

Constructive Feedback Using Pattern Smells

Christian Kohls*¹[0000-0001-5858-7068] and Rebecca Wirfs-Brock²[0009-0006-7215-3788]

¹ TH Köln - University of Applied Sciences, Köln, Germany
christian.kohls@th-koeln.de

² Wirfs-Brock Associates, Sherwood, OR 97140, USA
rebecca@wirfs-brock.com

Abstract. This paper explores how “pattern smells”—a concept inspired by (and reimagined from) software engineering’s code smells—can support more effective feedback throughout the lifecycle of pattern mining, writing, and maintaining their descriptions. Pattern smells name and classify recurring weaknesses in patterns, ranging from misconceptions, insufficient domain grounding, and gaps in analysis, to overly optimistic or incomplete utopic claims and lack of wholeness. Rather than focusing on fixing individual smells, the paper shows how they can guide different forms of feedback, including self-reflection by authors, iterative shepherding, writers’ workshops, and AI-generated feedback. The discussion maps specific smells to appropriate stages of feedback on patterns as they are written: early stages emphasize identifying whether real, whole solutions exist; intermediate stages refine understanding of solution details, forces, evidence, and context; and later stages improve clarity, structure, and presentation. The authors argue that explicitly naming such flaws helps participants in the pattern writing and review process articulate vague concerns, improve communication, and strengthen both pattern descriptions and pattern languages. Overall, pattern smells provide a practical framework for diagnosing weaknesses and improving the quality and communicability of patterns.

Keywords: pattern smells, patterns, pattern languages, pattern descriptions, shepherding, writers’ workshops

1 Introduction

We introduced the concept of pattern smells in our paper “Patterns and their Umwelt: How to make patterns come to life” [1]. The term “pattern smell” is borrowed from the software domain where “code smells” [2, 3] indicate that something is rotten in the code, alerting you that something is wrong that needs fixing. Pattern smells can help us to name and characterize what is not quite right with a pattern. They can clarify our thoughts when we have only a vague feeling that something isn’t right with the patterns.

We wrote more elaborate descriptions of pattern smells and their remediation in our paper “Pattern Smells, Get Rid of Them” [4]. The focus of this paper is different. We want to show how pattern smells can help clarify our critique and improve the quality of feedback in each of the different phases that are involved in finding, describing, applying, and living the patterns of good practice.

There are different times in the process of pattern mining, writing, and publishing [5], and throughout the process smells sneak in. There is some natural order to the phases, such as selecting a domain, selecting mining methods and fields, the actual pattern mining (including the identification of pattern candidates, looking for more evidence, analyzing the functioning of the patterns with both practical knowledge and theoretical reasoning), the pattern writing, the dialectic feedback in shepherding, the feedback from writers' workshops, the final reviews and edits for publishing, and ultimately monitoring and maintaining the patterns to address changes in context (as the world moves on) and to account for new data (which may support or contradict a pattern).

However, this natural order does not imply a strict linearity. Quite the contrary: very often it is in the writing process that we realize that patterns need to be merged or split. Moreover, writing and receiving feedback reveals knowledge gaps (for example, missing justifications for design decisions, or limitations of the pattern) and we need to go back to the field. When patterns are part of a collection or a language, adding new patterns can affect existing patterns. It is fair to say that finding, writing, and maintaining is a messy business. And as we focus on the good stuff, it is very easy to miss the smells that enter into our pattern descriptions.

This is where our paper aims to provide some aid.

We have described our pattern smells in more detail in "Pattern Smells, Get Rid of Them" [4]. Each smell has a name, is labeled with the difficulty of fixing it (easy/possible/takes effort/hard to fix) and how urgent it is to fix (required/urgent/recommended/optional), and includes a brief description and discussion, indicators of the smell, and a fix—heuristics for addressing the smell. Here is an example:

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). Testability and sociotechnical impact are also commonly overlooked forces. 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.

So far, we have described 30 smells and identified over 30 more (their brief descriptions appear in the appendix). Fig. 1 shows an overview of the smells we previously described in [4] and refer to in this paper.

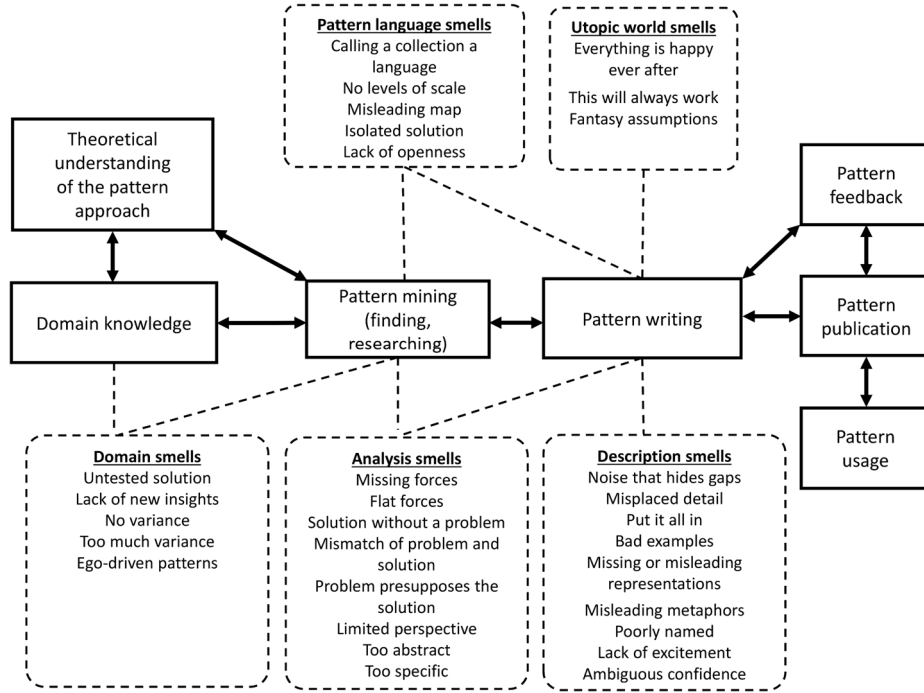


Fig. 1. Relation of pattern smells to phases. Each phase is in a box. The different smells are in dotted boxes. Smells are linked by dotted lines to the phases. Some smells are typically found in, but not limited to, specific phases. More smells will be added in future work. Each phase is connected by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

