

Pattern Smells, Get Rid of Them

Rebecca Wirfs-Brock¹[0009-0006-7215-3788] and Christian Kohls²[0000-0001-5858-7068]

¹ Wirfs-Brock Associates, Sherwood, OR 97140, USA
rebecca@wirfs-brock.com

² TH Köln - University of Applied Sciences, Köln, Germany
christian.kohls@th-koeln.de

Abstract. This paper introduces the concept of pattern smells. Like code smells, pattern smells tell you that something is wrong that could lead to deeper problems. For each pattern smell a description is given, common indicators are named, and heuristics to fix the smells are proposed. Smells can be helpful for identifying flaws in patterns, pattern descriptions, and pattern collections or languages. Knowing about specific smells, authors can work to remove them and improve their patterns. Equally important, named pattern smells introduce a vocabulary that the patterns community can use to discuss patterns and provide clearer feedback on work-in-progress.

Keywords: pattern smells, patterns, pattern languages, pattern descriptions, pattern theory

1 Introduction

The term “pattern smells” relates to the concept of code smells [1, 2]. A code smell is something in the code that feels odd—maybe even rotten. Code smells are remedied by reworking the code, improving its structure, and removing dependencies.

Smells indicate that something might be off—wrong or even dangerous. We can find pattern smells just as we can find code smells. Unlike code smells, however, pattern smells are not limited to the software domain. And while many code smells are easily fixed, sometimes remedying pattern smells requires significant effort.

There are many patterns published, many patterns applied, and many patterns in use. In our paper, “Patterns and their Umwelt” [3], we pointed out that one quality many patterns and pattern descriptions lack is enough consideration for the author’s, the reader’s, the designer’s and ultimately the user’s umwelt—that is, the unique environment and experience that is accessible to each individual. On this journey of understanding how umwelt affects patterns, we started to list common defects that we have seen in patterns and pattern descriptions. We reflected on why some patterns work while others fail (or at least are not used). The purpose of our analysis was not to show what is wrong with patterns. Quite the opposite! We want to see better patterns that can be used to help make the world a better place.

Sometimes, we feel that something is wrong or odd about a pattern but find it hard to express what bothers us. Descriptions of pattern smells make it easier to detect and identify typical shortcomings. By giving each pattern smell a name, it becomes easier

to talk about them. Knowing about smells helps authors to critically reflect on their writings, and everyone who provides them feedback can know what to look for to improve, and use a common vocabulary to describe what they find.

But identifying pattern smells is only half of the story. We also want to provide suggestions for how to resolve them. Wait, does this sound like patterns for remedying pattern smells? In a way, yes. However, we don't find it useful to have a full discussion of context and forces for each smell. Some are simple to understand and remedy (and don't need lots of explanation). Others are more nuanced. And the best way to resolve a smell is not always clear.

So we only provide suggestions, potential paths to resolve the smells—we think that this advice is in the form of heuristics rather than patterns.

In addition to the smells we identified in our previous paper, we also ran a focus group at PLoP 2025. In this focus group we collected typical smells that other pattern authors and experienced designers were aware of. This paper elaborates on the discussions of the smells we identified in that focus group as well as additional smells we identified as we wrote about pattern smells.

We think that there are different types of smells, and each comes into play at different times during understanding, analyzing, writing, reviewing, publishing and using patterns.

- Misconception smells: Lack of understanding about what patterns are.
- Domain smells: The author has only superficial understanding of the domain, making it hard to actually identify and organize distinct solutions. Sometimes the proposed solution is fishy.
- Analysis smells: The solution is okay but there is a lack of reflection and incomplete understanding of other perspectives.
- Description smells: Even when you get the solution, context, problem and forces right, you may still describe them poorly.
- Utopic world smells: Patterns that promise too much and place their solutions in a world that does not exist.
- Pattern language smells: These smells are about the interconnection of patterns, their granularity, their level of abstraction, and gaps or lack of wholeness. (We also suspect that pattern collections come with their own smells.)
- Wholeness smells: The solution or the description (or both) lack wholeness.
- Writers' Workshop smells: Feedback smells that fail to make the most of a Writers' Workshop.
- Shepherding smells: Feedback smells (and responses) that limit the potential of the shepherding process.
- Publication smells: Smells that do harm to the successful distribution of patterns and to quality standards.
- Usage smells: Smells that make a solution bad.

We draw on our own experiences of over 100 shepherding partnerships and participation in more than 300 writers' workshop sessions both at *PLoPs and in our work contexts. We integrate the knowledge about how to write good patterns (pattern languages, intro texts), book introductions, and the pattern language of shepherding [4].

We also take design knowledge and philosophy into account as well as the writings of Christopher Alexander.

The structure for describing each smell is as follows:

- Name: A name for the smell.
- Essence: The essence of the smell.
- Discussion: A brief discussion of how it makes the pattern odd.
- Indicators: Clues that make you feel that something is off about a pattern in this way.
- Fix: Heuristics for how a smell might be addressed. What works best for you and the specific smell can only be evaluated when you are faced with addressing your specific situation. Some smells can be easily fixed. Others require more effort. And some might be unfixable (for example, if the pattern description sells a bad solution).

There is one limitation to this paper that we are quite aware of. While we draw from many examples and experiences of our own, we rarely reference specific patterns. There are two reasons for this. First, if we were to start analyzing just a few patterns in detail, this paper would explode in length. Indeed, detecting and discussing the smells for a single pattern could be a paper on its own—for each pattern! Second, we don't want to point fingers at weak patterns. We respect the efforts that have been invested into creating each pattern description. And while many could be improved (nothing is ever perfect), they still provide useful guidance and invite readers to reflect. The aim of this paper is to provide tools and terminology that can be used to reflect on your own patterns and to give better feedback on other patterns. Hence, all examples are short, sometimes fictional, illustrations.

Each smell is labeled with a difficulty to fix (easy/possible/takes effort/hard to fix) and how urgent that fix is (required/urgent/recommended/optional). We do not justify these recommendations. They are based on our experiences; you may come to other conclusions.

2 How Smells Relate

Smells come at different times in the pattern mining, writing, feedback, and publication process. There are also smells that arise when applying the patterns but these are largely outside the pattern author's control (unless their poor pattern descriptions have caused them to be misapplied).

Some are easy to fix and some are really hard to fix. Also, their impact can be different; some smells are irritating, others can do serious harm.

The diagram (see Fig. 1) provides an overview of how the smells relate to the different phases of pattern development and usage. Some smells can occur in several phases. Pattern mining requires both a theoretical understanding of the pattern approach and domain knowledge. After mining the patterns, authors write pattern descriptions and may eventually publish them. The published patterns can be used and applied by others. Feedback can be given during the writing process as well as on the actual usage of the patterns. This is not a linear process. As we mine for patterns, we gain a deeper understanding of the domain and of patterns. Likewise, mining and writing about patterns

interact. And even publication is not the end of the story. Authors can continue to refine their patterns or add more patterns to a language. Ideally, one only moves forward to another phase when most of the smells of a previous phase are resolved. For example, if there are still misconceptions about patterns, these should be addressed before continuing to mine more patterns.

