Two patterns, a study and a message for the validation of our patterns

Apostolos V. Zarras
Department of Computer Science and Engineering, University of Ioannina
Greece
zarras@cs.uoi.gr

ABSTRACT

Each time we write a pattern we have to make sure that the pattern can be used by others and show how this can be done in practice. Moreover, we have to make sure that what we describe is really a pattern that concerns others and not just a one-shot solution to a problem that concerns only us. This paper revisits the issue of pattern validation. In this context, the paper reports in the form of patterns two well-established techniques that allow pattern authors to show the applicability and the generality of their patterns. Following, the paper investigates the extent to which authors apply these techniques in practice, in a study of 109 EuroPLoP papers, published from 2019 until 2021. The results of the study indicate that the pattern authors show the applicability and the generality of their patterns quite often, but not always. Therefore, the overall validation process can be further improved.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**.

KEYWORDS

Patterns, Validation, Known Uses

ACM Reference Format:
Apostolos V. Zarras. 2023. Two patterns, a study and a message for the validation of our patterns. In *28th European Conference on Pattern Languages of Programs (EuroPLoP 2023), July 05–09, 2023, Irsee, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3628034.3628047

1 INTRODUCTION

The following excerpt is a shepherd's comment to his sheep during the shepherding phase of a past EuroPLoP conference

> *"... But what comes to known uses, I want to be sure that the pattern is grounded to practice and comes from the experience of many. Pattern is a pattern because it has been observed three times independently in separate occurrences. If you see a solution, it is just noise. If you see the same solution twice in different products by different vendors, it might be a pattern, but it might also be coincidence. If you see the same solution thrice, it is a pattern. I still feel, that the known uses section really describes only one instance of the solution. So, I would very much like to see added other occurrences where the three-step refactoring detector has been observed. Patterns are not novel, they are not invented, they are discovered..."*

The shepherd's comment is pretty much inline with the definition of what is a pattern, given by Christopher Alexander and his colleagues, in their seminal book, back in 1977 [4].

> *"A pattern is a careful description of a perennial solution to a recurring problem within a building context, describing one of the configurations that brings life to a building. Each pattern describes a problem that occurs over and over again in our environment, and then describes the core solution to that problem, in such a way that you can use the solution a million times over, without ever doing it the same way twice."*

The message conveyed by the previous quotes is that a *valid* pattern description should illustrate the *applicability* and the *generality* of the pattern. Typically, we show these two key properties via *examples* and *known uses* of the pattern. Specifically, to show the applicability of the pattern we provide a detailed example that illustrates the solution of the pattern in a specific instance of the problem. On the other hand, to show the generality of the pattern we discuss known uses of the pattern in several different real-world instances of the problem.

In the literature, there are several popular pattern writing forms [11–13]. In some forms, examples and known uses are considered as mandatory elements of the pattern description, while in others this is not the case. For instance, in the Alexandrian form [4] a pattern description includes an archetypal example of the pattern, while the problem statement provides empirical background and evidence of the pattern validity. In the GoF form, a pattern description comprises a motivating example, sample code that shows how to use the pattern in the motivating example and a discussion of known uses. The POSA form [6] includes a motivating example, the example resolved based on the pattern solution and known uses. The Portland form [8] is more brief, comprising the problem statement and a few paragraphs that discuss the solution; examples and known uses are not mandatory in this form. The PEAA form [11] is also brief, it includes examples that show the use of the pattern in practice, but the examples are not necessarily from real-world situations. The Coplien form [7] focuses mainly on the context, the problem, the forces, the solution, and the resulting context. Similarly, the Fearless Change pattern form starts with an opening story and a summary, followed by the problem, the forces, the solution, and the resulting context [20].

EuroPLoP 2023, July 05–09, 2023, Irsee, Germany

ACM ISBN 979-8-4007-0040-8/23/07.
https://doi.org/10.1145/3628034.3628047

Regarding pattern writing guidelines, the situation is similar. Meszaros and Doble [23] provide a very interesting pattern language for patterns writing. According to this language, the mandatory parts of a pattern description are the name, the context, the problem, the forces, the solution, and the consequences. Examples and code samples are among the optional elements that can be used when necessary. Harrison's guidelines [14] focus mainly on how to write the problem, the solution, and the forces of the pattern. He also points out the importance of specifying the target audience. At the end of their paper, Harrison focuses on further details that could be useful for the readers, along with the description of the pattern. As part of this discussion he refers to code samples and examples. Wellhausen and Fiesser [27] focus mainly on the context of a particular pattern, the problem that is solved, the solution, and the consequences.

