Authoring with IDN Structural Patterns

David E. Millard
Electronics and Computer Science, University of Southampton
Southampton, UK
dem@soton.ac.uk

ABSTRACT

IDN Structural Patterns are a type of Design Pattern that offer three potential benefits to IDN Authors: they can inform them of common solutions to problems, they can provide ways to create complex structure quickly, and they can provide a lens to reflect on existing work. But how might they be successfully integrated into an authoring tool? In this paper I set out a design space for patterns in IDN Authoring, looking at cookbooks, patterns by design, domain specific languages, and structural parsers, and exploring whether they deliver those benefits, and also whether they support uncommon as well as common patterns. I show that no single approach delivers all of the benefits, but that combinations of methods could potentially do so, at the risk of increased cognitive load for authors. This initial work shows that there is significant potential in using patterns for authoring, but that more empirical work is needed to understand their affordances and interaction effects.

CCS CONCEPTS

• **Human-centered computing → Hypertext / hypermedia**; *Interaction design theory, concepts and paradigms*; • **Software and its engineering → Design patterns**.

KEYWORDS

hypertext, digital interactive narrative, design patterns, authoring

ACM Reference Format:
David E. Millard. 2022. Authoring with IDN Structural Patterns. In *Proceedings of June 28, 2022 (NHT'22).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In my 'Strange Patterns' chapter of 'The Authoring Problem' I set out the work that has been done over the last thirty years on understanding structural patterns in Hypertext and Interactive Digital Narratives (IDN) [16]. Following a predominantly structural approach several authors have explored the different structures that regularly occur in hypertext literature and I argue that these become *Design Patterns* [1] that hypertext and IDN authors can use as part of their own authoring process.

While it is intuitively true that understanding the community's common methods for solving problems would be useful to authors, in this paper I try to go further and set out a design space for how patterns could be incorporated into authoring tools, based on specific examples and inspired by similar approaches in the literature.

I review these approaches against three potential benefits:

- **Potential Benefit 1 (PB1)** - they can inform authors of common solutions to problems
- **Potential Benefit 2 (PB2)** - they can provide a way to create complex structure quickly
- **Potential Benefit 3 (PB3)** - they can provide a lens by which to reflect on an existing structure

IDN structural patterns normally refer to *Common Patterns* that occur in many different works. In 'Strange Patterns' I argue that this risks a post-structural critique that no patterns should be considered as fundamental or universal, and that doing so might actually be harmful in stifling innovation (especially in narrative games where the mechanics of interaction might be wildly different from those systems where the patterns were first observed). An alternative is to support *Uncommon Patterns* - which are those that are unique to a particular work, but which occur multiple times within that work [20]. So when exploring the design space for patterns in IDN authoring we will also consider whether the solutions lend themselves to uncommon patterns as well as common ones.

2 BACKGROUND

'Strange Patterns' sets out three types of IDN Pattern:

- **Micro Patterns** – these are the small building blocks of IDN, structures that are so core to the authoring experience that they seem invisible to us, even though they can define a whole form. Examples including Bush's *trails* [6], semantic web *triples* [12], *adaptive links* [8], or *storylets* [10].
- **Meso Patterns** – these are larger patterns that describe sub-structures within an IDN made up of multiple micro patterns. These sub-structures are often used to create specific effects or solve certain problems. Examples include *split/joins* [4], *phasing* [7], or *unchoices* [14].
- **Macro Patterns** – these are the largest type of pattern and describe entire IDNs or large sections of IDNs. Macro patterns capture the overall structure, and thus some part of the overall experience, especially in regard to the agency of the reader. Examples include *Canyons/Deltas/Plains* [18], or *Gauntlets* or *Spoke and Hub* [3].

In this paper I am focusing on Meso Patterns and their potential role in authoring. Micro patterns are so fundamental in our IDN systems that they tend to be baked into authoring environments already (you can do the same with Meso patterns, see Section 3.2 below). Macro patterns define the entire shape of an IDN; they

NHT'22, Barcelona, Spain,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

could be used in authoring and if so would be variations on the Meso approaches defined here.

