



RIIKKA KOULU, AMADEUS PETERS, JÖRG POHLE

Finding Design Patterns in Law: An Exploratory Approach

HIIG DISCUSSION PAPER SERIES

2021-03

ABSTRACT

Currently, the importance of design is recognized across society and advocated for in ethical guidelines surrounding the development of information systems. The concept of design patterns was originally introduced by design theorist Christopher Alexander, to identify and collect reusable design solutions for desired social interaction in physical spaces. The concept was then adopted by software developers to enhance problem solving through collecting functional design solutions. Our exploration is motivated by the notion of law as abstract architecture meant to regulate social interaction. We hope to transfer the concept of design patterns to law, as such analytical practice would facilitate the identification of recurring good solutions across different fields and, ultimately, the repeated practice of pattern recognition would contribute to law's proactive capabilities. We engage in an exploratory approach to describe the conceptual evolution of design patterns from architecture to software and to legal design. We discuss the process of pattern recognition as a collective iterative process and then proceed to identify three examples of legal design patterns in law. Following this exploration, we perceive design patterns particularly useful for identifying procedural structures and conclude by discussing the advantages of the approach for socio-legal research on proceduralization and automation of law.

KEYWORDS

Design patterns, legal design, legal theory, legal automation, proceduralization

AUTHOR INFO & AFFILIATION

Dr. Riikka Koulu, assistant professor (Social and Legal Implications of AI) at the Faculties of Social Sciences and Law at University of Helsinki and the director of University of Helsinki Legal Tech Lab.

Ass. iur. Amadeus Peters, PhD candidate at Justus-Liebig-Universität Gießen.

Dr. Jörg Pohle, postdoctoral researcher at Alexander von Humboldt Institute for Internet and Society, Berlin and head of research program "Data, Actors, Infrastructures".

For helpful comments and insights on earlier drafts, the authors would like to thank Julian Hölzel, Mareike Lisker, Alina Wernick, and the researchers at the University of Helsinki Legal Tech Lab for the discussions that have contributed to these ideas.

CONTENTS

1 TOWARDS DESIGN PATTERNS IN LAW?	3
2 INCARNATIONS OF DESIGN PATTERNS: FROM ARCHITECTURE TO SOFTWARE DEVELOPMENT	5
2.1 Design Patterns in Architecture and Urban Planning	5
2.2 The Diffusion of Design Patterns in Software Engineering and the Emergence of AntiPatterns	6
2.3 Design Patterns in Law, Legal Design and Normative Patterns	8
3 IDENTIFICATION OF DESIGN PATTERNS	10
3.1 Form and Functions of Design Patterns	10
3.2 Generation of Design Patterns	13
3.3 Criteria for Legal Design Patterns	15
4 FINDING DESIGN PATTERNS IN LAW	16
4.1 Example 1: De-escalation through Dialogue Design Pattern	16
4.2 Example 2: NGO Enforcement Design Pattern	19
4.3 Example 3: Moving Protection Upstream	20
5 DISCUSSION: LEGAL DESIGN PATTERNS AND PROCEDURALIZATION	23
6 CONCLUSIONS	24

1 TOWARDS DESIGN PATTERNS IN LAW?

Patterns are ubiquitous. With the current active development, artificial intelligence, machine learning and deep learning techniques have become some of the main approaches to discover repeating structures and regularities in data. These methods aim to mimic human neural networks and thus expand the concept of pattern recognition beyond its psychological meaning of a human cognitive process of matching information to memory. Recognition of patterns saves time but, more importantly, it enables us to navigate social complexity, e.g. through facial recognition and language development. In human cognition, learning to recognize patterns is a focal step towards development of problem-solving skills. This problem-solving dimension of patterns is repeated in architecture and in software engineering, where ‘design patterns’ has come to refer to reusable problem-solving models that can be employed to certain recurring classes of problems.

In this article, we examine the legal domain through the conceptual lense of design patterns. In a cross-disciplinary manner, we hope to translocate the concept as it is developed and understood in software engineering into the examination of legal structures. To this end, we perceive the legal system as a societal system implicitly focused on problem-solving activities. Although this approach is by no means new,¹ there is a lack of research on reusable legal structures that could be broadly disseminated as good solutions across different legal fields. Cumulated experiences of realizing “what works and what doesn’t” are typically not reflected upon analytically in legal scholarship, although individual lawyers might have a very good implicit understanding of repeating patterns based on their subjective experiences. Although the advantages of design patterns are often obvious to software engineers, they are not that obvious to lawyers – despite what could be perceived as inherent similarities between these domains in their problem-solving objectives. We propose a simple change of perspective: that design patterns do exist in law and that it is worthwhile to examine them as such instead of reinventing the wheel in each particular context.

But the simplicity of such change of perspective is deceptive. One does not simply borrow a concept from another field, ignorant of the field-specific knowledge-constructing interests that define how and for which purposes a borrowed concept is used in its field of origin. As is customary to most interdisciplinary research, transplanting concepts requires us to understand the theoretical, methodological and practical underpinnings in the field of origin and to ground the concept within the frameworks and practices of the field of destination – in this case, translating design patterns into the language of law. The challenge of such an approach is that it is by no means easy, calling for caution in the form of exploratory approach. Yet if successful, the awards well merit the jump – perceiving law as design may give us new insights into the internal workings of law and pave the way towards more proactive and more effective legal responses.

We argue that by enabling a better understanding of law as a problem-solving activity – and doctrinal self-reflection that ideally accompanies such a perspective – legal pattern recognition would facilitate both the identification of existing functional models and their subsequent use across legal fields. Identification of design patterns would enable transplanting model solutions that have already been tested and found functional, from contract law to administrative law, from environmental law to technology regulation, thus improving law’s problem-solving and self-reflecting capabilities. Furthermore, such an approach would enable recognition of those patterns that have been proven to be dysfunctional time and time again.² In

¹ For example, Roscoe Pound, the prominent American legal scholar of the early 20th century, famously proposed a theory of law based on social engineering, where law is seen as a societal mechanism for balancing competing interests. See e.g. William L. Grossman, *The Legal Philosophy of Roscoe Pound*, 44 YALE L.J. 605 (1935).

² Similarly see, Max Rheinstein, *Teaching Comparative Law*, 5 U CHI L REV 615, 617 (1938): “The statement that law is a means of social control and organization has almost become a commonplace. Not all of the implications of this proposition have been fully

short, a design pattern approach would save time and help complexity reduction within the legal domain. Legal scholars, practitioners and policy makers alike could benefit from the design pattern approach that would draw attention to re-usable solutions to legal and regulatory problems. This approach might prove to be a helpful heuristic tool for comparative law, which examines *inter alia* the possibilities of legal transplants between jurisdictions.³ Additionally, the design pattern approach might open up new insights into automation of legal decision-making processes by developing a shared vocabulary between legal and engineering experts.

Our objective is two-fold: first, we hope to connect and expand current and earlier research on legal design to include design patterns in the sense in which the concept is used across disciplinary boundaries. This objective contributes to methodological renewal of socio-legal scholarship. Second, we hope to elaborate a framework to enable recognition of design patterns and test this in practice through substantive examples. This objective contributes to theoretical renewal and connects the interdisciplinary research field of law, technology, and society to legal theory, teasing out the architectural similarities between law and engineering.

Our examination is motivated by the underlying theoretical framework of Niklas Luhmann's social systems theory. The framework guides our understanding of the mechanisms which the legal system utilizes to incorporate knowledge from other fields to its own operations, while maintaining its own distinct identity. At the same time, the framework motivates our objective of improving law's reflexivity through external stimuli. Hence, we do not claim there is no ontological difference between law and engineering, but we maintain that the functional similarities of the fields constitute untapped potential for collaboration.

We structure our argument as follows. In section two, we discuss in further detail the origins of design patterns and how the concept is used within software engineering. Following this, we briefly examine existing socio-legal literature on legal design and approaches, which are also referred to as "design patterns", focusing mostly on what we consider the most concrete, or closest-to-text level of patterns, i.e. reusable contractual clauses and applications of design thinking in the development of legal services. Here, the objective is to provide an overview of the related work on grassroots level as well as to point out the research gap on mid-level architectures of law we aim to remedy. In section three, we demonstrate how design patterns approach might play out in reinterpreting existing problem-solving structures in law, drawing from concrete examples of patterns we identify as *De-escalation through Dialogue*, *NGO Enforcement Design Pattern* and *Moving Protection Upstream*. Subsequently, in section four, we elaborate in

realized, however. For legal science, it implies the task of inquiring into the social function of every legal rule and institution. It means that nothing in the realm of law can be taken for granted, that every rule and institution has to justify its existence under two inquiries: First: What function does it serve in present society? Second: Does it serve this function well or would another rule serve better?" Regarding legal disciplines it should be noted that each legal field is defined by its own focal concepts, principles and protected goods or interests, which sometimes might limit transplantation across sub-disciplinary boundaries. It should be noted, however, that strict boundaries between legal sub-disciplines are perceived as artificial divisions, constructed mostly for heuristic purposes in legal education.

³ For example, Professor Frankenberg describes constitutional transfer, i.e. how globalization has facilitated translocation of constitutional norms, constructs, and institutions from local contexts into "global constitution", where regardless of jurisdiction most constitutional documents have adopted the same basic solutions. See, Günter Frankenberg, *Constitutional transfer: The IKEA theory revisited*, 8 I CON, 563 (2010). Similarly, UNCITRAL's model laws can be construed as an attempt to disseminate best regulatory practices across jurisdictions. However, more in-depth analysis of similarities between design patterns and comparative law methods are beyond the scope of this article, due to space limitations and the article's objective. Our examination is focused on elaborating the conceptual package related to design patterns and to develop interdisciplinary methodology across legal fields and between engineering and law, whereas comparative law tradition has developed more nuanced understanding of the implications and limitations of cross-jurisdictional legal transplants.

further detail, what the proposed design pattern approach would entail, followed by concluding remarks about the advantages and shortcomings of design pattern approach to law.

2 INCARNATIONS OF DESIGN PATTERNS: FROM ARCHITECTURE TO SOFTWARE DEVELOPMENT

In order to understand what is meant by design patterns, it is necessary to look into the concept's origins in urban planning and to explore how the concept was defined to consist of a context, a problem and a solution, and how it was adopted by and adapted to the field of software development. In the following section, we shed light to the origins and developments of the design pattern concept to demonstrate its objectives and theoretical underpinnings.

