Software Design Patterns of Computational Creativity: A Systematic Mapping Study

Porter Glines, Isaac Griffith, and **Paul M. Bodily**
Department of Computer Science
Idaho State University
921 S. 8th Ave, Pocatello, ID 83209 USA
glinport@isu.edu, grifisaa@isu.edu, bodipaul@isu.edu

Abstract

Software design patterns can be helpful in describing the architecture of a system. Our objective is to obtain a broad overview of the current state-of-the-art of software design patterns used in Computational Creative (CC) systems. We conducted a systematic mapping study using manual and snowballing search techniques. Only 7 primary studies are identified in the CC community that explicitly mention the use of design patterns. Within these primary studies, 14 design patterns are mentioned, 12 of which are user-interaction design patterns rather than software design patterns describing the architecture of the system. The small number of primary studies indicates a gap in CC literature regarding the use of software design patterns in CC systems and motivates the need for research to identify software design patterns specific to CC systems.

Introduction

As computationally creative (CC) systems strive to become more creative, they tend to require an increasing number of behaviors. CC systems strive to have behaviors such as self-evaluation, a knowledge-base, and ultimately an understanding of the world — all of which are added to push creative systems further along in the spectrum of creative systems (Ventura 2016; 2017; Glines, Biggs, and Bodily 2020).

Each desired system behavior adds another design challenge when creating a CC system. Each design challenge comes with an opportunity to introduce elements into the system that are not easily maintainable, reusable, or understandable. To avoid common design challenges, system builders can use software design patterns to implement common system behaviors.

Software design patterns, which we will refer to as "design patterns", are general, reusable solutions to commonly occurring problems in software development. In 1994, a team of researchers referred to as the "Gang of Four" identified the foundational 23 design patterns in their work "Design Patterns: Elements of Reusable Object-Oriented Software" that has since become a standard part of collegiate computer science curriculum (Gamma et al. 1994). Since then, many more design patterns have been identified like model-view-controller, delegation, blackboard, etc. Practitioners use design patterns with the goal of improving maintainability, scalability, reusability, understandability, among other quality attributes (Zhu 2009). However, there is debate as to the effectiveness of design patterns. Some studies have shown that design patterns can negatively affect quality attributes, but conclude that more research is needed (Khomh and Gueheneuc 2008). Though design patterns may negatively affect quality, they have shown to improve maintainability (Zhang and Budgen 2012). Overall, it appears that design patterns should be applied to the problem they solve while considering the consequences they can bring.

The potential benefits of design patterns motivates the desire to identify design patterns for CC systems. Ventura (2017) identifies a general architectural pattern for building a CC system for any arbitrary domain — see Figure 1. In the architectural pattern, a system builder first chooses a domain in which the system operates. Then internal and external representations of artifacts are designed, a knowledge base of the domain is collected, a conceptualization or model is chosen to generate artifacts, an aesthetic is chosen to influence how the system learns, and finally an evaluator for artifacts is designed. Note that the architectural pattern for CC systems describes easily separated components that feed into an overarching architecture: a knowledge base, generator (model), aesthetic, and evaluator. Ventura argues that it is worth spending "significant" time searching for existing implementations for these components rather than building them from scratch. This argument for more reuse in CC systems further motivates the desire for maintainable and easily communicable code underlying these systems.

The goal of this paper is to provide a broad overview to researchers and practitioners of the state-of-the-art in CC systems with regards to the software design patterns used to build them. To provide this broad overview, a systematic mapping study will be performed as described by Peterson et al. (2008; 2015). Our systematic mapping study consolidates the use of software design patterns from 30 papers that present CC systems. This paper aims to provide insights to researchers and practitioners regarding how and when design patterns are used in CC systems to facilitate building systems that are easier to maintain, reuse, and understand.

There are currently no studies providing an overview of design patterns used in CC systems. This paper aims to fill this gap in the literature.

Proceedings of the 12th International Conference on Computational Creativity (ICCC '21)
ISBN: 978-989-54160-3-5

Table 1: Research questions along with their rationale.