3 AUTHORING WITH PATTERNS

This section outlines five approaches to incorporating structural patterns into IDN and hypertext authoring tools, some of which are poorly explored in both tools and the literature. These are: cookbooks, patterns by design, templates, domain specific languages, and structural parsers.

3.1 Cookbooks

A 'cookbook' is a colloquial term for describing help documentation that presents a set of recipes for solving particular problems. Both Twine[1] and Inform[2] have an online cookbook (Inform calls this a 'Recipe Book') which describes a set of topics, and for each topic gives a description of the functionality used in the examples, followed by the examples themselves that demonstrate various principles and challenges. For example, Figure 1 shows the page from the Twine cookbook for simulating Dice Rolling using Twee (the markup language used within the Twine application).

"Dice Rolling": Harlowe (v2.0)

Summary

"Dice Rolling" demonstrates how to create the same effects of rolling various physical dice through using the `(random:)` macro and adding or subtracting numbers.

Twee Code

```
:: StoryTitle
Harlowe: Dice Rolling

:: Start
Rolling a 1d4: (random: 1,4)
Rolling a 1d6: (random: 1,6)
Rolling a 1d8: (random: 1,8)
Rolling a 1d10: (random: 1, 10)
Rolling a 1d12: (random: 1, 12)
Rolling a 1d20: (random: 1, 20)
Rolling a 1d100: (random: 1, 100)
Rolling a 1d4 + 4: (text: (random: 1, 4) + 4)
Rolling a 1d6 - 2: (text: (random: 1, 6) - 2)
Rolling a 2d6 + 10: (text: (random: 1, 6) + (random: 1, 6) + 10)
```

Figure 1: The Twine cookbook page for 'Dice Rolling'

While cookbooks could easily be used to describe structural patterns and give examples of how they can be created they instead tend to focus on behaviour, with both the Twine and Inform cookbooks focusing on how to manipulate story state to achieve particular effects. Even Tinderbox, which is more focused on knowledge management tasks and has a hypertext heritage (coming from Eastgate systems, the creators of Storyspace[5]), has a cookbook that focuses on actions and export, rather than structure[3].

Cookbooks are simple to understand and a community can also be involved in their creation, this makes them good at informing authors of solutions to common problems (PB1). They do not directly allow structure to be created in a tool (PB2) but examples can be cut and pasted and modified to fit. While they do not directly allow for reflection within a tool (PB3), they do provide a visible standard against which authors can compare their own work. As they only describe a finite set of common patterns, uncommon patterns are not supported.

3.2 Patterns by Design

An alternative approach is to embed certain patterns into the design of the authoring tools themselves. This is frequently done with micro patterns, where certain kinds of links or structures are supported in the interface, but can also be done with meso patterns.

The StoryPlaces authoring tool embeds two meso patterns: *locking*, where one node has to be read before another; and *phasing*, where a set of nodes are locked or unlocked together [17]. These become primary elements in the interface. For example, as shown in Figure 2 when editing a node in the 'narrative constraints' section the tool shows a list (initially empty) of nodes that would unlock this one, and provides a mechanism for looking up and adding nodes to this list. Similarly, the same page allows the node to be added to a Chapter (the StoryPlaces name for a phase), and to specify whether this node locks or unlocks any chapters.