2.1 Design Patterns in Architecture and Urban Planning

The concept of design patterns was introduced in the 1970s by architect and urban design theorist Christopher Alexander. He explored the idea that the success of architecture and urban planning in creating “living” buildings and towns – in contrast to what Alexander calls “dead” buildings and towns – depends upon the application of patterns of good design that have been around for thousands of years, though not recorded. In *The Timeless Way of Building*,⁴ the first of three parts,⁵ Alexander called for a pattern language as a means for describing such good design practices for architecture, urban design, community livability, and liveliness. He observed that things that are different in detail might become similar in general outline, that there is always repetition of patterns and that the number of the patterns out of which a building or a town is made is rather small.⁶ In designing buildings and towns, these patterns can then be adapted to specific contexts, varied, and combined and recombined to make an infinite variety of unique structures.⁷

In order to be able to talk about patterns, and not only consciously or unconsciously use them, they must be made explicit. For this purpose, Alexander proposes a pattern description structure composed of three related parts: 1) a certain context, 2) a problem, which is defined by conflicting forces that have to be resolved, and 3) a solution:⁸ “Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”⁹ In Alexander's understanding, the patterns are connected, they form a network, and it is this what he then calls a language:¹⁰ “Each pattern is connected to certain ‘larger’ patterns which come above it in the language; and to certain ‘smaller’ patterns which come below in the language. The pattern helps to complete those larger patterns which are ‘above’ it, and is itself completed by those smaller patterns which are ‘below’ it.”¹¹ Patterns thus can exist at all scales. It is both this scalability of patterns as well as the possibility of pattern-within-a-pattern structure that helps to prevent pattern users from applying patterns in a mechanical fashion. Instead, even after a pattern has been applied to solve a specific problem, the solution is still to a large extent malleable. Yet the structure of pattern description is a fundamental part of finding a pattern,

⁴ Christopher Alexander, *The Timeless Way of Building* (1979).

⁵ The series was published out of order, though, with volume 3 coming out in 1975, volume 2 in 1977, and volume 1 in 1979. Christopher Alexander et al., *A Pattern Language: Towns, Buildings, Construction* (1977 [hereinafter Alexander et al. 1977]); Christopher Alexander et al., *The Oregon Experiment* (1975).

⁶ Alexander, *supra* note 4, at 98.

⁷ Id. at 179.

⁸ Id. at 246–247.

⁹ Alexander et al. 1977, *supra* note 5, at x.

¹⁰ A pattern language thus resembles a “mental model” (Ludwig Wittgenstein), a “denkstil” of a “denkkollektiv” (Ludwig Fleck), an “epistemic culture” (Karin Knorr-Cetina), or an “epistemic regime” (Peter Wehling).

¹¹ Alexander et al. 1977, *supra* note 5, at xii.

and should also guide the identification of legal design patterns. But how then did Alexander identify patterns?

Alexander's theory of reoccurring architectural solutions can be exemplified through a mid-level pattern called *Small Public Squares*.¹² Here, Alexander et al. refer to empirical research that has shown that many squares built in modern cities are too large and end up "desolate and dead". They identify two forces associated with this problem: first, that there is a certain pedestrian density threshold for a place to seem deserted, and second, that human capacity to recognize faces and hear voices beyond some 25 to 30 meters is very limited. Based on this reflection and the abundant number of lively small squares in cities and towns from all times, they suggest that a public square should be much smaller than usually imagined or built, at no more than 15 to 25 meters across in the short direction. They then connect this design pattern with other patterns in the language: such a public square could serve as an *Activity Node*, a design pattern "above in the language", with *Activity Pockets* around the square where people can congregate and *Something Roughly in the Middle* to make the center of the square as useful as the edges, both referring to design patterns "below in the language". In this sense, hierarchical pattern languages also clarify the reasons and values behind certain design choices.

Design patterns are thus about making the implicit explicit, including the assumptions and the underlying rationale for their application in a given context. They are found in practice and defined largely intuitively.¹³ There is no one right answer but different design patterns may frame a problem in different ways, e.g. with different significance assigned to different competing forces, and thus propose different solutions. Hence, design patterns reflect disciplinary perspectives, paradigms and blind spots, cognitive schemas and existing institutions as well as value-laden decisions embedded in the problem presentations and proposed solutions.¹⁴ In architecture and urban planning, even high-level patterns are very tangible and can be understood easily through their material realization, as they exist in and depend on the very physicality of the world, whereas the more abstract architectures of law and software defy such easy recognition. Then again, neither in developing software nor in creating or applying law, one is tightly bound by the very materiality of the physical world. While this allows for much more flexible application of a design pattern, it also makes it harder to navigate. How then can the concept of design pattern be translocated into such abstract architectures?

2.2 The Diffusion of Design Patterns in Software Engineering and the Emergence of AntiPatterns

The shift of design patterns from tangible physical architecture to abstract mental architecture was undertaken by software development. As architecture detached from physical limitations of built environments, the importance of context became a decisive factor for the recognition of design patterns. In this section, we describe how the software engineers developed the concept further and how they introduced the distinction between functional and dysfunctional patterns that hoped to address the value-laden implications of calling something a pattern.

Alexander's hope that others would improve patterns,¹⁵ was heard and taken up in a field that had just recently rediscovered the "architecture" leitmotif, albeit in more abstract than material sense: the software

¹² Id. at 310–314. Mid-level here illustrates that it lies approximately in the middle between high-level, e.g. *region*, and low-level, e.g. *room*, design patterns.

¹³ See Alexander, *supra* note 4, at 246–276. For a systematic general approach, especially regarding the "neo-pragmatic resolution" ("Beschluss"), see Herbert Stachowiak, *Allgemeine Modelltheorie*, 50–56 (1973).

¹⁴ This even includes the choice of programming language because it influences one's point of view, see Erich Gamma et al., *Design Patterns: Elements of Reusable Object-Oriented Software* 4 (1994).

¹⁵ Alexander et al. 1977, *supra* note 5, at xv.

development community.¹⁶ The field applied the concept of pattern language to document design decisions taken in software development.

One of the decisive moments in the adoption of the pattern concept can be traced back to a presentation given in 1987 by Kent Beck, a software engineer and the creator of the extreme programming (XP) paradigm, and Ward Cunningham, a programmer and the inventor of the wiki. They introduced the notion of a design pattern language for developing user interfaces in the Smalltalk programming language¹⁷ that sparked great interest. This pattern language was then supplemented by the Portland Pattern Repository (PPR), a repository for software design patterns with a companion website, WikiWikiWeb – the world's first wiki, developed by Cunningham. Together, these developments created an early design pattern movement, which in 1994 led to the now-classic text on software design patterns by the so-called “Gang of Four”, *Design Patterns: Elements of Reusable Object-Oriented Software*.¹⁸ The authors had to concede, however, that from Alexander’s point of view, their book but presented a collection of related patterns, not a “complete” set of patterns, which thus do not form a pattern language.¹⁹ In the same year, the concept of design patterns was first applied to organizational analysis.²⁰ It has since spread to other fields, e.g. security, privacy analysis and design, human-computer interaction design and business process design, as well as taken up by other disciplines, e.g. pedagogy or communication studies, with the design pattern literature ever growing.²¹

The increasing application of design patterns in software development projects in the 1990s soon revealed problems related to context-dependency. Users of design patterns applied them outside the contexts they were originally designed for, and developers started classifying “everything as a design pattern or solvable by a specific set of design patterns before attempting to perform and complete their domain analysis”²², i.e. before they analyzed the problems in the very contexts or surroundings in which they occurred. The concept of AntiPatterns²³ thus emerged as the antithesis of design patterns to denote commonly proposed solutions that generate decidedly negative consequences.²⁴ While the pattern description structure contains a problem and a solution, with the problem usually being elaborated in terms of its context and applicable design forces, and the role of the solution being to resolve the design forces in a beneficial way, the essence of an AntiPattern is *two* solutions. The first solution presented is problematic as it generates overwhelmingly negative consequences. Thus, a second solution is added, which is called a ‘refactored

¹⁶ See generally Frederick P. Brooks, Jr., *No Silver Bullet. Essence and Accidents of Software Engineering*, 20 IEEE COMPUTER 10 (1987). For an overview of competing leitmotifs in the history of software development and the reemerging of the “architecture” leitmotif in the 1980s, see Hans Dieter Hellige, *Software Manufaktur – Software Engineering – Software Architektur: konkurrierende Leitbilder in der Geschichte der Softwaretechnik*, Artec working paper no. 221 (2019), https://www.uni-bremen.de/fileadmin/user_upload/sites/artec/Publikationen/artec_Paper/221_paper.pdf.

¹⁷ Kent Beck and Ward Cunningham, *Using Pattern Languages for Object-Oriented Programs*. Technical Report No. CR-87-43 (1987). Presented at the ACM OOPSLA-87 workshop on the Specification and Design for Object-Oriented Programming, <https://c2.com/doc/oopsla87.html>.

¹⁸ See generally Gamma et al., *supra* note 14.

¹⁹ Id. at 357.

²⁰ James O. Coplien, *A Development Process Generative Pattern Language* (1994), presented at the 1994 Pattern Languages of Program Design (PLoP) conference in Monticello, Illinois, <https://pdfs.semanticscholar.org/04d9/23750e0beea90c3ec84082ad4bcc1ac690b0.pdf>.

²¹ See closer section 2.3.

²² See generally William H. Brown et al. *Antipatterns: Refactoring Software, Architectures, and Projects in Crisis* 7 (1998).

²³ Not to be confused with the concept of “dark patterns”, which refer to deceptive user interfaces, which have been crafted to trick users into doing things they wouldn’t do if they were to be asked for an informed decision, see Harry Brignull, *Dark Patterns: Deception vs. Honesty in UI Design*, in 338 A LIST APART (2011), <https://alistapart.com/article/dark-patterns-deception-vs.-honesty-in-ui-design/>.