	Research Question	Rationale
RQ1	*What design patterns are mentioned in CC literature?*	The aim is to explore works published in the field of CC, identify what design patterns are commonly used, and inform researchers what those patterns are.
RQ2	*Are there software design patterns devised specifically for CC systems?*	The aim is to inform researchers, especially those new to the field, of field-specific design patterns.
RQ3	*Of the CC literature that mentions design patterns, what conferences and journals are represented?*	The aim is to inform researchers when and where these papers are being published to better direct research resources.

Table 2: Inclusion and exclusion criteria for selection of primary studies.

Inclusion criteria

- English language articles.
- Peer-reviewed conferences or journals articles.
- Articles published between January 2010 and September 2020.
- Studies that relate to the field of CC.
- Studies that mention a keyword identified as relating to design patterns, whether high level or otherwise.

Exclusion criteria

- Studies that do not mention a design pattern of any kind.
- Articles that present frameworks for building CC systems but do not identify or name a design pattern.

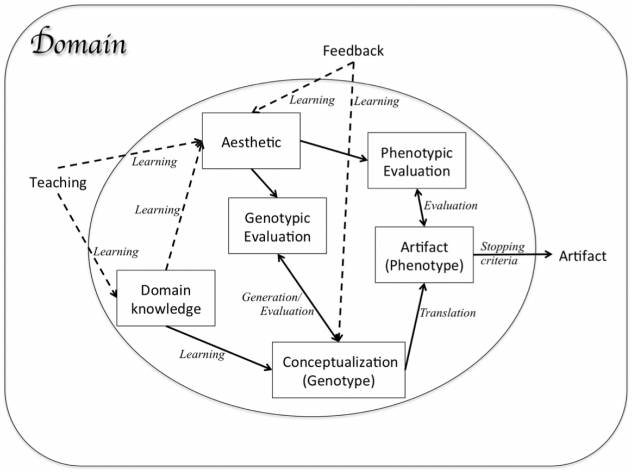

Figure 1: A diagram from Ventura's (2017) paper: "How to build a CC system". The paper describes a general approach to constructing a CC system; components described in the approach could have design patterns associated with them.

Methods

Systematic mapping studies are used to provide a broad but rigorous review of the literature with the goal of revealing gaps in the literature. The systematic mapping study in this paper is conducted as described by Peterson et al. (2008; 2015). Planning of the mapping study is described, including identified research questions and search procedures. The process by which papers are screened is described in the inclusion/exclusion criteria. Then, data extraction procedures are described. Extracted data is kept in a database to be queried for later analysis.

Planning Stage

Identified research questions and their motivations are shown in Table 1. For clarity, we define CC literature as any literature containing concepts related to CC as defined by Colton and Wiggins (2012):

> *[CC is] the philosophy, science and engineering of computational systems which, by taking on particular responsibilities, exhibit behaviours that unbiased observers would deem to be creative.*

Search Strategy The CC community has yet to establish a formal database of published literature facilitating advanced keyword search. Therefore, a manual search using the search term "design pattern" was performed on the proceedings of the International Conference on Computational Creativity (ICCC). The search term was changed to just "pattern" after an initial search yielded few papers. Once a starting set of primary studies was selected, a forward snowballing search was performed as described in Wohlin's (2014) guidelines.

Selection of Primary Studies Identified papers are then systematically marked to be included or excluded from the study. The criteria for including or excluding a paper is shown in Table 2. This criteria is first applied to study titles and abstracts; then to the introduction, conclusion, and methods sections; and finally, full papers are read to evaluate whether the study should be included as a primary study.

Identified Keywords To address RQ1 and RQ2, we use additional keywords to help detect the mention of design patterns. The 23 design patterns presented by the Gang of Four are included as keywords, e.g., "factory", "flyweight", and "mediator", as well as the three categories "creational

Proceedings of the 12th International Conference on Computational Creativity (ICCC '21)
ISBN: 978-989-54160-3-5

design pattern", "structural design pattern", and "behavioral design pattern". Additionally, a keyword is included for the "blackboard" design pattern. Generic keywords "pattern", "design pattern", and "software design pattern" are included.

Data Extraction

Papers identified as primary studies have the following information extracted from them: *title*, *year published*, *conference or journal*, and *pattern(s) mentioned*. This information is extracted and stored in a spread sheet[1] for later analysis.

