

The Use of Anti-Patterns in Human Computer Interaction: Wise or Ill-Advised?

JUDY VAN BILJON AND PAULA KOTZÉ

University of South Africa
and

KAREN RENAUD AND MARILYN MCGEE

University of Glasgow
and

AHMED SEFFAH

Concordia University

In this paper the tenability of anti-patterns in Human-Computer Interaction is explored. Patterns have been accepted as being useful in software development and more recently also in Human-Computer Interaction. A concerted effort is being made in Software Engineering to identify and document anti-patterns. Patterns and anti-patterns are essentially about transferring captured expert knowledge, therefore compatibility between the nature of anti-patterns and the nature of the learner's internal knowledge representation and processing is crucial. This paper addresses the differences and similarities between patterns and anti-patterns and how this impacts on the mental models and cognitive processing of patterns and anti-patterns. We present evidence from theories of mental modelling and reasoning that highlight possible significant dangers in the use of anti-patterns to teach novices human-computer interaction principles. If the notion that the current representation of anti-patterns is not supporting cognitive processing is correct, a new approach to structuring anti-patterns is needed. Recommendations are made towards a new specification technique for HCI anti-patterns.

Categories and Subject Descriptors: H5.2 [Information Interfaces and Presentation]: *User Interfaces* - User-centred design; Interaction styles; Theory and methods

General Terms: Design, Human Factors, Performance

Additional Key Words and Phrases: Human-computer interaction, Patterns, Anti-patterns, Mental models

1. INTRODUCTION

The use and value of patterns is accepted in the field of Software Engineering (SE) [Coplien 2004; Jézquel et al. 1999]. Patterns for Human-Computer Interaction (HCI) design were first discussed in the late nineties and since then the HCI community has increasingly promoted HCI patterns as a design tool for user interfaces [Zajicek 2004]. A user interface pattern, also called an HCI pattern, is generally defined as a proven solution for a common user or usability problem that occurs in a specific context of work [Dearden 2000, Bergin 2002]. HCI patterns for supporting the design process and as an alternative/complementary design tool to produce guidelines for designers is slowly gaining momentum [Van Welie and Van der Veer 2003]. Many HCI experts have devoted themselves to the development of HCI pattern languages and patterns, which are generally perceived to be a useful design tool [Borchers 2000].

An anti-pattern is something that looks like a good idea, but which backfires badly when applied [Cockburn et al. 2004]. The use of anti-patterns are gaining general acceptance in Software Engineering [Brown et al. 1998]. The idea of patterns and anti-patterns can be seen in other domains as well, such as in design guidelines commonly issued by system manufacturers, which can be categorised as describing what should be done (i.e. prescribe a design model) or describing what should not be done (i.e. discourage a particular implementation) [Sinnig *et al.* 2004]. This echoes the notion of patterns versus anti-pattern.

The use of anti-patterns, at first glance, seems to be as beneficial to the SE and HCI communities as indeed patterns are themselves. However, if one considers users' cognitive processing strategies and limitations, potentially fatal problems emerge in the use of anti-patterns. The remainder of this paper explores the nature of patterns and anti-patterns, considers their differences and similarities, and relates this to the nature of human information processing and

Author Addresses:

J.A. van Biljon, School of Computing, University of South Africa, P O Box 392, UNISA, 0003, South Africa; vbiljja@unisa.ac.za

P. Kotzé, School of Computing, University of South Africa, P O Box 392, UNISA, 0003, South Africa; kotzep@unisa.ac.za

K.V. Renaud, Department of Computing Science, The University of Glasgow, 17 Lilybank Gardens, Glasgow, G12 8RZ, United Kingdom; karen@dcs.gla.ac.uk

M. McGee, Department of Computing Science, The University of Glasgow, 17 Lilybank Gardens, Glasgow, G12 8RZ, United Kingdom; mcgeemr@dcs.gla.ac.uk

A. Seffah, Department of Computer Science, Concordia University, Montreal, Canada seffah@cs.concordia.ca

The research reported on here was conducted while the co-authors were visiting lecturers at the University of South Africa.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, that the copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than SAICSIT or the ACM must be honoured. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

© 2004 SAICSIT

reasoning. In doing so, we shed light on some possible scenarios for the misuse of anti-patterns in transferring knowledge to HCI novices. A case study of the use of simple anti-patterns is presented to support this argument.

If it is accepted that there are fundamental differences in the way that patterns and anti-patterns are processed, then this has implications for anti-pattern design. A new specification technique for HCI anti-patterns, which takes these human limitations into account, is proposed.

2. PATTERNS AND ANTI-PATTERNS

2.1 Patterns

It is generally accepted that patterns originated in the field of building architecture, when Christopher Alexander invented the idea of capturing design guidelines in the form of design patterns [Alexander and Silverstein 1977]. However, the idea of specifying the best way of doing things originated far earlier with the work of Taylor [1911] and was later refined by Frank and Lillian Gilbreth [1917]. The use of SE patterns was first introduced by Gamma et al. [1995] and since then the research and documentation of patterns in the field of Software Engineering has proliferated. The use of patterns in HCI was a natural progression from the use of patterns in SE, with HCI patterns being discussed at workshops since 1997 (CHI'97, INTERACT'99, HCI'00) [Dearden 2000]. Some important papers have been published on using patterns for designing user interfaces, an area which appears to be gaining more and more acceptance in the HCI world [Erickson 2000, Martin et al. 2001, Van Welie and Van der Veer 2003].