²⁴ The term has been first formally defined by Michael Akroyd, *AntiPatterns: Vaccinations against Object Misuse*, in Proceedings of the 1996 Object World West Conference (1996), though both discussions on dysfunctional design patterns and the term itself clearly predate Akroyd’s presentation.

solution', with a commonly occurring method in which the AntiPattern can be resolved and reengineered into a more beneficial form.²⁵

This leads us to the following: the distinction between a design pattern and an AntiPattern is not clear-cut. The applicability of a design pattern to a specific problem in a specific real-world context essentially depends on the matching of this real-world context with the design pattern's description of a context and a problem. This means the former's successful subsumption under the latter. This matching process is riddled with paradigmatic, cognitive and value-related challenges, such as the problem presentations and the proposed solutions themselves.²⁶ While examining the design pattern/AntiPattern distinction helps in shedding light on the criteria for assessing the quality of design patterns, the term AntiPattern should be used cautiously as it may easily slide into a strategic or even pejorative use. Yet the distinction may serve a purpose within the legal community, as it denotes the descriptive and normative dimensions embedded already in the concept of design patterns.

2.3 Design Patterns in Law, Legal Design and Normative Patterns

The concept of legal design patterns is not completely novel. Design patterns have, in fact, entered the legal realm through the back door, when computer scientists started reflecting law as one of the decisive forces of a contextual problem that was to be solved. The first field in software engineering where law – usually in the form of legal requirements – and design patterns met, was security, privacy and data protection engineering,²⁷ soon to be followed by the IP law and licensing practices' influence on design decisions,²⁸ and contract design.²⁹ For the time being, these discussions remain mostly in the margins of current socio-legal scholarship.

The concept and the function of design patterns should not be confused with “legal design”, although the terminology is often mixed up, especially in the legal tech field. Legal design originates in design thinking and focuses on the design of the design processes themselves, e.g. by employing “agile” methods or through participatory design approaches.³⁰ In addition, there is a second conflation of design patterns with “legal design”, where the latter refers to information or communication design, i.e. the presentation of law, legal concepts or provisions to users or affected people in a more accessible form,³¹ while in Alexander's as well as in the engineers' and software developers' understanding “design” relates to the very structure of what is to be built or created. As a consequence, the examples presented by, e.g., Rossi et al.³² are not design patterns in the sense envisioned by Christopher Alexander, neither in substance nor in their form. In this sense, the emphasis legal design places on information design and particularly on visualization techniques may also be interpreted as a counter-reaction to the customary representation of law as a text-based practice. This may

²⁵ Brown et al., *supra* note 22, at 9.

²⁶ On this problem, see generally Ian I. Mitroff and Tom R. Featheringham, *On Systematic Problem Solving and the Error of the Third Kind*, 19 Behavioural Science 383 (1974).

²⁷ See generally Luca Compagna et al., *How to capture, model, and verify the knowledge of legal, security, and privacy experts: a pattern-based approach*, in ICAIL '07: Proceedings of the 11th International Conference on Artificial Intelligence and Law 149–153 (2007). Luca Compagna et al., *How to integrate legal requirements into a requirements engineering methodology for the development of security and privacy patterns*, 17 Artificial Intelligence and Law 1 (2009).

²⁸ Imed Hammouda et al. (2010), *Open source legality patterns: architectural design decisions motivated by legal concerns*, in Proceedings of the 14th International Academic Mindtrek Conference: Envisioning Future Media Environments 207–214 (2010).

²⁹ See generally Erik F. Gerding, *Contract as Pattern Language*, 88 Washington Law Review 1323 (2013); Helena Haapio and Margaret Hagan, *Design Patterns for Contracts*, in Networks. Proceedings of the 19th International Legal Informatics Symposium IRIS 2016 381 (Erich Schweighofer et al. eds., 2016).

³⁰ See, e.g., Margaret Hagan, *Law By Design* (2017), <http://www.lawbydesign.co>

³¹ E.g. Arianna Rossi et al., *Legal Design Patterns: Towards a New Language for Legal Information Design*, in Internet of Things. Proceedings of the 22nd International Legal Informatics Symposium IRIS 2019 517 (Erich Schweighofer et al. eds., 2019).

³² *Id.* at 520–522.

have implications also on the identification of design patterns in law in comparison to architecture, where visual information is central to design, or to software development, which can incorporate text, visual information and programming languages. Yet in Alexander's sense, design patterns are ideas rather than their textual descriptions, which accentuates the need to adopt a more abstract than textual approach in order to identify potential patterns.

At the same time, there are similarities between current legal design work and design patterns in law regarding their functions: both repositories of contractual clauses and design patterns serve the goals of recognizability and reusability. In this sense, reusable contractual text snippets contribute to an increased understandability between contracting parties. And information design can be perceived to carry this objective as well, though primarily for lay people and by replacing textual presentations with visualizations. The added value of such repositories for legal practitioners is clearly visible.

There are also other legal concepts that resemble legal design patterns. A strain of Nordic legal thought seems to suggest a similar, more abstract approach to legal design patterns, namely the theory of *law as normative patterns in a normative field* developed in the early 1990s by the late Swedish Professor of private law Anna Christensen.³³ The theory aims to describe long-term legal evolution in the social dimension and to organize the chaotic multitude of legal norms through the identification of crucial normative patterns that underlie the legal system. Thus, Christensen's theory is about the creation of coherence by uncovering values below the surface level of legislation, case-law and legal doctrine. Building on theoretical understanding of the "functional relationship between law and the structure of society" and an analysis of the different areas of private law,³⁴ Christensen describes basic normative patterns that "resurface now and again in different connections, and yet maintaining a certain degree of constancy".³⁵ She distinguishes normative and functional patterns, a difference she exemplifies with principles such as "pacta sunt servanda" and "prevention", the first being "arisen from and responses to a pattern of social behaviors which is found to be morally binding", whereas the latter having a more functional and technical character.³⁶ In Christensen's theory, legal principles provide a gate to normative patterns, which reflect the competing interests, conflicts and tensions underlying them. Christensen's normative patterns do not form a hierarchy nor provide for a coherent system,³⁷ or more accurately, as Kaarlo Tuori describes in later analysis, normative patterns produce coherence only locally.³⁸

Anna Christensen's theory and the Nordic socio-legal research building on it provide important theoretical insights and notions that also lend support to our analysis. For our inquiry the design element of pattern recognition is decisive, as it connects the identification of patterns to their later reuse in the hopes of improving law's problem-solving. In this sense, we complement Christensen's approach by making the connection with legal patterns and architectural design.

In addition, some scholars use the metaphor of law as architecture to describe the artificial nature of legal concepts, objects and places.³⁹ Such comparisons have been voiced in relation to document writing

³³ See generally Ann Numhauser-Henning, *Understanding Law as Normative Patterns in a Normative Field*, in *Normative Patterns and Legal Developments in the Social Dimension of the EU* 27 (Ann Numhauser-Henning and Mia Rönmar eds., 2013).

³⁴ *Id.* at 31.

³⁵ See generally Anna Christensen, *Protection of the Established Position: A Basic Normative Pattern*, 40 *Scandinavian Studies in Law* 285 (2000).

³⁶ *Id.* at 289.

³⁷ *Id.* at 291–292.

³⁸ Kaarlo Tuori, *Law Beyond Law*, in *Normative Patterns and Legal Developments in the Social Dimension of the EU* 53, 55, 64 (Ann Numhauser-Henning and Mia Rönmar eds., 2013).

³⁹ Adolfo Giuliani, *Law, architecture and design* (2019), <https://blogs.helsinki.fi/adolfogiuliani/2019/09/11/law-architecture-and-design/>.

techniques,⁴⁰ trial and court design⁴¹ as well as to the impact courthouse architecture plays on access to justice.⁴² Although these approaches may advocate on the growing importance of design choices, they do not include suggestions for potential design solutions.

In conclusion, most existing legal approaches to design patterns or architectural design problems fall short of what this contribution aims for, i.e. the broader application of the design pattern concept to law itself, and making it fruitful for legal theory, especially to law-making, or in other words, logistics⁴³ or jurisprudence.⁴⁴

3 IDENTIFICATION OF DESIGN PATTERNS

As discussed, design patterns are heuristic tools that facilitate problem-solving. Their quality as tools can only be assessed with regards to their specific function in the process of problem-solving, with different functions demanding different forms.⁴⁵ For example, if the design pattern is geared towards recognition of a certain situation, the focus should be on the problem description. If the design pattern hopes to alleviate problem-solving, the focus should be on detailed description of that solution, including possible design options or implementation-specific pitfalls. In order to propose a suitable form of design patterns in law as well as a process to identify or generate such patterns, we deem it necessary to shed light on existing debates and experiences regarding the form and the identification or generation of design patterns in fields other than law. Simply put, in this section we hope to elaborate how patterns come about intuitively and how the subjectivity involved in this process of recognition entails that the identification and description of a pattern actually coincide. This merger of form and function helps us understand the methodological challenge of identifying design patterns in law.

3.1 Form and Functions of Design Patterns

The existing literature in both architecture and software engineering has identified a number of different functions that may be performed by design patterns, the most important being description, generalization, rhetoric, teaching, prescription and prediction.⁴⁶ These focal functions can be synthesized as follows:

- *Description*: design patterns can be used to describe the key characteristics of a place, situation or event, in a context-sensitive way, e.g. in order to serve as organizational or epistemic community's memory.

⁴⁰ See generally Jill J. Ramsfield, *Law as Architecture: Building Legal Documents* (2000).

⁴¹ See generally Linda Mulcahy, *Legal Architecture: Justice due Process and Place of Law* (2010).

⁴² See generally Fabien Gélinas et al., *Foundations of Civil Justice: Towards a Value-Based Framework for Reform* (2015). And since long, such ideas are followed very actively, see e.g. *Handbuch der Rechtsförmlichkeit* [Manual for Drafting Legislation] by the German Federal Ministry of Justice:

https://www.bmjbv.de/SharedDocs/Downloads/DE/Themen/RechtsdurchsetzungUndBuerokratieabbau/HandbuchDerRechtsfoermlichkeit_eng.pdf.

⁴³ See generally Dominique Remy, *Légistique – l'art de faire les lois* (1994); Ulrich Karpen, *Gesetzungslehre – neu evaluiert* [Legistics – freshly evaluated] (2nd edition 2008).

⁴⁴ See generally *Legisprudence – A new Theoretical Approach to Legislation* (Luc Wintgens ed., 2002).

⁴⁵ For an overview of how and how far different formats, including abbreviated patterns, affect the use of patterns in practical design settings, cf. Janet Finlay et al., *Pattern Languages in Participatory Design*, in *People and Computers XVI – Memorable yet Invisible*. Proceedings of HCI 2002 159 (Kristine Faulkner et al. eds., 2002).