2 Avoiding and Fixing the Smells

Ideally, we would avoid smells altogether. But smells are unavoidable. They are a side effect of identifying and writing patterns, if you will. For example, as we start writing patterns, we want to share our knowledge as best as we can—and so we start to *Put it all in*. And as our text gets longer, we start to have a *Lack of excitement* or create *Noise that hides gaps*. Moreover, things that are overripe have a tendency to rot and start smelling. If a description hasn’t been updated for some time, it is likely that it no longer represents the state of the art.

We can get rid of some smells if we counteract them carefully. We say “carefully” because sometimes fixing a bad smell causes another smell. But that is the name of the

game: fixing as best you can. In particular, we want to show how pattern smells can be minimized through actions taken to remove smells in response to a variety of feedback:

- Individual self-critique and reflection (single authors, or individuals of a group in charge of writing a pattern) during pattern mining
- Group self-critique and reflection (a group of pattern miners and authors discussing their own work) during pattern mining and writing
- Peers who provide feedback on the current state of the patterns
- Systematic feedback during shepherding (in several iterations)
- Systematic feedback in writers' workshops
- Observations and suggestions generated by artificial intelligence (AI)

In the upcoming sections we will relate the smells to different phases of pattern development. We introduce a short description of each smell in the sections in which it is first used. Some smells can be addressed in several phases. We explain each smell when we first refer to it and briefly recall it where it matters again.

3 Before the Mining Starts

Some smells can be avoided before pattern mining starts. Misconception smells indicate that the authors have some misperceptions about what patterns are. It is important to have a good understanding of what patterns are in the first place before trying to find and describe sound patterns. Domain smells occur when authors have chosen a domain in which they have only limited expertise. As we mine patterns, we should not rely only on our own expertise.

The following two smells are typical for authors who are domain experts but not yet pattern experts:

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

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

To avoid these smells, one should become familiar with the works of Christopher Alexander. Read at least a few of his patterns in *A Pattern Language* [6]. Moreover, we recommend reading *The Timeless Way of Building* [7] from cover to cover as it provides a strong foundation for understanding the theory behind patterns. One should also read POSA-5 (*Pattern-oriented software architecture. Volume 5*) [8] to get some theoretical background. Kohls [9, 10, 11] has used a path metaphor to explain the qualities of a pattern and provide suggestions for the writing process. Typically, qualities include wholeness [12, 13, 14], good abstractions [8, 15, 16], openness and variability [7], equilibration and fitness, encapsulation [12], and generativity [17, 18].

The other smells that occur even before the mining starts are related to incomplete domain knowledge. In such situations, sooner or later these smells will arise:

Lack of new insights: The identified solutions are not wrong but they don't offer any new insights into the domain. 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 more deeply. If your pattern is a refinement, then craft your pattern description to include both the unique challenges and how your specialized solution addresses them. The new insights can also come from a better description of the solution—a better discussion fostered by the pattern format. But it is important to label your pattern as offering a new way of describing knowledge rather than claiming it to be new knowledge. In our paper “Truth in labeling”, we discuss ways to indicate what new insights are presented in a pattern [19].

Ego-driven patterns: The pattern sells ideas of the authors rather than ideas seen in the field. Ask yourself whether you are promoting an idea because you like it and think it is the right thing to do or whether there is a common understanding, broad agreement, even evidence and research that the solution works.

4 As the Mining Begins

There are many different ways to identify patterns in any domain. Often, the recognition of a few recurring solutions is exactly what motivates authors to share those insights. As we start looking for more patterns and get a better understanding of the ones we have already identified, we learn more about the solutions and how they actually work. As we study further, we often integrate our own expertise with insights from peers and established principles. Such approaches are subjective, based on our personal expertise and experiences.

However, pattern mining can also be based on scientific research, using qualitative methods such as interviews, observations, and text or object analysis, as well as quantitative methods such as performance measurements or surveys.

Both approaches have value, and often the best patterns are based on both personal experience and systematic research.

Typical smells that occur at the beginning of mining:

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

Solution without a problem: It is unclear why the solution is needed. Maybe the pattern solves the problem of making something. But we also want to understand why we should make that thing in the first place. If a pattern is both “a thing” and “a process”, then we want to learn about the problem that the thing solves, and about the problem of creating that thing. Think about what would happen if the solution did not exist. What would be bad? That is the problem.

Ambiguous confidence: Missing or wrong statements of confidence. Explain to your readers what your level of confidence is and establish clear labels or indicators.

To avoid these smells, one can improve the mining process and select among various mining methods (research methods) accordingly:

Introspective approach / individual contribution: Self-observation and analysis of one's own projects to see which processes and designs have been successful or not [20, 21]. It is also called the retrospective approach [22].

Social approach / secondary contribution: Observation of the environment and the behaviors in the domain, interviews with experts who explain their experiences or patterns [20, 21].

