ADAPTING THE SOFTWARE DESIGN PATTERN MODEL FOR AI-ENABLED DESIGN COMPUTING

GEOFF KIMM[1], MARCUS WHITE[2] and MARK BURRY[3]
[1,2,3]*Swinburne University of Technology.*
[1]*gkimm@swin.edu.au, ORCID 0000-0002-9643-8968*
[2]*marcuswhite@swin.edu.au, ORCID 0000-0002-2238-9251*
[3]*mburry@swin.edu.au, ORCID 0000-0001-7743-7719*

Abstract. Exponential AI development requires an adaptation to new technology by traditionally reluctant architects and allied practitioners. This paper examines the potential of the software design pattern (SDP) model, used in software engineering to capture and reapply designs, as one underpinning. Patterns have creativity and pedagogical benefits in parametric modelling, yet consideration of AI and broader design computing as well as the derivation and versatility implied by an SDP model are underexamined. This research questions how, in an AI context, new patterns may evolve for varied AI levels and non-geometrical features. It is undertaken in the Unity game engine with critical application of two prominent extant patterns as a computational workflow design response to a real-world citizen engagement scenario. A novel, feature-agnostic pattern is derived with a simple AI model and is verified for other AI models. The work concludes design computing patterns can abstract existing pattern knowledge to flexibly evolve and apply across rapidly changing AI-enabled design computing contexts and thereby assist practitioners to positively respond to AI advances.

Keywords. Artificial Intelligence, Computational Design, Software Design Patterns, Architectural Practice, Unity 3D, Intelligent Agents.

1. Introduction

Computer scientist and entrepreneur Andrew Ng posits that "AI is the new electricity" and is developing exponentially (Ng, 2017). Yet as architect and Yale University emeritus professor Phillip Bernstein notes (2022, p. 131), architects typically maintain a sceptical resistance to the adoption of "new, disruptive technologies", and an enduring lag exists between AI development and its integration in architectural practice (Kimm & Burry, 2021). AI is continually evolving within a rapidly changing techno-savvy society and is in a persistent state of novelty (ibid.); since Ng spoke in 2017, broad AI models, like Midjourney or ChatGPT, and emerging industry software solutions in domains from early site analysis (see Sidewalk Labs' Delve) to generative design (see Evolve Lab's VERAS), have demonstrated potential to supplant architects' and urban designers' ("practitioners'") roles in workflow aspects. Yet full

ACCELERATED DESIGN, Proceedings of the 29th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2024, Volume 1, 49-58.

consideration of diverse project criteria requires the creative analysis and synthesis of the practitioner. These two contradictions indicate a need for support to practitioners when creating their own AI-enabled workflows. This paper examines the model of Software Design Patterns (SDPs) and assesses how it may be adapted to aid practitioners – typically inexpert programmers – in their own construction of AI-enabled design computing workflows and tools. "Design computing" is here used in the broad sense of Gero (1998) as use of "software with the relevant features and utilities to support some aspects of design activity."

2. Background and Context

The formal concept of a pattern as a reusable solution to a common problem, given independent of a specific context, was popularised with the publication of *A Pattern Language: Towns, Buildings, Construction* (Alexander et al., 1977). These Alexander patterns (APs) were adopted by the programming community, including in the seminal book *Design Patterns: Elements of Reusable Object-Oriented Software* which catalogued 23 SDPs (Gamma et al., 1995). As illustration, the *singleton* SDP ensures only one instance of a class may exist, thereby avoiding problems that can arise from concurrent access of resources such as database connections. Patterns help developers to leverage insights of past designers to avoid common pitfalls or reinventing the wheel, and to adopt a higher-level, abstract perspective to remain "in the design" for longer rather than prematurely contesting with the intricate specifics of implementing code (Freeman et al., 2020, Chapter 1; Shalloway & Trott, 2004, Chapter 5).

A systematic literature search showed limitations in use of pattern concepts in design computing. Pattern models are typically explicitly rooted in APs oriented by subsequent pattern-based approaches, and SDP consideration, if any, is narrowly based on the work of Gamma et al. As *gap 1*, although some publications referenced AI tangentially – for example Globa or Steenson (Globa, 2015, p. 61; Steenson, 2017, p. 76) – none focused on the application of patterns to the use of AI in design computing.