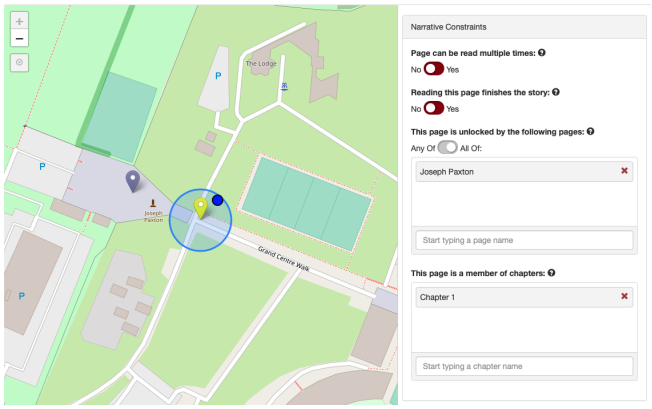

Figure 2: The StoryPlaces authoring tool, showing both locking and phasing built directly into the interface

Embedding patterns in the interface normalises meso patterns in the same way as micro patterns (StoryPlaces also embeds storylets) and thus encourages their use by framing them as part of the normal functioning of the authoring tool, thus indirectly informing authors about those patterns (PB1). It also makes the instantiation of the patterns seamless (PB2). However, in doing so it permanently pivots the authoring tool towards the use of those specific patterns, potentially at the expense of other patterns that might also be useful to users, meaning that their use for reflection is very limited (PB3), and similarly that uncommon patterns are not supported.

3.3 Templates

Templates are a declarative way of creating reusable material within an authoring environment. In note-taking tools such as *Obsidian*[4]

[1]Twine cookbook: https://twinery.org/cookbook/
[2]Inform cookbook: http://inform7.com/book/RB_1_1.html
[3]Tinderbox cookbook: http://www.eastgate.com/Tinderbox/cookbook/index.html
[4]Obsidian homepage: https://obsidian.md

they allow content and structure to be defined once, and then instantiated again and again to make the creation of elements with standard parts easier. They can also include macros that are dynamically filled by the system at the moment of creation. For example, Figure 3 shows a template for a 'Zettelkasten' style note in Obsidian using macros to insert the title, date, and time.

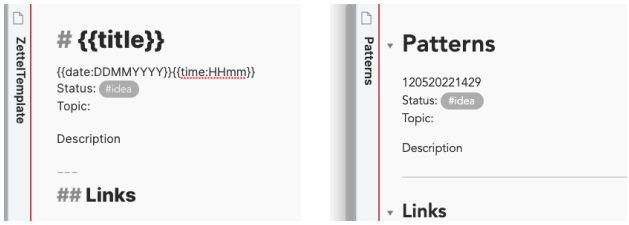

Figure 3: A template (left) and instantiated page (right) for a 'Zettelkasten' note in Obsidian

This style of template only handles local structure – i.e. outbound links from this node – but we can imagine templates where we might specify multiple nodes at once and the relationships between them. Such a structural template could be used for common structural patterns, for example split/joins, and in theory would both make the authoring process simpler (PB2) (as complex structures would be created in a single operation rather than many) as well as helping to educate new authors by exposing them to common ways of working (PB1). The existence of templates would also indirectly help authors reflect on the structure of their work (PB3). If new templates could be created by authors then it also supports uncommon patterns.

A simpler form of templates could be supported through structural cut and paste (where multiple nodes and their relationships could be selected, copied, and then pasted into the authoring environment). While this approach does not have the benefits of informing authors about common patterns (PB1), it does still allow for the easy creation of complex structure (PB2) and indirectly helps authors reflect on their work through their own cut and paste behaviour. Structural cut and paste is also a very straightforward and familiar way to support uncommon patterns.

3.4 Domain Specific Languages

Domain Specific Languages (DSLs) are computer languages for defining behaviour in a specific domain, with a higher level of abstraction than is found in more general programming languages and which use the concepts and terminology of that domain[15]. As a DSL is aligned closely with its problem space it becomes easier for people to connect what they want to do in that domain with what they need to write. DSLs are therefore a useful bridge between technical and domain expertise. They can also be independent of any given target platform or tool.

