(Don't) Join the Dark Side

An Initial Analysis and Classification of Regular, Anti-, and Dark Patterns

Alexander G. Mirnig and Manfred Tscheligi
Center for Human-Computer Interaction & Department of Computer Sciences
University of Salzburg, Salzburg, Austria
Email: firstname.lastname@sbg.ac.at

Abstract— **Patterns describe proven solutions to reoccurring problems. Anti-patterns describe solutions, which are proven not to work for solving reoccurring problems. Both concepts are well understood, documented, and employed in several different disciplines. Another, more obscure pattern concept, is that of "dark patterns". Dark patterns describe solutions used to trick users and are often considered to be anti-patterns. In this paper, we show that dark patterns have a different status and focus. Depending on the circumstances, a dark pattern can either be a regular pattern or an anti-pattern. Treating and documenting a dark pattern in the same way as a regular or anti-pattern could result in making malicious solutions easy to access and reproduce. We provide a review and delineation criteria for regular patterns, anti-patterns, and dark patterns in Human-Computer Interaction (HCI). This enables a more reflected knowledge transfer via patterns and protection of users from malicious designs.**

Keywords-basics on patterns; design patterns; anti-patterns; dark patterns.

I. INTRODUCTION

Design is usually not a blind, directionless activity, but happens with a certain focus. In HCI, 'design' is most often encountered in regards to user interface (UI) design. A good or well-designed UI should be readable and understandable for the intended user group, provide quick access to the most often used commands and functions, not obstruct the view onto other parts of the program that are not part of the UI, as well as satisfy the user depending on the specific application and context. As can be expected, there is no "one size fits all" solution to good UI design. General guidelines and knowledge on what constitutes sensible UI design do exist, but these require the hand of an experienced designer when it comes to covering specific cases and contexts, where tweaks and modifications in even the smallest details are often necessary – details, which general guidelines do not cover [1][2].

A pattern or design pattern is a documentation of a working solution to a particular (design) problem, embedded in its context and with concrete implementation examples. In contrast to patterns, an anti-pattern (sometimes also written 'antipattern') presents a solution that is proven not to work for solving a particular problem. A dark pattern describes a design solution intended to trick or otherwise deceive the user.

Both, regular and anti-patterns, are aimed at carefully documenting the solutions contained within them and are tied to approaches and structures that support this aim. With dark patterns, however, such an approach would arguably defeat the purpose behind naming and describing patterns, namely making the described solutions easy to access and reproduce. A specific level of information is certainly necessary, but if one wants to protect users from harmful designs, is providing step-by-step instructions on how to easily reproduce such designs really the right way or rather even counterproductive? This is the reason why a more detailed analysis of dark patterns and their relation to regular and anti-patterns is important.

In this paper, we provide such an analysis. We explore patterns, anti-patterns, and dark-patterns and the concepts behind them. We reflect on available literature in order to extract the basic characteristics of each type of pattern. We provide a minimal definition for each type of pattern and discuss these. As we will eventually discover, dark patterns carry their name only in the very loosest sense of the word, due to the lack of focus on reproducibility of the described solutions and other factors. In Section II, we provide a brief overview on related work to patterns, anti-patterns, and dark patterns. In Section III, we take a look at common definitions, structures, and examples for each pattern type, in order to extract the minimal requirements for a good or successful pattern of each of these types. The overall aim of the paper is to show the potential dangers of not clearly separating patterns, anti-patterns, and dark patterns and provide usable classification for all three pattern types to avoid these dangers. The paper concludes with a discussion of the results and future work in section IV, and an overall summary in Section V.

II. RELATED WORK

The term 'pattern' in this context was first coined by Christopher Alexander [3] to document techniques and solutions in architecture. His idea was to develop these small, standalone solutions with the eventual goal of making buildings by "stringing together patterns" [4]. Nowadays, the term generally refers to documented proven solutions to reoccurring problems in specific fields and contexts. Various pattern approaches exist and are applied in different disciplines, interaction design and software engineering being the most prominent among these [5][6].

 ISBN: 978-1-61208-534-0

In HCI, patterns have been adopted for capturing UI design solutions in several domains, such as web design [7], contextual User Experience design [8], or the automotive domain [9]. In this paper, we shall mostly focus on UI design patterns as they are used in the HCI community, in order to keep the analysis and discussion condensed, though the eventual results should be expandable into other domains.