This paper attempts to give more emphasis on the issue of pattern validation. To this end, the paper provides two patterns: Illustrate Applicability and Show Generality. The purpose of the patterns is to pinpoint the need, the mechanics, and the benefits of pattern validation, via examples and known uses. Moreover, the paper investigates *the extent to which pattern authors actually use these techniques*, in a study of 108 papers, published in previous EuroPLoP conferences.

The rest of the paper is structured as follows. Sections 2 and 3 introduce Illustrate Applicability and Show Generality. Section 4 discusses the setup and the results of the study. Section 5 concludes the paper with a more general discussion concerning the validation of our patterns and related issues that we should consider to achieve this.

2 ILLUSTRATE APPLICABILITY

Context

You are writing a pattern that gives a solution to a problem in a context. You have already described the context, the problem, the forces, the solution, and the consequences. Your pattern describes the core of the solution to the problem in hand at a certain level of abstraction so that the solution can be applied several times to solve different instances of the problem.

Problem

Being described abstractly can make the pattern hard for some people. So, you want to make sure that you described the pattern in a way that allows the readers to actually apply it in practice and show how this can be done in more detail.

Forces

Using a pattern can be difficult when the description of the pattern suffers from certain weaknesses like a very broad context, a vague problem statement and forces, a very abstract solution, a solution that does not match the problem, and so on.

Even if the pattern description is adequate, it is given at a certain level of abstraction. Mapping the abstract concepts (classes, interfaces, roles, etc.) of the pattern to the respective concrete concepts of a particular problem instance may not be easy for all people who intend to use it.

A very detailed pattern description may restrict the applicability of the pattern to very specific cases.

Solution

Illustrate the use of the pattern in a detailed example. Start with the description of a particular situation in which the pattern can be applied, i.e., an instance of the context. Describe the problem instance in the given situation. Then, describe the solution to this problem instance. If there are different variants of the solution it would be good to discuss them in the context of the given example. Illustrate all the different concepts, parts, elements, roles, etc. involved in the pattern. To make the example more clear you can provide a mapping between the abstract concepts of the pattern and the concrete concepts of the specific example.

Consequences

Giving a detailed example that demonstrates the use of the pattern is a sanity check that shows to you and the readers that the pattern can be actually applied in practice. Giving a good example that reflects the context, the problem and the solution of the pattern can help you to improve the description of the pattern itself. It allows you to give concrete details about the solution separately, without restricting the solution.

The example helps the reader to understand the intent of the pattern, witness a particular situation in which the pattern can be applied and see how to map general pattern-specific concepts, parts, elements, roles to respective situation-specific concepts, parts, elements, roles. etc.

The example can make the description of the pattern longer, more complicated and tiring for the readers.

Finding a good example that reflects the context, the problem and the solution of the pattern may not be easy. A superficial example will be just noise.

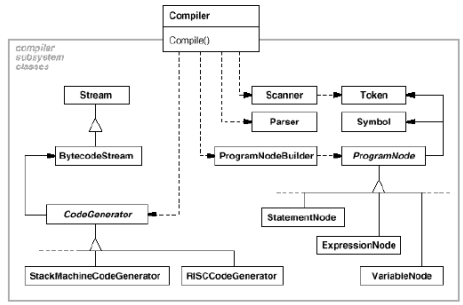
A good example that demonstrates a solution to a problem in a context does not establish the generality of the pattern, i.e., the recurrence of the pattern in several real-world situations.

Example

Facade is a well-known GoF pattern [13]. The intent of the pattern is to make the use of a subsystem easier via a higher-level interface. To show the applicability of the pattern the authors employ the example of a compiler subsystem that includes various classes. To make the use of these classes easy the compiler subsystem also includes a Compiler class that serves as a facade. The provided example includes an overall design that shows the relations between the involved classes, along with code samples that illustrate implementations of the classes (Figure 1).