Artifactual approach: Observation and analysis of project results [20]. Many of Alexander's architectural patterns have been found by this method: by studying existing buildings. Many software design patterns are developed using this approach [8] by contrasting and comparing similar systems [23].

Pattern mining workshops: Focus groups are used to collect, categorize, and summarize the experience of experts. Such discussions often show that there are different views on the same issues that can be harmonized in the course of discussion [21]. Iacob [24] describes design workshops in which "participants are encouraged to externalize any design problem, solution, or decision they consider relevant to their design process."

Holistic pattern mining: Iba and Isaku [25] describe a workshop format for a holistic pattern mining approach. Instead of mining individual patterns, the workshop aims at mining whole languages. Experiences are collected on sticky notes and clustered visually by a group of participants. This process reveals patterns and connections between them for a domain. This approach works best when experienced practitioners from the domain are also involved.

Gap identification: Hanmer [26] has written patterns for pattern mining that reflect on gaps identified in a pattern language and then distribute the writing among a group of authors.

Each of the methods can be used in a pragmatic, design-oriented approach or in a more rigorous scientific fashion. The scientific value of patterns has been discussed in Kohls and Panke [22] and Riehle, Harutyunyan, and Barcomb [27]. Iba has systematized the whole process of crafting pattern languages, including the mining [25] and writing process.

But it's not only the mining methods that count (i.e., how you mine and look for nuggets of wisdom) but also where you look (i.e., the domain area you mine). In scientific terms, this is the sampling strategy: which samples and how many of them you account for in your research. In practical terms, this means whether you are limiting your studies to a narrow field or expanding to a wider field of examples and known uses. For example, are the software patterns programming language specific, company specific, paradigm specific, or more universal? Another example from education: are the patterns specific to your own teaching, to your university, or more generally to all universities?

5 In the Middle of Mining

This is the most exciting, most insightful, and messiest part! You identify new patterns. You get a better understanding of how and why solutions work. And you realize how little you know, how many design contradictions remain.

This is where you analyze your findings from applying the methods described in the previous section. In addition to your own findings, you can refer to existing literature. By doing so, you can avoid these smells:

No variance: Without variance there is no generative design space. To fix this, show variance and describe ways of making distinct adaptations. Separate variations from the basic solution and label them explicitly.

Too much variance: With too much variance, the essence of form—the gestalt—is lost. To fix this, 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.

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

Too specific: An overly specific pattern description may describe a single instance rather than the universal form. Enumerate several examples of how the pattern could be applied. 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.

Mismatch of problem and solution: There is a problem, there is a solution, but they do not fit together. As you observe recurring solutions and distill them into patterns, this smell is best fixed by not touching the solution but by reflecting again on 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.

It is in the middle of mining that you have to make some hard decisions:

- Is an observed solution a singular phenomenon?
- When does a solution become a pattern candidate?
- When can you be sure that you have a solid and important pattern that is worth writing about?
- Do you really understand how the solution works or do you need to do further research?

6 At the End of Mining

There is no end to mining. Or, more precisely, the mining ground is never fully exhausted. We can always find more solutions and learn more about the solutions we have identified [28].

Still, at some point we have gathered enough expertise—our own and the insights of others—that we feel confident enough to write our initial patterns. Usually, we begin writing the first patterns while mining continues.

But before we make a decision to start writing a pattern, we should check the quality of its solution.

Wholeness smells mean that the patterns, or the pattern languages, are incomplete, not self-sustained, unbalanced, or lack harmony. While some of the smells only arise as we write the patterns, it is helpful to check before you start whether or not each of the pattern candidates is whole. Watch for wholeness smells, and for analysis smells that undermine wholeness by leaving the forces unbalanced. Before you start writing, check that none of the following occur:

Lack of quality: Patterns focus on the functional side and utility rather than inner beauty and harmony.

Ignoring the umwelt: The umwelt of the pattern authors, readers, designers, and users matters. Without acknowledging and understanding the different stakeholders and having the target group in mind, patterns will not help to unfold good solutions in new situations.

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

Missing forces: Important forces, the main drivers of the solution, are missing. Examine the solution for fitness to the problem as it is currently stated. 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.

You can have a whole form, a solution that has the quality without a name (QWAN), and that is alive. But you can also lose those qualities as you start writing about the pattern. A pattern description and the pattern are not the same. Any pattern description is only a mapping of the pattern. And the pattern is a generalization of multiple solutions; in such abstractions we can lose the wholeness that was once present in the more specific solutions.

7 As the Writing Begins

Writing is where you craft pattern descriptions that capture the knowledge about designing a solution. This knowledge was collected in the pattern mining process. You may identify some gaps as you start writing. You can always return to the mining field to learn more and fill the gaps. But there are also some smells that sneak in as you write and write and rewrite:

Noise that hides gaps: Some pattern descriptions miss the essential details and hide this by throwing in too many irrelevant details. This makes giving feedback very hard: You want to tell the authors to add more detail and get rid of detail at the same time. First trim (take out the noise), then expand (fill the gaps). To identify both noise and gaps, have a 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.

Put it all in: Put in all the details, and you guarantee that your pattern will be as boring as possible. 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. Likewise, 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.

Lack of excitement: The pattern is alright but it doesn't grab the reader's attention. Adjust your writing so that other designers feel invited to add the pattern to their designs by adapting 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.

Missing or misleading representations: Looking good but showing the wrong thing.

Bad examples: Examples are as important as illustrations. But one can pick the wrong examples and use them in the wrong way. Examples should be illustrative, too. 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). Use more examples as evidence at the end of the pattern description. 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.