Providing solutions to problems and giving guidance to novices and experts was traditionally done via guidelines. Using guidelines is subject to a number of problems [1][2]. They are often too simplistic or too abstract, they can be difficult to interpret by the designer, or they can even be conflicting with other guidelines, due to their general nature and the many different application contexts. Due to this same general nature, it can be difficult to identify which concrete problem(s) a guideline actually addresses. One particularity of patterns is that they are always focused on a certain problem. Where a guideline would give an overall answer to a question of the form "How do I do x?" a pattern would answer "How do I solve x?".

Patterns are less holistic but more specific, with a focus on providing a completely retraceable solution to a specific problem. According to Van Welie and van der Weer [6], this makes them even potentially better tools than guidelines. Patterns usually contain more specific knowledge than guidelines, but with a much narrower thematic focus. Depending on the abstraction level of a pattern [10], it can contain little to no guidance towards any greater overall task the problem might be a part of. Patterns can thus be seen as complementing guidelines and other means of general guidance. It is also possible for a pattern to contain information from several guidelines, but only the parts pertaining to a particular situation or problem [11].

Pattern creation or "mining", as it is often called (e.g., [5][17]), is a lengthy and structured process, requiring designers who were actually able to solve a certain problem to retrace their steps and carefully document how they arrived at their solution in several iterations. The goal is to fully document the solution finding and implementation process embedded in its context, so that the solution can be faithfully reapplied in a similar or even different context, if possible. Contemporary pattern approaches still follow Alexander's general intention of individual patterns working together as solution elements to larger problems. Patterns are, therefore, rarely standalone, but are collected in collections or repositories, which are either published as paper volumes or online.

Where patterns describe working solutions to reoccurring problems, anti-patterns do the opposite; they describe solutions to reoccurring problems that are proven to not work. The basic idea is the same as with regular patterns – carefully document the solution process as well as its embedded context. The overall goal, however, is to *avoid* the solution the anti pattern describes rather than its implementation. Appleton [12] describes anti-patterns as descriptions of lessons learned instead of the best practices described by regular patterns.

A third type of patterns, although not as well documented as the former two, is that of dark patterns. Brignull [13] defines dark patterns as descriptions of design solutions, which "*appear[s] to have been carefully crafted to trick users into doing things […] and they do not have the user's interests in mind.*" What might be desirable and good design in one instance could very well be a dark pattern in another – otherwise, e.g., spoofing would not work as well as it (sadly) often does.

So the distinction between regular and dark patterns is not as clear-cut, as it might seem at first glance. Similarly, it is sensible to expect a dark pattern solution to work at least moderately well for its envisioned purpose or it would not warrant the attention. In this case, however, it would be incorrect to label it an anti-pattern, as anti-patterns document solutions that do *not* work well in the first place. This somewhat muddy situation is reflected in the literature. To provide an example, in their 2014 DIS Paper, Greenberg et al. [15] define dark patterns as anti-patterns in a wider sense, whereas darkpatterns.org [14], a website dedicated to expose deception and malicious design practices, explicitly separates dark patterns from anti-patterns as their own pattern category.

III. ANALYSIS

In the following three sections, we provide common concepts, structure templates (where available), and examples of patterns (also referred to as 'regular patterns' in order to not confuse them with the latter two types), anti-patterns, and dark patterns. We then use these to derive commonalities for each pattern type. At the end of each subsection, we transform these commonalities into a brief list of minimal requirements for each pattern type. The analysis is a high-level one, with focus on common concepts. It is not intended to be an encompassing and detailed meta-analysis of all available pattern literature.

A. Regular Patterns

Since a pattern describes a proven solution to reoccurring problems, this means that each pattern starts from a problem, which requires a solution. The solution described in a pattern needs to be a reliable and proven one. If it worked only once, then it is not a good solution for a pattern. The general rule for what constitutes a solution as proven is commonly known as the *rule of three* [5]. If a solution has worked to solve the problem in at least three cases, then it is considered a working solution. This is not a hard rule, but it has generally been accepted in most pattern approaches.