Foremost in the literature are the 13 Woodbury patterns (WPs) that help designers create geometry through interacting with the particulars of the parametric design software package GenerativeComponents and, more generally, as Woodbury (2010, p. 186) writes in his book *Elements of Parametric Design*, "help designers learn and use propagation-based parametric modeling systems". In the *controller pattern* (CP), for instance, a full geometric model may be controlled by a simpler linked model so that a user may interact with a model in a "clear and simple way" (ibid. p.191).

Qian (2009, p. 191) demonstrated that WPs support self-directed and formal learning, script development and the potential to introduce novel solutions, and collaborative design and practitioner communication. Globa (2015) affirms and extends the findings of Qian, although proceeds from Woodbury's book and not Qian directly. Globa tested design support through reuse of knowledge by empirical comparison of WPs, representing an abstraction reuse approach, with reuse of specific scripting solutions. The outcomes of workshop-based studies, in which participants used Grasshopper, showed that abstract programming solutions facilitate design scripting: they aid user understanding of the essential logic and modelling processes of scripting, encourage production of complex script and model outputs, and assist translating ideas into designs and exploration of novelty. Additionally, Globa (2015,

pp. 280-282) reasons these benefits would extend to other design computing domains.

Yu and Gero (2015) provided other empirical evidence of patterns by coding observed workshop design activities to the distinct stages of a design process ontology, thus inferring pattern existence though ready transitions between design process states. They conclude that if inferred patterns are extant or developed on the fly is unanalysed in their work, and urge that "if we can generalise these transitions to design patterns it would be of assistance to architects in conceptualizing their scripting process."

The model of SDPs is heterogeneous and there are a number of gaps between it and the reviewed literature. This paper focuses on two such: how patterns arise, and their flexibility across computational environments.

Gap 2 concerns pattern derivation. An SDP is a response to a "recurring" design problem (Buschmann et al., 1996, Chapter 1; Freeman et al., 2020, Chapter 13) and should express "successful designs" (Gamma et al., 1995, p. 1). The part of experience of software designers in deriving patterns is clear: patterns come from the "collective experience of skilled software engineers" and "prior experience" of software designers that is "learned by designers and users over the years" (Buschmann et al., 1996, Chapter 1; Gamma et al., 1995, p. 1; Shalloway & Trott, 2004, pt. 1).

Qian demonstrated pattern derivation through forming a large corpus of design examples in participant observer workshops followed by interview coding and other analysis. The process, although sound, cannot serve to identify pattern use in rapidly evolving contexts or in the informal settings of practice. The method of Yu and Gero could in principle identify pattern existence in even a singular workflow, but does not define patterns. Woodbury (2010, p. 189) asserts the importance of the experience of the designer: "To write a pattern is to listen to yourself and your colleagues." In the works surveyed, the particulars of how an individual designer may derive a pattern are left unstated. As Qian (2009, p. 191) concluded, the question of "how a pattern… can be polished and enriched by designers' active involvement and communication" remains open. The "pioneering spirit" of Pemberton and Griffith (1998), who derived patterns for collaborative workspaces and technology, offers a possible response – a pattern once found could prompt the identification of another pattern per new needs.

Gap 3 concerns pattern versatility. An SDP typically has enduring use across diverse computational environments; the singleton SDP, for instance, is as applicable to C++, released in 1985, as it is to Rust, released in 2015. In contrast, idiomatic SDPs are particular to a given language (Buschmann et al., 1996, sec. 1.3). An idiom useful for one programming paradigm is not necessarily applicable to another paradigm.

The patterns models surveyed tend towards idiomacy. Globa extended WPs from GenerativeComponents to Grasshopper but they are still within the domain of propagation-based parametric modelling systems. Furthermore, focus is on geometrical primitives rather than on manipulation of more general built environment features such as structural members, circulation paths, houses, or styles.