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

Misleading metaphors: Metaphors are used to relate a familiar structure to an unfamiliar one (in this case the solution structure). If the analogy only covers part of the structure, then the metaphor gives a wrong impression. Get rid of the metaphor if it isn't

working for your readers. Avoid overusing or mixing of metaphors. Augment metaphors with concrete examples and details which aid understanding.

There is good information about pattern writing. Doug Lea has compiled a checklist for writing good patterns [29]. There is also a pattern language for writing patterns that provides guidance [30]. In addition to these patterns, Harrison [31] collected further patterns for advanced pattern writing. Wellhausen and Fießer [32] address first-time pattern authors and reflect on their own experience in writing patterns. They point out that patterns are not written in the sequence they are presented; very often the writing starts with the solution, followed by the problem, consequences, forces, and context. Vlissides [33] identifies seven habits of effective pattern writers: taking time to reflect, adhering to a structure, being concrete early, keeping patterns distinct and complementary, presenting effectively, iterating tirelessly, and collecting and incorporating feedback.

8 As You Are Writing

Experiment with different styles of writing and pattern formats. Each format will require you to provide specific information about the patterns. Make sure you can fulfill those demands before deciding on a specific pattern format.

As the writing continues, at some point you may submit your work to a pattern conference. Then, a shepherd will be assigned to your paper and you will start a discourse about your work.

We recommend involving artificial intelligence before you first submit. You can ask an AI to identify the smells we described [4]. Upload your own texts along with our smell descriptions and see what the AI suggests. Not all suggestions will be useful. However, addressing this additional feedback *before* the shepherding starts can improve the shepherding experience. Focus on improving what you most care about by eliminating or reducing smells before you submit your paper.

There are some smells you should be aware of as you decide to write a pattern and as you continue writing. Sometimes, we get overexcited as we write. Utopic world smells refer to patterns that promise too much:

Everything is happy ever after: Negative consequences and limitations are missing. 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 make the designer aware of them. The resulting problems should be smaller in scale; hence the overall situation after applying a pattern should be better, not perfect.

This will always work: Assuming that the pattern works in more contexts than the evidence supports. 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?

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

A good pattern description does not claim to provide more than it actually does. If a description is based on personal knowledge, that's fine, but don't assume that it works for everyone. Some patterns are aspirational and foresee a future that does not yet exist. Such patterns may describe reasonable solutions that haven't been tested. However, when this is the case, it is important to label them as speculative. Moreover, such speculative solutions need stronger argumentation and backup from principles and theories to compensate for the missing evidence that the proposal will actually work.

Try to avoid the *Untested solution* smell. While this smell often occurs at the beginning of the mining process, it is not uncommon for the details section to call for, well, details. And in the description of these details, it may be that you add some details that are not tested yet. That is, a solution may have been tested as a whole, but some of its details have not. This leads to the untested solution smell: The pattern description proposes solution details that have never been tested. To fix this smell, 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)?

But even when we describe existing solutions, there is always a temptation—maybe a necessity—to add some fiction. Sometimes we fabricate justifications and explanations as we write. Sometimes we elaborate on things that we have insufficient experience with or incomplete knowledge of.

There is a fine line between aspirational and fictional patterns. While you can provide evidence for observed patterns, you need to make strong arguments for aspirational patterns and label them as such. If possible, you should support these kinds of patterns with other research to increase confidence in the proposed solution. However, once a solution is put into the world, we often get a better understanding of what works and what fails. It is sometimes hard to say in advance what is an achievable vision and what is pure fiction.

Avoid confusing aspirational patterns with those with tested solutions. If you explicitly label a pattern as speculative, it cannot be criticized as untested; the label grants a grace period in which to show that the pattern is a good idea. Perhaps every pattern description has elements of true observation, visions of how things could be and how things should be.

It might seem that writing about solutions you have seen many times is easier. But be cautious about blindly trusting the soundness or quality of your observations. Even though you have seen the solution repeatedly, it doesn't mean that it is a good solution.

And so it happens that sometimes we are confident in our pattern but less so in our writing. Smells can help us to identify, understand, and name what is wrong with a pattern description: the smells of *Flat forces*, *Noise that hides gaps*, *Misplaced detail*, or *Solution without a problem*.

To avoid *Misplaced detail* (information under the wrong heading, or at the wrong level of detail), 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.

Ideally, authors use these insights to improve their work before submission or publication. But sometimes it helps to be honest about the work-as-it-is. One can add statements like:

- There are more forces but additional mining and research are required to fully understand how and why this works.
- These are the known uses: ... A more detailed description will be added.
- There are only two known uses the authors are aware of but it is assumed that additional instances can be found in the field.
- Whether this pattern is actually two different patterns is subject to discussion.

Being honest about what we don't know is better than making stuff up. In fact, the strictness of common pattern formats can lead to some speculative descriptions that are not labeled as such. Because authors feel obligated to fill out all sections, they invent to deliver. Our patterns are neither universal truths nor pure fiction. Some knowledge gaps can be resolved as we continue to write. Others may be resolved only through future research. That is fine. Acknowledging this is better than inventing fiction and passing it off as fact.

Finally, the writing should be polished before you submit. Spend time editing. Can you condense the content and get rid of unnecessary details and sentences? Does it feel better to have many little patterns or one big pattern? Do you use language appropriate for the target audience? Is everything in the right section? Do you get lost in details? Is your writing too abstract? Do your patterns generate excitement? Avoid the *Lack of excitement* smell or all your efforts will be wasted because no one will care about your pattern.