Patterns in each particular field have their own specialized vocabulary and structure. There does not appear to be a general consensus within the HCI community about what patterns are, how they should be presented, what their purpose should be and how they should be used [Borchers and Thomas 2001]. However, there does seem to be general agreement on some core concepts, which are consistent with Gamma's definition stating that a *problem* has a set of goals and constraints collectively referred to as *forces*, which occur in a specified context. The *context*, in this case, is the recurring set of situations within which the pattern applies. The pattern is a *generalised solution that can be applied to resolve the forces*, the core solution to the recurring problem [Gamma et al. 1995]. Pattern formats range from purely narrative approaches to more structured approaches. A comprehensive list of HCI patterns is available from Sally Fincher's Pattern Form Gallery [Fincher 2003] and Van Welie's collection [Van Welie 2004]. HCI patterns are defined in different categories including task representation, dialogue, navigation, information, status representation, layout, device aspects and physical interaction, user-profile, and overall system architecture [Dearden 2000]. HCI patterns differ from SE patterns in the specification of scope, level of abstraction and specification format, but both HCI and SE patterns are primarily concerned with knowledge transfer [Borchers 2000].

The designer's receptivity to the pattern creator's envisaged transfer of pattern-encapsulated knowledge will depend on how well the pattern is formulated. The quality of the pattern does not depend on the technical brilliance of the implemented design, but rather on the quality of the mental model the user constructs as a result of the way in which the pattern is structured and presented. This internalised mental model will be matched against future design problems encountered by the designer, and used if the problem matches the potential solution proffered by the model. If the model is sufficiently well captured there is a better chance of the designer identifying it and using it. Hence, the efficacy of any design pattern's knowledge transfer process depends on how well the issues in the pattern are communicated to the HCI designer *at the first encounter*, which is when the pattern is first understood and internalised.

The tricky problem in the formulation of effective patterns therefore lies in ensuring that the formulation satisfies the needs of naïve designers. The dilemma is that experts often omit essential details, simply because they assume knowledge of those facts. The efforts of many researchers in the field of HCI patterns have been aimed at closing this communication gap. Over the last few years, the idea of anti-patterns has gained importance in SE pattern research [Mahernoff and Johnston 1998, Van Welie and Van der Veer 2003]. Anti-patterns document what usually goes wrong in software development, and how one can avoid these mishaps [Cockburn et al. 2004]. At first glance this appears to be a perfectly sensible progression from patterns themselves. When an expert is involved in any project he/she will often act as a brake on the ambitious but ill-advised efforts of inexperienced designers. Hence it appears to be sensible to capture the knowledge of 'what not to do' as well as the knowledge of how things ought to be done. The following section describes and contemplates anti-patterns.

2.2 Anti-patterns

The idea of patterns in HCI developed from the idea of patterns in SE, although there are marked differences [Borchers 2000]. In order to identify something as a pattern it is necessary to identify a particular pattern, show that it exists in a successful system and also show that those patterns are absent in unsuccessful systems [Long 2001]. Likewise, the presence of anti-patterns must be identified in unsuccessful systems, and their absence shown in successful systems in order for them to qualify as anti-patterns. The basic rationale in publishing anti-patterns is to identify recurring design flaws for the purpose of preventing other people from making the same mistakes [McCormick 2004A].

An anti-pattern has two possible forms: it either *provides knowledge on how to go from a problem to a bad solution* or shows *how to go from a bad solution to a good solution* [Cockburn et al. 2004]. The former will be referred to as a *simple* anti-pattern and the latter is commonly called an *amelioration* anti-pattern. If described properly,

an anti-pattern also tells the designer why the bad solution looks attractive, why it turns out to be bad, and what positive patterns are applicable in its stead. Anti-patterns therefore concentrate on presenting negative solutions [Brown et al 1998, McCormick 2004A]. Simple anti-patterns are thus not very useful to the designer, behaving as a mere example of what can go wrong; the amelioration pattern is constructive and useful to the designer since it shows how the bad solution can be refactored.

In the theory of pattern languages, actually developed more extensively in computer architecture rather than in building architecture, the concept of ‘anti-pattern’ plays a central role. Unfortunately, the solution space is not one-dimensional; a pattern may have a number of associated anti-patterns, and a pattern used in a different context may be an anti-pattern. Furthermore, the negative of an anti-pattern is not necessarily a pattern – it could merely be another anti-pattern. For example, consider a design guideline such as: *Use blue as a background colour*. The negative of this guideline (simple anti-pattern) would be: *Do not use anything other than blue as a background colour*. A negative of this anti-pattern could be: *Use something other than blue as a background colour*, which is simply another simple anti-pattern. Since patterns are far more comprehensive than guidelines the definition of the negative of a pattern is non-trivial.