3. Research Questions and Method

The lead author designed and conducted a research-through-design (RtD) investigation to test SDP potential at those three gaps. RtD is concrete research through the action of the practitioner to create an experimental object within and regulated by a

specific practice context (Herriott, 2019). The object is a means to uncover and convey knowledge, not the end goal. Two complementary research questions on pattern derivation and versatility in an AI-attuned design computing practice context were considered. On derivation, what potential exists to evolve new patterns from an existing pattern? On versatility, may such patterns be applicable to different types of AI and features other than geometry alone?

An extensible model of AI is essential to address the aspect of practitioner use of diverse AI. This research uses an adaptation to design of a computer science model of intelligent agents (IAs) that perceive, decide, and act (Kimm, 2022). Beyond the agent scale, the model considers AI from a teleological perspective of the designer providing value to the client rather than analysing the underlying AI technical mechanics directly. It offers a framework of increasing AI sophistication that, rather than imposing an arbitrary, inflexible partitioning of AI and not-AI, sets out a continuum from AI that reflexively responds to stimuli to AI with a complex internal world model.

Initial patterns are taken from WPs for their well-described, prominent, 13-strong pattern catalogue of demonstrated practitioner benefit. Two are selected in the next section for their model clarification and abstraction ability.

The practice context which enables consideration of features other than geometry is an engagement with Aurecon – a global design, engineering, and advisory company – and its online community engagement software tool commissioned by a government client which sought citizen stakeholder preferences for the design of a harbourside park in central Sydney, Australia (https://harbourpark.sydney/). Website users may select and position on a bare site three to five 2D park feature graphics from a library of fourteen cultural, health/safety, landscaping, pedestrian, or recreational features (Figure 1). An aim, per the tool website, was to permit a user to create a design that "will contribute valuable information that will help us determine how the park will look and feel based on the features and activities that matter to you." Graphics are animated on short loops – for example, a person at an exercise station is shown using it – and resize to give a sense of perspective. The graphics are otherwise invariant: the user is presented with little scope to indicate any preferences in this static framework.

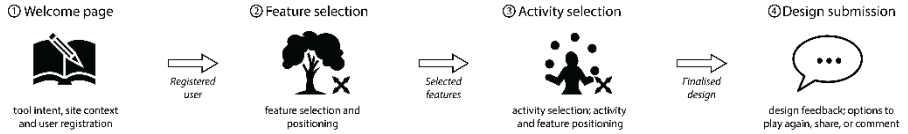

Figure 1 The workflow of the harbourside park app.

The research computational starting point is an initial study on capturing citizen street tree planting preference, only imprecisely given, and then making a definite move in a search space of appropriate trees. A user interacting with this may trace a desired tree massing profile on the screen. The prototype then analyses the path of the user's indicative 2D line, using the *$1 Unistroke Recognizer* algorithm (Wobbrock et al., 2007), and suggests one option from a curated library of 3D precedents (Figure 2).

A design goal was thus formulated for reimaging the harbourside park app (HPA):

How may the user have more latitude to convey their preferences

imprecisely or incompletely, and how may they receive guidance as they develop their design?

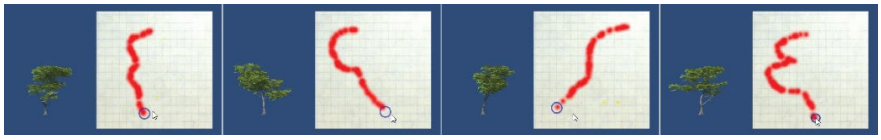

Figure 2 The base computational exploration.

The Unity game engine was selected as the research platform for its industry use, capacity for design analysis, and accessibility to non-programmers (Huang et al., 2021). It supports non-parametric modelling and, via C#, flexible exploration of AI.

Prototype evaluation was by interactive demonstration to and interview of Aurecon employees. As the research focus is the pattern exploration process, rather than high prototype utility and usability, a response to the design goal need not be unconditionally successful but must demonstrate in its seriousness that the investigation took place in a bona fide design computing practice scenario. Prototype evaluation is thus restricted.