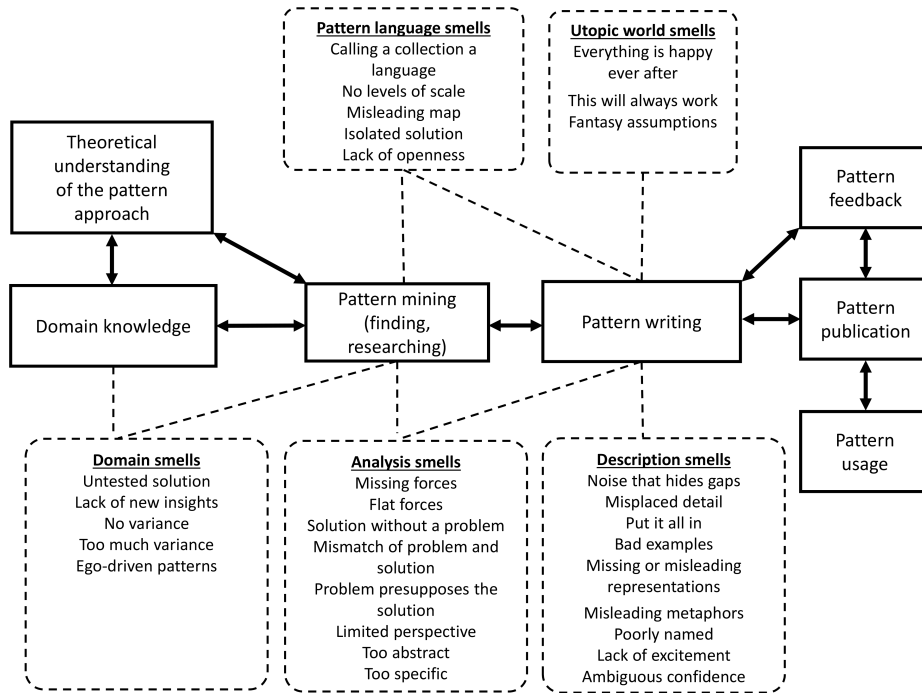


Fig. 1. Relation of pattern smells to phases. Each phase is in a box. The different smells are in dotted boxes. Smells are linked by dotted lines to the phases. Some smells are typically found in, but not limited to, specific phases. More smells will be added in future work. Each phase is connected by a double arrow to indicate that this is not a linear process.

Misconception smells are rooted in the lack of understanding of what patterns are. They will lead to problems in all phases, and it is hard to give any advice other than to learn more about patterns or get help from a mentor or experienced pattern writer.

Wholeness smells are about the original deeper quality of the solutions, for example dead solutions or a missing quality.

Domain smells indicate a lack of good solutions or the understanding of them. There can be a deficiency in the solution itself (a poor or weak solution) or a shortcoming of the pattern description (for example, the description fails to express when the solution works or doesn't work, or it does not provide enough examples as empirical support). It can sometimes be hard to tell whether the solution is weak or simply appears to be weak due to poor descriptions.

Analysis smells can still occur even if the authors have identified a good solution. With analysis smells, we can see that the solution works (or at least get a glimpse of it) but something is missing in or obfuscating the solution. You can see the solution but some forces or other aspects of a pattern analysis are lacking. Sometimes a thorough analysis isn't presented because the authors know the solution too well. And because they do, they leave out some information that is obvious to them, which leads to poor descriptions.

Description smells can be based on a lack of analytical information or poor writing. You cannot describe what you have not completely analyzed. However, even with a careful analysis of the solution and a deep understanding of the domain, one can still produce poor pattern descriptions that lack conciseness, clarity, precision, or excitement. Both analysis and description smells are fixable. The solution is there, the pattern is there. You just need to better analyze, understand, and describe it.

The **utopic world smells** occur when authors are overconfident or they promise too much with their patterns. They may misidentify something as a potent solution because they are unaware of negative consequences, pitfalls, and real-world limitations. This can be due to a lack of understanding of what patterns are (real-world stuff with inner beauty), missing domain knowledge (what might be barriers or showstoppers), insufficient analysis, or sloppy writing. While patterns and pattern descriptions should be exciting, they should also be realistic.

Pattern language smells apply to the scope and structure of a language, such as gaps, inappropriate granularity, duplicate or wrong relations.

Writers' Workshop smells and **shepherding smells** occur when feedback is given for the written patterns. The smells are often caused by other people, not the authors.

Publication smells are questionable practices after a version of a pattern description is written and ready to be shared, such as not soliciting feedback, or ignoring it.

Usage smells are the results of misapplying published patterns. While fixing them is outside the authors' control, newly applied patterns change designers' understanding of the domain. Hence, pattern authors may consider creating future pattern versions. These revisions can account for misuses and clarify some parts of pattern descriptions, such as appropriate contexts.

In this paper we will only describe in detail those smells related to pattern mining and writing. These include domain smells, analysis smells, description smells, utopic world smells and pattern language smells.

The following sections present smells for each of the categories. Some smells might cause other smells. And some smells belong to multiple categories. Although there seems to be a somewhat linear progression of smells that occur throughout the pattern mining and discovery process, we find the interplay of smells to be rather complex. Smells influence each other.

3 Domain Smells

You get the impression that the authors are not experts in the domain and have not talked to domain experts either. You read the solution but you don't buy it. Or can't believe it. Or you see some flaws.

3.1 Untested solution (Hard to fix by testing, easy to fix by labeling, urgent to fix)

Essence: The pattern description proposes a solution that has never been tested.

Discussion: Patterns can and should be tested [5, 6]. The more often a pattern is a success, the more trust we can put in it. Some proposed patterns are not rooted in experience. This is okay in principle, as Alexander himself states that patterns can be deduced. For example, you can analyze bad solutions and come up with a better proposal. But the term "proposal" means that this idea is untested. Good patterns require real-world examples. However, since patterns are abstract entities that can only manifest in real instances, they are *also* theoretical constructs. Writing a pattern description is a creative and original act.

Indicators: No real-world examples. Lack of known uses. The pattern description contains proposals and promises rather than evidence. Another indicator is missing information about the mining methods (how you found the patterns) and the mining ground (where you found the solutions).

Fix: Report on tested solutions. Cite several known uses. Describe the mining ground and mining methods (i.e., research methods). How did you look for the patterns? Where? What was your sample size? Is the pattern based on your own experience only? What qualifies you to see a pattern (what experience in seeing do you have)?

Label your patterns: If parts of the pattern are speculative or the whole solution has never been implemented, make it very clear that you used the pattern approach as an analysis framework to propose a new solution. Do not call it a tested pattern. Instead, it might be labeled as a proto-pattern.

3.2 Lack of new insights (Takes effort to fix, recommended to fix)

Essence: The identified solutions are not wrong but they don't offer any new insights into the domain.

Discussion: Common knowledge about things can be written as patterns, or be part of a pattern description, but what is the point? What is relevant depends on the situation of the designer. Tables are certainly relevant in the world. But is the description of a physical Table as a pattern relevant? Not for most people who want to use a table in the design of their living rooms. But if you are writing the pattern Table for carpenters, a description of materials, good table measures, and principles of stability could be interesting. Understanding the environment of your target group is an important consideration when you write a pattern.

Sometimes, a pattern description is a refinement of an existing pattern. It can integrate additional and novel challenges that the more general pattern did not address. If this is the case, then you have a compelling reason to write a version of that pattern specifically for that new domain.¹

Indicators: You read the pattern and you wonder “so what?” You heard it all before.

Fix: Get rid of the pattern if there isn’t anything new. Otherwise, indicate what is new. You can add new insights by analyzing the problem and its context deeper. If your pattern is a refinement, then craft your pattern description to include both the unique challenges and how your specialized solution addresses them.

3.3 No variance (Possible to fix, optional to fix)

Essence: Without variance there is no generative design space.

Discussion: A pattern that shows no variance is less useful. By definition, recurrent structures are also patterns—and that includes structures that repeat themselves without variation (such as fabric patterns or wallpaper). But such patterns are not that interesting to designers. A designer needs to be able to design; that is, she needs to be able to shape the final form. A more interesting design pattern is one that lets you create the same thing a million times over without ever doing it the same way twice. You don’t want to have the form of a single house; you want to know how to generate the forms of many different houses.

This smell can occur when the authors do not have enough expertise in the field. Their experience is based on too few examples. To see and understand a pattern one needs to understand both its essence and its variations. Sometimes variants and options are described, but not labeled as such. When they are not labeled, everything seems essential to the solution. Readers should know what is optional or what alternatives exist.

Indicators: Solution appears overly prescriptive, e.g., do all these specific things; or, the solution contains conditional or alternative steps.

Fix: Show variance and describe ways of making distinct adaptations. Separate variations from the basic solution and label them explicitly.

3.4 Too much variance (Possible to fix, recommended to fix)

Essence: With too much variance, the essence of form—the gestalt—is lost.