⁴⁶ For descriptions of individual functions see, Gamma et al., *supra* note 14, at 3; Brendan G. Cain et al., *Social patterns in productive software development organizations*, 2 *Annals of Software Engineering* 259, 282 (1996); Marshall P. Cline, *The pros and cons of adopting and applying design patterns in the real world*, in 39 *Communications of the ACM* 47 (1996). Elisabeth Bayle et al., *Putting It All Together. Towards a Pattern Language for Interaction Design: A CHI 97 Workshop*, in 30 *SIGCHI Bulletin* 17, 19 (1998); Jan O. Borchers, *Interaction design patterns: twelve theses, Position paper presented at the CHI2000 Patterns Workshop*, The Hague (2000). Andy Dearden and Janet Finley, *Pattern Languages in HCI: A Critical Review*, in 21 *Human-Computer Interaction* 49, 75–80 (2006).

- *Generalization*: design patterns can be used to support generalizing across places, situations and events, while retaining an element of concreteness.
- *Rhetoric*: design patterns can be used as a *lingua franca*, a common way of talking about design issues that is accessible to analysts and designers (of whatever disciplinary backgrounds), the users as well as those affected by the situation. For their rather formal nature, they can serve as a technical lexicon.⁴⁷
- *Teaching/learning*: design patterns, especially pattern languages, provide a structured approach to learn the essential concepts for designing successful, i.e. user-friendly, reusable and maintainable systems – or, in Alexander’s words, “living” systems with “a quality without a name”.
- *Prescription*: design patterns can be used to prescribe solutions to commonly encountered problems in a particular design domain. Thus, patterns might be used as a way of presenting guidelines for technical, organizational or socio-technical design.
- *Prediction*: design patterns can provide a ‘what-if’ mechanism for reflecting on the possible impact of changes to a place, a situation or an event. Design patterns thus allow for assessing design alternatives to as well as trade-offs and limitations of particular solutions in a place, a situation or an event.

These synthesized functions describe the added value of design patterns, which serve several purposes from context-sensitive description to generalization and from teaching to prescribing solutions. For our exploratory approach, the emphasis is placed on this balancing between description and generalization, as we perceive these functions central to our objective of developing a methodological framework for identifying design patterns in law.

Yet it is important to emphasize the inherent limitations of design patterns that have been identified in earlier research. These limitations simultaneously reflect the advantages described above, among them their contextuality, their value-ladenness, a need for a material basis as well as the non-predictiveness of design patterns’ organization:⁴⁸

- *Contextuality*: a design pattern is always grounded in a domain. It has no meaning outside the domain to which it applies.
- *Value-ladenness*: design patterns explicitly reflect design values and are thus not neutral. Recording and selecting patterns are value-laden activities that reflect the assumptions, goals, priorities and rationales of the recording and selecting actor.
- *Materiality*: a prescriptive pattern is always manifest in a solution; a descriptive pattern’s existence is proven by its recurring, identifiable presence in artifacts. Design patterns thus capture design practice, not theoretical or conceptual thinking.
- *Non-predictiveness*: the organization of design patterns in architecture and software development – and most probably in law as well – is not predictive, in the sense that it could predict characteristics of yet unknown elements by their position in the structure. This is in stark contrast to other domains, such as chemistry, where the organizing structure, the periodic table, laid bare the ‘holes’ that were gradually filled by previously unknown elements and at the same time facilitated these elements’ discovery.

The argument on what are the necessary components of design patterns already started with Alexander’s work. The list he proposed included a name, a certain context, a problem, which is defined by conflicting

⁴⁷ For comparison, see S.L. Star, *The structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving*, in Distributed Artificial Intelligence 37 (Morgan Kauffmann and Susan Leigh Star eds., 2010).

⁴⁸ See generally Tiffany Winn and Paul Calder, *Is this a pattern?*, in 19 IEEE Software 59 (2006). Dearden and Finley, *supra* note 46, at 60–61.

forces that have to be resolved, and a solution.⁴⁹ But the examples of design patterns he provides to the reader are much more encompassing: “A meaningful, concise *name* identifies the pattern, a *ranking* indicates the validity of the pattern, a *picture* gives a ‘sensitizing’ and easily understood example of the pattern applied, and the *context* explains which larger patterns it helps to implement. Next, a short *problem statement* summarizes the competing ‘forces’, or design trade-offs, and a more extensive problem description gives empirical background information and shows existing solutions. The subsequent *solution* is the central pattern component. It generalizes the examples into a clear, but generic set of instructions that can be applied in varying situations. A diagram describes this solution and its constituents graphically, and *references* point the reader to smaller patterns that can be used to implement this pattern.”⁵⁰ A similar inconsistency can be noticed in the software engineering debate, though they all agree that concrete examples are very important, “because they help you see the design in action.”⁵¹

The conflict over the character of the examples is particularly illustrative of the value-laden nature of design patterns, and at the core of this debate is the fundamental question of who are the ultimate users, the target audience of patterns: it was Alexander’s explicit goal to enable inhabitants, i.e. users, themselves to design their environments.⁵² The examples provided by, e.g., Gamma et al. serve a fundamentally different purpose and address a different audience: they are for developers only, to convey practical expert knowledge to fellow practitioners. The underlying tension is – and this seems to be particularly important for any reflection on design patterns in law – the degree of directness of the connection between the design work and the user experience: in urban design and architecture, the connection is rather direct, but not so in software engineering. In human-computer interaction (HCI), there are much more commonalities with architecture than with software engineering in this regard,⁵³ and thus a stronger focus on examples that are comprehensible for users – and more generally, a form of design patterns that allows for user participation in the design work.⁵⁴

The issue of the assumed target audience is connected with the expected level of the scientific evidence and rigor needed for a design pattern to be accepted within that community. The epistemological foundations acceptable for practitioners, or even for stakeholders, do not necessarily fulfill the expectations of scientific communities. Design patterns are ultimately tools to facilitate abstract reflection behind design choices. The struggle over whether, where and how to include information regarding the motivation, the rationale as well as the empirical basis is another illustrative point for the form’s dependency on the design patterns’ functions. While design patterns can serve a descriptive function, they often go well beyond that and thus must allow for providing “a window on the thought processes behind choosing the solution”.⁵⁵ And that’s what information on the design pattern’s motivation and rationale is for. It helps designers make better design decisions by teaching them the reasoning behind the how and why a design pattern got accepted as a good or successful one.⁵⁶ It helps them bridge different abstraction levels and think about both the problem and its solution, understand and balance conflicting forces, and adapt designs in the face of

⁴⁹ Alexander, *supra* note 4, at 247.

⁵⁰ Jan O. Borchers, *A pattern approach to interaction design*, in 15 *AI & Society* 359–376, 361 (2001).

⁵¹ Gamma et al., *supra* note 14, at 6–7.

⁵² The other two among the three related criteria that Alexander saw as central are generative design and life-enhancing outcomes, Finlay et al., *supra* note 45, at 161. In his keynote address at the 1996 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications (OOPSLA ’96) in San José, California, Alexander heavily criticised the software development community for having a too narrow focus on the communication and reuse of design patterns within a professional community.

⁵³ Borchers, *supra* note 50, at 1.

⁵⁴ At the same time, HCI scholars have to admit that their own empirical user research suggests that users do not find the Alexandrian format particularly useful, cf. Janet Finlay et al., *supra* note 45, at 172.

⁵⁵ See generally Gerard Meszaros and Jim Doble, *A pattern language for pattern writing*, in *Pattern Languages of Program Design* 3, 529–574, 529 (Robert C. Martin et al. eds., 1997).

⁵⁶ See Dearden and Finley, *supra* note 46, at 78–80.

change.⁵⁷ This struggle also reflects different disciplines' epistemological backgrounds, and is thus especially instructive for the debate in legal scholarship.

Simply put, design patterns are inherently subjective. What then are the consequences when approaching design patterns in law from a normative perspective? Here, Christensen's theory paves the way by building on law's polycentricity and lack of hierarchy, connected with an understanding of the legal system as "a medium where different moral and legal patterns appear and dissolve, float, clash and form new patterns."⁵⁸ In this sense, patterns are subjective only to the extent the exact formulation of the problem and the conflicting forces take. In any case, as patterns reside in and can be accessed through the surface level of legal materials and doctrinal thought, they remain connected with the legal system's rationale regardless of the subjectivity of the identification process. One could say, the existing patterns are rendered visible through legal practices that interpret this material repeatedly over time.

To conclude, there is no uniform definition of what exactly constitutes a design pattern, although most authors agree on the basic elements of name, context, problem description, conflicting forces and a solution. The literature in software development and human-computer interaction perceives patterns as descriptions of "design in action"⁵⁹ and emphasizes that patterns are not neutral as such but do in fact reflect design values, which become apparent through the pattern's rationale.⁶⁰ Patterns should be formulated in general terms to facilitate the reuse of successful patterns but at the same time this reusability is lost if patterns are formulated too abstractly. Hence, the context of a pattern becomes a central tool for establishing the boundaries of a given pattern. For the purposes of our examination, an exploratory approach to identifying legal design patterns beyond reusable fragments of legal text, we focus on these uncontested basic elements of context, problem, conflicting forces, rationale and the solution, complemented with concrete examples. It should be noted, however, that the conceptual ambiguity of design patterns, although it may render the identification process more difficult, also provides interpretative flexibility necessary for such an exploratory endeavor.

3.2 Generation of Design Patterns

How then does one go about identifying or generating design patterns in abstract architectures such as law? In software development, the generative process is often framed in terms of collaborative discursive practice that taps into the participants' practical experiences and intuition to locate the functional solutions that already exist. In this sense, patterns are rather "found" than created, as the objective is to observe "design in action" as discussed in the previous section. This language of discovery and observation is repeated in software development, which highlights the practical orientation, i.e. patterns are derived from practice rather than theory and grounded in everyday experience.⁶¹ Some authors use the metaphor of "mining"

⁵⁷ Winn and Calder, *supra* note 48, at 60–62.

⁵⁸ Numhauser-Henning, *supra* note 33, at 28.

⁵⁹ Gamma et al., *supra* note 14, at 6.

⁶⁰ Dearden and Finley, *supra* note 46, at 61.