4. Results

Two WPs were selected as pattern exploration seeds for their capacity to map user design intent expression to some point in the search space of a design computing tool, as implied by the design goal. The controller pattern (CP) was described above; in the *reactor pattern* (RP), an object responds to another object as an "interactor point" within a common modelling environment. Each simplifies the set of inputs to a model and is conceptually abstract rather than narrowly geometry oriented. Nonetheless, they, although applicable to placement of park features, for instance, do not fully address the design goal that deals with the non-concrete domain of flexible preference expression.

4.1. APPLYING THE WOODBURY CONTROLLER PATTERN

The initial computational study embodies the CP to a limited degree: the specification of a tree massing as a direct one-to-one mapping of sketched profile preference to sampled tree elevation fits the *abstracting* CP classification of Woodbury (2010, p. 192) of a "simple version of the main model that suppresses unneeded detail". The *transforming* mode of the CP, which "changes the way you interact with a [full] model" (ibid. p192), suggests a brokering model may be inserted into the full model to allow nuanced expression of the feature preferences within the wider environment.

Accordingly, the initial prototype was updated such that a user could express a tree preference – shading provided by a tree – distally by interacting with the model environment rather than proximately by direct selection. The processing of the library of trees was redeveloped so that it may be searched by shading profile rather than elevation profile as in the initial prototype. In the iterated tree workflow (Figure 3), a user clicking on a grass canvas places a new tree and can draw the desired shading profile. The system displays the best-match 3D tree and the profile stored within the $1 gesture recogniser. Any instantiated tree can be selected and repositioned by the user.

The resulting prototype has limitations as a simplified model of environment

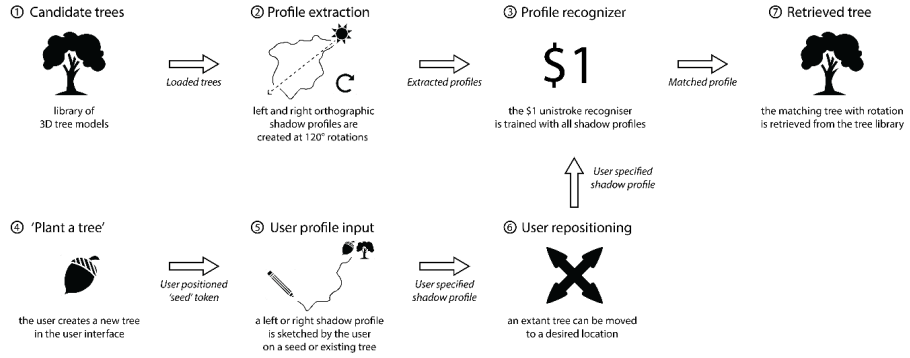

Figure 3 The workflow of the prototype following consideration of the controller pattern.

conditions and user interaction. First, shadow sketching is supported only for drawing on flat ground. Second, due to preprocessing, only one time of day can be supported. Third, the sketching method permits a user to specify one tree feature preference with a greater dimensionality than can be captured readily in a list of choices, yet the treatment within the prototype, once that sketch is analysed by the gesture recogniser, is precise – the prototype never provides options where ambiguity of choice may exist.

4.2. APPLYING THE WOODBURY REACTOR PATTERN

The HPA provides guidance only on submission of a design and a user cannot change any aspect of a feature to a preferred state by indirect or direct interaction. To test the RP for this, a matrix was developed to analyse the potential for RP application if taking features, rather than geometry, as the material of that pattern. The matrix sampled tree, light, and path features from the HPA to model how a feature may respond to the proximity of a RP interactor point which, rather than being an abstract position in space with no other associated quality, is itself a feature in this treatment. The RP, rather than operating on geometry that responds in its position, direction, or scale, here provides a schema for features to dynamically moderate their design role.

Matrix feature response analysis reveals a deficiency of direct use of the RP. If treating the RP interactor point as a feature, rather than as an abstract position in space, the interactor point must itself change. As example from the matrix, a path feature as an interactor correctly induces in nearby trees a change to low branchfall risk. Conversely, a tree moving as an interactor near to a path causes a path change, such as to preserve a clear root buffer zone, yet is itself not changed: it takes on no attribute of low branchfall risk, and thus a risk to public safety is not improved. Branchfall in this example is a proxy for any interactor feature attribute that requires moderation.