Discussion: If there are too many differences between the examples a pattern builds on, then there is no common form. Often the solution description is a mix-up of different

¹ Rebecca wrote patterns with Joseph Yoder and other colleagues specifically for Adaptive Object-Model Architectures. Several of the patterns they wrote were variants of patterns found in *Design Patterns* [7], describing solutions tailored to adaptive systems. One such pattern was Adaptive Object-Model Builder [8], a variant of the Builder Pattern.

forms sharing some properties. A red apple and a red car both share REDNESS. But are they really instances of a REDNESS pattern? Such confusion often occurs when two solutions share several properties (being RED, LARGE, FAST, ...) and especially when they share the same effect or goal.

Two different pathways can lead to the same destination. That does not make those two ways the same pattern. Each of the two paths has its own environments, requirements, and consequences. However, if the designer follows the same trail and makes some detours of her own volition, then she is following the same pattern, with some variant steps. Being open for such changes makes the pattern more alive, not less.

Indicators: Form appears to be indistinct or seems to be a mix of disjoint things.

Fix: Divide and conquer. One way to do this is to try to identify distinct problems and not focus on any commonality of their solutions. The result may be several patterns.

3.5 Ego-driven patterns (Hard to fix, urgent to fix)

Essence: The pattern sells ideas of the authors rather than ideas seen in the field.

Discussion: The main purpose of an ego-driven pattern is to celebrate the works or achievements of the author. Ego-driven patterns convey self-importance of the author and have the primary goal of shining a bright light on the author's experience. Paying attention to the needs of their readers isn't a primary concern.

Indicators: Defensive wording; unfounded claims to uniqueness or innovation; obscure jargon or text that is "obvious" to the author but not known to others in the same field; advertising one's own views and values; putting your very individual views into the "objective" pattern description. No revisions (because: I—and only I—know what is perfect. So, no need for revisions).

Fix: Rework ego-driven patterns to make them comprehensible to and enjoyable for their intended audience. Understanding and appreciating that what the intended readers need differs from the current pattern descriptions is an important first step. Then the ego has to be diminished and focus turned to effective communication. Receiving constructive advice and gratefully accepting it can be difficult and emotional for the author, especially if they feel attacked or threatened by critique. There aren't quick, easy fixes for revising ego-driven patterns. They often need extensive revision and a willingness by the author to step back and accept constructive criticism and to make changes.

A pattern is not a speaker's corner.

4 Analysis Smells

You see a good solution but some details and answers are missing.

4.1 Missing forces (Possible to fix, recommended to fix)

Essence: Important forces, the main drivers of the solution, are missing.

Discussion: A pattern may list several forces, but miss one or more important forces that compel you toward the solution. These might be forces from a perspective that you have not considered (for example, security is a commonly overlooked force in software patterns). Consequently, since your forces have critical gaps, your reader may not be convinced that your solution is appropriate, let alone a good one.

Indicators: Hard to justify solutions; to others the solution may appear arcane, overly complex, or lacking something.

Fix: Examine the solution for fitness to the problem as it is currently stated. Play devil's advocate. Ask: What might drive you to consider other solutions? Why is it that your proposed solution is more appropriate? Under what conditions is it better than the alternatives? Or worse? This analysis can reveal missing forces that will either lead you to favor your solution, or to add discussion of trade-offs that were made when coming to the solution.

4.2 Flat forces (Possible to fix, recommended to fix)

Essence: Forces are listed without explaining their priorities, weights, conflicts, and relations.

Discussion: Often forces are presented as if they were equally important, independent, and in simple opposition. As a result, the “forces field” becomes flat and undifferentiated: the reader cannot tell which tensions are critical, which are secondary, or how trade-offs are actually made. Patterns are not about two forces in neat conflict, but about *many interacting forces with different strengths* and several conflicts going on at the same time. When the description treats them merely as a list, or relies on simple text that fails to emphasize importance, the reader cannot grasp why the solution is structured the way it is. It is not shown how the forces shape the solution. The pattern may be correct, but its rationale remains opaque.

Indicators: Forces are presented as a flat list with no indication of relative importance or strength. Obvious or trivial forces are explained at length, while subtle or critical ones are mentioned only briefly. The text does not show how one force affects or constrains another; interactions between forces are missing. The problem appears to be framed as “two forces in conflict,” even though the situation is clearly more complex. This smell is often accompanied by either over-explanation of the obvious or under-explanation of what is essential. Important insights are buried in uniform prose, while minor or self-evident forces are given equal weight. The reader is left without a sense of priority, direction, or structure of the problem space.

Fix: Add a discussion and make explicit which forces dominate, which are secondary, and which are merely background conditions. Describe how forces influence, amplify, or constrain one another rather than presenting them as isolated items.

Use structure and rhetoric to highlight the essential: start with the most important forces, use formatting or language (“crucial”, “the key tension is...”) to guide the reader.

Smaller or more technical forces may be better discussed in the solution section rather than competing for attention in the description of the forces.

Re-check the forces field. Can the reader see why this solution—and not another—emerges?

4.3 Solution without a problem (Possible to fix, required to fix)

Essence: It is unclear why the solution is needed.

Discussion: Very often we perceive solutions at work and want to share them as patterns. But it is easier to see a solution than it is to understand the underlying problem. After all, the solution makes the problem disappear. This sometimes leads to patterns that tell you *how* to create something but do not tell you *why* to create something. The problem of implementing something (the *building* process) is different from the problem that led to needing that solution in the first place. Not only should patterns solve problems; they should also raise awareness of problems.

Indicators: “How to...” problem descriptions often point to the problem of finding a good solution structure and then implementing that structure. Such problem statements, however, do not tell you why a solution is needed or what is bad in the world without the solution. Missing contexts, backgrounds, and forces are typical for this smell.

Fix: Think about what would happen if the solution did not exist. What would be bad? That is the problem. Sometimes the core problem is already discussed in the forces section. Each force adds to the problem, making it difficult to find a solution. Some forces, or conflict between forces, are the original problem. There are some forces that drive you to do something in the first place and other forces that shape how you proceed. Make sure that the problem fits the solution.

4.4 Mismatch of problem and solution (Possible to fix, required to fix)

Essence: There is a problem, there is a solution, but they do not fit together.

Discussion: The idea of fitness between two entities is illuminated in Alexander’s early work *Notes on the Synthesis of Form* [9]: “It is based on the idea that every design problem begins with an effort to achieve fitness between two entities; the form in question and its context. The form is the solution to the problem; the context defines the problem.” [9, p. 15]

Each step in a design process is a decision that can fit or misfit. Along a path each step is a decision towards a direction. Each step has to fit into its current environment. A misfit indicates that not all of the forces are in balance. A solution does not fit the problem if its form, or parts of it, are not contributing to the resolution of the problem as a whole. There is some stuff in the solution that does not address the problem at all; and some aspects of the problem remain unaddressed.

Indicators: There is more in the solution than the problem asks for. The proposed solution does something, but it does not solve the stated problem.

Fix: As you observe recurring solutions and distill them into patterns, this smell is best fixed by not touching the solution but by reflecting again about what problem the solution actually solves. If you got the problem right but you don't know how to solve it, then it is not a pattern.

4.5 Problem presupposes the solution (Easy to fix, recommended to fix)

Essence: Revealing the solution in the problem is self-referential and uninformative.

Discussion: Referring to the solution in the problem statement is tempting. Often a problem seems too abstract if you don't mention the later solution. People don't understand where you are headed. But revealing the solution in the problem is self-referential. A better way to preview the solution is to have a summary or synopsis.

Indicators: The problem statement describes parts of the solution. Often it uses names, concepts or structures that are only understandable or meaningful if you already know the solution.

Fix: Describe the problem as if you don't know the solution yet. Use unbiased wording that would allow other solutions as well. The specific solution of a pattern is not only caused by the problem but also by the forces. Hence, the same problem with a different set of forces could lead to different solutions. And therefore, the problem statement itself should not suggest or forecast a solution.

4.6 Limited perspective (Possible to fix, recommended to fix)

Essence: Solution and description are correct but do not draw a complete picture.

Discussion: This smell is best explained with the parable of the blind men and the elephant. In this parable a group of blind men encounters an elephant. Each has a different account of its shape. One person touches the trunk and says the elephant is like a snake. The next touches a leg and says the elephant is like a pillar. One touches the tail and feels a rope. Another one touches the elephant from the side and compares it to a wall. And there are even more possible descriptions! All of them are right, and yet incomplete if you want to describe the elephant as an integrated whole. Each description just captures a part and focuses on this perspective.