As mentioned previously, one of the main ideas behind pattern approaches in general is to describe only that single solution instead of giving general guidance. At the beginning of the pattern mining process, the pattern writer retraces each step that leads to the eventual solution until s/he has a complete description of every single step, which led to the solution, including the exact context the solution was embedded in, as well as contextual forces and other variables. The term 'writer' might suggest only one individual, but it is not unusual for several individuals to be involved in a pattern mining and writing process.

In order to ensure a good end product of such an involved process, a successful pattern should usually satisfy a number

 ISBN: 978-1-61208-534-0

of requirements in order to be considered of sufficient quality. In a meta-study on pattern requirements and guidelines, Wurhofer et al. [8] defined the following requirements for patterns, based on the work of Niebuhr et al. [18], McGee [19], Khazanchi et al. [20], Borchers [10], and Dearden et al. [21]:

a) Findability: A pattern needs to be easily findable within a pattern collection or language. If it already requires considerable effort to find a pattern in the first place, then that defeats the aim of patterns to provide easier access to specific information.

b) Understandability: The described solution must be understood by its users. A solution, which is not understood, can hardly be implemented correctly (or at all).

c) Helpfulness: The described solution must be feasibly realizable within the reader's available resources. It must furthermore contain enough information, so that the reader can realize the solution in practice.

d) Empirical Verification: The pattern solution should be supported by empirical data. A solution supported by empirical data is of higher quality than one, which is based only on individual experiences and/or observations.

e) Overall acceptability: This is an additional criterion to capture the subjective component of whether or not a reader agrees with a pattern solution or not, regardless of the presence or absence of deficiencies in any of the other quality requirement categories.

To ensure that a pattern satisfies these and similar quality criteria, they are often written according to predefined structures or templates. Such templates contain fields for all the essential information for a successful pattern in a certain domain. Gamma et al. [5] proposed a detailed structure in their influential work about design patterns, which consists of 13 fields, tailored towards documenting object oriented software solutions. Tidwell [7] proposes a slightly simpler and more generally suited structure, which consists of the fields *Name*, *Examples*, *Context*, *Problem*, *Forces*, *Solution*, *Resulting Context*, and additional *Notes*.

In another pattern collection, Tidwell [22] even proposes a rather minimalistic pattern structure containing only the four categories *What*, *How*, *Why*, and *When*. This structure expresses the minimal requirements of a pattern, in that it needs to address a problem via its solution (the *What*), describe the solution and the steps that need to be taken (the *How*), a justification and explanation of why the solution works as it does (the *Why*), and an explanation of the context and conditions for successful reapplication (the *When*).

Mirnig et al. [11] propose a general pattern structure intended for use across disciplines. This pattern structure is very similar to Tidwell's and consists of only five mandatory elements: *Name*, *Problem Description*, *Context* and/or *Forces*, *Solution*, and *Examples*.

B. Minimal Regular Pattern Requirements

Based on these observations, we can conclude that a successful pattern should at least contain the following elements:

a) Means of reference: Name, Type, Keywords, and similar elements serve to distinguish a solution description from others, help build references between solutions, which are dependent on other solutions or problems, and aid in finding or re-finding the particular solution in a collection or database containing several patterns. Corresponds to the criterion of findability. At least one such means of finding and reference should be contained in every pattern.

b) Problem description: Patterns are not general guidance documents but always targeted at a specific problem. This problem must be described or explicitly mentioned at least briefly, to let the reader decide whether the pattern is of use in the particular case or not.

c) Context description: Since patterns provide solutions for very concrete problems, these problems need to be described in the context the solution occurred in. Depending on the context, some solutions are not feasible or have different effects than they would have in other contexts. Ideally, this context description includes a detailed listing of the forces influencing the solution, but not necessarily. The basic requirement is a description detailed enough to let the reader decide whether the solution can be applied in the particular context or not.

d) Solution description: The solution is arguably the most important part of a pattern. It must be described, not merely mentioned, ideally from the identification to the problem to the fully working implementation of the solution in a step-by-step manner.

e) At least one example: In order to satisfy the general requirement of giving practical guidance, the pattern should contain at least a description of one case of a successful solution implementation.