DSLs are a more complex method for defining patterns, as they can include parameterised elements. This makes them more powerful than simple visual templates (PB2), but also means they might be less effective at communicating patterns to authors (PB1). Their complexity means that they are also not very good at helping authors to reflect on their creations (PB3).

```
function flashback(nodes_for_option_A, nodes_for_option_B) {
  locks(nodes_for_option_A, nodes_for_option_B);
  locks(nodes_for_option_B, nodes_for_option_A);
}

let player_a_option_a = story.NewPage(...);
let player_a_option_b = story.NewPage(...);
let player_b_option_a = story.NewPage(...);
let player_b_option_b = story.NewPage(...);

flashback([player_a_option_a, player_b_option_a],
          [player_a_option_b, player_b_option_b]);
```

Figure 4: An example of a Typescript DSL (taken from [20]) that defines a Flashback

DSLs are a good solution for uncommon patterns. Figure 4 shows a snippet of the Typescript DSL used in [20] to define a *Flashback* uncommon pattern: a set of four nodes, with four lock relationships between them (where the lock relationship has been previously defined using the same DSL).

3.5 Structural Parsers

A final potential approach would be to identify patterns in the author's structure as it emerges. This is similar to the way that a spatial parser works in spatial hypertext [19], in that case the parser is looking for visual and layout similarities (such as items in a row that become a list, or items with the same colour that become a set); with structural parsing it would be analysing the graph of link connections and looking for structures that match well known patterns. In graph theory this is called the *sub-graph isomorphism problem* [21] and there are numerous potential solutions competing for computational efficiency [2]. Once patterns have been identified they could be highlighted to the author and labelled or described.

Spatial hypertext was first presented as a tool for *information triage*, allowing structure to emerge rather than being imposed from the start – ideal when that structure is not known in advance [13]. A structural parser might therefore help the author to understand what sort of story they were creating (in the case of macro structures) or what kinds of approaches they were using within the story to manage its progress (the meso structures) even if they didn't know this when they began (PB3). They also draw attention to common patterns and therefore expose those approaches to users (PB1), but they don't enable them to create complex structure more quickly (PB2) (although they might if coupled with structural cut and paste).

Structural parsers that look for repeating (in addition to well known) patterns could also support uncommon patterns, as they could draw attention to structures that reoccur in an authors work. Searching for common sub-graphs within a larger network is a different problem known as *frequent sub-graph discovery*, with a range of potential solutions that vary in their search strategy, inputs, and completeness [11].

4 DISCUSSION

Table 1 shows a summary of the five methods (and the cut and paste template variant) mapped against the three potential benefits, the final column also shows whether they support uncommon patterns

Table 1: The methods mapped to the potential benefits and their applicability to uncommon patterns

	PB1 Inform Users	PB2 Complex Structure	PB3 - Reflection	Uncommon Patterns
Cookbook	Yes	Indirect	Indirect	No
By Design	Indirect	Yes	No	No
Templates	Yes	Yes	Indirect	Possible
Cut and Paste	No	Yes	Indirect	Yes
DSL	Indirect	Yes	No	Yes
Parser	Yes	No	Yes	Yes

as well as common ones (Templates shows 'possible' as this is only supported if new templates can be generated by the author).

4.1 User/Tool Alignment and Cognitive Load

One of the challenges that authors face in the authoring process is aligning themselves with their tools, with misalignment possible in four dimensions: conceptual, ontological, expertise, and workflow [9]. This misalignment can place significant cognitive load on authors.

Introducing patterns into authoring impacts all of these dimensions, and could make that cognitive load worse. It requires authors to conceptualise their work through the patterns, it requires them to learn the terminology and use (ontology) of the patterns, it requires additional expertise in the form of the skills needed to manipulate and use the patterns within the tool, and it requires them to adapt their workflow to include patterns.