9 The Shepherding Begins

Before a pattern paper is accepted for a conference and discussed in a writers' workshop, it is shepherded by an experienced pattern author. There is even a pattern language for shepherding [31]. The shepherd builds a personal relationship with the authors and provides iterative feedback on the evolving pattern paper. This non-anonymous peer review is based on a trusted relationship where authors can ask shepherds

questions and shepherds can be candid in their critique and suggestions for improvement.

There are different phases in the shepherding process: initial feedback, feedback closer to the program committee review, and feedback close to the final submission. At each stage, authors and shepherds should do their best to get rid of smells and improve both the patterns and the writing. There should be at least three rounds of feedback, and often many iterations and additional revisions.

9.1 Initial Feedback

As a shepherd, you should not jump into commenting on details before the big questions are answered. You should not suggest how to better describe a solution if there is no solution at all. So, in the first round, start by identifying misconception smells, domain smells, wholeness smells, and utopic world smells.

Spot the issues that would make further detailed feedback a waste of your time. That is, if you have the feeling that one of the proposed patterns is not a pattern at all, say so right away and do not elaborate. It is the job of the author to provide supporting information, not yours. And it certainly does not make sense to comment on whether the statements of the description are in the right section (e.g., context, problem, or forces) or whether they could be stated more clearly, if the thing described isn't a pattern.

These smells should be addressed:

- Misconception smells (missing understanding of what patterns are)
- Domain smells (lack of new insights, ego-driven patterns)
- Wholeness smells (solutions that are incomplete, unbalanced, or lack harmony)
- Utopic world smells (promising too much)

If you cannot clearly see the solutions or you have the feeling that each pattern contains multiple solutions that are not self-contained, then you might face an inexperienced pattern author who does not fully understand the design theory behind patterns. So, you might suggest that they look at good examples of pattern descriptions that are easy to understand as good models to emulate. You can also point out specific pattern language smells.

You can also start by looking for the wholeness of each pattern. If you cannot see the gestalt, then you cannot get an idea of the solution. You cannot even judge whether it is a good or bad solution, or whether there is enough evidence for both the pattern and the statements in the pattern description. Therefore, it does not make sense to provide feedback on details of the solution or details of the description. You have to be able to see the whole of the solution. And the value of it.

Do you understand what their patterns are? Without this understanding you cannot provide reasonable feedback. So let the authors know: I cannot see what you mean! To fix this, the authors may need to add examples, illustrations, or be more precise. You can also suggest that they split or merge patterns if there is too much/little variance or too abstract/specific detail.

Make a quick assessment of strengths and weaknesses of their patterns. Let the authors know which patterns worked best. And tell them why they worked. Then ask them to write the other patterns in the same style.

If all of the patterns are hard to see and understand, you have to decide whether the authors have enough valuable domain expertise to mine and better describe their patterns. Is there domain expertise the author has to offer? Is there potential for finding good patterns during the shepherding process?

9.2 Content Feedback During Shepherding (Second Round Plus Iterations)

At this stage you should have a paper version that actually contains patterns! (If your author took your first round of comments seriously or if they are an experienced pattern author, that is.) So, you can sniff for some analysis and description smells (problems in how the pattern is written down). Analysis smells lead you to ask for more elaboration.

Make sure the authors get this second round of feedback before they submit an updated version for program committee review. Authors should have enough time to integrate at least some of the shepherd's comments before they submit for the review that will decide whether their paper is accepted.

Start with analysis smells. They indicate that not all information about the solution and its context is elaborated. Look for gaps and ask authors to add more information:

- *Missing details*: Details essential to understanding or applying the solution are left out.
- *Missing forces*: The 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 they have not considered (security, for example, is often overlooked in software patterns).
- *Flat forces*: Ask them to explain their priorities, weights, conflicts, and relations.
- *Lack of evidence*: Claims about the solution are not supported by known uses, data, or research.
- Tips for further reading (literature related to their pattern)
- Why should I use the solution? Avoid the *Solution without a problem* smell.
- Why is the solution shaped that way?
- When should I use the pattern? When should I not use it? Sniff for the *This will always work* smell: assuming that the pattern works in more contexts than the evidence supports.
- No utopia! Utopic world smells refer to patterns that promise too much. *Everything is happy ever after*: Negative consequences and limitations are missing. Ask the author: What are the consequences? What are the limitations?
- *Bad examples*: Do the examples help to understand the pattern and its variance? Consider the examples in the text and check their function and purpose. Do they explain, do they illustrate, do they connect solutions? Are they good models? Do they show variation?

- *Missing or misleading representations*: Ask the author to illustrate the structure, either with diagrams, photos, examples, stories—whatever makes the gestalt perceivable. We should focus more feedback on diagrams, illustrations, and photos. These artifacts illustrate the form of the pattern; and bad examples or illustrations can miscommunicate the essence. Annotate what’s important, use good visual layout principles, avoid “clip art” or “AI slop”, and add brief text explanations. If possible, use a photo instead of a badly drawn illustration.
- *Isolated solution*: A pattern should not be an island. Ask the authors, “Do you know about these other patterns?” Suggest how they relate to their pattern.

While the analysis smells indicate that there should be more information (details), the description smells often lead to reducing information and focusing on those details that matter. Ask the author to be concise and cut out content:

- Move things to the appendix.
- Be more precise.
- Match the pattern descriptions to the needs of the intended audience.
- Authors should take out their egos.
- Remove made-up things (the lies).
- Back up claims.
- Add appropriate context discussion to avoid the *Problem presupposes the solution*: Revealing the solution in the problem is self-referential and not informative.
- Remove *Noise that hides gaps*: Some pattern descriptions miss the essential details and hide this by throwing in too many irrelevant details. This makes giving feedback very hard: While you may want to tell the authors to add more detail, they also need to get rid of some details at the same time.
- Avoid *Put it all in*: Put in all the details, and you guarantee that your pattern will be as boring as possible.
- *Misplaced detail*: When the authors have mixed up context, problem, or forces, you can already provide suggestions for restructuring. However, if some of the elaborations are just under the wrong heading, this can be easily fixed. It matters more that every perspective is covered than that each is in the right place.