It should be noted that none of the examined templates and structures contained written documentation of the solution status as "proven" as a requirement. Corresponding to the criterion of empirical verification by Wurhofer et al. [8], the assumption is that a pattern ideally contains more than one example in order to show that it worked in more than one case. However, the rule of three or other potential standards in this regard are rarely explicitly mentioned or enforced in pattern templates or structures. For this reason, the status as proven or the number of successful solution applications is also not included in the list of minimal pattern requirements above.

C. Anti-Patterns

If patterns are the "Dos", then anti-patterns are the "Don'ts". Anti-patterns are documentations of bad or nonworking solutions to problems. Appleton [12] distinguishes between two kinds of anti-patterns:

Type 1: Those that describe a bad solution to a problem, which resulted in a bad situation.

Type 2: Those that describe how to get out of a bad situation and how to proceed from there to a good solution.

The second type of anti-pattern is also known as an "Amelioration Pattern" [17]. Type 2 or amelioration patterns skirt the boundaries between pattern types and are – depending on their level of detail – more of a combination of a type 1 anti-pattern (description of the bad solution) and a

 ISBN: 978-1-61208-534-0

corresponding regular pattern (description of the working solution). Anti-patterns are not as widely used as regular patterns, although they are arguably just as useful as regular patterns, in that they document solutions that, according to Coplien [24], might "look like a good idea, but which backfire badly when applied." Like regular patterns, the negative nature of the anti-pattern's solution might not be obvious, and the anti-pattern serves to make this fact explicit.

Despite this, anti-patterns are not always documented in the same level of detail as regular patterns are. For example, Github's list of anti-patterns [23] consists of only five elements, each one to two lines long, with only two of them containing actual reasons for why the solution is considered an anti-pattern.

The Portland Pattern Repository Wiki [17], on the other hand, provides a detailed template very similar to that of a regular pattern, outlining the components a well-written anti-pattern should feature. The structure proposed by this template is very similar to that of most regular pattern approaches. The main differences are references to other anti-patterns and positive patterns (in case it is a Type 2 or amelioration pattern), together with two context sections.

In this paper, we want to understand anti-patterns as more than a simple listing of things not to do, since a simple listing does not ensure understandability, non-reproducibility, verification, and other factors tied to the concepts the term 'patterns' carries. We shall call those, which satisfy these factors *genuine* anti-patterns, and those, which do not (i.e., simple listings or incomplete anti-patterns) *nongenuine* anti-patterns.

D. Minimal Anti-Pattern Requirements

Keeping in line with regular pattern requirements and quality criteria, the following would be sensible high-level expectations from any (genuine) anti-pattern: (1) ensure (non-)reproducibility of the solution; (2) foster understanding why the solution does not work as intended; (3) provide distinction between the desired and actual outcome; (4) make the description accessible to experts and novices. Taking these into consideration and by matching them to the discussed anti-pattern approaches, we can conclude that a successful anti-pattern should at least contain the following elements:

a) Means of reference: An anti-pattern needs to be easily found and be able to be referenced, so the same standards as for regular patterns apply.

b) Problem description: An anti-pattern provides a solution to a problem, just like a regular pattern does. Since the distinction lies in the (in-)appropriateness of the solution and since the reader needs to be able to decide whether the anti-pattern is relevant for him/her, the same standards as for regular patterns apply.

c) Context description: Just like in a regular pattern, whether or not a solution works or can be considered "good", depends on the application context and influencing factors. Therefore, the same standards as for regular patterns apply.

d) Solution description: Unlike solution descriptions in regular patterns, the focus in not on reproducibility of the described solution. However, anti-patterns can describe well-intentioned bad solutions, so the instructions should be detailed enough, so that the individual steps can be retraced. This way, it is easier to pinpoint where the solution went wrong (start, middle, end). Therefore, similar standards as for regular patterns apply here as well.

e) Result description: An anti-pattern describes a solution, which does not work well or which does not work as intended. In order to adequately do this, the pattern needs to contain a description of the result of applying the pattern solution in the particular context(s), in order to allow the reader to compare the desired with the actual result.

f) At least one example: Similar to regular patterns, the anti-pattern should contain at least one example case. In an anti-pattern, however, the focus is not on reproducing the solution. Therefore, the example should serve to justify the implicit or explicit assumption that the solution described by the anti-pattern leads to the described result.