The same can happen for any pattern. If you experience a solution from a specific perspective again and again, it is easy to interpret your perspective on the thing as the thing itself. Is a red brick house that you always see from the front really only made of brick? Such limited perspectives often come into play when a solution fails to consider multiple parties. For example, a pedagogical setup might be a good solution for the teacher. But does it also solve the problem for the student? Or rather, the needs of different students? Ask one group of students who live nearby and they will interpret an intervention differently than students who have long commutes. The same limited perspective often happens in software design patterns: does the solution help the developers, the organization or the customer? Is it suitable for different programming languages, architecture models, and platforms?

It is impossible to include every possible perspective. Yet it is important to realize that the experiences, skills and attitudes of people not only lead to different interpretations but also imply partial views of the solutions and how to build them. What is achievable for one might be difficult for others.

Indicators: The pattern description contains many assumptions based on the world view of the authors. Examples are repeatedly given, but only from the specific setups and environments familiar to the authors.

Fix: Writers' Workshops and Shepherding can propose specific fixes, as new perspectives are shared. However, we recommend that authors search for more perspectives as they are initially writing about their patterns. Interview domain experts from other organizations. Talk to the different stakeholders that are involved in the solution. Have your pattern reviewed by peers and colleagues. Fundamentally, a shift of mindset is required. Admitting that you will never have the whole picture lets you dig deeper and run a more thorough analysis even if you think that you know it all.

4.7 Too abstract (Hard to fix, required to fix)

Essence: A pattern that omits too many details does not tell you anything about the essence of its form.

Discussion: To be open and allow for variance, we need to make some generalizations. However, if the description is too abstract, no one else can apply the pattern. In order to unfold the structure of the pattern, one needs to know the critical design decisions and the crucial elements that make the pattern form. An abstraction is a mapping of some part of the world that intentionally omits irrelevant features. However, if we choose an abstraction that loses essential information, nothing is gained by abstraction. Abstraction should be used only "in moderation" [10, p. 19]. Rising [11] points out that there might not be one right level of abstraction. Representations of patterns should illustrate the structural quality of recurrent solutions, i.e., the fundamental relations between the elements of the pattern [12]. A pattern description that is formulated in terms that are too abstract assumes too much prior knowledge on the part of the reader; hence, the reader cannot unfold the meaning in their Umwelt or they may miss important points.

Indicators: Difficulty applying the pattern or knowing what constitutes a good solution to the problem the pattern addresses.

Fix: Start by identifying distinct examples of how the pattern is applied. Analyze what is common to them. If there are few commonalities among examples, you may need to write one or more patterns that are more concrete. After doing this, you may find you can now see how to rewrite your original pattern, perhaps by adding more contextual details, explaining some important forces, showing illustrative examples, and from those examples, improving your general solution.

4.8 Too specific (Possible to fix, recommended to fix)

Essence: An overly specific pattern description may describe a single instance rather than the universal form.

Discussion: A pattern is too specific if it prescribes every single design step and decision to be made. When this is the case, especially if the description includes lots of detail, one can find it impossible or tedious to adapt the pattern to their specific situation. To be open and allow for adaptation, a pattern needs to illustrate the structural quality of a recurrent solution, without over constraining the solution and how to accomplish it.

Indicators: The solution offers only one example and one right way to do it.

Fix: Enumerate several examples of how the pattern could be applied (these may or may not all end up in your pattern description as examples). Then, using these examples as a guide, revise your solution to accommodate them. Strip out any unessential constraints and details that were in your original pattern solution.

5 Description Smells

You see a good solution; everything seems to be there. Except it isn't. Or maybe it is hard to find. Or it's overwhelming. Or boring.

5.1 Noise that hides gaps (Possible to fix, recommended to fix)

Essence: Some pattern descriptions miss critical details and hide this by throwing in too many irrelevant details.

Discussion: If a pattern contains too much irrelevant detail, it becomes hard to see the essence of a pattern. What is important not to miss, what is optional? Even more dangerous, overwhelming detail makes it hard to spot gaps. Being flooded with information confuses readers and it risks obscuring the lack of essential information. With so much there, how would you notice when something critical is missing?

An experienced designer knows when to break the rules and improvise. But inexperienced designers find it much harder to distinguish between essential and optional but recommended steps.

Indicators: Hard to distinguish what information is important. You feel overwhelmed by the amount of text and still can't see the picture or are left with open questions.

Fix: First trim (take out the noise), then expand (fill the gaps). To identify both noise and gaps, have a designer who is representative of your target audience verbally walk through the pattern and explain what they are thinking as they imagine trying to apply this pattern to a semi-realistic situation. Observe where they get distracted (due to unessential details) or make missteps (due to missing cues or forming false assumptions). Revise your solution and repeat this exercise with others.

5.2 Misplaced detail (Possible to fix, recommended to fix)

Essence: Parts of the description are too detailed, other parts are missing details.

Discussion: Different pattern sections require different levels of abstraction and detail, but authors sometimes put the wrong level of detail in the wrong place. Context may become overly specific, forces too shallow, or the solution excessively verbose. The result is a pattern that is either tedious to read or too vague to apply, despite containing all the “right” parts.

To use a path metaphor [13]: sometimes a change of direction is needed to get you to your destination; sometimes there are different options and you can either describe all of these options (boring!), describe just one option (but how does the designer know that there are others and that the decision for one option is not essential?), or none (let the designer figure it out). The worst kind of specificity is extraneous detail. Good: “After you see the town house, turn right”; Better: “Immediately after you see the town house, turn right”; Bad: “Move straight on and pass the coffee shop, move straight on and pass the bakery next to the coffee shop, after the coffee shop and bakery, you see a bookstore, still move straight ... and then you will see the town house, there you turn right.”

The bad description is not wrong. It’s just full of extraneous details and there is a significant risk of missing the essential direction to turn right after the town house. The “good” description may be lacking the necessary detail that you should turn immediately after the town house. A “better” description adds that extra bit of detail in just the right place. Too much noise on the one hand, and leaving out the important bits might also indicate that the author still has some confusion about her own *umwelt* (like not seeing potential dangers or making false assumptions).

Indicators: Context sections are overloaded with operational detail. Forces are described in generic terms. The core of the solution is hidden in solution details.

Fix: Experiment with different places to put the details. Try removing some details and see if things are clearer. Revise, rewrite, and have the pattern reviewed by experienced pattern authors and several designers who are your target audience.

5.3 Put it all in (Takes effort to fix, urgent to fix)

Essence: Put in all the details, and you guarantee to make your pattern description as boring as possible.

Discussion: This smell is a paradox. On the one hand, we want to see a thorough analysis of the solution. On the other hand, we want the pattern description to be clear, concise, comprehensive, and delightful to read. But often, more is less. If a pattern includes too much, the essence easily gets lost. The pattern becomes confusing.

The problem and forces can be overwhelming. True, we want to see many forces, and we want to see their conflicts to understand the problematic part of the context. But do not overdo it. The challenge is finding a good balance.

With only one force there is no tension. The way to go is clear. With two contradicting forces, at least you have a problem. With multiple forces, the problem becomes

complex and calls for a pattern (a patterned solution structure). With too many forces, you **put it all in**, and it is hard to understand which forces are most pressing.

Indicators: Everything is there, but nobody cares. The amount of text is overwhelming. One can no longer see the forest for the trees. Reading the pattern becomes boring at some point.

Fix: Outsource some information.

You can move some information into sections separate from the pattern. Make the detailed reflection and discussion an extra section or an appendix. Label skippable sections [14].

A context should quickly grab your attention. The context should cover the right situations. But not all of them, as there might be too many. Rather, focus on the important bits and leave the rest out.

5.4 Bad examples (Takes effort to fix, required to fix)

Essence: Examples are important. But one can pick inappropriate examples and use them in the wrong way. Examples should be illustrative.