Therefore, an adaptation of the RP must be reciprocal: any feature engaged in the model should operate both as an interactor and a reactive result. Counterintuitively, given the criticism of one-sided interaction directly above, this symmetry indicates that exploration of reactive response in the reactor pattern might, for a suitable subset of features, be realised in the developing prototype through a reactive response in either of the moving interactor or the stationary features. Moreover, a focus on the reactive

response of a moving feature facilitates consideration of features that are static for reason of being preexisting site conditions (e.g. overshadowing) or of being fixed in an earlier stage of the design process (e.g. primary pedestrian circulations).

The workflow was updated so that a tree species is selected for a site location by its appropriateness (Figure 4). Each tree library tree is thus tagged with its characteristics. In this test, the attribute set is limited to shade tolerance, soil condition, and branchfall risk. These three measures represent the many possible factors and potential trade-offs that could be incorporated, including considerations of hydrology, heat island effect, or bird life. The notional tags do not reflect the actual species requirements; however, tag selection was curated to create a crude congruence between the tagged requirements and the appearance of the tree models. The Unity implementation is seen in Figure 5.

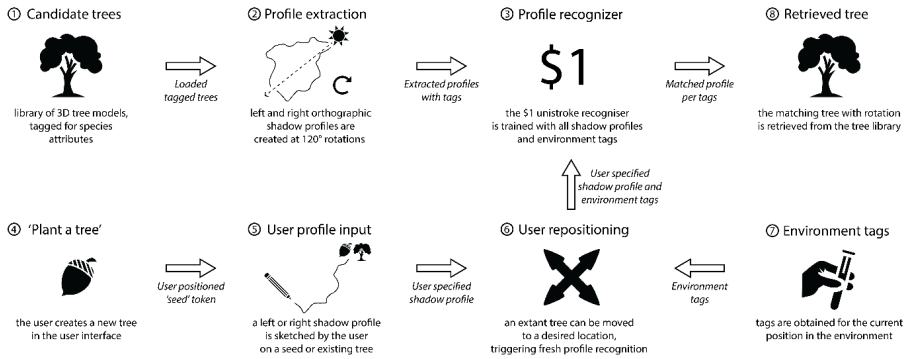

Figure 4 The workflow of the prototype following consideration of the reactor pattern.

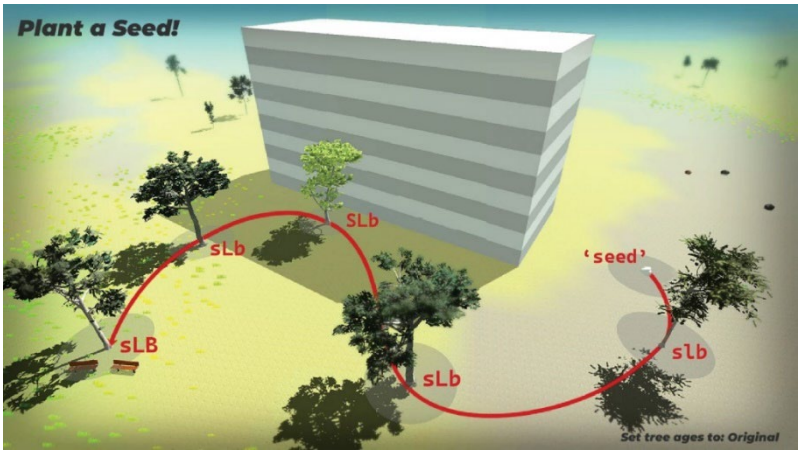

Figure 5 The progress of one tree in the final prototype using the workflow of Figure 4.

Despite limitations noted previously, workshop feedback from two Aurecon leaders in digital innovation and engagement indicated the outcome of consideration of the CP and RP in the prototype design process met the design goal and could usefully

extend the HPA workflow, particularly for diverse government-based projects and establishing social licence. Nonetheless, separate informed comment by an academic and landscape architect formally interviewed on the CP stage results warned against uncritical use: "design is much more comprehensive than just placement of objects."