E. Dark Patterns

A dark pattern describes a design solution, which "*appear[s] to have been carefully crafted to trick users into doing things … and they do not have the user's interests in mind.*" [13]. Unlike anti-patterns, this definition by Brignull et al. does not leave room for well-intentioned solutions, which did not work out as intended. The definition found on darkpatterns.org, an adaptation of the previous definition, makes this even more explicit: "*Dark Patterns ... are not mistakes, they are carefully crafted with a solid understanding of human psychology, and they do not have the user's interests in mind.*" [14]

Where a pattern describes a well-working solution and an anti-pattern describes one, which does not work well (or as well as it was intended to work), a dark pattern describes a solution, where the *intention* behind it is a negative one. Documenting a solution as a dark pattern is a way of exposing often well-hidden malicious practices (e.g., hidden costs in "free" services or disguised advertisements). There is no direct requirement of the solution having to work well (pattern) or not (anti-pattern). Greenberg et al. [15] and Zagal et al. [16] also highlight the intentionality of a dark pattern as the main distinguishing characteristic from an anti-pattern. Nonetheless, they combine both dark patterns and anti-patterns in a broader sense in their work.

However, it would be reasonable to expect a dark pattern to be more important or more dangerous, if the solution it describes worked well rather than the opposite. After all, if a design intended to trick the user does not work very well as per its intended use, then that solution is less dangerous than one, which works very well in tricking users.

So in a way, it would seem more sensible to consider dark patterns to be closer to regular patterns instead of anti-patterns. Taking the proposed minimal recommendations we found for regular patterns and applying them to dark patterns would also be misguided, however, as the focus of regular patterns lies in their reapplicability – the exact opposite of dark pattern solutions, which should *not* be reproduced [13]-[16].

As we can see, the distinguishing characteristic of a dark pattern is not the quality of its solution, and neither is it the

 ISBN: 978-1-61208-534-0

level of detail of its solution description. It is rather the *intention* behind the design solution and its status of undesirability, which makes a particular solution a dark pattern solution. Dark patterns are, much like shallow anti-patterns, often simply documented as brief statements of the solution implementation, followed by a list of examples. The focus is more on warning the user and exposing malpractices.

F. Minimal Dark Pattern Requirements

Considering that a dark pattern is not about reproducing a solution, but a statement as to how and why a particular solution is malicious and should be avoided, we arrive at the following minimal requirements to satisfy these aspects:

a) Means of reference: If a dark pattern should carry the name 'pattern' for a reason, then it should also satisfy the general pattern requirement of being easily referenceable, in order to build a pattern collection or language. Therefore, similar standards as for regular and anti-patterns apply.

b) Solution description: Just like a regular pattern or an anti-pattern, a dark pattern is about a particular solution implementation. This solution needs to be described in enough detail, so that the reader can recognize it.

c) Solution goal or intention: The focal points of dark pattern solutions are the malicious intentions behind the solution implementation. While the intention in regular or anti-patterns is, in most cases, simply the intention of solving the problem, malicious goals can be manifold and not always known to the reader (phishing, spoofing, credit card fraud, etc.). Therefore, the intention needs to be made explicit in dark patterns.

d) Undesirability statement: The fact that the solution with its respective goal is an undesirable one might not be obvious to every reader, depending on his or her background, experience or legal knowledge. A dark pattern should, therefore, contain a statement about the undesirability of the solution. This also clearly demarcates it as a dark pattern.

e) Undesirability justification: More important than the undesirability statement itself is an appropriate justification as to why the intention behind the described solution is undesired in a particular context (or all of them). This justification might often be obvious or already implicitly contained in the solution description, but it is nevertheless very important for three intuitively plausible reasons. First, a dark pattern should expose practices that skirt or cross legal and/or moral boundaries. They should not be based on one's subjective sensibilities regarding aesthetics or other nonrelevant factors. Second, moral codes are still subjective in a wider sense and not uniform across societies, so an intersubjectively traceable reference should be provided. Third, legal constraints are similarly not uniform across nations, so an adequate reference or justification should be provided.

f) At least one example: Similar in form to regular patterns and anti-patterns, examples have an entirely different function for dark patterns. They should warn users from interacting with the designs presented in the examples section. The focus should be on quantity over quality, since the designs need not be reproduced.