Discussion: Every pattern should come with examples. Hence, we say “yay!” to examples in a pattern description. However, there are different types of examples and this is often overlooked. An example can be used to show variation, as evidence, or as illustration. These are different purposes, and all are useful (as long as we know what we’re seeing):

- **Show variation:** Several examples can show important variations (and be identified as such); they show the missing (abstracted) parts. Examples can be given with simple descriptions or photos.
- **Evidence:** Known uses are examples used as evidence. They should only back up the pattern (showing that there are multiple occurrences); they can be shorter or just referenced.
- **Illustration:** Examples that are used to illustrate certain parts or aspects of a pattern are often just sketched out (that is, not fully described). However, it is useful to have at least one example explained in detail (its structure and the way to create it; the process and thing). A running example can be picked up and continued through many patterns.

An example should be a good model. It should be a clearly understood instance, an exemplar—one that is simple enough, not mixing in too many things. While examples for evidence must not be invented, examples for illustration can either be constructed or refer to existing cases.

Indicators: Unclear example type, missing variations, embellished example (saying/showing too much).

Fix: Have one example at the beginning to set the stage and support understanding of the solution. The example can show the solution (a picture or illustration) or focus on

the problem (a story or scenario). At the end of the pattern description, cite further examples as evidence.

Use examples to illustrate implementation details. Use examples to show variations.

Use a running example to connect different patterns. A running example is a complex solution that benefits from the use of several patterns.

5.5 Missing or misleading representations (Hard to fix, required to fix)

Essence: Looking good but showing the wrong thing.

Discussion: Sometimes it is easy to show the structure of a pattern, for example by making an illustration or showing a photo. If the structure is not observable in physical space—such as behavior sequences, organizational structures—then another mapping is required.

Diagrams, stories, metaphors, and models are ways to represent the essential structure. A good model shows a simple yet whole instance of the pattern. Christopher Alexander highlights the importance of diagrams: “If you understand the need to create independent diagrams, which resolve, or solve, systems of interacting human forces, you will find that you can create, and develop these diagrams piecemeal, one at a time, in the most natural way, out of your experience of buildings and design, simply by thinking about the forces which occur there and the conflicts between the forces” [9]. If you can’t draw a diagram, then it is not a pattern.

However, as important as representations are, they can also be misleading. A poorly chosen image or crudely drawn diagram can be distracting (at best) or off-putting (at worst).

Some authors begin each pattern with an illustration or representative image. These images or illustrations are not intended to merely be decorative. They are there for a reason: as iconic representations that aid the reader in vivid recall of the pattern.

Pattern descriptions often include one or more diagrams or images that illustrate the solution. They, too, can easily show the wrong stuff. A diagram can be confusing when it shows extraneous or incidental detail, isn’t adequately labeled, doesn’t highlight what is important, is illegible, or is too “cute”. Good illustrations depict the essence of the solution. But even with a great illustration, it is helpful to accompany it with text that explains more about the illustration. Good illustrations should be recognized and applauded; bad ones should be discarded. Here are some examples that use aesthetically pleasing illustrations but sometimes illustrate the wrong structure or use a confusing metaphor: <https://refactoring.guru/design-patterns/catalog> [15].

Indicators: Too much time spent trying to understand a diagram. Readers either skip over diagrams or spend too much time puzzling out their meaning. Diagrams may emphasize the wrong things or lack any emphasis or distinguishing features. Illustrations can be distracting, poorly drawn, or misleading—figures that are decorative rather than informative.

Fix: Add labels; add a key to a complex diagram; simplify and remove extraneous detail; annotate what’s important; use good visual layout principles; avoid “clip art” or

“AI slop”; add brief text explanations. If possible, use a photo instead of a badly drawn illustration. Read Takashi Iba’s work on good illustrations and practice [16]! Aim for consistency across patterns in a pattern collection or language.

5.6 Misleading metaphors (Easy to fix, recommended to fix)

Essence: Metaphors are used to relate a familiar structure to introduce an unfamiliar one. If the analogy only covers part of the unfamiliar structure, then the metaphor gives a wrong impression.

Discussion: Metaphors and analogies are useful to explain new concepts and solutions. They help us relate them to something we are already familiar with. That’s a good thing. Unless it’s not. If the structural similarities between the metaphor and the new structure are weak, or the mapping between them feels overly contrived, then the metaphor is a failure at best, or will lead to false assumptions at worst. But, even knowing these things, we may fail to see these failures because we do like metaphors and are biased towards them.

Indicators: Overly simplistic metaphor (leading to no insights or deep understanding), awkward connections between familiar and unfamiliar concepts, misleading connotations.

Fix: Get rid of the metaphor if it isn’t working for your readers. Avoid overuse of or mixing of metaphors. Augment metaphors with concrete examples and details which aid understanding.

5.7 Poorly named (Possible to fix, urgent to fix)

Essence: Pattern names are vocabulary. A poor pattern name is difficult to use or creates wrong expectations.

Discussion: Semiotics is the study of signs in society; a sign can be a two-part relationship between a signifier and a signified. For example, “tree” is a signifier and that which is a tree is the signified. Pattern names should be instructive: “As an element of language, a pattern is an instruction which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces wherever the context makes it relevant” [12, p. 247].

Noble, Biddle and Tempero [17] consider patterns as signs and discuss the semiotics of design patterns. Pattern names should express what structure they signify. Names should not only be expressive; they should be evocative and easy to use in conversation. By giving names to patterns, we can refer to the patterns themselves as well as their implementations. This gives designers an important vocabulary for discussing both existing designs as well as design ideas: “Their names collectively form a vocabulary that helps developers communicate better” [18]. This shared vocabulary can also improve the communication between different stakeholders, such as developers and clients, or members of an interdisciplinary team. A pattern language can “create the vocabulary for a lingua franca, a common language, between designers and users” [19, p. 29].

Indicators: Readers realize that the pattern is something they did not expect when they read its name. Sometimes the name seems more like a title. It may be hard to pronounce. Even worse is when the name is long and difficult to use in conversation or in a design description.

Fix: Ensure that the name is a good sign for the signified solution. Choose a name that is expressive, evocative, and easy to understand. Avoid names that can mislead or allow misinterpretations. Check that you can use it in design descriptions, such as: “In this house we will have Light on Two Sides of Every Room”; “Let’s define an Adapter for this low-level interface”; or “In this Learning Landscape we will have Learning Alcoves, Learning Isles and Learning Corners.”

5.8 Lack of excitement (Hard to fix, urgent to fix)

Essence: The pattern is alright but it doesn’t grab the reader’s attention.

Discussion: There is nothing bad about the pattern or its solution. It might even pose a novel solution to a complex problem. But the pattern does not grab nor resonate with the reader. This can be for many reasons. The author may have assumed that the reader will just pick up the pattern, see its obvious benefits, and then use it. Or assume that the reader is just like the author. The pattern description may not make any effort to tell the reader what’s important. The writing may be too academic—or shall we be honest and call it a dull, dry read. There may be too many words and citations. Or maybe the pattern solution is sparse and important aspects of the solution are not explained. Instead of writing for readers who need more, the pattern description may capture only the gist of the idea. Or, wording and formatting get in the way of understanding.

The reader should resonate with the pattern, either by being able to apply it by themselves or by using it in design specifications. A reader should not wonder: Is it for me? Can I do it? Do I have the skill? Do I want to do it?

Indicators: There is no magic. The pattern description does not cause an inner “wow!” when read by its intended audience. A pattern should not only be practical; there should also be *enjoyment* in the here and now, right in the moment.

Fix: Adjust your writing so that other designers feel invited to add the pattern to their designs by adjusting the pattern’s structure according to their context and personal aesthetics. A designer should feel free to add her own character to the design!

If possible, show the magic of the solution, or its impact, by using photos or compelling stories.

5.9 Ambiguous confidence (Easy to fix, recommended to fix)

Essence: Missing or wrong statements of confidence.

Discussion: Christopher Alexander used stars ** in *A Pattern Language* [20] to indicate the confidence the authors had in the pattern. Subsequently, we have seen stars in pat-

tern descriptions with different meanings. Sometimes they indicate the amount of evidence (the pattern is based on many solutions), the importance, or the author's belief in the validity of the pattern. But these are very different perspectives: do the stars indicate empirical support, values or beliefs? There is too much ambiguity.

Indicators: Use of the stars in pattern names without explaining their meaning. No statements of confidence.