⁶¹ For example, Peter Coad suggested that "patterns are found by trial and error and by observation", Peter Coad, *Object-oriented patterns*, in 35 Communications of the ACM 152, 153 (1992). And he and Mark Mayfield discussed "discovering" patterns from experience, Peter Coad and Mark Mayfield, *Addendum to the proceedings of OOPSLA '92*, in 4 ACM SIGPLAN OOPS MESSENGER (1993). Especially the early design pattern community, which dissociated itself from the object oriented software community in the mid-1990s, pursued a rather aggressive line against theory-based design pattern generation: "Original ideas (and particularly those that emerged from non-practicing disciplines such as academics) by definition had no record of accomplishment. [...] Instead of originality, the community embraced experience: experience that was embodied in true expertise", James O. Coplien, *The Culture of Patterns*, in Computer Science and Information Systems 1, 6 (2004). For a counter-argument based on an understanding of patterns as theories from a constructivist epistemic perspective, see Christian Kohls, *The Theories of Design Patterns and their Practical Implications exemplified for e-Learning*, PhD dissertation 129–136 (2013), especially p. 135: "It is important to point out, once again,

patterns from existing designs,⁶² a formulation that has later on been adopted more broadly,⁶³ whereas others clearly base their patterns on observations of common design solutions.⁶⁴

Hence, pattern mining starts with identification of good practice. Patterns are not intended to state obvious solutions to trivial problems or to cover every possible design decision, but to capture “big ideas”.⁶⁵ A pattern should capture insights about the design that can inform even an experienced designer; explaining not only *how* a problem can be solved but also *why* a design choice is appropriate to a particular context.

It is important to notice that there is a temporal quality in the generation process. From a descriptive point of view, the human-made structure of the world, including law, is composed of “some collection of patterns which keeps on repeating over and over and over again.”⁶⁶ Repetition over time renders patterns more easily observable and recognizable,⁶⁷ with the repetition being predictable because of the patterns being mental models⁶⁸ and part of an epistemic regime.⁶⁹ This being said, patterns should also embody a timeless quality, presenting a solution that is applicable regardless of particular platform or current technology.⁷⁰ In other words, repetition over time helps identify existing good practices but the formulation as a design pattern needs to detach from the observational temporal context.

Against this backdrop, we deem it plausible to hypothesize that the combination of one perceiving “design in action” with the very recurrence of good solutions in practice may lead to procedural legal structures being easier to identify as design patterns than substantive legal structures.⁷¹ At the same time, we must admit that the intuitiveness embedded in the identification of design patterns may be at odds with law’s self-image of being rather anti-intuitive, although we can always argue that intuitive problem-solving is an apt description of what lawyers generally do in practice.⁷² This orientation to practice reconnects the identification of legal design patterns with the established practices used in the software community.

In the software patterns community, a specific practice of “writers’ workshops” and “shepherding” has evolved to support the development of patterns and pattern languages,⁷³ with both workshop and author

that the generalization from single cases, the reasoning about causalities of working forms and the judgement between relevant and irrelevant features is speculative. [...] As such, patterns are theories of good practices.”

⁶² Gerard Meszaros (1995), *Workshop Summary: Patterns for Decision Making in Architectural Design*, in Addendum to the Proceedings of '95, 6 ACM SIGPLAN OOPS MESSENGER 132, 133 (1995); Martijn van Welie and Gerrit C. van der Veer, *Pattern Languages in Interaction Design: Structure and Organization*, in Proceedings of the IFIP TC13 Conference on Human-Computer Interaction (INTERACT '03) 527, 532 (Matthias Rauterberg et al. eds., 2003).

⁶³ See Dearden and Finley, *supra* note 46, at 63.

⁶⁴ Many of the patterns offered by Jennifer Tidwell, *Interaction design patterns*, in PLOP'98 Conference on Pattern Language of Programming (1998); and in Martijn van Welie's pattern library (<http://www.welie.com/patterns/index.php>, retrieved February 20, 2020).

⁶⁵ Winn and Calder, *supra* note 48, at 65.

⁶⁶ Alexander, *supra* note 4, at 100.

⁶⁷ Bayle et al., *supra* note 46 at 20; Winn and Calder, *supra* note 48, at 56.

⁶⁸ van Welie and van der Veer, *supra* note 62, at 532.

⁶⁹ Epistemic regimes are “stable arrangements of practices of knowledge production and social structures in which these practices are carried out”, Jochen Gläser et al., *Scientific fields as epistemic regimes: new opportunities for comparative science studies*, 3 (Technical University Technology Studies Working Paper TUTS-WP-3-2018, 2018), <https://www.ssoar.info/ssoar/handle/document/60196>.

⁷⁰ Dearden and Finley, *supra* note 46, at 64.

⁷¹ Prima facie easier identification of procedural patterns seems to be in contrast with the Nordic discussion of law as normative patterns that builds on Ann Christensen's work on substantive private law. See generally Christensen, *supra* note 35, Numhauser-Henning, *supra* note 33, at notes 25 & 27. On closer inspection the discrepancies can be explained by the different objectives of the approaches. Our approach places emphasis on the reusability and proven success over time, whereas the normative patterns approach highlights the importance of patterns for describing broader trends of legal evolution.

⁷² See Herbert Burkert, *Das Versprechen des Formalen – Einige Bemerkungen zur Beziehung zwischen Recht und Informatik*, in *The Digital Economy – Anspruch und Wirklichkeit*. Festschrift für Beat F. Schmid 47, 51–54 (Katarina Stanoevska-Slabeva ed., 2004).

⁷³ Dearden and Finley, *supra* note 46, at 84.

“shepherds”. An author shepherd is a special sort of (not anonymous) reviewer, whose job is to help the author make the best work possible in a short time by overseeing at least one, but preferably more revision cycles.⁷⁴ A writers’ workshop shepherd does the same for authors preparing papers for the workshop as well as chairing and facilitating the workshop, maintaining a constructive atmosphere and ensuring that the comments are always constructive, as the authors don’t participate in the discussion of their papers but are present and listen and then take the comments of the workshop into account in finalizing their paper for publication.⁷⁵

Thus, the generation process and pattern description become entwined, echoing Alexander’s original notion that a design pattern “is both a process and a thing; both a description of a thing which is alive, and a description of the process which will generate that thing.”⁷⁶ And pattern mining is a social process. To conclude, there is no single right way to generate a pattern, although once formulated there is the expectation it will be recognized by others, that “it makes sense” based on experience and intuition. What, then, does this mean from the perspective of generating legal design patterns?

3.3 Criteria for Legal Design Patterns

As discussed, the generation of patterns is typically seen as a collaborative and discursive process that aims to identify existing good practices that have been proven successful over time. What are the criteria we are using to identify design patterns in law and how do we assess success?

Design patterns include an internal and external perspective. Identification of legal design patterns requires looking into repetition in legal practice, which is context-specific by default, but the reconfiguration of this practice into a design pattern necessitates a jump in the level of abstraction. The move from legal practices to abstract patterns is made more difficult by the fact that patterns take a form fundamentally different compared to legal materials and doctrinal self-understanding, which operate based on distinctions between different legal fields.⁷⁷ As the ultimate objective of our exploratory approach is to enable the transfer of successful solutions across fields, pattern identification should not build on these established categorizations. Still, legal design patterns need also to be recognizable to practitioners and legal scholars who rely on law’s internal self-description. Hence, identification of legal design patterns needs to incorporate both internal and external perspectives. Finding patterns requires us to look at solutions as they are repeated in legal structures but from a different perspective of architectural design, to detach one’s observations from fundamental differentiations made in legal doctrine in order to introduce new possible ways to approach law’s organization. The oscillation between internal and external perspectives in pattern recognition has also been recognized in the Nordic discussion on Christensen’s theory on normative patterns.⁷⁸

We do not assume there will be completeness to legal design patterns nor are we convinced that this would be possible.⁷⁹ In software engineering, pattern languages entail a promise of a comprehensive set of

⁷⁴ Richard P. Gabriel, *Writers’ Workshops & the Work of Making Things* 51 (2002).

⁷⁵ Dearden and Finlay, *supra* note 46, at 84.

⁷⁶ Alexander, *supra* note 4, at 247.

⁷⁷ This notion is similar to Numhauser-Henning and Rönnmar’s interpretation of Christensen’s theory, where normative patterns can be distinguished amidst the multitude of legal norms. See Ann Numhauser-Henning and Mia Rönnmar, *Introduction*, in *Normative Patterns and Legal Developments in the Social Dimension of the EU 3*, 6 (Ann Numhauser-Henning and Mia Rönnmar eds., 2013).

⁷⁸ See Tuori’s observation that Christensen’s theory ultimately encompasses an external perspective, as it aims to uncover the “law beyond law”, Tuori, *supra* note 38, at 54.

⁷⁹ Similarly, Christensen, *supra* note 35, at 292: “The various normative patterns do not constitute a coherent system. They have come about at different times, for different types of relations, and in order to meet different kinds of needs. Each normative pattern has its own defined core territory or source of origin, but can also be transposed to other situations. A conflict between patterns with different contents may then arise.”

workable solutions for a specific context. Although such coherence might seem enticing for law, it would entail an assumption of completeness that is impossible to maintain in the legal domain. Some legal problems might be solved by a single pattern, whereas for others multiple alternative solutions might exist. Also, legal design patterns might have diverse audiences, ranging from judges, legislators and practitioners to public organizations and legal scholars, with varying needs and demands, which cannot be catered to simultaneously. Hence, design criteria should not aim for identification of complete sets of patterns but instead focus on generation and collection of different patterns, followed by elaboration of their relations and improvement of generation methodology.

4 FINDING DESIGN PATTERNS IN LAW

To exemplify what we mean by architectural or mid-level design patterns within the legal domain, we describe three examples of design patterns in law in the following section. The first example draws inspiration from conflict management theory and alternative dispute resolution (ADR). The pattern named here as *De-escalation through Dialogue* aims to solve the broadly discussed problem of procedure design. Built on the observation that adversarial dispute resolution is typically detrimental to the parties' future relations, the pattern aims to structure dispute resolution in a way that de-escalates the situation through interest-based dialogue. In turn, the second example discusses choice of enforcement mechanisms to ensure compliance with legislation that aims to protect abstract goods, such as environmental values, the public's interest in fair competition or "the" consumer. As ensuring compliance is usually dependent on interested party seeking enforcement, finding representation for abstract goods constitutes a structural design problem. The identified pattern *NGO Enforcement* facilitates the identification of a societal group for the purpose of representation. The third example then concerns the definition of the scope of protection granted by law to particular legal interests, which in their original scope defy or impede legal protection mainly due to them being fuzzy and their infringement being hard to detect and prevent. The pattern named here as *Moving Protection Upstream* targets this problem by reasonably extending the scope of legal protection, whether in their social, temporal, spatial or material dimensions, so that the extended scope allows for easier detection and prevention of infringements.