However, the different methods have these requirements to different extents. For example, the Cookbook and Cut and Paste approaches are familiar from many other applications, and authors should have transferable skills, and already think and work in this kind of way. By Design also requires no special expertise to deploy, but it does require the author to conceptualise their story using the patterns and adapt their workflow for the tool (e.g. in StoryPlaces authors must become familiar with the idea of *Chapters*, conceptualise their narrative design in terms of locking and unlocking chapters, and adjust their workflow so that the creation of chapters is at its heart).

A DSL is the most sophisticated method, and requires significant expertise to deploy effectively, however the conceptualisation, ontology, and workflow are driven by the author, and so – assuming they have those skills – it might actually introduce the least complexity, and arguably is the approach that most empowers the author.

4.2 Combining Approaches

So far we have discussed the methods for including patterns in authoring independently, but in reality they could be combined and used together in a single tool.

Templates, Cut and Paste, and the Structural Parser are all declarative in nature and could potentially work together. Templates and a Structural Parser both inform users about patterns (PB1), Templates and Cut and Paste enable the easy creation of of complex structure (PB2), and a Structural Parser supports reflection directly (PB3). In addition, the Cut and Paste and the Structural Parser enable the support of uncommon patterns.

A combined approach thus achieves all of the benefits, although it also aggregates all of the cognitive load from each approach.

By Design and DSLs are alternative methods that are not really compatible with Templates, Cut and Paste, or a Structural Parser, but they could be combined with a Cookbook, which could help the conceptualisation problem for By Design, and provide substantial support for the complex DSL approach, and mitigate to some degree the lack of reflection inherit in both methods.

Just because the methods are compatible does not mean that they are effective, and more direct empirical work is needed to evaluate the individual methods, and to understand their interaction when combined in a single tool.

5 CONCLUSION

In this paper we have briefly covered the sorts of structural patterns typically found in IDN and then set out three potential benefits (PBs) of using them within authoring: PB1 - that they can inform authors of common solutions to problems, PB2 - they can provide a way to create complex structure quickly, and PB3 - they can provide a lens to reflect on the existing structure in an author's work.

We then explored six methods for incorporating patterns into IDN authoring tools. By using them in a *Cookbook* with explanations and examples, building them *By Design* into the authoring interface, supporting them through *Templates*, allowing *Structural Cut and Paste*, supporting them through a *Domain Specific Language (DSL)*, or highlighting them to the author with a *Structural Parser*. By mapping these methods to the benefits we have shown that none in isolation provide all of the potential benefits, nor do they all support uncommon patterns (local patterns that only apply within a given piece of work), but that they could be combined together in different ways to do so – but at the risk of increasing the author's cognitive load.

Further work is required to implement some of the methods, and evaluate them in isolation and in combination, in order to understand both their individual affordances, and their interaction effects. IDN Authoring remains a significant challenge for the research community, and is a barrier to more sophisticated interactive fiction and narrative games. Patterns have the potential to help break down this barrier and contribute to more powerful and usable future tools.

ACKNOWLEDGMENTS

Thank you to Callam Spawforth and Sofia Kitromili, whose PhD work contributed significantly to some of these ideas.

REFERENCES

[1] Christopher Alexander. 1978. *A Pattern Language: Towns, Buildings, Construction: 2* (illustrated edition ed.). OUP USA, New York.

[2] Zubair Ali Ansari, Jahiruddin, and Muhammad Abulaish. 2021. An Efficient Subgraph Isomorphism Solver for Large Graphs. *IEEE Access* 9 (2021), 61697–61709. https://doi.org/10.1109/ACCESS.2021.3073494

[3] Sam Kabo Ashwell. 2015. Standard Patterns in Choice-Based Games. https://heterogenoustasks.wordpress.com/2015/01/26/standard-patterns-in-choice-based-games/

[4] Mark Bernstein. 1998. Patterns of Hypertext. In *Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia : Links, Objects, Time and Space—structure in Hypermedia Systems: Links, Objects, Time and Space—structure in Hypermedia Systems*. ACM, 21–29.