Fix: Get rid of the stars. Sometimes it is better to make no statement about things you aren't sure about. Otherwise, understand and explain to your readers what you mean by the stars and be consistent with their use.

You may also find a better system for characterizing the importance, level of trust, degree of normative values and tested empirical evidence for your domain.

6 Utopic World Smells

A great solution, a pleasure to read. But does it work for me? It seems that the patterns are perfect for a perfect world...but not for the world we live in—with all its constraints, politics and limitations.

6.1 Everything is happy ever after (Easy to fix, urgent to fix)

Essence: Negative consequences and limitations are missing.

Discussion: A pattern should solve a problem as a whole. But even *A Pattern Language* [20] does not claim that a single pattern solves everything. That is why patterns depend on other patterns. A pattern in isolation can never do the entire job and nothing comes without having to make trade-offs, choosing among alternatives, or incurring some cost. Using a particular pattern may eliminate other viable options; you can't have your cake and eat it too. Costs (such as money, time, sacrifices) are always a negative consequence. If you can't live with the costs, retreat from the solution! A consequence should make the cost more specific. What do you have to give up to achieve this? What negative side effects are there? There are always consequences.

Indicators: No consequences at all, just good consequences, promising too much, missing links to related patterns that solve consequent problems.

Fix: Add consequences. Every action you take has consequences. Putting a design into the world is an intervention that has consequences—both positive and negative. Describe both.

If solving one problem leads to new problems, then you can refer to smaller patterns that address these problems. That is why pattern languages offer a more holistic approach to solving problems. New problems do not remain unresolved.

However, you may not always be able to point to existing solutions. In such cases, it is better to at least point out the new problems and let the designer be aware of them. The resulting problems should be smaller in scale; hence the overall situation after applying a pattern should be better, not perfect.

6.2 This will always work (Takes effort to fix, recommended to fix)

Essence: Assuming, without evidence, that the pattern works in more contexts.

Discussion: Some patterns assume that a solution always works in every situation and for everyone. However, culture, the skill of designers and the attitudes and values of users of solutions (solution beneficiaries and performers) are part of the context. The same problem in different contexts requires different solution forms.

Indicators: A pattern that refers to a solution that has only been applied by the authors. Or a single solution instance that is generalized and presented as a universal approach: Just do as we do. A solution that promises too much. Or is only achievable by superheroes. (Or those who have way too much time and resources). Keeping quiet about important constraints, limitations or counter indicators.

Fix: Expand the sample size and look for similar solutions elsewhere. It is okay to start from your own solution (which you know best) but you should discover it elsewhere, too, before you claim that it is a pattern.

Reflect on the assumptions and requirements for the patterns. When should it not be used? What are counter indicators? Keep the resources and capabilities of your target audience in mind. Can they actually build the pattern?

Do not only look for positive examples. Look for examples where the pattern failed. This will not invalidate the pattern. Rather, you will get a better understanding of the context (and the limits of the context). Add alternatives that could work in other contexts. Name and list alternatives if you know about them. The context tells you when to use a pattern. However, it often does not tell you when NOT to use the pattern. Unlike medical drugs that come with a note about possible side effects and when to not use it, this is often skipped in pattern descriptions. You can also make readers aware of an overdose as the Patterns of Project Management Risk Reduction [21] does.

6.3 Fantasy assumptions (Takes effort to fix, required to fix)

Essence: The pattern makes unjustified and quite unrealistic assumptions about an idealized world, especially the behavior of humans.

Discussion: Patterns can be aspirational and foresee a better world by pure reasoning and deductive argument. While this is the exception, it is not a smell as long as this is clearly indicated. The smell sneaks in when there are assumptions about resources, politics, physical laws, or human behaviors that are unrealistic from the start. For example, if the solution depends on perfect weather conditions (assuming the sun is always shining), unlimited computing power (assuming there are no costs for using tokens), or unbiased human actions (assuming rational behavior), this is a smell. If only everyone would donate a dollar every month, if only everyone used 10% less energy, if only there were more time...this is all wishful thinking. This smell is dangerous because it can also be found in patterns based on existing solutions. It might well be that in the observed solutions conditions have been ideal but it is wrong to extrapolate such perfect circumstances without justification.

Indicators: All sorts of wishful thinking and variants of “If only...” statements, such as “All we have to do is... [take the bike, read the manual, donate a little],” or “Assuming that...”. While creative thinking sometimes uses “What if...?” thinking to open up alternative solutions, the “If only...” thinking is the opposite: it sticks to one particular solution that requires unrealistic preconditions. While such limitations technically belong to the context, the smell is often hidden in the solutions—the recommended actions and configurations make assumptions. For example, if the solution suggests using “safe passwords” as the main security measure, it makes the assumption that users will use safe passwords whenever possible. But most users only do so when it is convenient.

Fix: When you spot a hidden assumption, check first whether it is justified. Not every assumption is wrong. But often, solutions only work under certain narrow conditions. That’s why there is a context description for patterns. Try to make assumptions explicit in both context and solution descriptions. Reconsider your known uses and ask whether they had ideal conditions (or contexts).

7 Pattern Language Smells

The individual patterns are fine but they don’t work together or are hard to combine with other solutions.

7.1 Calling a collection a language (Easy to fix, urgent to fix)

Essence: There is nothing wrong with collections, just don’t call them languages (yet). Because a pattern language promises more.

Discussion: It is fine to create a pattern collection, but don’t conflate it with a pattern language. A pattern language contains richly interconnected patterns and emphasizes the relationships between them. The patterns in a pattern language are used together to solve a larger problem (larger than the problems solved by individual patterns in the language) in a specific domain. Using one pattern leads to using another, and another. The solution unfolds; the patterns in the language work together to create a solution.

Although a pattern collection contains related patterns, the patterns in it are often used independently. Using one doesn’t lead naturally to considering the use of another. Patterns in collections are related through some common characteristic or trait (for example, the Creational Patterns in the *Design Patterns* [7] book), not because they work together to solve a problem.

Pattern collections are easier to create and maintain than pattern languages. A pattern language promises more—a vocabulary for thinking about a larger problem and then composing several pattern solutions that work together to solve it.

A good example of a language composed of several smaller ones can be found in *Object-Oriented Reengineering Patterns* [22]; each chapter is a small pattern language focused on a specific re-engineering activity.

Indicators: Difficulty in describing how the patterns work together to solve a problem. Links between patterns are weak (or non-existent or based only on common properties). All patterns are at the same abstraction level.

Fix: Label your grouping of patterns according to the way it is best used; don't conflate a collection with a language. Both pattern collections and pattern languages have value; they are just used differently. Be honest about whether you have assembled a pattern collection or a pattern language.

Turning a pattern collection into a language involves understanding (and developing) solutions that use a related set of patterns that work together. It might involve adding larger-scoped or smaller-scoped patterns, adding new patterns, referencing patterns from other collections or languages, or even removing particular patterns that lack strong connections to others.

7.2 No levels of scale (Hard to fix, optional to fix)

Essence: The size of the description should match the size of the solution; there are always larger solutions that build on smaller solutions.

Discussion: In a pattern language you need levels of scale. The scope and granularity of the solution are hierarchical. From big to small. The length of the pattern descriptions also varies. The big pattern is brief, as it delegates work to the patterns of lower levels. The patterns at the lowest levels are also often brief, as they have a much smaller scope. A network of patterns suggests that there are patterns made of other patterns (one whole is made of sub-wholes). These parts must be smaller patterns, as they are part of a larger whole.

Indicators: All patterns are of the same size, all descriptions are of the same size, all sections are of the same size. Same size can refer to the length of the description but more importantly to the scope of the pattern, or the granularity of a pattern.

Fix: Check that the scope, granularity, and detail of the description match the scope, granularity, and detail of each pattern (or the solutions the pattern is based on).

7.3 Misleading map (Takes effort to fix, recommended to fix)

Essence: The map does not show how and why related patterns are connected, or the ways the patterns are used together.

Discussion: If you have a pattern language, it is desirable to show the relations between various patterns. One way to do this is through a "pattern map". As the number of patterns in a language grows, a map showing all the patterns and potential relations can become complex and unwieldy to navigate. A map should be organized to show how strongly related patterns are connected, and common ways patterns are used together. When a single map becomes too large or disorganized, or the pattern connections aren't carefully managed, a map can be confusing and less than useful.