At this stage research into software design anti-patterns abounds [Cockburn et al. 2004], but none of the major HCI pattern collections make any reference to anti-patterns [Van Welie 2004, Tidwell 2004, Fincher 2004]. The reason for the absence of defining research into the use of HCI anti-patterns may be due to the youth of HCI patterns in general. If anti-patterns follow the historic trend of patterns, we might expect to see an escalating interest in the use of anti-patterns in HCI. However, we should consider, *before* the anticipated arrival of HCI anti-patterns, whether this natural next step is indeed the beneficial innovation it appears to be.

In conclusion, patterns and anti-patterns exist to capture experts’ expertise, and to communicate knowledge. Hence, in deciding on the reception we should be extending to anti-patterns, we should take a closer look at how patterns and anti-patterns achieve this knowledge transfer.

3. KNOWLEDGE TRANSFER USING PATTERNS AND ANTI-PATTERNS

One of the main challenges for HCI as a discipline is to develop effective techniques for communicating the knowledge of HCI experts. This kind of knowledge and expertise can be accumulated about solutions that work in particular circumstances. Most traditionally scientific knowledge is passed on reasonably successfully in textbooks. There are few areas of interaction design, however, that can genuinely be said to be built on comparable scientific principles. In engineering and craft fields the traditional way of learning has normally been the accumulation of skills and knowledge via a skilled craftsman as an apprentice. Such apprenticeships in interaction design are rare. Alternatives are therefore required for passing on expertise in these fields. One such technique is the use of design patterns.

Patterns are essentially a tool for capturing and transferring experience in a standard and understandable way. Apart from the transfer of expert knowledge, patterns also improve knowledge processing by chunking concepts together. According to [Chang and Tuovinen 2004] it is this combination of elements for storage in the long-term memory, and the later efficient retrieval and utilisation of the pattern that enhance cognitive capacity. The next section considers how mental models are structured and used in problem solving.

3.1 Mental Models

Mental models are small-scale psychological representations of real, hypothetical, or imaginary situations [Craik 1943]. They are constructed to anticipate events, to reason and to underlie explanation. Cognitive scientists have since argued that the mind constructs mental models as a result of perception, imagination and knowledge, and the comprehension of discourse [Johnson-Laird et al. 1998]. It is therefore reasonable to assume that we construct mental models to represent patterns and anti-patterns in a problem context. Mental models are more than just mental pictures or imagery. Some models cannot, in fact, be visualized at all. Models can represent abstract notions, such as negation and ownership. It is the concept of modelling *negation* that has relevance when considering the internalising and processing of anti-patterns.

Mental models can be thought to represent explicitly what is true *but not* what is false. This is partly due to the fact that the capacity of working memory is limited. Individuals tend to minimize the load on working memory by constructing mental models that represent only what is true. To focus only on models of what is true is a sensible way to deal with limited processing capacity – but it does lead to illusions. Reasoners are assumed to make ‘mental footnotes’ about the propositions that are false but these footnotes are sometimes not taken into account, this can lead to systematic errors.

Falsity is a semantic notion where negation is a syntactic one. This means that a negative assertion can be true or false. For example, consider the following premise: *This is a solution to a particular design problem*. The negation of this premise would then be: *This is not a solution to a particular design problem*. The negation of a premise is kept as a ‘mental footnote’ in the designer’s mind, whereas the solution itself is kept as a mental model. An anti-pattern *is* a solution – albeit a bad one. Hence the solution itself is stored as a mental model, with a footnote stored about the ‘badness’ of the solution. If the designer ‘loses’ the footnote the anti-pattern solution could be incorrectly used as a solution.

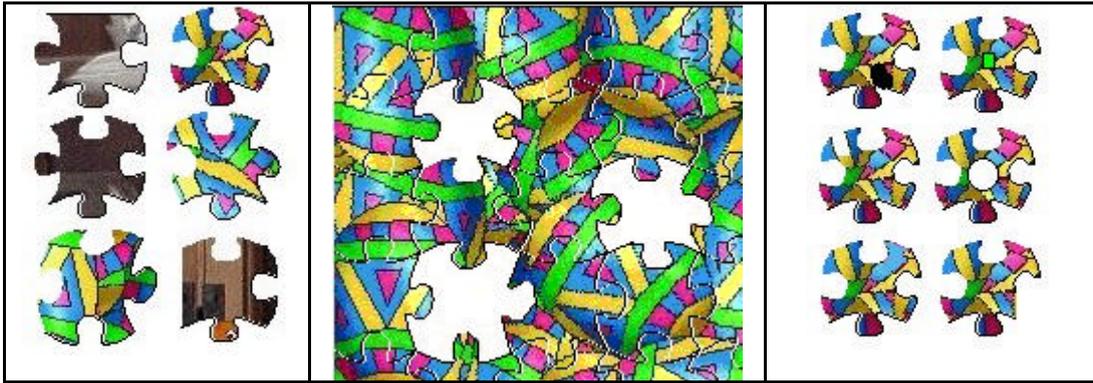


Figure 1: Design problem with patterns (left), anti-patterns (right)

In focusing on stored models it is easier to reason if only one model is available than if multiple models are available [Johnson-Laird and Byrne 2000]. Hence a mental model based on *one* pattern is superior to a number of mental models, some of which are based on anti-patterns. This leads to our argument that counter-examples can be used effectively only once the pattern (correct model) has been well-established.