All three examples were formulated based on a common understanding of design patterns by the authors. The form of presentation in the following three examples varies not only because the design patterns differ or the authors' style differ, but because it is intended to illustrate that the presentation of a design pattern is not fixed, but open and can be shaped in different ways. We are well aware of the fact that no presentation ever follows the way it was produced.⁸⁰ The first example is structured around an observation in practice, in a similar manner as in writers' workshops described above. The second example intends to derive a design pattern from intuitive observations. Finally, the third example is structured according to how design patterns are usually presented in books or collections in fields other than law.

4.1 Example 1: De-escalation through Dialogue Design Pattern

A recurring problem in law consists in finding an appropriate conflict management mechanism to resolve a conflict between two parties. Appropriateness of the mechanism, however, depends on the conflicting forces in Alexander's sense, or, conflicting interests of the parties in the legal sense. Such interests might include the conflicting parties' mutual interest in the preservation of their long-term relationship after the

⁸⁰ This fact was specifically noted for organizational decisions by Niklas Luhmann, *Recht und Automation in der öffentlichen Verwaltung* 49–60 (1966), chapter "Funktions- und Arbeitsteilung zwischen juristischen und automatisierten Entscheidungsbeiträgen (I)"; and for the process systems of the legal system by Niklas Luhmann, *Legitimation durch Verfahren* 66–67 (1969).

conflict has been resolved. As established in ADR literature, civil procedure in the courts typically does not provide for preservation or restoration of the long-term relationship of parties involved. This results from the court's objective to reach a final decision on the legal issue at hand or to otherwise dispose of the case, for example, by the claimant withdrawing their claim. The objective of resolving the case through final decision typically requires the parties to assume adversarial positions in relation to each other with the result of escalating the dispute further. In addition, the rather formalized nature of the interaction between the opposing parties in the court may further escalate tension. The problem is thus how to structure the dispute resolution process in a way that would preserve the parties' future relationship. It has been observed that parties have better chances of avoiding escalation when they are able to openly discuss their interests with each other in a constructive manner. Hence, the proposed design pattern *De-escalation through Dialogue* aims to provide a solution to structure the dispute resolution process in a way that restores the parties' pre-dispute relationship, which logic reflects what has been said in the ADR literature.

We can also identify patterns "below" the higher-level *De-escalation through Dialogue* pattern. For example, *Neutral Third to Prevent Standstills* would include a neutral third party to the process in order to help the parties remain on speaking terms also when discussing heated or otherwise difficult topics. It should be noted that there are other functions and therefore other patterns on how the neutral third party participates in the process. A different role might be found in more institutionalized and adjudicative resolution processes, where the neutral third party assumes the authority to decide the contested issues between the parties.

There are conditions for using the *De-escalation through Dialogue*. For example, the parties would need to trust each other and be sufficiently motivated to engage in the dialogue-based process and face the risk of losing time and resources in case the process does not provide resolution of the conflict. Also, one of the consequences of using the pattern is that there are no procedural safeguards to mitigate power imbalances between the parties, thus meaning that the context of the pattern is limited to parties who are relatively equal. It should be noted that it is the parties' interests, not the society's or the neutral third's that is decisive here. The pattern is feasible only in situations where parties have the same interests regarding the preservation of their long-term relationship. The interest for dialogue needs to be genuine in a sense that it cannot be imposed. In ADR literature this is often framed as the voluntary and non-binding nature of mediation-like resolution mechanisms, as the fear of losing legal standing in the final decision would likely deepen adversarial positions. In this sense, if the conditions of good faith motivation are not met, we might consider *De-escalation through Dialogue* as an AntiPattern. This might be the case in situations where the interest for long-term relationship does not hold true for both sides, e.g. when a party commits to dialogue in bad faith to play time or to avoid litigation.

The *De-escalation through Dialogue* pattern can be found in various forms of conflict management in the society, which in the ADR literature is often framed as a characteristic of mediation processes.

The first example deals with court-based mediation in cases concerning child custody and visitation rights. Custody cases are a typical example of a conflict category, where conflicting parties – typically the child's parents – have a strong interest to preserve a functioning relationship after the decision, as interaction with the other parent cannot be completely avoided. The parents' negotiated settlement is often seen as preferable to adjudicated court judgment, as the parents are best equipped to take into consideration the unique characteristics and best interests of their child. Custody conflicts can also be notoriously difficult due to the emotional stakes, repeated processes, and mistrust. During the 2010s, Nordic countries have institutionalized *sui generis* court-based mediation procedures for custody cases that the parties can choose to enter either instead of or during a formal court process without the danger of waiving their legal rights

or standing.⁸¹ When compared to other forms of court-based mediation, custody case mediation is unique because the process design includes two facilitators: the mediator, a judge with mediation training, is accompanied by an expert on the child's perspective, typically a child psychologist or a social worker. The expert's role is to facilitate a conciliatory discussion between the parents so that they can work out an amicable solution together, e.g. by addressing the emotional and psychological needs of the parents and the child, by directing the discussion through questions and comments, by negotiating with the parents both individually and together and helping them to formulate their opinions and understand each other better. Originally initiated as a pilot experiment but then institutionalized, the expert-facilitated child mediation is broadly considered successful, by parties and judges as well as by the legislator and legal scholars.

Another example of *De-escalation through Dialogue* can be found in internal complaint boards, a form of out-of-court settlement mechanisms procedure advocated by the EU as a feasible solution for dealing with consumer-related disputes across sectors, e.g. in energy, telecommunications and payment services.⁸² Although the role of dialogue is not explicated, most complaint management guidelines point towards the communicative dimension of the process. This emphasis can be seen in ISO standards on external and internal complaint management in organizations, which advocate for a "customer-focused approach" that translates into responsive resolution process and increased complainant satisfaction. Concretely, the organization should actively communicate with the complainant and genuinely attempt to resolve the issue.⁸³

The dynamic is also repeated in labor market conciliation, which aims to preserve the long-term relationship between the trade unions in situations of industrial action. In labor conciliation, the conflicting parties are the employer and employees as a collective, e.g. the union as the assumed representation of collective employee interests. These *sui generis* conflict management procedures are not typically considered to be mediation as such, because of the unique characteristics and needs of long-term labor relations. In Finland, the state-appointed National Conciliator derives her authority from legislation, which also obligates the social parties to notify any industrial action in advance that initiates the conciliation process. The parties also have legal obligation to participate in the conciliation process, although there is no obligation to agree to the Conciliator's settlement offer. The form of the process is not structured by legislation or other frameworks but instead it is defined by the Conciliator's facilitative role, the parties' in casu needs and the focus on negotiated settlement.⁸⁴

To conclude, we can identify different applications of the *De-escalation through Dialogue* pattern across fields and conflict management mechanisms. Due to our objective here and because the advantages of dialogue-based dispute resolution are widely discussed in ADR literature,⁸⁵ it suffices at this point to notice the transferability of the solution.

⁸¹ See generally Kirsikka Salminen, *Asiantuntija-avusteinen tuomioistuinsovittelu ja konflikti lapsesta*, in 5 Defensor Legis (2013); Sanna Koulou, *Lapsen huolto – ja tapaamissopimukset: Oikeuden rakenteet ja sopivat perheet*, PhD dissertation (2014). In Finland, the legal basis is *Laki lapsen huollosta ja tapaamisoikeudesta* 361/1983 [Lapsenhuoltolaki] [Child Custody Act] January 1, 1984, *Suomen säädöskokoelma* [SDK], at ch 3a, 17a-d §, which includes a reference to general provisions on court-based mediation, expert's duties defined in preparatory document HE 186/2013.

⁸² See generally Marta Cantero Gamito, *Dispute Resolution in Telecommunications: A Commitment to Out-of-Court*, in 25 European Review of Private Law 387 (2017).

⁸³ ISO standards 10003:2018 (external) and 10002:2018 (internal).

⁸⁴ See generally Kari-Pekka Tiitinen et al., *Mediation of the Collective Interest Disputes* (1999). See also, National Conciliator's Office, *Mediation in Labour Disputes*, <https://valtakunnansovittelija.fi/en/mediation-in-labour-disputes> (accessed 19.2.2020).

⁸⁵ See generally e.g. Carrie Menkel-Meadow, *Whose Dispute Is It Anyway: A Philosophical and Democratic Defense of Settlement (In Some Cases)*, in 83 GEO L.J. 2663 (1995); Carrie Menkel-Meadow, *The Trouble with the Adversary System in a Postmodern, Multicultural World*, 38 Wm. & Mary L. Rev. 5-44 (1996).

4.2 Example 2: NGO Enforcement Design Pattern

Another recurring problem in law consists in finding an appropriate enforcement mechanism to ensure compliance with laws serving protected goods such as property, the environment and health. Before describing the design pattern that we will call *NGO Enforcement Design Pattern*, we will provide three examples in which this design pattern has been applied.

Our first example comes from the Aarhus Convention, which addresses environmental protection issues.⁸⁶ Article 6 of this convention specifies the way in which the public has to be informed about environmental decision-making procedures. To ensure compliance with Article 6, the Convention foresees that all “members of the public concerned having a sufficient interest (...) have access to a review procedure before a court of law and/or another independent and impartial body established by law, to challenge the substantive and procedural legality of any decision, act or omission subject to” Article 6 of the Convention.⁸⁷ Those “members of the public concerned” are to be understood as non-governmental organizations (NGOs) promoting environmental protection.⁸⁸ Hence, the enforcement of violations of Article 6 depends on whether NGOs exercise their right to sue.

Our second example comes from Germany, where we take a closer look at the Act Against Unfair Competition (“UWG”, i.e. “Gesetz gegen den unlauteren Wettbewerb”).⁸⁹ According to Section 1, the Act shall not only serve the purpose of protecting competitors, consumers and other market participants against unfair commercial practices, but also protect the interests of the public in undistorted competition. To this end, the UWG allows to sue for elimination whoever engages in an illegal commercial practice pursuant to the UWG.⁹⁰ This right to sue is granted to “associations with legal personality which exist for the promotion of commercial or of independent professional interests”.⁹¹ Here, too, an NGO is given the opportunity to assert abstract interests – the interest of the public in undistorted competition – through legal action.