Results

There are a total of 458 papers within the eleven ICCC conference proceedings from 2010 to 2020. Within the conference proceedings, a pilot search was performed with the search term: "design pattern". This preliminary search yielded only four results. Thus the search was expanded by using the more generic term: "pattern". This new search yielded 195 papers. Of these resulting studies, six satisfied the inclusion/exclusion criteria and were identified as primary studies and as a starting set to perform a snowballing search.

The forward snowballing search yielded an additional 86 new papers from the starting set of six studies. Of the newly found papers, one satisfied the inclusion/exclusion criteria.

The combined search results yield the following seven primary studies: (Compton and Mateas 2015; Concepción, Gervás, and Méndez 2019; Goel 2015; Kreminski et al. 2020; Petrovskaya, Deterding, and Colton 2020; Abdellahi, Maher, and Siddique 2020; Chang and Ackerman 2020).

Findings Regarding Research Questions

RQ1: What design patterns are mentioned in CC literature? There are 14 design patterns mentioned in the primary studies:

1. Strategy design pattern
2. Instant feedback design pattern
3. Mutant shopping design pattern
4. Chorus line design pattern
5. Simulation and approximating feedback design pattern
6. Entertaining evaluations design pattern
7. No blank canvas design pattern
8. Limiting actions to encourage exploration design pattern
9. Modifying the meaningful design pattern
10. Saving and sharing design pattern
11. Hosted communities design pattern
12. Modding, hacking, teaching design patterns
13. Turn-taking pattern
14. Biologically inspired design patterns

[1]Link to data: `https://tinyurl.com/y4w7najp`

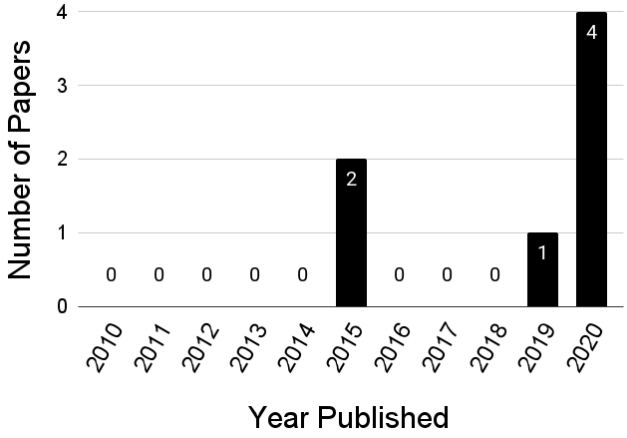

Figure 2: Number of primary studies published per year that mention design patterns and are related to CC. We can see that 2020 sees the most studies published; however, there are only seven CC studies mentioning design patterns. With so little data, it is impossible to state whether there is a trend.

Only one design pattern mentioned is a traditional "Gang of Four" design pattern, namely the strategy pattern. The strategy pattern is used in a story generation system to select and apply heuristics used to weave generated story plot-lines together (Concepción, Gervás, and Méndez 2019). Design patterns 2 to 13 are user-interaction design patterns, e.g., the instant feedback pattern where users observe an artifact, make a change, and observe the result of the change at a glance. We note that user-interaction design patterns inform system architects of how users will interact with the system. However, they do not inform system architects of how objects and classes interact within a codebase, i.e., how a codebase itself is designed, like a software design pattern would. Biologically inspired design patterns describe generic patterns by which biology is used as inspiration to solve a problem. Like user-interaction design patterns, biologically inspired design patterns do not inform system architects of how objects and classes interact.

RQ2: Are there software design patterns devised specifically for CC systems? The 11 user-interaction design patterns identified by Compton and Mateas (2015) are presented in the context of CC but are not specific to CC. Out of the seven primary studies, there are no identified design patterns that are specific to CC.

RQ3: Of the CC literature that mentions design patterns, what conferences and journals are represented? The one primary study found during the snowballing search came from the 2020 AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE). The remaining six primary studies come from various years of the ICCC. As shown in Figure 2, five out of the seven primary studies were published within the last two years (2019 and 2020).

Discussion

With only seven primary studies, the results indicate that design patterns are not often mentioned in CC studies. In other words, the results indicate a gap in CC literature in regards to the use of design patterns in CC systems.