For example, if you say to a designer ‘don’t use red print on a green button’, the designer has to think about the green button with red print on it before storing the footnote that this is an unwise course of action. We tend to internalise what we focus on so when the designer thinks of the green button with red print that is often what he or she tends to use. The moral of this type of story is that an experienced designer should ‘ask for what you want, *not* what you don’t want’. The truth of this is borne out by the fact that negative information is often ignored in decision making, which is why novices often fail to learn from experience. Mental model theory postulates that human reasoners can, in principle, see the force of counterexamples, and indeed people are able to construct those [Johnson-Laird et al. 1998] but the benefits need to be carefully weighed against the potential disadvantages.

Having considered the essence of patterns, anti-patterns and mental models, we now take a look at knowledge transfer from expert designers to novices.

3.2 Knowledge Transfer

The key issue is whether the use of anti-patterns is a desirable mechanism for transferring knowledge and if so, what the requirements are for supporting cognition. The use of patterns in knowledge transfer is well-established. For example, when bank tellers are trained to recognize counterfeit money they are not shown examples of counterfeit money, which, at first, appears the obvious way to go about teaching them the difference. On the contrary, they spend a great deal of time handling only valid currency (pattern matching). Only once they are completely familiar with this is a counterfeit note (anti-pattern) slipped into the pack. Their knowledge of the look and feel of valid notes (the pattern) facilitates immediate identification of the counterfeit note (anti-pattern).

Consider Figure 1, where the designer is faced with a design problem, and compares each of the design patterns on the left of the image to the problem domain until one is found that fits in terms of *problem, forces* and *context of use*. The designer compares the problem domain to the possible pattern solutions, and identifies a pattern. The anti-patterns on the right are close enough to the correct pattern to be mistaken as patterns by an inexperienced designer, but each has marked deficiencies that make it unsuitable.

When the user carries out a pattern-matching process he or she is mentally working through a tickable checklist, and rejects the pattern if some aspect of the pattern does not match the requirements. Anti-patterns may have many of the checklist items ticked off but will fail in some respect and be rejected. The presence of the anti-patterns can serve to strengthen the pattern-recognition process for the experienced designer because it allows the user to refine the pattern-matching process. However, if the inexperienced user is being exposed to anti-patterns one has to guard against these being used instead of the correct pattern. The anti-pattern may indeed be used if one of the following happens:

- The designer does not remember that it is an *anti-pattern* [Long 2001].
- The designer incorrectly matches the anti-pattern to the problem domain. This could be caused by the designer not understanding the rationale for the anti-pattern, as it is currently defined.
- The designer has no correct pattern to hand or is unable to recall it. Being exposed to a range of anti-patterns more often than the correct patterns will result in a stronger memory trace for the anti-patterns (regardless of whether or not to the relative goodness or badness of the pattern was stored). This could result in retrieval of the anti-pattern more readily than the pattern in some instances due to the availability bias [Johnson-Laird and Savary 1996]. This would not be problematic if the designer has attached a footnote to the memory of the anti-pattern as a reminder that it is this anti-pattern that should be avoided.

If this footnote is not encoded at all, encoded incorrectly or encoded but not retrieved with the anti-pattern, then each of these circumstances could result in the anti-pattern being incorrectly selected as the correct solution. To prevent anti-patterns from being used in place of patterns we therefore have to address these three issues. The first and second

can be addressed by improving the way anti-patterns are formulated, and this will be addressed in Section 5. The third can be addressed by ensuring that designers are exposed to good patterns first – and often – *before* they are exposed to anti-patterns. Only once these ‘good’ design solutions have been well and truly established in the designer’s mind should he/she be exposed to anti-patterns.

An anti-pattern is more than the negative of the pattern. However, if stating instructions in the negative is counterproductive to knowledge transfer, it has implications for the anti-pattern concept. The following section discusses a case study carried out to determine the effects of first exposure to either patterns or anti-patterns.

4. CASE STUDY: EXPERIENCES WITH NEGATIVITY

In order to test our ideas we carried out a case study. The aim of the case study was to find the difference in the effectiveness of knowledge transfer between patterns, anti-patterns and the combination of an anti-pattern followed by a pattern. Since guidelines are simpler to use and to test effects of than patterns [Wesson and Cowley 2003], anti-patterns and patterns are represented by negative and positive guidelines and not by complete patterns in this case study. The rationale is that if there is a difference in the comprehension and processing of guidelines and anti-guidelines, this difference is bound to be compounded in the use of patterns and anti-patterns. Using simple instructions also avoids interrelationship dynamics.

A class of 66 Honours students took a course in Database and Internet technology. One of the lectures they were given concentrated on e-commerce concepts and issues and presented various concepts both in negative and positive format. The students were required to form groups of 3 students each and to develop an e-auction site given the guidelines in sections 4.1 to 4.3.

4.1 Guideline

First the whole class was given the positive guideline and the results (whether they applied the guideline or not) were recorded. The guidelines were:

- Use an email address instead of a site-specific user identification.
- Put an explicit logout button on the site.

4.2 Negative Guideline

Next they were exposed to the following negative guidelines and the results (whether they applied the guideline or not) were recorded:

- Don’t expect users to log in if they just want to browse (The positive guideline would be: Browsers need not log in).
- Don't use techno-speak (The positive guideline would be: Use universally accessible terminology).