To sum up, a regular pattern provides the reader with clear reasoning and context as to why and how a certain solution solved a particular problem. In the same spirit, a dark pattern provides the reader with a clear reasoning and context as to why and how a certain solution is undesirable from a legal and/or moral standpoint.

TABLE I. MINIMAL REQUIREMENTS PER PATTERN TYPE

Requirement	Patterns	Anti-Patterns	Dark Patterns
Reference means	X	X	X
Problem	X	X	
Context	X	X	
Solution	X	X	X
Goal/Intention			X
Result		X	
Undesirability statement			X
Undesirability justification			X
Example(s)	X	X	X

G. Minimal Requirements - Summary

When comparing the minimal requirements for the three pattern types we can see that, the only requirements all three have in common are reference means, solution description, and examples. Problem statement and context description are only relevant for regular patterns and anti-patterns. Anti-patterns require an additional result description, in order to show how the solution does not work well or as well as intended. Dark Patterns, having a different focus, require an additional statement about the solution intention, the undesirability of it, and a justification for said undesirability. Since they do not focus on reproducibility of the solution, problem statement and context description are not required for dark patterns. An overview of the minimal requirements for each pattern Type is provided in Table 1.

IV. SUMMARY AND DISCUSSION

From this preceding analysis, we derive that there are two dimensions, which govern the separation between regular patterns, anti-patterns, and dark patterns. These two levels are **completeness** and **focus**.

The *completeness* of a pattern determines whether it is genuine or non-genuine. Since completeness means fulfilling the minimum requirements outlined above, it is reasonable to state that only genuine patterns could be considered good or high quality patterns. There is no guarantee, however, that a

genuine pattern is automatically of high quality, as its content may still be lacking. This depends on the pattern mining and writing processes and cannot be dictated by structural requirements alone.

The *focus* of a pattern finally decides whether the solution is a dark pattern solution or not. For the distinction between anti-patterns and regular patterns, the intentions behind the solution are irrelevant. Anti-patterns can be well intentioned with unintended side effects or misguided from the start, whereas regular patterns do not infer any legally or ethically relevant intentions beyond simply wanting to solve the particular problem. Thus, patterns and anti-patterns are focused on the solution and how well it works. We call these **solution-centered patterns**. Dark patterns are focused on the intentions behind a pattern solution. We call these **intention-centered patterns**. In their genuine form, patterns, anti-patterns and dark patterns are separate, non-overlapping categories. Only in their non-genuine form there is an (potential) overlap between anti-patterns and dark patterns. Regular patterns and anti-patterns share their status as solution-centered patterns. Only dark patterns are in the separate category of intention-centered patterns.

A. Intentions Matter

As we have learned, requirements for genuine dark patterns are different from both pattern and anti-pattern requirements. Furthermore, reproducibility is not a factor, and viability of the solution is a secondary rather than a primary factor. The solution might be easy or difficult to reproduce. It might be a solution that works well, moderately well, or not even all that well. But this does not really matter as to whether the solution description constitutes a dark pattern. What matters is the intention behind a problem solution. Consider phishing emails as an example case. There are more and less convincing phishing attempts – the more convincing ones are usually grammatically well written and spoof domain names, as well as corporate designs in some cases. Whether they are well done or not, the intention behind them is still a malicious one – be it obtaining personal information without a user's consent, stealing passwords, committing monetary fraud, or a combination of these.

The deciding factor in whether a solution is a dark pattern solution or not, is the intention with which it is implemented. This can mean that a dark pattern solution is newly developed for a certain nefarious purpose, or that a well-working and proven solution is appropriated and reused with malicious intent. This also serves as another clear delineation criterion from anti-patterns, as it might well be that an anti-pattern solution might lead to private date being made public with all its negative consequences (identity theft, credit card fraud, public shaming, etc.). If the intention behind the solution was, however, a positive one and the solution simply misguided for whatever reason, then the pattern is clearly an anti-pattern and not a dark pattern.