Known Uses

All of the patterns in the seminal book of Gamma et al. [13] include examples and code samples that illustrate the usage of the patterns in respective problem instances. The use of detailed examples and code samples for the illustration of patterns is further employed in several other well-formed pattern catalogs like the Fluent C catalog of C programming patterns [25], the xUnit automated testing patterns catalog [21], the Enterprise Application Architecture (EAA)




```
class Compiler {
   public:
   Compiler();
   virtual void Compile(istream&, BytecodeStream&);
};

void Compiler::Compile (
   istream& input, BytecodeStream& output
) {
   Scanner scanner(input);
   ProgramNodeBuilder builder;
   Parser parser;
   parser.Parse(scanner, builder);
   RISCCodeGenerator generator(output);
   ProgramNode* parseTree = builder.GetRootNode();
   parseTree->Traverse(generator);
}
```

Figure 1: Excerpt from the compiler example that illustrates the applicability of the GoF Facade pattern [13].

patterns catalog [11], the object-oriented re-engineering patterns catalog [9], the distributed control systems pattern language [10] and so on.

Related Patterns

Validate Applicability can be seen as a refinement of the Running Example pattern discussed by Meszaros and Doble [22]. In particular, Validate Applicability provides a more detailed discussion of the problem, the forces, and the solution.

3 SHOW GENERALITY

Context

You are writing a pattern that gives a solution to a problem in a context. You have already described the context, the problem, the forces, the solution, and the consequences. To make the pattern easier to understand you also added a detailed example that explains how to use the pattern in a particular situation.

Problem

Still, what you described as a pattern may concern only you and no one else. Others will not use your pattern unless they are convinced that the pattern provides a solution to a recurrent problem that concerns them.

Forces

There can be several reasons for which a solution to a problem in a context may not be a pattern. The context of the problem may be too narrow. The problem may not be important for others than the author of the pattern. The problem may not be recurrent or it may be very rare. The solution to the problem may be fixed, regardless of the circumstances. Even if what you have is really a pattern, the general intent and the consequences of the pattern may not be entirely clear for the readers.

Solution

In the pattern description, provide convincing evidence that shows the generality and the usefulness of the pattern. Such evidence comes right out from the pattern creation process that you employed.

Often, patterns are observed in different real-world situations (a.k.a., known uses). The usual practice is to report at least three known uses (Rule of Three). If you observed your pattern in real-world situations, provide details about the particular context, the problem instance and the solution instance that resolved the problem. You should further discuss the practical consequences in this particular situation.

Evidence about the generality and the usefulness of the pattern can be derived from other sources too, like workshops, focus groups, interviews with experts, surveys, case studies, controlled experiments and other pattern mining and scientific research methods [18, 19, 26]. If you employed such a method for creating your pattern, provide details about the overall process that you followed (meetings, workshops, interviews, participants, etc.) and report the results that you obtained.

You can provide evidence that shows the generality and the usefulness of the pattern in dedicated pattern form sections (e.g. known uses). However, if the pattern form that you use does not have such sections you can provide the evidence wherever else it seems appropriate in the pattern description.

Consequences

Providing evidence that shows the generality and the usefulness of the pattern gives confidence to you and the readers that what you have is really a pattern and not a one-shot solution to a problem.

Providing evidence that shows the generality and the recurrence of the pattern further allows the reader to understand the general intent of the pattern, witness different situations in which it can be applied, see similarities and differences between problem instances and respective solutions.

Discussing in detail known uses, expert opinions, personal experiences, case studies, empirical results, etc. can make the description of the pattern longer, more complicated and tiring for the readers.

In some cases, it may be difficult to provide evidence that shows the pattern generality for confidentiality reasons. Moreover, there may be only anecdotal evidence that shows the pattern generality.

Examples

Showing generality with known uses. To show the generality of the Facade pattern the GoF refer to three real world cases. Firstly, they refer to the Objectworks Smalltalk compiler system as the main

inspiration of the pattern. Secondly, they discuss the involvement of the pattern in the ET++ application framework. Finally, they refer to the facades used in the Choices operating system. In the excerpt that is provided below, we can see that the authors do not simply refer to the specific cases. On the contrary, they discuss in detail the usage of the pattern in these contexts.