9.3 Final Shepherd Round

The last suggestions should be about improving the text by

- Revising to add excitement. *Lack of excitement*: The pattern is alright but it doesn’t catch the reader’s attention.
- Renaming the pattern to reveal the essence of the solution. This is often related to the *Misleading metaphors* smell. If a metaphorical name is used, it should reflect the essential structure of the pattern.
- Asking for clarification (very detailed and specific feedback, i.e., what does this or that sentence mean?)
- Suggesting simple spelling or grammatical edits.

Pattern names are vocabulary. A poor pattern name is difficult to use or creates false expectations. Help the authors to avoid the *Poorly named* smell. Ask authors to make sure that the name is a good sign for the signified solution. A name should be expressive, evocative, and easy to understand. Point out names that can mislead or allow misinterpretations. Check whether 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.”

10 Feedback in the Writers’ Workshop

The writers’ workshop takes place at all *PLoP conferences. It is another round of feedback from several peers. An elaborate discussion about the elements and backgrounds of a writers’ workshop can be found in Coplien and Woolf’s “A Pattern Language for Writers’ Workshops” [34] and Gabriel’s *Writers’ Workshops and the Work of Making Things* [13]. More recently, Gabriel has written “A Pattern Language for Writers’ Workshops” [35].

Some papers present several patterns that are part of a larger pattern language or collection. Other papers may describe patterns that are not yet organized into a language or that extend or refine existing published patterns. Regardless, a writers’ workshop discussion should start with the patterns and keep the focus on them.

First ask: Which patterns work best? Which are whole?

Do you understand the solutions? If the authors present a pattern language in their paper, do we understand the pattern language? Do we understand the whole of the language?

Address pattern language smells both at the beginning to make the discussion easier (for example, if a pattern is hard to read because it contains several solutions and should be reworked into several patterns), and at the end to provide suggestions for future development and refinement of the pattern language. Common pattern language smells are:

- *Calling a collection a language*: A pattern language contains richly interconnected patterns and emphasizes the relationships between them. There is nothing wrong with collections, just don’t call them languages (yet). A pattern language promises more—a vocabulary for thinking about a larger problem and then providing several patterns that work together to solve it.
- *Misleading map*: A map should be organized to show how related patterns are connected, and the 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 quickly become confusing and less than useful.
- *No levels of scale*: The size of the description should match the size of the solution; there are always larger solutions that build on smaller solutions. Discuss whether 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).

- *Isolated solution*: A pattern should not be an island. Patterns can be nested, overlapping, competing, and collaborating. In pattern languages, the patterns themselves introduce connections to other patterns.
- *Lack of openness*: Openness refers to the flexibility and adaptability of patterns as well as their compatibility with other patterns outside of the language.

Once you have discussed the overall structure of the collection or language, then focus on each pattern. If you don't understand the style of writing, point that out.

Start with the pattern that needs the most attention! If they all need attention, pick one as an example and discuss it in more detail. Give tips for that one example and illustrate how to change it.

Point out these smells as appropriate: *Noise that hides gaps*, *Misplaced detail*, *Bad examples*, *Missing or misleading representations* (such as images or illustrations), *Putting it all in*, and *Problem presupposes the solution*.

Much of the discussion should focus on the pattern solution. Does the solution have wholeness? Is it actionable and testable? What do you think about the solution? Do you understand it? Do you feel comfortable building it? Do you feel its potential?

Does the solution balance the forces?

If you encounter a pattern where you cannot clearly see the solution, say so right away. There can be two reasons for this. There might actually be no solution. In this case, it is not worth spending time discussing the pattern any further. Just say that you need more evidence to buy into the proposed solution. But there are also other cases where there might be a solution, but you cannot yet see it. In this case, it is useful to provide feedback on how to make the solution more understandable.

For each pattern, establish at the beginning of the discussion whether you see the quality without a name. Does it feel whole to the participants? Do you buy the solution? Or is it too perfect? That could be a *Lack of roughness* smell: the solution is presented as too finished to leave designers room to adapt it. We do not believe in absolute perfection. A pattern can never be perfect. It needs to be rough in order to let designers make their own decisions.

Next, discuss whether the solution fits the problem and context. Are the forces resolved? Which other forces are resolved by the solution? Can you identify the *Flat forces* smell?

From now on, most of the discussion should be about analysis smells, similar to those smells that might have been detected during shepherding. These include *Missing details*, *Missing forces*, *Flat forces*, *Solution without a problem*, *This will always work*, *Everything is happy ever after*, and *Missing or misleading representations*.

The writers' workshop discussion will reveal remaining smells that the authors should ideally address in their final submission.

11 Revision After the Writers' Workshop

The writing of patterns is itself part of research and overlaps with pattern mining. As we write, we gain a deeper understanding of what the patterns are. Hence, shepherding and the writers' workshops contribute not only to the quality of writing but to the quality of our understanding of the patterns. In each of the phases, different stakeholders are involved, and each is qualified in different ways to identify and fix smells. Hence, the dialectic discourse that occurs both in shepherding and writers' workshops contributes to the substance of patterns. Therefore, the final submission should integrate these insights.

Ideally, authors take notes during the workshop about what is said. Authors should take the feedback they received into account as they revise their work. It might be that some workshop participants didn't understand what the paper was saying because they were not the target audience. That's okay. If more feedback is needed, authors can share their work with other experts and incorporate their feedback before they make their final submission.