Our third example is the German Injunctions Act,⁹² which gives the right to take legal action against companies that have unlawful general terms and conditions to registered associations serving consumer protection. We intuitively identify here a pattern similar to the two aforementioned examples.

To identify the specific characteristics defining the design pattern recurrent in all three examples, it is helpful to take a step back and look at the broader goal pursued by the three regulations: when an issue is regulated by law, most people will usually comply with these rules. Nevertheless, some will not adhere, which makes enforcement mechanisms necessary. We may assume that, on a very abstract level, there must be a class of design patterns aiming to solve the problem of enforcing the law, the *Enforcement Design Patterns* class. Yet the differences in protected goods, participating actors etc., their particular characteristics as well as the conflicting forces that characterize the problem of enforcing laws on those goods, in relation to those actors etc. clearly demand for differentiating subclasses within this class. As mentioned in section 3, it is possible to draw different distinctions that then constitute different sets of subclasses. One intuitive

⁸⁶ Convention on Access to Information, Public Participation in Decision-Making and Access to Justice in Environmental Matters, opened for signature June 25, 1998, <https://www.unece.org/fileadmin/DAM/env/pp/documents/cep43e.pdf> (entered into force Oct. 30, 2001) [hereinafter Aarhus Convention].

⁸⁷ Aarhus Convention Art. 9 paragraph 2.

⁸⁸ Aarhus Convention Art. 2 paragraph 5 and Art. 9.

⁸⁹ Gesetz gegen den unlauteren Wettbewerb [UWG] [Act against Unfair Competition] July 1, 1896, REICHSGESETZBLATT [RGBl].

⁹⁰ UWG Sec. 8 subsection 1.

⁹¹ UWG Sec. 8 subsection 3 number 2.

⁹² Gesetz über Unterlassungsklagen bei Verbraucherrechts- und anderen Verstößen [Unterlassungsklagengesetz] [UKlaG] [Act on injunctions for the protection of consumers' interests] January 1, 2002, Bundesgesetzblatt I [BGBl].

distinction with regard to the illustrated examples is the good that is protected by all three regulations. The law in the first example serves the protection of the environment, in the second it aims at upholding a fair competition, and the third example's law seeks to protect consumers from unlawful contracts. In all three cases, the protected good has an abstract nature. No one is (necessarily) directly affected by a non-compliant practice. Thus, there is no "natural" claimant. An enforcement mechanism however requires somebody to take action in regard to the non-compliant practice. There are many different bodies that can be called upon to enforce the law. In our examples, NGOs promoting the protected abstract good are called upon. Other bodies can also serve the legal protection of abstract goods. The stability of the financial markets is *inter alia* legally protected by criminalizing insider trading.⁹³ The body taking action in this case is the public prosecutor. Another example is compliance with data protection laws, which is overseen and enforced by independent supervisory authorities.⁹⁴

Based on the observations made previously, we can identify the following characteristic features for the NGO enforcement design pattern. The pattern aims at creating an enforcement mechanism to facilitate the protection of abstract goods. The problem that this design pattern aims to solve is the lack of entities – individuals, legal persons etc. – that are affected in their own rights with the consequence that nobody will take action against the infringement. The solution is thus to give a specific entity the necessary competence. In contrast to other possible solutions, the *NGO Enforcement Design Pattern* chooses to assign this power to an NGO that is committed to the abstract good.

Once the *NGO Enforcement Design Pattern* is identified as such, it can be examined more closely at various levels in contrast to other design patterns. It is worth noting that NGOs have no legal obligation to monitor the application of regulations. Therefore, NGOs will not automatically become active should any violation of the law occur. Rather, they will decide by their own values, norms and strategic goals which violations they want to address. In addition, NGOs have a lack of public control because of their private nature, but at the same time they are deeply rooted in society and in the very public which is interested in the abstract good. These are just some observations one might make about the *NGO Enforcement Design Pattern*. Our aim here is not to examine this pattern in detail, but to show that the identification of the patterns allows many overarching investigation approaches, the results of which can then also be used for each example given.

4.3 Example 3: Moving Protection Upstream

The third example concerns the reasonable extension of the scope of protection granted to particular legal interests, whether in their social, temporal, spatial or material dimensions, in all cases where the legal interest's original scope would be hard or impossible to protect by other legal means. We will call this design pattern *Moving Protection Upstream*. Its best-known manifestation is arguably the move from ex-post to ex-ante regulation, e.g. when legislators aim to prevent harm instead of simply repairing damages, i.e. the scope's extension in its temporal dimension.

The context in which this pattern can be found regularly is the legal definition of legal interests' scopes in legislation.

⁹³ This was the official justification for the law in Germany, see Deutscher Bundestag [Parliament of the Federal Republic of Germany], Drucksache 12/6679, at 33, <http://dipbt.bundestag.de/doc/btd/12/066/1206679>; see also Chunliang Yun, Die Strafbarkeitsgründe des Insiderhandelsverbots 249 (2016).

⁹⁴ See Regulation (EU) 2016/679 of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), 2016 O.J. (L119) [hereinafter GDPR], Chapter VI.

The problem addressed by this legal design pattern is that the particular interest itself is considered unsuitable for being linked with an effective legal protection mechanism, for one or both of the following reasons, while the harm is of a kind that cannot be undone or healed, and compensation would be insufficient or inappropriate. Firstly, it may be hard to detect, establish or prove an infringement of this legal interest. Depending on the field of law in which this pattern can be observed, this hardness may affect the victims, law enforcement agencies, courts or even the infringers themselves. For example, the victims may be unable to properly defend themselves, take legal action effectively or report a crime, law enforcement agencies may find it hard to investigate or establish whether an infringement actually took place, while courts may face difficulties to properly prove it beyond reasonable doubt. It may also affect infringers, or possible infringers: the law's function of regulating behavior, or more specifically in this case: of preventing infringements of legally protected interests, strongly depends on their detectability. If infringements are not detectable, the law that aims at protecting the particular interests will not be able to influence the behavior of both obliging and malicious actors, though for different reasons: the former by voluntarily refraining from infringing on the said interests, the latter for the threat of punishment. Secondly, it may be hard to prevent an infringement from happening, with the run-up to the infringement being of a slippery slope type and lacking "natural" or other barriers that have to be overcome. Thus, the separation between being compliant and non-compliant, between not infringing and infringing is fuzzy and essentially undefendable.

The conflicting forces that characterize the problem are, first, the interest in the effectiveness of the protection mechanism, second, that limiting or narrowing the scope of protection is considered unacceptable and, third, that extending the scope of protection intensifies the interference in legal subjects' rights and freedoms. Both rights' holders, i.e. those protected by the law, and legal subjects, i.e. those on which the law imposes duties or whose freedoms are restricted in order to protect particular legal interests, have an interest in the effectiveness of the protection mechanism. The interest of the former is obvious and direct, the latter's interest is more indirect. In this case, it is the justifiability of restricting freedoms or imposing duties that is at stake, i.e. the law's justifiability depends on its effectiveness in protecting the legal interest. Secondly, if society considers a legal interest worth protecting and has decided to protect it, artificially limiting or narrowing the scope of its protection in order to make its infringement easier to detect and pursue with criminal and civil legal action or easier to prevent would explicitly expose those very legal interests and thus run counter to society's decision to protect it. Thirdly, extending the scope of protection may further restrict freedoms of and impose more duties on legal subjects, which would demand for a more comprehensive justification.

The solution is to balance the conflicting forces by searching for a reasonable extension of the scope of protection that allows for effectively protecting the particular legal interest, i.e. to move protection upstream. This extension could affect the scope's social, temporal, spatial or material dimension. In the social dimension, the scope could be extended to include actors, i.e. legal entities, with the appropriate skills, knowledge and resources, including powers that enable them to mitigate the problem. In the temporal dimension, the extension of the scope can be done by exchanging ex-post with ex-ante mechanisms, while in the spatial dimension, the scope could be extended by moving from a situation to a space in which the situation usually occurs or may occur. The scope's extension could also affect its material dimension, most commonly by moving from a focus on an action to be prevented to an instrument that is causally related to the action or the action's negative effects.

For each of the four dimensions we will provide an example of the *Moving Protection Upstream* design pattern.

The first example concerns the scope extension in its social dimension. In the last few years, there has been a growing acknowledgment that regulating operators or users of tools, machines or other technical artifacts and their usage behavior may be less effective than formerly expected. An important reason for this approach's relative ineffectiveness might be that the technologies' part in governing their usages can be considerably greater than the users' part, up to technologies essentially determining their use. Thus, a considerable move can be observed towards regulating manufacturers, e.g. of medical devices⁹⁵ or in the field of AI⁹⁶, i.e. actors controlling the design and construction of the tools, machines or other technical artifacts including those characteristics that are causally related to the problems that the law aims to solve, prevent or mitigate.

The *Moving Protection Upstream* design pattern finds its most widespread application in its temporal configuration, i.e. where the scope is extended by moving from ex-post to ex-ante regulation.⁹⁷ A case in point is data protection law. At the same time, it is an example of a double extension. When regulating modern information processing in order to mitigate negative effects on individuals was first discussed in the 1960s, it was generally accepted that it was the use of personal data that created or exacerbated the problem, while at the same time there was a clear consensus that "a regulatory scheme that focuses on the end use of the data by governmental or private systems might be a case of too little, too late."⁹⁸ Thus, data protection law's first move was from a use-oriented towards a collection- and processing-oriented protection approach. The second move of protection upstream in data protection law followed soon, in 1977, from regulating the processing of personal data to regulating the design and implementation of technical and organizational means that are used to process such data.⁹⁹ While this attempt has been fruitless in influencing system design and implementation, as it was deliberately undermined by a coalition between the German Federal Commissioner for Data Protection, a few German Laender commissioners and interested parties from industry,¹⁰⁰ a similar provision was included in the EU General Data Protection Regulation of 2016.¹⁰¹

The third example concerns the spatial extension of a scope of protection. In order to protect the work of a parliament, which is composed of situations like meetings, debates and votes, the legal protection is very commonly applied to the parliament building or buildings. And the protection mechanisms are then implemented at the building's entrances, with guards, security installations and security checks. In many countries, the zone of protection is even more extended than to the buildings themselves, with laws defining zones around them where, e.g., demonstrations are generally prohibited.¹⁰²

The most well-known examples of protection scopes' extensions in their material dimension are probably regarding weapons on the one hand and chemical substances on the other hand. Though the primary legal interest in this case is in the life and health of people, in many (civilized) countries the scope has long since

⁹⁵ E.g. see Council Directive 93/42/EEC of 14 June 1993 concerning medical devices, 1993 O.J. (L 169/1).