4.3. A SYNTHETIC NOVEL PATTERN

Critical application of the encapsulated knowledge of two WPs to the design problem evolved a novel method. That solution is here expressed as a pattern using the four prime descriptors urged by Woodbury: "Name, What, When and How" (2010, p. 189).

Name: Polymorphic Feature (PFP).

What: Allow a feature to change its own nature according to its context.

When: Useful if feature solution space position information is initially incomplete.

How: Model a feature with an IA interface. Develop one IA to update model attributes per dynamic environment conditions, bounding the solution space search by domain knowledge and any original user intent. Implement new IA models as required.

4.4. CROSS-DOMAIN APPLICATION OF THE NOVEL PATTERN

To test the PFP and its capacity to harness diverse AI, the simple, reflexively responsive IA of the prototype was replaced with an IA of a more complex internal world model. This, as a proof of concept, was carried out manually using commercial large language model and text-to-image model services. For each new tree position, the prototype output a ChatGPT prompt customised with context specific tags (in italics below):

> Determine a commonly used tree species for a *sunny* city park location with *loam* soil in Sydney, Australia. You must consider factors of urban planning, botany, landscape architecture, and other relevant disciplines. Then write a text-to-image (Midjourney) prompt of a *mature* example of that species. The prompt should follow this formula: "/imagine prompt: [subject], [your descriptive keywords or phrases], [background] [art style] --ar [width]:[height]". Use a low poly style, and a plain white removable background is essential. Adjust the aspect ratio ("--ar") as appropriate for the dimensions of your selected tree.

Figure 6 Midjourney outputs for the quoted prompt (two leftmost images), prompted juvenile examples (two centre images), and a reference blue gum photo (rightmost image).

An example generated Midjourney prompt for a recommendation of *Eucalyptus globulus* (blue gum), which also appears in the City of Sydney tree species list (swi.nu/trees), is below with output and variations in Figure 6:

> /imagine prompt: Towering Blue Gum Tree, silver-blue leaves shimmering in the sunlight, providing an iconic silhouette in the city park. The low poly style enhances its grandeur. Plain white removable background for transparency. --ar 4:5

5. Discussion and Conclusion

This investigation has shown an adaptation of the Software Design Pattern (SDP) model for the construction of AI-enabled design computing workflows and tools. Aspects of SDPs have been applied by others in design computing and in depth for the subfield of propagation-based parametric modelling, most prominently as Woodbury patterns (WPs). Those uses have clear design space exploration and pedagogical benefits but consider the SDP model as one precedent of many and, as one gap, do not specifically consider AI. Consequently, further gaps exist of which two were questioned here: the derivation and versatility of patterns in an AI context.

Adapting the SDP model, the lead author demonstrated pattern derivation utilising the intuition and experience of the designer. Critical application of two WPs, as conceptual framings for a design computing tool, guided evolution of an initial computational study. Significantly, this direction occurred though clear congruence with the design problem as well as points of divergence. Incompatibilities prompted exploration of novel reaches of the design solution space, in contrast to the pattern chaining of Pemberton and Griffith in which a pattern is linked to its ancestors by its consonance alone. The resultant polymorphic feature pattern (PFP) is not asserted to be novel in the sense of Boden's H-creativity and it does not need to be: patterns capture past experience. Nonetheless, it is a P-creative, novel addition to this design computing practice scenario. This work is a complement to that of Yu and Gero and their call to generalise to patterns the design process transitions they inferred. It is, however, a single datapoint that illustrates a generalisation mechanism; group participant studies could offer valuable empirical evidence but the challenges of unadulteratedly recording design process decisions to the degree done so here would be considerable.

The investigation exhibited versatility of patterns in respect to technology domains and to built environment considerations. The former had two aspects. In the selection of WPs, the design process exploited the encapsulated knowledge of past designers in parametric computing and applied it to the distinct Unity environment. Further, it demonstrated how a pattern may apply across an AI continuum of intelligent agents (IAs) by developing the PFP with an IA model of mere instinctual responsiveness and then showing how it can apply with contemporary, sophisticated AI models. While those outputs were not perfect – see the juvenile blue gum images of Figure 6 – they do provide a robust proof of concept. Regarding built environment considerations, this research shows how a pattern can operate not only on geometric elements, as in the precedents, but also broadly on actual built environment features. The derived novel PFP hence aids a freedom of action in expressing a feature with subsequent guidance – it facilitates exploring what-if scenarios on a micro scale.