The image shows a web page with a black border and the title "Welcome, please enter your details" in bold black text. Below the title are four input fields: "Name:" with a yellow highlight, "Surname:", "Password", and "Confirm Password".

Figure 2: Web Page based on Anti-Pattern

4.3 Anti-Pattern Followed by Positive Guideline

Finally, the group was exposed to a simple anti-pattern followed by a positive guideline and the results (whether they applied the guideline or not) were recorded. It should be noted that whereas the user context in the case of the guidelines and negative guidelines was variable and certainly not manipulated by the lecturer, in this case the students had been exposed to a bad solution (an anti-pattern) in a previous assessment. Students were given the following guidelines:

- *Anti-Pattern*: Students were given the task of writing JavaScript to validate a password. An HTML page was provided for the data entry and the students were instructed to embed JavaScript in the HTML to validate the password. No user instructions about the requirements for the password were displayed on the page, as shown in Figure 2. They were instructed to display a message if the password did not meet strong password requirements, or if the password and confirmed password were different.

- *Guideline*: In a subsequent lecture students were told that they should display instructions next to the password field so that users would know what was required. It is important to note that the lecturer did not refer to their previous experience, but only presented this guideline in a positive way. It was hoped that the students' previous experience with the anti-pattern – waiting for the user to make a mistake before telling him/her what was required – would make the desirability of the new pattern obvious.

When the projects were marked the results as depicted in Table 1 was observed.

		Done	Not Done
Negative Guideline	Allow users to browse item before they are required to log in	11	11
Negative Guideline	Use techno-speak. <i>This includes the use of jpg, gif, SQL Warnings, Error Messages with Exception printouts</i>	16	6
Guideline	Use only an email address as a user id	20	2
Guideline	Put an explicit logout button on the site	19	2
Anti-Pattern then Guideline	Put instructions next to the password field	3	19

Table 1: Results from case study on the knowledge transfer of positive and negative guidelines

The two *negative guidelines* had limited success in influencing behaviour. The first was emphasized in the lecture, with examples given from real-life E-Commerce websites. Even so, only half the class took the message to heart. The use of techno-speak is perhaps inevitable amongst Computing Science students, but since it was discussed in lectures one would have expected students to have been more aware of this as a potential HCI problem. The *guidelines* can be expected to be very well established in these students' minds since they are all avid computer users and many of them order items from e-commerce sites. The most popular means of identification on these sites is the email address and sites usually have a logout button. The *anti-pattern followed by the guideline* produces an interesting effect. Students received this instruction in the same lecture as the previous guidelines, but the influence of the previously experienced anti-pattern was so strong that most of them were unaffected by the positive guideline.

The positive and negative guidelines were given separately. The guidelines were selected to be similar in complexity but it remains a limitation as some guidelines may have been easier to follow. These results indicate that the positive guidelines appear to be more effective in knowledge transfer – but only if the guideline is the designer's first encounter with the design problem. There may, of course, be other factors which influenced the students, as mentioned in the discussion above, and the sample size is too small to draw solid conclusions from the case study. However, the finding that positive guidelines are more effective in supporting knowledge transfer than negative guidelines is in keeping with the theories of mental models (as discussed in section 3.2). This goes some way towards substantiating the claim that anti-patterns and patterns cannot be expected to perform in the same way in transferring knowledge to novices.

This case study highlights another need which emerges in the study of the use of patterns and anti-patterns in HCI practice: that of designer context (the pattern or anti-pattern user's previous exposure to the design problem for which the pattern is being offered as a solution). If the first exposure to a particular design problem is an anti-pattern, then the presentation of the correct pattern will have to be emphasized so that the designer does not remember the anti-pattern as the correct solution to the problem. If the designer being exposed to the anti-pattern is an experienced designer with experience of the particular design issue then the anti-pattern will probably have a positive effect. If, on the other hand, the designer is inexperienced, the anti-pattern may be remembered and recalled as a pattern – and it will be very difficult to eradicate this and to replace it with the correct *pattern*.

The next section will consider a more positive formulation of anti-patterns to better support correct identification of these as anti-patterns when a designer is seeking a solution to a design problem.

5. COMPARING PATTERN AND ANTI-PATTERN SPECIFICATIONS

As a basis for comparison the pattern template according to the CHI 2003 workshop report was chosen [Fincher 2003]. No anti-pattern template or guidelines could be found in either of the following major pattern collections [Van Welie 2004, Tidwell 2004, Fincher 2004] and therefore an anti-pattern template from SE is used [Brown et al. 1998]. McCormick [2004B] and [Howe 2004] also provide anti-pattern templates from SE, which first give the solution and then state what is wrong with it. The structure of patterns and anti-patterns are traditionally defined in the form of templates. The attributes that these templates provide are shown in Table 2.

Patterns and anti-patterns support the same architectural principles, as specified by Faridul [2003] namely abstraction, encapsulation, separation of concerns, cohesion and coupling. Patterns and anti-pattern specifications are remarkably similar. What can be done to facilitate the correct recording of the anti-pattern, as an anti-pattern – and to assist the designer in rejecting it as a possible solution to the problem?

