Context Familiarity in the use of Computational Design Tools

An experimental study across expertise levels

Xavier Garnavault[1]*, Aurélie de Boissieu*[2]
[1,2]*ULiège*
[1,2]*{xavier.garnavault|aurelie.deboissieu}@uliege.be*

While computational design tools and practices have been developing at a fast pace, their adoption in architecture remains relatively limited. This paper investigates how users of various profiles engage with pre-existing computational design logic, focusing on whether a lack of familiarity with a given toolset constitutes a significant limitation to its effective use. It examines how context familiarity influences interactions with a parametric façade-generation script implemented across three distinct interfaces. By combining automated logging of user behaviour with survey-based feedback, the study highlights both an initial "learning phase" and the gradual convergence of behaviour over time—underscoring that once users grasp the underlying computational logic, skill disparities diminish. Rather than emphasizing tool-specific proficiency, the results point to how interface constraints and parameter framing affect exploration and engagement. These findings reveal that well-structured parameters and interface transparency are as critical as prior expertise, suggesting that even novices can successfully engage with Computational Design when provided intuitive, open-ended environments. As a pilot exploration studying behavioural analytics based on automated data logging, this work opens avenues for further research into how interface design, parameter clarity, and user-learning trajectories can foster broader industry uptake of computational methods.

Keywords: *Computational Design, Computational Thinking, Behavioural Analytics, Design Cognition, Pilot Study.*

INTRODUCTION

While Computational Design tools and practices have been developing at a fast pace over the past years (Deutsch, 2019; Bernstein, 2018; Menges and Ahlquist, 2011; de Boissieu, 2022), their adoption in industry remains limited (Deutsch, 2019; de Boissieu, 2022). Despite ongoing technological advancements, Computational Design remains a niche practice.

Generative design and AI-driven approaches have already been explored in architectural contexts (Bernstein, 2022; Chaillou, 2022; Leach, 2021; Heumann and Davis, 2019). However, while expert roles and skillsets have evolved, a critical mass of practitioners remains unfamiliar with computational thinking and its application during the design stage.

In response, new strategies and toolsets—such as "scripts as a service"—have emerged through simplified interfaces or platforms (e.g. Hypar, TestFit). While these platforms attract attention within expert communities, their broader adoption remains limited.

This paper aims to investigate how users of varying expertise interact with predefined Computational Design tools, and whether a lack of familiarity with a given toolset limits their effective use. Such insights into how users engage with computational design logic can support tool development aimed at broader adoption.

The scientific context and related works are presented in the next section. Section three introduces the experimental methodology. Sections four and five present the main results: a characterization of user interactions, followed by an analysis based on context familiarity. The discussion addresses the implications, limitations, and perspectives for future research.

RELATED WORK

While most digital tools in architecture support existing practices, Computational Design introduces a paradigm shift through the integration of computational thinking (Menges and Ahlquist, 2011; Carpo, 2017; Caetano, 2020). It is important to distinguish between "computer-aided" and "computational" approaches in architectural design (de Boissieu, 2022).

Despite the significant potential of Computational Design (Menges and Ahlquist, 2011; Bernstein, 2018; Carpo, 2017; Denning and Tedre, 2019), its adoption remains limited (Stals, 2019; Gardner, 2019, 2022; de Boissieu 2020). Key barriers include:

- its perceived complexity (Stals, 2019)
- the difficulty of learning new tools in a professional setting (Gardner, 2019; Gardner, 2022; de Boissieu, 2020)
- lack of available time (Gardner, 2019)
- resistance to changing established design methodologies (Stals, 2019; de Boissieu, 2020)

Visual programming environments, such as Grasshopper, have lowered entry barriers by allowing users to create scripts using graphical interfaces. Nonetheless, these tools and environments still require users to learn new systems and are more designed for creating rather than using existing algorithms. Thus, computational thinking remains mostly associated with expert users (Deutsch, 2019; de Boissieu, 2020).

Recent platforms such as Hypar, TestFit or ShapeDiver attempt to address this by providing simplified, often cloud-based interfaces that allow non-specialists to use CD tools without developing them. While still niche, this model, focusing on using rather than authoring, shifts the accessibility conversation. This study aims to investigate that shift: not just in interface accessibility, but how users interact with and adapt to predefined computational logic.

Studies that examine the adoption of digital tools in construction and architecture typically focus on:

- **Visualization**: including Virtual Reality (Ergun, 2019), Augmented Reality (Silcock, 2021), and game environments (Hegazy, Yasufuku, and Abe, 2019).
- **Control and Simulation**: particularly in relation to BIM tools, compliance checking (Beach, Hippolyte, and Rezgui, 2020), simulation (Sun, 2021), or construction site monitoring (Raimbaud et al, 2019).
- **Collaboration**: including studies on digital coordination and data exchange (Meng, 2020; Shih, 2017; Calixte, 2021).

Some research also explores tool ergonomics for non-expert design participants, such as future inhabitants (Kwiecinski, 2023). However, very few studies focus on how architects and engineers unfamiliar with scripting engage with pre-authored computational tools.

A few notable works (Davis, 2014; Chen, 2019; Ataman and Dino, 2021; Gardner, 2019; Stals ,2019; Ercan, 2015) have addressed aspects of

Computational Design usability and accessibility, but a gap remains in understanding how unfamiliar users approach pre-authored computational tools, especially across varying interfaces.