Authors should reflect on what people liked, misunderstood, thought unclear, or felt was missing. What did the commenters find intriguing? Or surprising? Or insightful? Don't wait too long to revise a paper; otherwise, the impressions will fade.

Authors should decide which smells to really work hard to get rid of. Sometimes one has to live with a particular smell to achieve something else good. Also, a pattern can smell in one context but be fine in another. Here is an example: if you only write for yourself or your organization, it's perfectly fine to be less rigorous about the mining ground. But claiming your personal patterns to be universal is a smell; the smell is not the pattern; the smell is in the claim about the pattern. Ideally, the final published paper should make the intended use of the patterns clear.

At the end, a final AI check might be done to see what smells remain in the final text. One should not indiscriminately use AI feedback. Reflect on it. Sometimes it's pure nonsense; sometimes it points to smells that still exist and should be addressed.

One smell that can be fixed with the help of the writers' workshop is *Limited perspective*. Sometimes, the solution and description are correct but do not draw a complete picture. Writers' workshops and shepherding can add new perspectives. However, we recommend that authors search for more perspectives as they are initially writing about their patterns.

12 Conclusion

In this paper, we have shown how pattern smells can be avoided in the making of patterns and pattern descriptions—from pattern mining to writing and incorporating feedback.

However, new smells can occur after the final submission and publication. A once shining and exemplary pattern can get smelly as time passes. Why is that? The context may have changed and the solution isn't as fitting as it used to be. Or we have gathered

more experience about the domain, rendering certain forces and assumed causal relations between context, problem, forces, and solution invalid. That is, the real-world test of re-applying a pattern in the field may show that some of the statements about the pattern are empirically wrong or less predictable than claimed. If such new data comes in, the pattern needs to be updated. Otherwise, it no longer captures the phenomena of the world appropriately, and that is a smell.

There are several more types of smells that also likely need attention:

- Shepherding and mentoring have smells (such as being too authoritative, unresponsive, or unhelpful).
- Writers' workshops have smells (such as an overfocus on introductions and abstracts and minor editing suggestions rather than patterns themselves, repeating yourself, getting sidetracked, or advertising your own experiences instead of helping the authors).
- The application of patterns has smells (such as using patterns for the sake of patterns, maximizing the use of patterns, overengineering, or strictly following a pattern without adaptation).

These smells call for further investigation in the future. For now, however, we find avoiding smells in the mining and writing of patterns to be the most urgent.

We believe that our list of smells will grow over the next several years. We plan to use this approach in our own work and see how knowing and using these smells improves our own writing and reviewing processes. In the near future, we hope to learn both from our own experiences and from the feedback of others who also try to identify smells throughout the writing and publication process. We want to learn to what extent knowing smells can help. Then, we hope to provide a revised guideline with smells and how to use them for giving even better feedback.

Acknowledgments. We thank our shepherd, Ademar Aguiar, who gave us sound advice and encouragement. We also acknowledge our fellow writers' workshop participants who candidly gave us helpful feedback.

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

Appendix: Additional Smells

The following additional smells have been identified. A full discussion of these smells is left for future work.

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.

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

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

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” [36] highlights that patterns should be grounded in existing instances. However, this disregard is only to protect from 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 of a format should be considered and explained.

Writers' workshop smells

Not enough time for patterns: The group discusses title, abstract, and introduction 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.

Shepherding 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 raising confidentiality and copyright concerns, 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.

Neil Harrison's *The Language of Shepherding* [37] offers remedies for many of these smells.

Usage smells

Patterns for the sake of patterns: Inexperienced designers think that the number of patterns used 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: A pattern may remain popular long after it has stopped being useful. Environmental changes may call for different solutions. Not all patterns are timeless.

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.

References

1. Wirfs-Brock, R., Kohls, C.: Patterns and their Umwelt: How to Make Patterns Come to Life. In: Proceedings of the 32nd Conference on Pattern Languages of Programs, People, and Practices (PloP '25), Article 8, pp. 1–15. The Hillside Group, USA (2025). <https://doi.org/10.64346/PloP2025p14>
2. Beck, K.: Code Smells. <https://wiki.c2.com/?CodeSmell>. Accessed 9 September 2026.
3. Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison-Wesley, Reading, MA (1999).
4. Wirfs-Brock, R., Kohls, C.: Pattern Smells, Get Rid of Them. In: Proceedings of the Viking-PloP 2026 Conference on Pattern Languages of Programs, People, and Practices. To appear (2026).
5. Quibeldey-Cirkel, K.: ETHOS: A Pedagogical Pattern. In: J. Coldewey, P. Dyson (eds.) Proceedings of the 3rd European Conference on Pattern Languages of Programming and Computing, pp. 261–268. Universitätsverlag Konstanz, Konstanz (1999).
6. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York (1977).
7. Alexander, C.: The Timeless Way of Building. Oxford University Press, New York (1979).
8. Buschmann, F., Henney, K., Schmidt, D.C.: Pattern-oriented software architecture. Volume 5: On patterns and Pattern Languages. John Wiley & Sons, West Sussex (2007).
9. Kohls, C.: The structure of patterns. In: Proceedings of the 17th Conference on Pattern Languages of Programs (PloP '10), Article 12, pp. 1–10. ACM, New York (2010). <https://doi.org/10.1145/2493288.2493300>
10. Kohls, C.: The Structure of patterns: part II – qualities. In: Proceedings of the 18th Conference on Pattern Languages of Programs (PloP '11), Article 27, pp. 1–18. ACM, New York (2011). <https://doi.org/10.1145/2578903.2601079>
11. Kohls, C.: The path to patterns: introducing the path metaphor. In: Proceedings of the 17th European Conference on Pattern Languages of Programs (EuroPloP '12), Article 9, pp. 1–16. ACM, New York (2012). <https://doi.org/10.1145/2602928.2603085>
12. Alexander, C.: Notes on the Synthesis of Form. Harvard University Press, Cambridge, MA (1964).
13. Gabriel, R.P.: Writers' workshops and the work of making things: Patterns, poetry. Addison-Wesley, Boston (2002).
14. Coplien, J.O.: Software Patterns. SIGS Books & Multimedia, New York (1996).
15. Lea, D.: Christopher Alexander: An Introduction for Object-Oriented Designers. ACM SIGSOFT Software Engineering Notes 19(1), 39–46 (1994).
16. Gabriel, R.P.: Patterns of software: Tales from the software community. Oxford University Press, New York (1996).
17. Appleton, B.: Patterns and Software: Essential Concepts and Terminology (2000). <https://www.bradapp.net/docs/patterns-intro.pdf>. Accessed 1 October 2026.
18. Hvatum, L., Wirfs-Brock, R.: Exploring the Generative Nature of Patterns. In: Proceedings of the 30th Conference on Pattern Languages of Programs (PloP '23), Article 12, pp. 1–17. The Hillside Group, USA (2023).