	HCI patterns [Fincher, S., 2003]	Anti-patterns [Brown et al. 1998]
Problem	✓	
Solution	✓	✓ (anti-pattern solution)
Refactored solution		✓
Variations		✓
Context of use	✓	✓
Name	✓	✓
Illustration	✓	✓ (background)
Forces	✓	root causes & unbalanced forces
Consequences	✓ and benefits	✓ causes and symptoms
Diagram	✓	
Evidence	✓	✓ anecdotal evidence
Related patterns	✓	✓
Example	✓	✓
Rationale	✓	
Relationship between the problem and the solution	✓	

Table 2: Comparison of pattern and anti-pattern templates

In the previous section we argued for the inexperienced designer being made familiar with patterns before they are exposed to anti-patterns. Hence we have to ensure that the anti-pattern is inextricably linked to the correct pattern in the user’s mind. The pattern forms the foundation and the anti-patterns merely act to strengthen it, not to act as alternatives.

The typical anti-pattern specification starts with anecdotal evidence, suggests the anti-pattern solution that could have caused the problem, suggests a refactored solution – a pattern, to correct the problem, and then tenders benefits and consequences of the refactored solution. It is difficult for the designer to link this with the correct solution when it is time to design a new system, and this could lead him/her to choose the anti-pattern as a solution to the problem. What we suggest is that the anti-pattern specification makes a stronger link to the correct pattern, as shown in Figure 3.

This new specification emphasises the correct pattern. The specification first explains the problem domain, and the correct pattern to apply. It then says which anti-pattern was applied, how it can be re-factored to solve the problems, and then provides a link once more to related correct patterns. The problem description required in our specification should allow the designer to make the mental connection between the previously captured pattern and this anti-pattern.

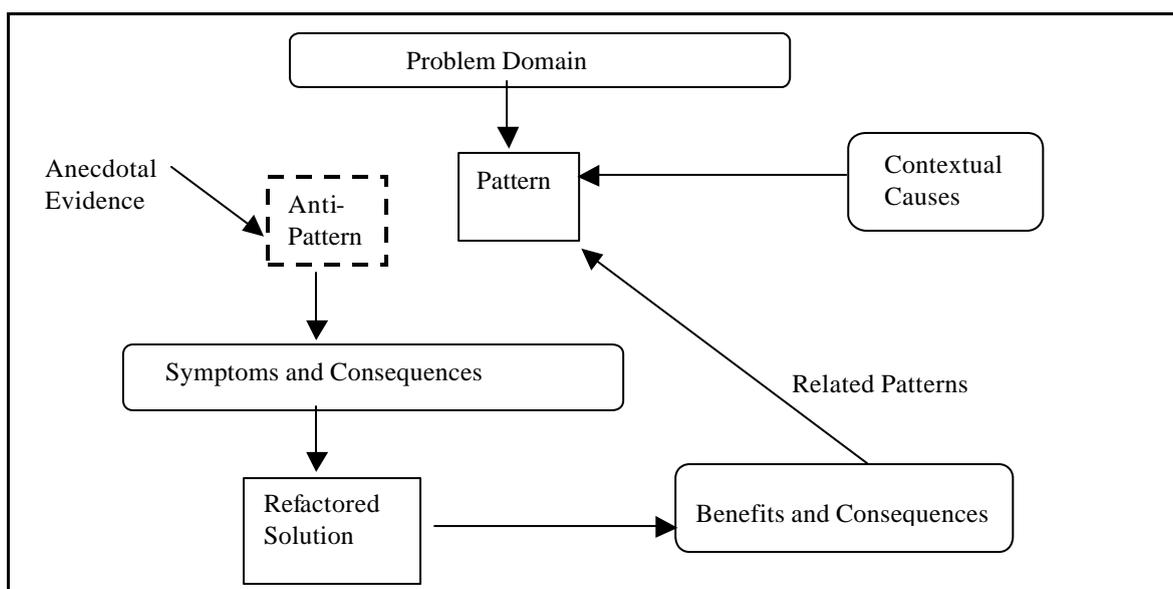


Figure 3: Improved Anti-Pattern Specification

5.1 A Better Anti-Pattern Specification

Consider the anti-pattern demonstrated by Figure 2. The user is not informed as to the password requirements and thus spends too much time defining a password and repeatedly being told that the password does not meet some pre-defined constraint. A pattern specification that captures good practice could possibly be the following:

5.1.1 *Pattern: Easy Web Password Choice*

Problem:

How to facilitate and ease the efficient choice of acceptable passwords for secure web sites.

Pattern:

- Put clear instructions right next to the password entry text that spell out exactly what the constraints are.
- Test the password at the client – using a scripting language such as Java Script, which speeds up the validation process and hence the entire process.
- Test for all constraints before reporting back all problems to the user, so that the user does not end up in a cycle of defining password, getting message, redefining to fix that error, getting new error message and so on.

Example:

Password definition for Web site enrolling process where particular constraints apply that might not be known by the user. These constraints are usually prescribed to ensure that more secure passwords are chosen.

Context:

A password chosen by the user to protect the security of a website.