> *"... The compiler example in the Sample Code section was inspired by the ObjectWorks Smalltalk compiler system [Par90]. ..... In the ET++ application framework [WGM88], an application can have built-in browsing tools for inspecting its objects at run-time. These browsing tools are implemented in a separate subsystem that includes a Facade class called "ProgrammingEnvironment." This facade defines operations such as InspectObject and InspectClass for accessing the browsers .... The Choices operating system [CIRM93] uses facades to compose many frameworks into one. The key abstractions in Choices are processes, storage, and address spaces. For each of these abstractions there is a corresponding subsystem ....."*

Showing generality via Iba's method. Iba's method is a well established pattern mining method [18], that involves collecting feedback and experiences from individuals about a particular problem, analyzing and clustering the collected data to discover common pattern seeds and finally specify the resulted patterns. An example of applying Iba's method concerns for the creation of patterns for improving foreign language skills when studying abroad can be found in [16]. In this effort, the authors conducted interviews to acquire practical experiences from four individuals who have studied abroad regarding important things for improving foreign language skills.

Showing generality with scientific research methods. The creation and validation of patterns via scientific research methods has been discussed in detail by Riehle et al. [26]. The idea is to assume patterns as theories that are created and validated with well established research methods like qualitative surveys, grounded theory, case studies, controlled experiments, etc. An specific example discussed in [26] concerns the use of multiple case studies to derive patterns for the design, implementation and management of user experience design in the context of product lines.

Known Uses

Showing generality with known uses. The discussion of known uses is an essential part of the description of the GoF patterns [13]. In particular, for each pattern the authors report at least two real world cases in which the pattern has been observed. In the Fluent C patterns catalog [25], each pattern is supported by at least three known uses. The authors of the distributed control systems pattern language [10] also provide at least three known uses for each pattern. In the object-oriented re-engineering patterns catalog [9] the authors refer to known uses, but this is not done for all of the patterns.

Showing generality via Iba's method. Iba and Iba [16, 17] employed Iba's method to mine patterns for improving foreign language skills when studying abroad. Yamakage et al. used Iba's method to create patterns for creative living [28], and patterns for online education [28].

Showing generality with scientific research methods. Rubem Barbosa-Hughes [5] introduced a pattern approach for the identification of opportunities for personalisation and automation of user interactions for the IoT. The author validated the generality of the proposed approach in an empirical study that involved 23 participants. Mor et al [24], reported assessment patterns for online professional development. They created and evaluated the generality of the patterns using mixed qualitative research methods like reviews, surveys and interviews. Harutyunyan and Riechle [15] introduced patterns for component approval in FLOSS governance. To this end, the authors employed a qualitative survey research method, involving interviews with industry experts.

	Published	Papers that report patterns	Other papers
2019	39	32	7
2020	35	28	7
2021	35	31	4
Corpus	109	91	18

Table 1: EuroPLoP papers 2019-2021.

4 SOME FACTS AND FIGURES

The goal of this study is to evaluate the extent to which pattern authors show the applicability and the generality of their patterns. To this end we manually inspected a corpus of papers that have been published in previous EuroPLoP conferences. Specifically, the corpus of the examined papers consists of 39 papers published in 2019 [1], 35 papers published in 2020 [2] and 35 papers published in 2021 [3] (Table 1). 18 of these papers have been excluded from the study because they do not report patterns. Instead, the excluded papers discuss issues that concern the specification of patterns, or the usage of patterns in specific contexts. Overall, we looked in these papers for examples that illustrate the use of the patters. Moreover, we looked in the papers for evidence that show the generality and the usefulness of the patterns. We searched for examples and generality evidence independently from the pattern forms that have been used by the authors to specify the patterns, i.e., we did not focus on specific pattern form sections.

Table 2, summarizes the results of the study. Specifically, the table gives the frequency of papers that show both the applicability and the generality of the patterns, the frequency of papers that illustrate at least the applicability of the patterns, the frequency of papers that show at least the generality of the patterns, and the frequency of papers that do not consider any of these properties.

The results of the study reveal the following key observations:

- Usually, about half of the papers include illustrating examples. In the overall corpus, the frequency of papers that validate at least the *applicability* of the patterns is *medium*. Over the years the frequency is quite stable ranging from 50.00% to 56.38%
- The frequency papers that show the *generality* of the patterns is *medium-high*. In particular, the frequency of the papers in the whole corpus is 79.12%. 2020 is a notable year with

	Frequency of papers that validate both the patterns' Applicability and Generality	Frequency of papers that validate at least the patterns' Applicability	Frequency of papers that validate at least the patterns' Generality	Frequency of papers without validation of the patterns
2019	50.00%	59.38%	78.13%	**9.38%**
2020	46.43%	50.00%	85.71%	10.71%
2021	40.63%	56.25%	71.88%	9.38%
All	**46.15%**	**56.04%**	**79.12%**	**9.89%**

Table 2: Frequencies of papers that show (a) applicability and generality, (b) at least applicability, (c) at least generality, (d) none of the two.