19. Wirfs-Brock, R., Kohls, C.: Truth in labeling. In: 33rd Conference on Pattern Languages of Programs, People, and Practices (PLoP 2026). Writers' Workshop version (2026).
20. Kerth, N.L., Cunningham, W.: Using Patterns to Improve Our Architectural Vision. *IEEE Software* 14(1), 53–59 (1997).
21. DeLano, D.E.: Patterns Mining. In: Rising, L. (ed.) *The Patterns Handbook*, pp. 87–96. Cambridge University Press, Cambridge (1998).
22. Kohls, C., Panke, S.: Is that true...? thoughts on the epistemology of patterns. In: Proceedings of the 16th Conference on Pattern Languages of Programs (PLoP '09), Article 9, pp. 1–14. ACM, New York (2009). <https://doi.org/10.1145/1943226.1943237>
23. Rising, L. (ed.): *The Patterns Handbook*. Cambridge University Press, Cambridge (1998).
24. Jacob, C.: A Design Pattern Mining Method for Interaction Design. In: Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '11), pp. 217–222. ACM, New York (2011). <https://doi.org/10.1145/1996461.1996523>
25. Iba, T., Isaku, T.: Holistic Pattern-Mining Patterns. In: 19th Conference on Pattern Languages of Programs (PLoP 2012), Tucson, Arizona. Writers' Workshop version (2012).
26. Hanmer, R.S.: Pattern mining patterns. In: Proceedings of the 19th Conference on Pattern Languages of Programs (PLoP '12), Article 23, pp. 1–10. The Hillside Group, USA (2012).
27. Riehle, D., Harutyunyan, N., Barcomb, A.: Pattern Discovery and Validation Using Scientific Research Methods. In: E. Wallingford, U. Zdun, C. Kohls (eds.) *Transactions on Pattern Languages of Programming V*. LNCS, vol. 14630, pp. 226–253. Springer, Berlin, Heidelberg (2025). https://doi.org/10.1007/978-3-662-70810-1_6
28. Wirfs-Brock, R., Hvatum, L.: The Value of Pattern Collections. In: Proceedings of the 31st European Conference on Pattern Languages of Programs, People, and Practices (EuroPLoP 2026), Irsee, Germany. To appear (2026).
29. Lea, D.: Pattern Writing Checklist. The Hillside Group. <https://hillside.net/index.php/pattern-writing-checklist>. Accessed 1 October 2026.
30. Meszaros, G., Doble, J.: A Pattern Language for Pattern Writing. In: R.C. Martin, D. Riehle, F. Buschmann (eds.) *Pattern Languages of Program Design 3*, pp. 529–574. Addison-Wesley, Reading, MA (1997).
31. Harrison, N.: Advanced Pattern Writing. *Pattern Lang. Program. Des.* 5, 433 (2006).
32. Wellhausen, T., Fießer, A.: How to write a pattern? a rough guide for first-time pattern authors. In: Proceedings of the 16th European Conference on Pattern Languages of Programs (EuroPLoP '11), Article 5, pp. 1–9. ACM, New York (2011). <https://doi.org/10.1145/2396716.2396721>
33. Vlissides, J.: Seven Habits of Effective Pattern Writers. In: *Pattern Hatching: Design Patterns Applied*, pp. 145–152. Addison-Wesley, Reading, MA (1998).
34. Coplien, J.O., Woolf, B.: A Pattern Language for Writers' Workshops. In: N. Harrison, B. Foote, H. Rohnert (eds.) *Pattern Languages of Program Design 4*, pp. 557–584. Addison-Wesley, Reading, MA (2000).
35. Gabriel, R.P.: A Pattern Language for Writers' Workshops. <https://breathturn.com/Files/WritersWorkshopPL.pdf>. Accessed 1 October 2026.
36. Foote, B.: The Introduction to Pattern Languages of Program Design 3. In: R.C. Martin, D. Riehle, F. Buschmann (eds.) *Pattern Languages of Program Design 3*. Addison-Wesley, Reading, MA (1997).
37. Harrison, N.B.: The Language of Shepherding. In: D. Manolescu, M. Völter, J. Noble (eds.) *Pattern Languages of Program Design 5*, pp. 507–530. Addison-Wesley, Boston (2006).