Forces:

- Every system has different password requirements.
- Web users are seldom trained in the use of the web site.
- The user needs to know what is expected.
- Passwords are not echoed to the screen because of security constraints so the user cannot check that password requirements have been met by visual inspection.
- If the user's chosen password violates some constraint a message will have to be produced and the password re-defined.
- Web users do not like wasting time and may abandon sites that waste their time by making them re-define passwords

Resulting Context:

The user is more likely to define a new password correctly the first time, and if they don't they will have all the required information to hand to define it correctly the second time. Thus the frustration levels are limited and there is a greater likelihood that good passwords will be provided.

To augment this and capture the consequences of an anti-pattern, we include the following anti-pattern as depicted in Figure 2.

5.1.2 *Amelioration Anti-Pattern*

Problem:

How to facilitate and ease the efficient choice of acceptable passwords for secure web sites.

Pattern:

- Put clear instructions right next to the password entry text that spell out exactly what the constraints are.
- Test the password at the client – using a scripting language such as Java Script, which speeds up the validation process and hence the entire process.
- Test for all constraints before reporting back all problems to the user, so that the user does not end up in a cycle of defining password, getting message, redefining to fix that error, getting new error message and so on.

Anti-Pattern:

Write JavaScript code to catch each error in turn. Report each error as it is detected. Allow the user to try again.

Attraction:

Simplicity of password validation process because as an error is detected it is reported and error reporting is simple.

Symptoms and Consequences:

- Web logs demonstrate that users leave the site without completing the enrolling process perhaps as a consequence of becoming frustrated by the password choice process.
- Users, when observed, don't know what kind of password to choose.
- Users define unacceptable passwords which seems to occur because they struggle with the validation process

Refactored solution:

- Apply pattern solution as given above.
- If you have the email addresses of users who abandoned the site, email them advertising the site and asking them to revisit the page. An e-commerce site may need to offer some inducement to lure them back.
- Put a “Help” button on the page and elicit comments about the enrolment process.

Benefits and consequences:

- Well-informed users choose valid passwords.
- Having all the information assists users in choosing more secure passwords.
- Users experience less frustration which benefits their overall usage of the site.
- Fewer users leave the site in the enrolling process and go on to do shopping using the web site.

This example is a simple one, but serves to demonstrate that the anti-pattern can be specified in such a way that the pattern is inextricably bound up with it. This increases the likelihood that the designer will encode the pattern even when he or she has previously unwittingly applied an incorrect solution to the problem, which turns out to require the use of an amelioration anti-pattern.

6. CONCLUSIONS AND FUTURE WORK

Anti-patterns are studied in order to avoid pitfalls but can actually create pitfalls in knowledge transfer, if not applied appropriately. The idea that the use of anti-patterns in knowledge transfer may be a dangerous strategy if applied incorrectly has been noted before [Cockburn et al. 2004]. The contribution of this paper is to ask if this may be due to the way in which the user’s mental model represents negation and what the implications are for HCI anti-patterns. Anti-patterns based on the templates for patterns will not necessarily provide the same benefit because the cognitive processing of anti-patterns has to deal with negation. This important fact has been largely ignored, and should be taken into account in considering the efficacy of anti-patterns in knowledge transfer.

The current information overload necessitates strategies of knowledge management, and the elimination of redundant, confusing and incorrect information, as done in anti-pattern design, is a valid strategy. However, if the specific implementation of this strategy has unexpected side effects it becomes a knowledge transfer anti-pattern in which case the implementation has to be reconsidered.

Hence we make a strong recommendation that designers be exposed to patterns repeatedly until the pattern is well established. Only then should anti-patterns be used to strengthen the pattern in the designer’s mind. We have proposed a new way of specifying anti-patterns to emphasise and strengthen the links to the correct patterns, and also recommend that inexperienced designers preferably make use of patterns rather than anti-patterns until such time as they have gained sufficient experience to be immune to the possible negative effects of anti-patterns. What emerged from the case study requires a larger and more substantial study in order to confirm or deny our assertions about the value or detriment of stating knowledge in the negative, and the implications it has for anti-patterns in HCI. If these findings are substantiated in a more comprehensive study the implications for anti-pattern design in HCI as well as in SE will have to be considered.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers of this paper for their extremely helpful comments.

REFERENCES

- ALEXANDER, C. S., SILVERSTEIN, M. 1977. *A Pattern Language*. Oxford University Press.
- BERGIN, J. 2002 *Fourteen Pedagogical Patterns*, <http://csis.pace.edu/~bergin> (last accessed 22.05.04)
- BORCHERS J. 2000 Interaction Design Patterns: Twelve Theses, Position Paper, *Workshop: Pattern Languages for Interaction Design. CHI 2000*. 2-3 April. The Hague, Netherlands.
- BORCHERS J. O., THOMAS, J. C. 2001 Patterns: What’s in it for HCI? In *Panel Extended Abstracts of the Computer-Human Interaction Conference on Human Factors in Computing* New York
- BROWN, W. J. MALVEAU, R.C., MCCORMICK III, H W. MOWBRAY T. J, 1998, *Anti Patterns - Refactoring Software, Architectures and Projects in Crisis* 1998, Wiley, ISBN 0471197130
- CHANG, D., TUOVINEN, J.E. 2004. The Meeting of Gestalt and Cognitive Load Theories in Instructional Screen Design. *ICEIS 2004, 6th International Conference on Enterprise Information Systems, Proceedings, Volume 5, 53-58*
- COCKBURN, A., BARUZ, A., ENGELUND, A., HANES, P.B., BROWN, C., SISKI, C., OLSON, D. 2004. *Anti Pattern*. <http://c2.com/cgi/wiki?AntiPattern> (last accessed 20.05.04)
- COPLIEN, J. O. 2004 Organizational patterns: Beyond technology to people. *ICEIS 2004, 6th International Conference on Enterprise Information Systems, Proceedings, Volume 5, 1S-15*
- CRAIK, K. 1943 *The Nature of Explanation*. Cambridge: Cambridge University Press
- DEARDEN, A., 2000. Patterns and Participation: The relevance of Christopher Alexander’s ideas for HCI. *Proceedings of the BCS HCI Group/IFIP WG13.2 Workshop on HCI Patterns*. London, UK

