doctoral dissertation, University of Regensburg, urn:nbn:de:bvb:355-epub-307682. Retrieved February 28, 2016, from http://epub.uni-regensburg.de/30768/

Conte, A., Fredj, M., Hassine, I., Giraudin, J.-P., & Rieu, D. (2002). A tool and a formalism to design and apply patterns. In Z. Bellahsene, D. Patel, & C. Rolland (Eds.), Proceedings of the 8th International Conference on Object-Oriented Information Systems, OOIS 02 (pp. 135146). London: Springer.

DeLano, D. E. (1998). Patterns Mining. In L. Rising (Ed.), The Patterns Handbook: Techniques, Strategies, and Applications (pp. 8795). Cambridge, UK: Press Syndicate of the University of Cambridge.

Harrison, N. B. (1999). The Language of Shepherding - A Pattern Language for Shepherds and Sheep. Retrieved from http://hillside.net/index.php/the-language-of-shepherding

Krischkowsky, A., Wurhofer, D., Perterer, N., & Tscheligi, M. (2013). Developing Patterns Step-by-Step A Pattern Generation Guidance for HCI Researchers. In A. Zimmermann (Ed.), Proceedings of the 5th International Conferences on Pervasive Patterns and Applications, PATTERNS 13 (pp. 6672). ThinkMind.

Kruschitz, C., & Hitz, M. (2009). The Anatomy of HCI Design Patterns. In P. Dini, W. Gentzsch, P. Geraci, P. Lorenz, & K. Singh (Eds.), Proceedings of Computation World 2009: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns (pp. 202207). Los Alamitos, CA: IEEE.

Kruschitz, C., & Hitz, M. (2010). Are human-computer interaction design patterns really used? In Proceedings of the 6th Nordic Conference on Human-Computer Interaction Extending Boundaries, NordiCHI 10 (pp. 711714). New York: ACM.

Martin, D., Rodden, T., Rouncefield, M., Sommerville, I., & Viller, S. (2001). Finding patterns in the fieldwork. In W. Prinz, M. Jarke, Y. Rogers, K. Schmidt, & V. Wulf (Eds.), Proceedings of the seventh conference on European Conference on Computer Supported Cooperative Work, ECSCW01 (pp. 3958). Norwell, MA: Kluwer Academic Publishing.

Nielsen, J. (1993). Usability Engineering. Amsterdam et al.: Morgan Kaufman.

Nielsen, J. (1994a). Enhancing the explanatory power of usability heuristics. In C. Plaisant (Ed.), Conference Companion on Human Factors in Computing Systems, CHI 94 (pp. 152158). New York: ACM.

Nielsen, J. (1994b). Heuristic evaluation. In J. Nielsen & R. Mack (Eds.), Usability Inspection Methods (pp. 2562). New York: Wiley & Sons.

Obrist, M., Wurhofer, D., Beck, E., & Tscheligi, M. (2010). CUX patterns approach: Towards contextual user experience patterns. In Proceedings of the 2nd International Conferences on Pervasive Patterns and Applications, PATTERNS 10 (pp. 6065). ThinkMind.

Ratzka, A. (2008). Steps in Identifying Interaction Design Patterns for Multimodal Systems. In P. Fobrig & F. Patern (Eds.), Proceedings of the 2nd Conference on Human-Centered Software Engineering, HCSE 08, and 7th International Workshop on Task Models and Diagrams, TAMODIA '08 (pp. 5871). Berlin, Heidelberg: Springer.

Sears, A. (1997). Heuristic Walkthroughs: Finding the Problems Without the Noise. International Journal of Human-Computer Interaction, 9(3), 213234.

Tidwell, J. (2011). Designing Interfaces (2nd ed.). Sebastopol, CA: OReilly Media.

Van Welie, M., & Trætteberg, H. (2000). Interaction patterns in user interfaces. Retrieved February 28, 2016, from http://hillside.net/plop/plop/plop2k/proceedings/Welie/Welie.pdf

Van Welie, M. (2001). Task-Based User Interface Design. Doctoral dissertation, Vrije Universiteit Amsterdam, SIKS Dissertation Series No. 2001-6.

Van Welie, M., & Van der Veer, G. C. (2003). Pattern languages in interaction design: Structure and organization. Retrieved June 12, 2013, from http://www.idemployee.id.tue.nl/g.w.m.rauterberg/conferences/interact2003/INTERACT2003-p527.pdf

Wellhausen, T., & Fiesser, A. (2012). How to write a pattern? A rough guide for first-time pattern authors. In Proceedings of the 16th European Conference on Pattern Languages of Programs, EuroPLoP 11 (Article No. 5). New York: ACM.

View publication stats