Indicators: Many connections between patterns; unclear organization; hard to orient to the map: where should you start, or how might you progress through understanding and applying the patterns?

Fix: Show only strong local connections, not all possible relations between patterns. Organize patterns in your map according to sub-problems they are used to solve. Make it clear how patterns naturally follow from other patterns. One way to do this is to use directional cues (arrows drawn downward from patterns that naturally lead to other groupings of patterns).

Organize the patterns in your language into subunits. Consider labeling and bounding them. An example redrawn from *Object-Oriented Reengineering Patterns* [22] (see Fig. 2) shows five patterns from the chapter “First Contact,” illustrating one way to show relationships. This map also conveys a sense of iterative application of particular pattern groups.

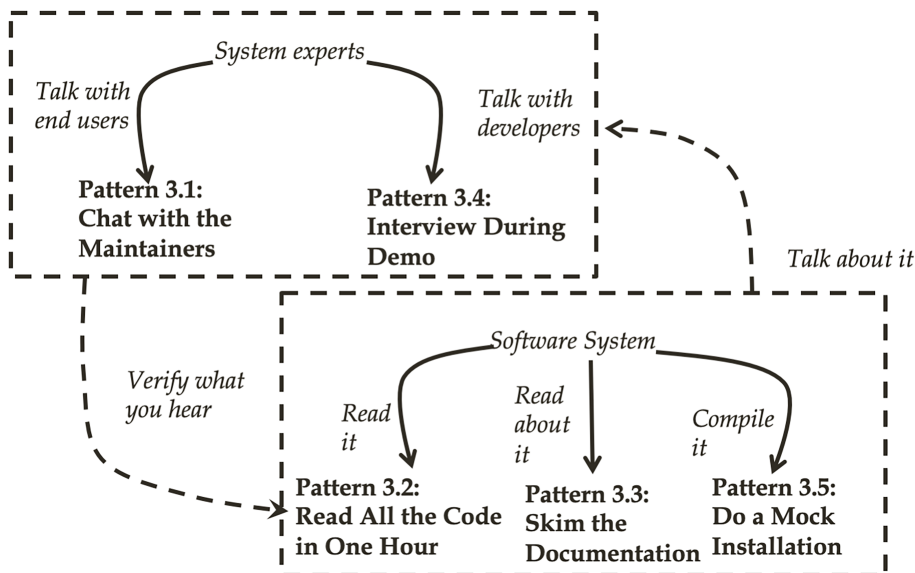


Fig. 2. A map showing relationships between five First Contact patterns in *Object-Oriented Reengineering Patterns* [22].

Does your map provide orientation or is it misleading? Is it whole and easy to perceive? A book page might be too limiting for a map. Think about having a wall map, or foldouts in a book.

7.4 Isolated solution (Possible to fix, urgent to fix)

Essence: A pattern should not be an island. Patterns can be nested, overlapping, competing, and collaborating.

Discussion: In pattern languages the pattern descriptions themselves explain connections to other patterns. This is done by “related patterns” sections or by referring to other patterns by their names. If the context of a pattern refers to other patterns, this indicates that its application makes sense in the context of those patterns. If the resulting context does not resolve all forces, it can show where to proceed by pointing to more patterns: “Each pattern then, depends both on the smaller patterns it contains, and on the larger patterns within which it is contained” [12, p. 312]. Moreover, connections between patterns provide orientation in the pattern vocabulary. They make explicit that a “pattern uses another pattern, a pattern refines another pattern, or a pattern conflicts with another pattern” [23]. A pattern description that stands alone, in isolation, without any referential information lacks connection. It is missing its wider *umwelt*.

Indicators: No mention of existing patterns or pattern literature or related patterns.

Fix: Refer to existing patterns or pattern languages. Write patlets—brief descriptions of a sentence or two—for those other patterns that are not written yet and reference them.

7.5 Lack of openness (Hard to fix, urgent to fix)

Essence: A lack of flexibility and adaptability of patterns as well as incompatibility with other patterns.

Discussion: In order to adapt to the specifics of a situation, a pattern is more open if it allows a variety of sub-patterns (open to the replacement of sub-parts). In addition, a pattern that does not narrow the options for how to proceed is more open (open to different next steps). A resulting context that can be addressed by more than one sub-pattern is more open. Likewise, a context that is broader (e.g., that includes more preceding or following design steps) makes the pattern more open to combination with other patterns. Without sufficient openness, the pattern cannot co-evolve with other solutions and along with the user’s *umwelt*. Openness does not only matter at the time of creating a system, but also impacts its maintenance.

Indicators: The solution depends on a very specific setting rather than diverse environments and continuous change.

Fix: Assess whether any parts of the context can be generalized without losing necessary constraints. For example, if the context is about specific systems (computer systems, applications, organizations) ask, what would be the more general category of that system?

8 More Smells

We have identified even more smells that occur in a pattern’s lifecycle. They all could benefit from more detailed descriptions. However, in this section we limit ourselves to sharing only their essence. In future papers we will provide more detailed discussions. We have no illusion that our list is complete. Hopefully, it is a good start.

8.1 Wholeness smells

Unwhole: You can perceive, or imagine, solutions that lack wholeness. You can lose wholeness and the essence of a solution, in the capturing, in the making, and in the using.

Lack of quality: Some patterns only focus on the functional side and utility rather than revealing their inner beauty and harmony. In Christopher Alexander's view, when a place becomes more alive, it is more whole and has a "quality without a name" [12].

Missing generativity: The pattern provides no guidance on how to unfold the solution and build it with piecemeal growth. There is no discussion on how to generate the solution in various contexts.

No properties of life: One cannot find any fundamental properties of life [24] in the patterns. Their impact on society and the community is unclear. The patterns don't make you feel good.

8.2 Misconception smells

Principles as patterns: A principle shows a direction or a goal but does not define a recurrent structure other than the recurrent outcome. Very different solution patterns can use the same principle.

Phenomena as patterns: A specific solution that entails a pattern is not the pattern itself. Selling a unique implementation as *the* pattern is misleading.

Patterns as tricks: Out-of-the-box approaches that create the illusion of solving a problem but ignore their impact on the environment. Immediate remediation without considering negative long-term consequences.

Patterns never change: Suggesting that a pattern is the ultimate solution. However, as the environment changes, patterns have to change as well.

Novelty required: Some patterns are criticized because they offer nothing new. Academics outside of the pattern community often complain that patterns are only based on existing solutions and therefore contribute nothing. But patterns are generalizations and the analysis of them frequently produces new insights.

Novelty disregarded: Some patterns are criticized because they provide new solutions. The "disregard for originality" [25] highlights that patterns should be grounded in existing instances. However, this disregard is only meant to protect against speculation, fiction, and untested proposals. It is not against new observations.

Only one format: Currently there are a few established formats for pattern descriptions. But there might be other ways to communicate patterns effectively. The choice for a format should be considered and explained.

8.3 Writers' Workshop smells

Not enough time for patterns: The group discusses title, abstract and introductions for a long time and there is not enough time for all of the patterns.

Too much for a workshop: Some papers contain too many pages and it is impossible to give proper feedback.

Linear discussion: Following the structure of the paper rather than focusing on the parts that need most attention. Often the sections are written in a different order and feedback should also be given in an order that prioritizes the most urgent issues.

Side discussions: Sometimes the discussion gets off-track and no one stops it. At *PLOP conferences, many writers' workshops get so side-tracked by interesting side discussions that they lose sight of the paper. And we are so easily distracted.

Alphas and silent participants: In some workshops a few talk a lot while others say almost nothing.

8.4 Shepherd smells

Weird introduction: Shepherding is a mentoring process, and there is a personal relation between shepherd and authors. Shepherds should introduce themselves humbly.

Authoritative shepherd: Shepherds who prescribe and demand rather than suggest, tutor, and ask the right questions.

Unresponsive parties: Some shepherds take their time in contacting authors and responding to emails. Likewise, some authors take their time to integrate the feedback, and don't let the shepherd know what their status is.

Unfocused shepherd: Feedback that ignores the major problems is pointless. Too much feedback and too much detail can be confusing. Experienced shepherds prioritize their comments.