- ERICKSON T., 2000 *Lingua Franca for Design: Sacred Places and Pattern Languages*. *Proceedings of DIS 2000* (Brooklyn, NY, August 17-19, 2000). New York: ACM Press, 2000, pp 357-368
- FINCHER, S., 2003. Perspectives on HCI Patterns: Concepts and tools (introducing PLML) *CHI2003 workshop report, Interfaces*, British HCI Group, ISSN 1351-119X, Human-Computer Interaction
- FINCHER, S., 2004. *HCI Pattern-Form Gallery*, <http://www.cs.ukc.ac.uk/people/staff/saf/patterns/gallery.html> (last accessed 17.05.04)
- FARIDUL, I., 2003 Investigating XML as Language for HCI Patterns Representation. *Master Thesis*, Concordia University
- GAMMA E., HELM R., JOHNSON R., VLISSIDES, J., 1995. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley
- GILBRETH, F. B., GILBRETH, L. 1917. *Applied Motion Study*, NY, Sturgis and Walton Co.
- HOWE, R., 2004. Proposed CTG *Anti-Pattern Template*, <http://home.earthlink.net/~ronhowe1/antiptn/antitmpl.htm> (last accessed 02.08.04)
- JÉZÉQUEL J., TRAIN M., MINGINS C. 1999. *Design Patterns and Contracts*. Addison-Wesley
- JOHNSON-LAIRD, P.N., SAVARY, F. 1996. Illusory inferences about probabilities. *Acta Psychologica*, 93, 69-90.
- JOHNSON-LAIRD, P., BYRNE, R. 2000. http://www.tcd.ie/Psychology/Ruth_Byrne/mental_models/theory.html
- JOHNSON-LAIRD, P.N., GIROTTO, V., LEGRENZI, P. 1998. *Mental models: a gentle guide for outsiders*. www.si.umich.edu/ICOS/gentleintro.html (last accessed 17.04.04)
- LONG, J. 2001. Software Reuse Antipatterns. *Software Engineering Notes* vol 26 no 4 ACM SIGSOFT
- MAHERNOFF M.J., JOHNSTON L.J. 1998. Principles for a usability-oriented pattern language. *Australasian Computer Human Interaction Conference*, Nov 29-Dec 12, Adelaide, pp132-139
- MARTIN, D., RODDEN, T., ROUNCFIELD, M., SOMMERVILLE I., VILLER S. 2001. Finding Patterns in the Fieldwork. In *Proceedings of ECSCW' 01*. 18 Sept. Bonn, Germany
- McCORMICK, H.W. 2004A. *What is an AntiPattern*. <http://www.antipatterns.com/briefing/sld003.htm> (last accessed 30.07.04)
- McCORMICK, H.W. 2004B. *AntiPattern Template*. <http://www.antipatterns.com/briefing/sld007.htm> (last accessed 30.07.04)
- SINNIG, D., GAFFAR, A., REICHART, D., FORBRIG, P., SEFFAH A. 2004. Patterns in Model-Based Engineering. *Computer-Aided Design of User Interfaces*. Island of Madeira, Portugal, January 13-16
- TAYLOR, F.W., 1911. *The Principles of Scientific Management* NY, Harper and Bros
- TIDWELL, J., 2004. *COMMON GROUND: A Pattern Language for Human-Computer Interface Design*. http://www.mit.edu/~jtiddwell/common_ground_onefile.html (last accessed 07.05.04)
- VAN WELIE, M. 2004. *Web Design Patterns*. <http://www.welie.com/patterns/> (last accessed 06.05.04)
- VAN WELIE, M., VAN DER VEER, G.C. 2003. Pattern Languages in Interaction Design: Structure and Organization, INTERACT 2003, Zürich, Switzerland, September 2003. M RAUTERBERG, M MENOZZI, J WESSON Eds. IOS Press
- WESSON, J., COWLEY, L. 2003. Designing with Patterns: Possibilities and Pitfalls. In *Proceedings of the 2nd Workshop on Software and Usability Cross-Pollination: The Role of Usability Patterns*, INTERACT 2003, Zürich, Switzerland, September 2003, M RAUTERBERG, M MENOZZI, J WESSON Eds. IOS Press
- ZAJICEK, M. 2004. A Methodology for Interface Design for Older Adults. *ICEIS 2004, In 6th International Conference on Enterprise Information Systems, Proceedings*, Volume 5, 81-88