	Known Uses	Iba's Method	Scientific research methods
2019	20	2	3
2020	15	2	6
2021	19	2	2
All	**54**	**6**	**11**

Table 3: Different generality validation methods.

a 85.71% of papers that validate generality, while in 2019 and 2021 the frequency is not lower than 71.88%. Table 3 gives more details about the different kind of methods that have been used to show generality. By far, the most common method is to report known uses. Iba's method and other scientific research methods are less frequent.

- Fortunately, the papers that *do not include any systematic validation* for the patterns are *not frequent*. Nevertheless, the frequency of such papers *is not negligible*. Overtime it ranges from 9.38% to 10.71%, while overall it is 9.89%.
- The frequency of papers that validate both the applicability and the generality of the patterns is *medium*. Specifically, in the overall corpus the frequency of the papers is 46.15%, while over the years the frequency varies from 40.63% to 50.00%.

5 TAKEAWAY MESSAGE

This paper introduced two patterns that describe well established pattern validation techniques, via examples and known uses, respectively. The intent of Illustrate Applicability is to demonstrate the applicability of a pattern via a detailed example, while the idea behind Show Generality is to show the generality of the pattern with evidence that concern known uses, workshops, focus groups, interviews with experts, surveys, case studies, controlled experiments, etc.

The results of the study reported in this paper suggest that the authors actually use these validation techniques in practice. Nevertheless, the results also show that there is room for improvement. As authors, we should provide evidence of the applicability and the generality of our patterns, no matter which form we choose for writing them. If the form does not provide specific parts for examples and known uses, we can incorporate these elements in other parts of the pattern like the problem and the solution. Applicability and generality are equally important for having a valid pattern. As authors, we should assess both of these aspects when validating our patterns. As shepherds we have to insist more during the shepherding process on the validation of the patterns applicability and generality and highlight to the authors the necessity for good examples that show how to use their patterns, and convincing known uses that strengthen the foundation of their patterns. Overall, as a community (authors, shepherds, and everybody else involved in the pattern writing process) we have to give some more attention to the issue of patterns validation.

The patterns reported in this paper focus on *"what"* should the pattern authors do to illustrate the applicability and the generality of their patterns. Additionally, more detailed patterns that describe *"how"* to write good examples and known uses may also be useful. Another interesting issue for future research would be the identification of smells and threats to pattern validity. To this end, a larger corpus of existing patterns from various sources (e.g., PLoP, EuroPLoP, AsinaPLoP, VikingPLoP) should be studied. Studying a larger and more diverse corpus of papers shall further allow to investigate trends in the validation of patterns applicability and generality overtime.

ACKNOWLEDGMENTS

Many thanks to the shepherd of the paper, Rosana Teresinha Vaccare Braga for her feedback during the shepherding process. Also, many thanks to the members of the writers workshop for their valuable comments and suggestions.

REFERENCES

[1] 2019. *Proceedings of the 24th European Conference on Pattern Languages of Programs, EuroPLoP 2019, Irsee, Germany, July 3-7, 2019.* ACM.
[2] 2020. *EuroPLoP '20: European Conference on Pattern Languages of Programs 2020, Virtual Event, Germany, 1-4 July, 2020.* ACM.
[3] 2021. *EuroPLoP'21: European Conference on Pattern Languages of Programs 2021, Graz, Austria, July 7 - 11, 2021.* ACM.
[4] Christopher Alexander, Sara Ishikawa, and Murray Silverstein. 1977. *A Pattern Language: Towns, Buildings, Construction.* Oxford University Press.
[5] Rubem Barbosa-Hughes. 2019. A pattern approach for identification of opportunities for personalisation and automation of user interactions for the IoT. In *Proceedings of the 2019 European Conference on Pattern Languages of Programs (EuroPLoP).* ACM, 8:1–8:9.
[6] Frank Buschmann, Kevlin Henney, and Douglas C. Schmidt. 2007. *Pattern-oriented software architecture, 4th Edition.* Wiley.
[7] James O. Coplien. 1996. *Software Patterns.* SIGS Books.
[8] Ward Cunningham. 1994. *The CHECKS Pattern Language of Information Integrity.* c2.com/ppr/checks.html
[9] Serge Demeyer, Stephane Ducasse, and Oscar Nierstrasz. 2002. *Object-Oriented Reengineering Patterns.* Morgan Kaufmann.
[10] Veli-Pekka Eloranta, Johannes Koskinen, Marko Leppänen, and Ville Reijonen. 2014. *Designing Distributed Control Systems: A Pattern Language Approach.* Wiley.