[5] Mark Bernstein. 2016. Storyspace 3. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media (HT '16)*. Association for Computing Machinery, New York, NY, USA, 201–206. https://doi.org/10.1145/2914586.2914624

[6] Vannevar Bush. 1945. As We May Think. *The Atlantic* (July 1945). Section: Technology.

[7] Charlie Hargood, Mark Weal, and David E. Millard. 2016. Patterns of Sculptural Hypertext in Location Based Narratives. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media*. ACM.

[8] G Kaplan and G Wolff. 1990. Adaptive Hypertext. In *Proceedings of the Intelligent Systems Technical Symposium*. IBM Endicott, NY.

[9] Sofia Kitromili, James Jordan, and David Millard. 2020. What Authors Think about Hypertext Authoring. In *ACM Conference on Hypertext and Social Media (13/07/20 - 15/07/20)*. ACM, 9–16. https://doi.org/10.1145/3372923.3404798

[10] Max Kreminski and Noah Wardrip-Fruin. 2018. Sketching a Map of the Storylets Design Space. In *Interactive Storytelling (Lecture Notes in Computer Science)*, Rebecca Rouse, Hartmut Koenitz, and Mads Haahr (Eds.). Springer International Publishing, Cham, 160–164. https://doi.org/10.1007/978-3-030-04028-4_14

[11] Varun Krishna, N. N. R. Ranga Suri, and G. Athithan. 2011. A Comparative Survey of Algorithms for Frequent Subgraph Discovery. *Current Science* 100, 2 (2011), 190–198.

[12] Ora Lassila, Ralph R. Swick, World Wide, and Web Consortium. 1998. Resource Description Framework (RDF) Model and Syntax Specification.

[13] Catherine C. Marshall, Frank M. Shipman III, and James C. Coombs. 1994. VIKI: Spatial Hypertext Supporting Emergent Structure. In *Proceedings of the 1994 ACM European Conference on Hypermedia Technology ECHT 1994*. ACM Press, 13–23. https://doi.org/10.1145/192757.192759

[14] Peter A. Mawhorter, M. Mateas, Noah Wardrip-Fruin, and A. Jhala. 2014. Towards a theory of choice poetics. In *FDG*.

[15] Marjan Mernik, Jan Heering, and Anthony M. Sloane. 2005. When and How to Develop Domain-Specific Languages. *Comput. Surveys* 37, 4 (Dec. 2005), 316–344. https://doi.org/10.1145/1118890.1118892

[16] David E. Millard. 2022. Strange Patterns: Structure and Post-Structure in Interactive Digital Narratives. In *The Authoring Problem (in press)*, Alex Mitchell Ulrike Spierling Charlie Hargood, David E. Millard (Ed.). Springer-Verlag.

[17] David E. Millard, Charlie Hargood, Yvonne Howard, and Heather Packer. 2017. The StoryPlaces Authoring Tool: Pattern Centric Authoring. In *10th International Conference on Interactive Digital Storytelling (14/11/17 - 17/11/17)*.

[18] David E. Millard, Charlie Hargood, Michael O. Jewell, and Mark J. Weal. 2013. Canyons, deltas and plains: towards a unified sculptural model of location-based hypertext. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media (HT '13)*. Association for Computing Machinery, New York, NY, USA, 109–118. https://doi.org/10.1145/2481492.2481504

[19] Thomas Schedel and Claus Atzenbeck. 2016. Spatio-Temporal Parsing in Spatial Hypermedia. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media (HT '16)*. Association for Computing Machinery, New York, NY, USA, 149–157. https://doi.org/10.1145/2914586.2914596

[20] Callum Spawforth, Nicholas Gibbins, and David Millard. 2018. Uncommon Patterns - Authoring with Story Specific Structures. In *Authoring for Interactive Storytelling Workshop - ICIDS 2018*.

[21] J. R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *J. ACM* 23, 1 (Jan. 1976), 31–42. https://doi.org/10.1145/321921.321925