AI shepherd: Shepherds who use AI to generate feedback (or worse yet, to rewrite the author's work), and take a back seat in the discussion. This is a poor form of interaction. Certainly, authors can (and should) use AI for feedback themselves. Besides the potential copyright violation, such shepherds risk being replaced by machines.

Defensive author: Authors who use their time to write defensive answers to their shepherds instead of improving the paper. If content needs defending, it should be done in the paper. If positions need to be explained, they should be found in the paper.

These smells can easily be fixed by reading Neil Harrison's *The Language of Shepherding* [4].

8.5 Usage smells

Patterns for the sake of patterns: Inexperienced designers think that the quantity of used patterns adds to the quality of systems. But patterns only add value when they are needed.

Misapplied patterns: Related to the previous smell, this smell happens if designers choose patterns that do not fit their problem. Context, problem, and forces should all be matched to one's own situation.

Overengineering: Using a pattern that fits but is not needed because the system does not need its benefits.

No adaptation/unfolding: Patterns are not prescriptions that should be followed step by step. Designers have to adapt the pattern to their specific needs and unfold the solution in their contexts.

Use without responsibility: Every design relies on the decisions made by the designer. Learning from existing solutions does not relieve us from our responsibility to make and justify our own design decisions.

Patterns fix it forever: Many patterns help improve system maintenance. However, evolving systems may call for new solutions. A pattern that once played an important part in the system needs to be evaluated for its fitness every now and then.

Using outdated patterns: Over time a good pattern becomes more popular. But over time environmental changes may call for different solutions. Not all patterns are timeless.

8.6 Publication smells

Publish and forget: The author takes great care and improves the quality of their patterns. The dissemination of patterns, however, is a weak spot. More effort is needed to distribute and advertise patterns to their target audiences.

No reports of application: If a pattern has been published but nobody uses it or reports on their experiences, then the pattern has limited use. Its value is not corroborated and it cannot evolve.

No revisions after Writers' Workshops: A lot of effort was put into giving feedback to the authors. It is disrespectful for authors to ignore the feedback and publish the work without updates.

Only one description: To make the patterns accessible to a wider audience, different versions of a pattern can be published: the detailed description, summaries, checklists, posters, card games, etc.

9 Conclusions

At this time, we have collected a number of smells that we have frequently encountered when we read, use, apply, and reflect upon patterns. However, we have not yet tested to what extent these smells (or the awareness of the smells) can actually help us to write better patterns or give better feedback in workshops. We will now start experimenting with referring to the smells and seeing whether this will improve our discussions.

While we have good heuristic understanding of the potential fixes to smells, we have not yet considered the strengths of the smells. To overstress this metaphor, which are the smells that bring tears to our eyes, i.e., which smells are stronger and more harmful than others? Also, any attempt to get rid of one smell may generate new smells. Think of using soap that smells even worse than the odor it removed. When is it better to leave a smell rather than attempt to remove it?

These smells have not been tested in the wild. Can they actually help the shepherding process? Can they enlighten the discussions in writers' workshops? And can they be used to automate a first round of feedback using generative AI? There is also a need for guidance on how to use the pattern smells for providing constructive human feedback in the shepherding process and in writers' workshops. This will be the subject of a follow-on paper for the upcoming EuroPLoP. Moreover, the smells presented in this paper are limited to the scope of mining, writing and publishing pattern descriptions. The other smells will be described in future work.

We have only just started on our quest of identifying smells and ways to fix them. We believe we should also focus on the bright side, and name pattern delights: the typical situations that make you smile and open your heart when you read patterns or see them in action.

Acknowledgments. The authors would like to acknowledge our shepherd, Michael Weiss, who generously gave us his time, sound advice and encouragement. We also acknowledge our fellow writers' workshop participants who candidly gave us great feedback, and Lise Hvatum, who thoughtfully critiqued our penultimate version and gave us good advice.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this paper.

References

1. Beck, K. Code Smells. <https://wiki.c2.com/?CodeSmell> Accessed 9 September 2026.
2. Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code*. Reading, MA: Addison-Wesley.
3. Wirfs-Brock, R. & Kohls, C. (2025). Patterns and their Umwelt – How to Make Patterns Come to Life. *HILLSIDE Proceedings of the 32nd Conference on Pattern Languages of Programs, People, and Practices (PLoP)*.
4. Harrison, N.B. (2006). The Language of Shepherding. In D. Manolescu, M. Völter, & J. Noble (Eds.), *Pattern Languages of Program Design 5*. Boston: Addison-Wesley.

5. Harrison, N.B. (1998). Potential Pattern Pitfalls, or How to Jump on the Patterns Bandwagon Without the Wheels Coming Off. In L. Rising (Ed.). *The Pattern Handbook* (pp. 87–96). Cambridge: Cambridge University Press.
6. Buschmann, F., Henney, K., & Schmidt, D.C. (2007). *Pattern-Oriented Software Architecture. Volume 5: On Patterns and Pattern Languages*. West Sussex: John Wiley & Sons.
7. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
8. Welicki, L., Yoder, J., & Wirfs-Brock, R. (2009). Adaptive Object-Model Builder. In Proceedings of the 16th Conference on Pattern Languages of Programs (PLOP '09). Association for Computing Machinery, New York, NY, USA, Article 4, 1–8.
<https://doi.org/10.1145/1943226.1943231>
9. Alexander, C. (1964). *Notes on the Synthesis of Form*. Cambridge: Harvard University Press.
10. Gabriel, R. P. (1996). *Patterns of Software: Tales From the Software Community*. New York: Oxford University Press.
11. Rising, L. (2007). Understanding the Power of Abstraction in Patterns. *IEEE Software*. 24 (4). 46–51.
12. Alexander, C. (1979). *The Timeless Way of Building*. New York: Oxford University Press.
13. Kohls, C. (2012). Patterns and Paths - Introducing the Path Metaphor. *EuroPLOP 2012 - 18th European Pattern Languages of Programs Conference*. Writers' Workshop version. Irsee.
14. Meszaros, G., & Doble, J. (1997). A Pattern Language for Pattern Writing. In R.C. Martin, D. Riehle, & F. Buschmann (Eds.), *Pattern Languages of Program Design 3* (pp. 529–574). Reading, MA: Addison-Wesley.
15. Refactoring.Guru: Design Patterns Catalog. <https://refactoring.guru/design-patterns/catalog>. Accessed 29 September 2026.
16. Iba, T., Banno, Y., & Ando, H. (2022). Principles of Pattern Illustration Design. In *Proceedings of the 26th European Conference on Pattern Languages of Programs (EuroPLOP '21)*. Association for Computing Machinery, New York, NY, USA, Article 31, 1–25.
<https://doi.org/10.1145/3489449.3490009>
17. Noble, J., Biddle, R., & Tempero, E. (2006). Patterns as Signs: A Semiotics of Object-Oriented Design Patterns. *Systems, Signs & Actions*. 2 (1). 3–40.
18. Vlissides, J. (1997). Patterns: The Top Ten Misconceptions. *Object Magazine*. March 1997.
19. Borchers, J. (2001). *A Pattern Approach to Interaction Design*. Chichester: Wiley.
20. Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press.
21. Cockburn, A. (1997). Patterns of Project Management Risk Reduction. EuroPLOP 1997.
<https://riehle.org/community-service/hillside-group/europlop-1997/p22final.pdf>
22. Demeyer, S., Ducasse, S., & Nierstrasz, O. (2003). *Object-Oriented Reengineering Patterns*. Retrieved from <https://open.umn.edu/opentextbooks/textbooks/object-oriented-reengineering-patterns>
23. Noble, J. (1998). Classifying Relationships Between Object-Oriented Design Patterns. *Australian Software Engineering Conference (ASWEC)*. 98–107.
24. Alexander, C. (2002). *The Nature of Order, Book 1: The Phenomenon of Life*. Berkeley, CA: The Center for Environmental Structure.
25. Foote, B. (1997). The Introduction to Pattern Languages of Program Design 3. In R.C. Martin, D. Riehle, & F. Buschmann (Eds.), *Pattern Languages of Program Design 3*. Reading, MA: Addison-Wesley.