[11] Martin Fowler. 2003. *Patterns of Enterprise Application Architecture.* Addison Wesley.
[12] Martin Fowler. 2006. Writing Software Patterns. www.martinfowler.com/articles/writingPatterns.html.
[13] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns - Elements of Reusable Object-Oriented Software.* Addison-Wesley.
[14] Neil B. Harrison. 2004. Advanced Pattern Writing Patterns for Experienced Pattern Authors. www.europlop.net/sites/default/files/files/1_2003_Harrison_AdvancedPatternWriting.pdf.
[15] Nikolay Harutyunyan and Dirk Riehle. 2020. Industry Best Practices for Component Approval in FLOSS Governance. In *Proceedings of the 2020 European Conference on Pattern Languages of Programs (EuroPLoP).* ACM, 33:1–33:12.
[16] Haruka Iba and Takashi Iba. 2019. A pattern language for improving foreign language skills when studying abroad. In *Proceedings of the 24th European Conference on Pattern Languages of Programs, EuroPLoP 2019, Irsee, Germany, July 3-7, 2019.* ACM, 13:1–13:9.
[17] Haruka Iba and Takashi Iba. 2020. Patterns for Gaining Language as Native Speakers Do: A Pattern Language for Improving Foreign Language Skills when Studying Abroad, Part 2. In *Proceedings of the 2020 European Conference on Pattern Languages of Programs (EuroPLoP).* ACM, 23:1–23:7.
[18] Takashi Iba and Taichi Isaku. 2016. A Pattern Language for Creating Pattern Languages. In *Proceedings of the 20th European Conference on Pattern Languages of Programs (EuroPLoP).* ACM.
[19] Christian Kohls and Panke Stefanie. 2009. Is that true…? - Thoughts on the epistemology of patterns. In *Proceedings of the 16th Conference on Pattern Languages of Programs (PLoP).* ACM.
[20] Mary Lynn Manns and Linda Rising. 2005. *Fearless Change: Patterns for Introducing New Ideas.* Addison-Wesley.
[21] Gerard Meszaros. 2003. *xUnit Test Patterns: Refactoring Test Code.* Addison Wesley.
[22] Gerard Meszaros and Jim Doble. 1997. *A Pattern Language for Pattern Writing.* Addison-Wesley Longman Publishing Co., Inc., 529–574.
[23] Gerard Meszaros and Jim Doble. 1997. Pattern Languages of Program Design 3. Chapter A Pattern Language for Pattern Writing, 529–574.
[24] Yishay Mor, Karen Donner-Asscher, and Jimena Pereyra. 2020. Assessment patterns for online professional development: Patterns from IIEP's Virtual Campus. In *Proceedings of the 2020 European Conference on Pattern Languages of Programs (EuroPLoP).* ACM, 32:1–32:24.
[25] Christopher Preschern. 2022. *Fluent C.* O'Reilly.
[26] Dirk Riehle, Nikolay Harutyunyan, and Ann Barcomb. 2021. Pattern Discovery and Validation Using Scientific Research Methods. arXiv 2107.06065 - To appear in Transactions of Pattern Languages of Programming V.
[27] Tim Wellhausen and Andreas Fiesser. 2012. How to Write a Pattern?: A Rough Guide for First-time Pattern Authors. In *Proceedings of the 16th European Conference on Pattern Languages of Programs (EuroPLoP).* ACM, 5:1–5:9.
[28] Misaki Yamakage, Miku Minami, Sora Hatori, Takashi Iba, and Mitsuki Saito. 2021. Natural & Creative Living Patterns, Part 1, Patterns for Creative Living: Patterns for Creative Living. In *Proceedings of the 2021 European Conference on Pattern Languages of Programs (EuroPLoP).* ACM, 25:1–25:9.