Perhaps unsurprisingly, the design patterns most mentioned in CC literature are user-interaction design patterns. These patterns are particularly relevant to the CC community in helping design co-creative systems. While user-interaction design patterns do not describe how to design the system codebase, like a more traditional "Gang of Four" design pattern would, they do provide guidance for building the overarching system architecture. Only three papers (Chang and Ackerman 2020; Abdellahi, Maher, and Siddique 2020; Kreminski et al. 2020) present systems where a user-interaction design pattern is used, indicating that many papers presenting co-creative systems do not disclose the use of a user-interaction design pattern.

The lack of results could indicate that the CC community is unaware that they are using design patterns. In this case, CC researchers would be less likely to mention them. Ultimately, our results indicate that we cannot gain a true understanding of the pervasiveness of design patterns in CC without reviewing the actual CC systems. This suggests the need for an empirical study on CC systems as future work. An additional avenue of research would be the evaluation of Ventura's architectural pattern for building a CC system — identifying common approaches (or patterns) for implementing each of the components. The suggested future work would achieve an understanding of the patterns used in building CC systems and furthers the idea that Ventura identified an architectural pattern.

We see design patterns in the CC community as an opportunity for pedagogical benefit — a way to bring in new CC community members and programmers by facilitating systems that are easier to reuse and understand.

Conclusion

This systematic mapping study aimed to provide a broad overview of the use of design patterns in papers presenting CC systems. However, the results show that there is a lack of discussion in CC literature regarding design patterns. This motivates future work to identify design patterns in CC systems to gain a full understanding of how design patterns are used in building CC systems. The results also motivates the need to open a conversation on design patterns in the CC community to create CC systems that are easier to maintain, reuse, and understand, facilitating more collaborative research.

References

Abdellahi, S.; Maher, M. L.; and Siddique, S. 2020. Arny: A co-creative system design based on emotional feedback. In *11th International Conference on Computational Creativity*.

Chang, J., and Ackerman, M. 2020. A climate change educational creator. In *11th International Conference on Computational Creativity*.

Colton, S.; Wiggins, G. A.; et al. 2012. Computational creativity: The final frontier? In *20th European Conference on Artificial Intelligence*, volume 12, 21–26.

Compton, K., and Mateas, M. 2015. Casual creators. In *Proceedings of the Sixth International Conference on Computational Creativity*, 228–235.

Concepción, E.; Gervás, P.; and Méndez, G. 2019. Evolving the ines story generation system: From single to multiple plot lines. In *Proceedings of the 10th International Conference on Computational Creativity*, 220–227.

Gamma, E.; Helm, R.; Johnson, R.; and Vlissides, J. M. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*.

Glines, P.; Biggs, B.; and Bodily, P. M. 2020. A leap of creativity: From systems that generalize to systems that filter. In *Proceedings of the 11th International Conference on Computational Creativity*, 297–302.

Goel, A. K. 2015. Is biologically inspired invention different? In *Proceedings of the Sixth International Conference on Computational Creativity*, 47–54.

Khomh, F., and Gueheneuc, Y. 2008. Do design patterns impact software quality positively? In *2008 12th European Conference on Software Maintenance and Reengineering*, 274–278.

Kreminski, M.; Dickinson, M.; Osborn, J.; Summerville, A.; Mateas, M.; and Wardrip-Fruin, N. 2020. Germinate: A mixed-initiative casual creator for rhetorical games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, 102–108.

Petersen, K.; Feldt, R.; Mujtaba, S.; and Mattsson, M. 2008. Systematic mapping studies in software engineering.

Petersen, K.; Vakkalanka, S.; and Kuzniarz, L. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64:1–18.

Petrovskaya, E.; Deterding, C. S.; and Colton, S. 2020. Casual creators in the wild: A typology of commercial generative creativity support tools. In *11th International Conference on Computational Creativity*.

Ventura, D. 2016. Mere generation: Essential barometer or dated concept? In *Proceedings of the Seventh International Conference on Computational Creativity*, 17–24.

Ventura, D. 2017. How to Build a CC System. In *Eighth International Conference on Computational Creativity*, 253–260.

Wohlin, C. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 1–10.

Zhang, C., and Budgen, D. 2012. What do we know about the effectiveness of software design patterns? *IEEE Transactions on Software Engineering* 38(5):1213–1231.

Zhu, Z. 2009. Study and application of patterns in software reuse. In *International Conference on Control, Automation and Systems Engineering*, 550–553.