B. What is a Pattern?

This brings us to the issue of whether a dark pattern justifiably carries the term 'pattern' in its name at all. Describing a dark pattern solution at the same level of detail as a regular pattern or anti-pattern can lead to the opposite of what a dark pattern should do. A dark pattern should warn both users and designers from malicious solutions. They should not encourage such designs. If a dark pattern describes the malicious solution in great detail, however, then it does just that by making it more accessible and easier to (re-)implement. In order to be protected from a dark pattern solution, one needs to know what it looks like, what the intentions behind it are, and where it is or can be encountered. Knowing how to reproduce the malicious solution is hardly relevant at all in this context.

The only time in which it seems appropriate to conflate dark patterns and anti-patterns is when we talk about non-genuine patterns. Non-genuine patterns are patterns only in a wider sense, as they are problem solution descriptions of some sort, but without the level of detail, reproducibility focus and accessibility of genuine patterns. So if it is only appropriate to conflate dark patterns with other pattern types when they are incomplete, which essentially lowers their quality potential, there is little reason for dark patterns carrying 'pattern' in their name. However, it seems inappropriate to police the use of the term 'dark pattern' too strictly, as it is already widely used and usually understood in a somewhat consistent way. But we want to stress that dark patterns should not be considered patterns in the same way that regular patterns and anti-patterns are. The focus and purpose of dark patterns are decidedly different, and the 'pattern' in 'dark patterns' should be used with care.

C. A View Ahead and the Dangers of Knowing Too Much

Informing user-centered design solutions and protecting the user from malicious intentions is often a difficult balancing act. Knowledge transfer is important, as is design, which caters to individual user needs and requirements. Well-documented and well-working solutions – especially those focused on trustworthiness, acceptance, and similar factors – are in constant danger of being "hijacked" by those with sinister intents. We cannot realistically expect to come up with user-centered designs, which are completely safe from being used with malicious intent. Neither can we expect phishing, scamming, spoofing, and other forms of cyber crime to disappear anytime soon.

What we can do, then, is to be more careful when collecting, summarizing, and editing information. This keeps the knowledge transfer more focused by including necessary and omitting unnecessary information. Treating regular patterns, anti-patterns, and dark patterns as complex concepts with concrete purposes and requirements lessens the danger of a dark pattern containing instructions on how to easily reproduce its solution. Similarly, it lessens the chance of a regular pattern solution being used with malicious intent without anybody noticing. While it is probably true for dark pattern solutions that knowing too much can be bad, the opposite can be said to be true for knowledge *about* dark patterns. By knowing exactly what the purpose and requirements of a particular pattern type are, the patterns themselves can be more easily molded to fit that, and only that, particular purpose. This, in turn, raises their effectiveness, while at the same time reducing the potential

 ISBN: 978-1-61208-534-0

of misuse, misdocumentation, or over documentation. Thus, preserving knowledge and protecting the user need not always be at odds.

V. CONCLUSION

Dark patterns are different from both regular patterns and anti-patterns due to their focus. Dark patterns are not patterns in the sense that they describe solutions embedded in their context with a focus on (non-)reproducibility, but serve more as warnings. They, therefore carry the name 'patterns' only in a very loose sense of the word. In order to satisfy quality requirements, which are often associated with patterns in the tradition of Alexander [3][4], Gamma et al. [5], and others, we provided a minimal definition for genuine dark patterns, thus bridging the gap between dark patterns and other pattern types as much as possible. A fundamental difference in focus and requirements between the pattern types still remains and the 'pattern' in 'dark patterns' should be used with care.

Future work will focus on refining dark pattern structures and centralization of information collection about malicious practices for use both within and outside of HCI. The definitions provided in this paper should serve to structure pattern approaches within and outside of HCI, especially regarding the sometimes neglected concepts of anti-patterns and dark patterns, as well as inspire more careful and focused handling of user-centered design knowledge.

ACKNOWLEDGMENT

The financial support by the Austrian Science Fund (FWF): I 2126-N15 is gratefully acknowledged.

REFERENCES

[1] A. Dix, G. Abowd, R. Beale, and J. Finlay, "Human-Computer Interaction," Prentice Hall, Europe, 1998.