This pattern investigation and framework developed in a real-world project context. Stakeholder workshop feedback substantiated a reasonable design computing practice scenario took place. Although the primary research focus was at a design thinking level,

the pattern found has a measure of external validity despite its narrow spectrum of designer experience, insofar stakeholders favourably judged it in terms of its method and the design goal. The generative AI reuse also supports validity, as does background PFP use for the prototyping within a past study of different focus (Kimm et al., 2023).

The adaptation of the SDP model in this paper signposts a way to further, broader adoption of patterns in industry by which design computing practitioners can be supported in their professional service to clients and society. They, with such patterns and as AI continually evolves, can apply existing knowledge flexibly across computing contexts and in consideration of features other than bare geometry. This research enlivens the toolkit available to practitioners to construct their own design computing workflows. It proposes a viable, enriching direction for further research into how architects may positively respond to AI advances and sustain their creative practice.

References

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language*. Oxford.

Bernstein, P. (2022). *Machine learning: Architecture in the age of AI*. Routledge.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-oriented software architecture. [Vol. 1], A system of patterns*. Wiley.

Freeman, E., Robson, E., Sierra, K., & Bates, B. (2020). *Head first design patterns*. O'Reilly.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns: Elements of reusable object-oriented software*. Pearson Deutschland GmbH.

Gero, J. S. (1998). Research in design computing: An artificial intelligence framework. *International Conference on Artificial Intelligence for Engineering*, (pp. 5-12).

Globa, A. (2015). *Supporting the use of algorithmic design in architecture: An empirical study of reuse of design knowledge*. PhD thesis, Victoria University of Wellington.

Herriott, R. (2019). What kind of research is research through design. *IASDR 2019*.

Huang, X., Kimm, G., & Burry, M. (2021). Exploiting game development environments for responsive urban design by non-programmers. *26th CAADRIA*, 2, (pp. 689-698).

Kimm, G. (2022). Classes of AI tools, techniques, and methods. In I. As, P. Basu, & P. Talwar (Eds.) *Artificial Intelligence in Urban Planning and Design* (pp. 61-83). Elsevier.

Kimm, G., & Burry, M. (2021). Steering into the Skid: Design Augmentation to Arbitrage Human and Artificial Intelligences. *40th ACADIA*, 1, (pp. 698-707).

Kimm, G., White, M., & Burry, M. (2023). Extending Visuospatial Analysis in Design Computing. 28th CAADRIA, (pp. 655-664).

Ng, A. (2017). 'AI is the new electricity'. *AI Frontiers Conference*, Santa Clara, Calif, 3 Nov. Available at: www.youtube.com/watch?v=JsGPh-HOqjY (Accessed: 3 Sept. 2023).

Pemberton, L., & Griffiths, R. N. (1998). The timeless way: Making living cooperative buildings with design patterns. *CoBuild'98*, 1, (pp. 142-15).

Qian, Z. C. (2009). *Design patterns: Augmenting design practice in parametric CAD systems*. PhD thesis, Simon Fraser University.

Shalloway, A., & Trott, J. (2004). *Design Patterns Explained: A New Perspective on Object-Oriented Design*, Second Edition (2nd edition.). Addison-Wesley Professional.

Steenson, M. W. (2017). Christopher Alexander: Patterns, Order, and Software. In *AI: How Designers and Architects Created the Digital Landscape* (pp. 21-76). MIT Press.

Wobbrock, J. O., Wilson, A. D., & Li, Y. (2007). Gestures without libraries, toolkits or training. *20th ACM Symp. on User Interface Software and Technology* (pp. 159-168).

Woodbury, R. (2010). *Elements of parametric design*. Routledge.

Yu, R., & Gero, J. (2015). An Empirical Foundation for Design Patterns in Parametric Design. *20th CAADRIA* (pp. 551-560).