⁹⁶ See generally *Artificial Intelligence – A European Perspective* (Max Craglia ed., 2018), <https://ec.europa.eu/jrc/en/publication/artificial-intelligence-european-perspective>.

⁹⁷ For a generalisation of the perspective see Liv Jaeckel, *Gefahrenabwehrrecht und Risikodogmatik: Moderne Technologien im Spiegel des Verwaltungsrechts* (2010).

⁹⁸ Arthur R. Miller, *Personal Privacy in the Computer Age: The Challenge of a New Technology in an Information-Oriented Society*, in 67 *Michigan Law Review* 1089, 1221 (1969).

⁹⁹ para. 6 of the German Data Protection Act of 1977. Gesetz zum Schutz vor Mißbrauch personenbezogener Daten bei der Datenverarbeitung – Bundesdatenschutzgesetz [BDSG] [Federal Data Protection Act] January 1, 1978, Bundesgesetzblatt I [BGBl], at 201, § 6.

¹⁰⁰ Jörg Pohle, *Das Scheitern von Datenschutz by Design: Eine kurze Geschichte des Versagens*, in 32 *FIFF-Kommunikation* 41 (2015).

¹⁰¹ GDPR art. 25 "Data protection by design and by default".

¹⁰² E.g. in Germany, see Gesetz zur Zusammenführung der Regelungen über befriedete Bezirke für Verfassungsorgane des Bundes [BefBezG] [Restricted Areas Act] December 8, 2008, Bundesgesetzblatt I [BGBl.], at 2366, para. 2.

been extended from the law's sole focus on the action, i.e. the infringing or killing act, to the weapon, i.e. the tool with which such an action is or may be conducted, which is considered justifiable as the definition of "weapon" does usually not refer to general purpose but single purpose tools. A similar approach has been taken regarding chemical substances that are technically harmless, but can be, and often are, used to produce drugs, explosives, or equally harmful products.¹⁰³ In those cases, regulation has been extended to include the production of and trade in such substances.

5 DISCUSSION: LEGAL DESIGN PATTERNS AND PROCEDURALIZATION

Two potential advantages emerge from the fact that procedural patterns are easier to identify within law. Firstly, the temporal dimension inherent in pattern generation means that the legal design patterns might lend their descriptive force to the examination of proceduralization by rendering this multifaceted phenomenon more visible. Within socio-legal scholarship, proceduralization often adopts a negative connotation in the sense that it takes place at the expense of substantive law. To borrow Thornhill's description, proceduralization refers to new forms of governance that place emphasis on multi-stakeholder participation in law creation institutionalized through procedures, which leads to the procedure itself becoming decisive for the legitimacy of governance.¹⁰⁴ In this sense, proceduralization of governance structures reflects social interaction, as the architectural design of the process strives for desired forms of participation. This observation incorporates yet again the idea that law's abstract architecture is not necessarily that much different from physical architecture, as both structures ultimately aim to promote desired social interaction.

Mark Dawson discusses "the procedural turn" within the EU's social law, which he attributes to the diversity and functional complexity of the transnational context.¹⁰⁵ According to Dawson, the procedural paradigm questions the previous conceptualizations of the relationship between law and its social environment and suggests a more reflexive and responsive approach, pointing to context-specificity.¹⁰⁶ Also Getliffe discusses the connection between responsive law and participation through proceduralization. In the context of EU environmental law, she argues that although proceduralization may alleviate the EU's democracy deficit and improve policy-making; these mechanisms should not replace top-down policy initiatives.¹⁰⁷ Procedural emphasis may lead to the disempowerment of vulnerable groups, as Nieminen argues in the context of the European Court of Human Rights' judgment on the Muslim veil ban. She demonstrates how the court's procedural emphasis together with its contextualized approach to rights inadvertently erodes the protection against discrimination of vulnerable groups.¹⁰⁸

¹⁰³ Cf. Single Convention on Narcotic Drugs, 1961, as amended by the Protocol amending the Single Convention on Narcotic Drugs, March 30, 1961, 976 U.N.T.S. 105; Regulation (EU) 98/2013 of 15 January 2013 on the marketing and use of explosives precursors, 2013 O.J. (L 39), and Regulation (EC) 1907/2006 of 18 December 2006 concerning the Registration, Evaluation, Authorisation and Restriction of Chemicals (REACH), establishing a European Chemicals Agency (ECHA), 2006 O.J. (L 396), respectively.

¹⁰⁴ "Above all, it has been widely claimed that, as the material presence of the people in government is diminished, legitimacy for laws can only be secured through a multi-level, mediated incorporation of the people – which needs to be institutionalised through multiple procedures. Through the later twentieth century, in fact, procedure itself came widely to replace the exercise of constituent power as a founding legitimational framework of government." See Chris Thornhill, *The global legal system and the procedural construction of constituent power*, in 5 *Global Constitutionalism* 405, 406 (2016).

¹⁰⁵ See generally Mark Dawson, *New Governance and the Transformation of European Law: Coordinating EU Social Law and Policy 2* (2011).

¹⁰⁶ Id. at 110. For a more theoretical, but earlier account, see generally Gunther Teubner, *Substantive and Reflexive Elements in Modern Law*, in 17 *Law & Society Review* 239 (1983).

¹⁰⁷ Kate Getliffe, *Proceduralisation and the Aarhus Convention - Does Increased Participation in the Decision-Making Process Lead to More Effective EU Environmental Law*, in 4 *Environmental Law Review* 101, 115 (2002).

¹⁰⁸ See generally Kati Nieminen, *Eroding the protection against discrimination: The procedural and de-contextualized approach to S.A.S. v France*, in 19 *International Journal of Discrimination and the Law* 69 (2019).

Considering the well-grounded critique of proceduralization, the question emerges whether our design pattern approach inadvertently contributes to its negative implications. While this may hold true, the approach might also bring a more nuanced understanding of this complex and multifaceted phenomenon. As our examples show, proceduralization comes with problem-solving capabilities, which also explains the ubiquity of the phenomenon. Simply put, it is context dependent in which cases proceduralization provides desired outcomes and in which not – in the language of design patterns, the distinction between design patterns and AntiPatterns may be helpful for establishing this context-dependency.

Secondly, the design pattern approach might lend itself to new insights on the automation of legal processes, as it emphasizes the problem-solving capabilities of law as societal architecture in comparison to software architecture. Here, the advantage lies in the interdisciplinary nature of design patterns, proven by the adoption of the approach across disciplines. By establishing a joint language between software engineers and legal domain experts, design patterns might facilitate knowledge exchange necessary for successful software development and deployment.

6 CONCLUSIONS

The main argument we have put forward in this article is that the design pattern approach would bring to the front law's problem-solving capabilities and encourage new forms of self-reflexivity. Ideally, the approach would contribute to a better understanding of law as an abstract architecture that can be improved upon by proactive design. Ultimately, design patterns can provide new heuristic and methodological tools for both socio-legal scholarship and interdisciplinary research between law and computer science.

The generation of design patterns, or more accurately the identification of recurring solutions and their formulation as design patterns, is an inherently collective process, as wider knowledge bases facilitate recognition of functioning solutions across different domains. A design pattern description needs to include the name of the pattern, the design solution it puts forward, a description of the problem and the conflicting forces at play, and the context and conditions for the pattern's applicability. In addition, a design pattern should have been proven successful through repetition over time. In our exploratory approach, legal design patterns are more readily found in procedural structures than in substantive rules due to the requirement of successful repetition. The requirement of proven success diverts gaze into temporal repetition that is clearly visible in procedural structures.

What is the role, then, for design patterns within law? It should be noted that design patterns entail both descriptive and prescriptive dimensions. Description is a fundamental quality of recognition of a pattern, yet the form needs to include a solution, which can and should be interpreted as a normative stance. By looking into the development of design patterns within software engineering, we observe the power configurations inherent in calling something a design pattern. This normative dimension and power relations included in the distinction between patterns and AntiPatterns need to be examined in further detail through normative theories.

In order to expand this exploratory examination of legal design patterns, three other venues for future research emerge. Firstly, the design pattern approach needs to be consolidated by a more comprehensive examination. This includes the identification of central patterns across legal fields and elaboration of their interconnections in order to assess potential categorizations, interdependencies and hierarchies between patterns. In addition, it is necessary to elaborate on the differences between design patterns and AntiPatterns to improve the recognition of unnecessary or even harmful solutions. One potential example

of a legal AntiPattern might be found in technology governance, where industry self-regulation is repeatedly advocated as a solution to the legislator's alleged inability to react to technological development. The fact that shortcomings of self-regulation as governance strategy are well known suggests that the pattern does not meet the criterion of proven success over time. For a comparable reason, we have abstained from including references to the discussion in law and economics on how externalities are to be regulated most efficiently. Contrary to the widespread reference to proven success in practice, success criteria in law and economics are virtually always derived from neoclassical or new institutional microeconomic theory, i.e. ideology, not practice. Furthermore, examination is needed on the different target audiences of legal design patterns to map out the added value for legal education and socio-legal scholarship, for practitioners across legal fields as well as the legislative and executive branches of government. Ultimately, the added value of legal design patterns is the promise of new heuristic and methodological tools for the examination of law in practice.

Secondly, it is necessary to further develop the identification process based on collaborative iteration. The focus here should be on the development of sound methodology that would then reap advantages of collaborative pattern generation and contribute to an improved transferability of the generated patterns. As research on social epistemology has shown, the combined efforts of multiple agents generally contribute to better outcomes in knowledge production.

Thirdly, the design pattern approach might provide new venues for the examination of law's self-reproduction, as law reacts to societal needs by problem solving. The temporality dimension present in pattern generation also comes with the promise of new insights into proceduralization, which ultimately is about establishing long-lasting epistemic structures within the legal system. In this sense, the design pattern approach might prove to be particularly useful for socio-legal research on automation of legal decision-making, as automation is dependent on proceduralization and necessitates architectural system design.