[2] M. J. Mahemoff and L. J. Johnston, "Principles for a Usability-Oriented Pattern Language," In Proc. Australian Computer Human Interaction Conference OZCHI'98, IEEE Computer Society, 1998, pp. 132–139.

[3] C. Alexander, "A Pattern Language: Towns, Buildings, Construction," Oxford University Press, New York, USA, 1997.

[4] C. Alexander, "The Timeless Way of Building," Oxford University Press, New York, USA, 1979.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software." Pearson, 1994.

[6] M. Van Velie and G. C. van der Veer, "Pattern Languages in Interaction Design: Structure and Organisation," In Proc. Ninth Int. Conf. on Human-Computer Interaction, IOS Press, 2003, pp. 527-534.

[7] J. Tidwell, "Common Ground: A Pattern Language for Human-Computer Interface Design," http://www.mit.edu/~jtidwell/interaction_patterns.html, retrieved: January 2017.

[8] D. Wurhofer, M. Obrist, E. Beck, and M. Tscheligi, "A Quality Criteria Framework for Pattern Validation," International Journal on Advances in Software 3, no. 1&2, IARIA, 2010, pp. 252-264.

[9] T. Kaiser, A. G. Mirnig, N. Perterer, A. Meschtscherjakov, and M. Tscheligi, "Car User Experience Patterns: A Pattern Collection in Progress," In Proc. Eighth International Conference on Pervasive Patterns and Applications (PATTERNS 2016), IARIA, 2006, pp. 9-16.

[10] J. Borchers, "A Pattern Approach to Interaction Design," AI & Society 12, Springer, 2001, pp. 359-376.

[11] A. G. Mirnig et al., "User experience patterns from scientific and industry knowledge: An inclusive pattern approach, International Journal On Advances in Life Sciences 7, no. 3&4, IARIA, 2015, pp. 200-215.

[12] B. Appleton, "Patterns and Software: Essential Concepts and Terminology," http://www.bradapp.com/docs/patterns-intro.html, retrieved: January 2017.

[13] H. Brignull, "Dark Patterns: Deception vs. Honesty in UI Design," http://alistapart.com/article/dark-patterns-deception-vs.-honesty-in-ui-design, 2011, retrieved: January 2017.

[14] H. Brignull, M. Miquel, and J. Rosenberg, Dark Patterns Library. http://darkpatterns.org, retrieved: January 2017.

[15] S. Greenberg, S. Boring, J. Vermeulen, and J. Dostal, "Dark Patterns in Proxemic Interactions: A Critical Perspective," In Proc. 2014 conference on Designing interactive systems (DIS '14), ACM (2014), pp. 523-532.

[16] J. Zagal, S. Bjork, and C. Lewis, "Dark Patterns in the Design of Games," In Proc. Foundation of Digital Games, 2013. http://www.fdg2013.org/program/papers.html, retrieved: January 2017.

[17] The Portland Pattern Repository Wiki. http://c2.com/cgi/wiki, retrieved: January 2017.

[18] S. Niebuhr, K. Kohler, and C. Graf, "Engaging patterns: Challenges and means shown by an example," Engineering Interactive Systems, Springer, 2008, pp. 586–600.

[19] K. McGee, "Patterns and Computer Game Design Innovation," In Proc. 4th Australasian conference on Interactive entertainment, RMIT University, 2007, pp. 1–8.

[20] D. Khazanchi, J. Murphy, and S. Petter, "Guidelines for Evaluating Patterns in the IS Domain," In Proc. MWAIS, AISeL, 2008, paper 24.

[21] A. Dearden and J. Finlay, "Pattern Languages in HCI: A Critical Review," Human-Computer Interaction 1, Lawrence Erlbaum Associates Inc., 2006, pp. 49–102.

[22] J. Tidwell, "Designing Interfaces," 2nd Edition, O'Reilly, Sebastopol, CA, USA, 2011.

[23] Anti-patterns on Github. https://github.com/angular/angular.js/wiki/Anti-Patterns, retrieved: January 2017.

[24] J. O. Coplien, "Software Patterns," SIGS Books, New York, NY, USA, 1996.

[25] W. J. Brown, R. C. Malveau, H. W. McCormick, and T. J. Mowbray, "AntiPatterns," 1998, Wiley.

 ISBN: 978-1-61208-534-0