METHOD

To investigate how users interact with existing Computational Design (CD) logic, we conducted an experiment with fourteen engineer-architects. Seven were current students, and seven were recent graduates, reflecting a range of familiarity with 2D/3D modelling, BIM tools, and scripting environments such as Grasshopper, with varying levels of expertise in computational thinking.

Computational design logic

A parametric façade-generation algorithm formed the backbone of the experiment. From a base surface and adjustable parametric openings, this logic randomizes panels across the façade following a set of defined parameters. This script was selected for its:

- **Relevance**: Façade panelling is a common architectural task.
- **Open-Endedness**: Participants could pursue diverse design outcomes by tweaking parameters like panel dimensions, spacing, and randomization seeds.
- **Manageable Complexity**: While fully parametric, the logic remained accessible to a broad user group.

Participants had to complete three façade-design exercises during the experiment, each with unique dimensions and placements but using the same underlying algorithm. This ensured a consistent basis for comparing interface-driven behaviours.

The three interfaces

To explore how interface context influences user adoption of pre-authored computational logic, we exposed the same façade algorithm via three distinct means: a command-line interface, a form-based interface, and a visual programming definition.

Although all three interfaces accessed the same underlying logic, they differed in how they exposed and constrained parameter manipulation: the command-line required precise text input with minimal constraints; the form-based interface used labelled sliders and fields with bounded ranges; and the visual programming option offered partial visibility into the script via a Grasshopper definition with connected nodes and controls.

Despite sharing the same algorithm, each interface afforded different interaction styles and levels of transparency, allowing us to compare how users engaged with the parametric logic under varying conditions.

Experimentation protocol

We structured the protocol to gather both quantitative and qualitative data from user interactions:

- **Preliminary survey**: Each participant answered questions about their background, with a specific focus on prior exposure to CD tools and programming mindsets. This helped categorize skill levels (e.g., BIM-oriented, CAD-only, advanced scripting).
- **Interaction with the computational design (CD) tool:** Each participant tackled the three façade-design exercises in a random sequence of interfaces. No strict time limit was imposed; participants decided when each design was "complete." This phase captured both **qualitative** (user impressions, ease or difficulty, through speak aloud method) and **quantitative** (parameter-change logs) data.
- **Final Survey:** Upon completing all three exercises, participants filled out a questionnaire to gauge on a scale of 1-10:

1. **Accessibility (user-friendliness)** – initial difficulty for new users.
2. **Ease of Use** – overall difficulty of operating the tool.
3. **Comprehension of the CD Logic** – the extent to which they understood the underlying façade algorithm.
4. **User Satisfaction** – whether they felt the tool allowed them to achieve desired design outcomes.
5. **Usability** – how likely they could see such a tool integrating their daily work practices.

Data collection and indicators

An automated logging system recorded every parameter modification—logging the timestamp, name of the parameter, and the updated value. Across fourteen participants and three exercises each, a total of 92,990 data points were collected.

From the cleaned dataset, we derived three key performance indicators of user behaviour, aggregated in one-minute intervals for comparative analysis:

- **Iteration rate**: Number of parameter changes per minute. This reflects how intensively participants manipulated values at any given time.
- **Switch rate**: Number of times per minute participants changed *which* parameter they were editing (e.g., from "Panel Width" to "Seed Randomness"). A high switch rate often indicates broad exploration of multiple parameters within short intervals.
- **Novelty rate**: Number of *new* parameter-value combinations generated per minute. This provides a partial gauge of how diverse the design exploration was, distinguishing between minor tweaks and truly distinct design configurations.

These indicators formed the basis for statistical comparisons across participants, interface types, and levels of experience. By correlating the logs with survey feedback, we captured both quantitative interaction patterns and qualitative user perceptions, enabling a comprehensive analysis of non-experts' engagement with a pre-authored computational design logic. An extensive data visualization workflow was developed to perceive initial findings in the data. All the elements of this workflow (code and figures generated) as well as the computational design script, interfaces and logging tools used in the experiment and the data collected can be found on the author's GitHub (github.com/XGar). The following sections detail our results, including overall interaction trends and how prior familiarity shaped participant behaviours.

OVERALL CHARACTERIZATION OF INTERACTIONS

In this section, we present key findings about user interactions with the façade-generation logic. We begin with participants' software backgrounds, then assess how interface order influenced their impressions and interaction patterns.

Characterization of participants' software proficiency

The preliminary survey (see Method) revealed that most participants were proficient in basic 2D and 3D modelling (e.g., AutoCAD and SketchUp), but less experienced with Rhinoceros and Grasshopper, the main software tools employed in this experiment. Based on these self-assessments, three user profiles were identified:

- **"Computational designer"**: Grasshopper proficiency and experience. These subjects also typically had experience in BIM or other 3D modelling.
- **"BIM modeler"**: Little to no proficiency in Grasshopper, but at least intermediate Revit

or ArchiCAD proficiency (all also had 3D/CAD experience).

- **"3D modeler"**: No or little experience in Grasshopper, Revit, and ArchiCAD, but stated proficiency in Rhino, AutoCAD, or SketchUp.

Impact of interface order on user ratings

Initial survey data suggest that the order in which participants encountered each interface strongly influenced their perception. When the visual programming interface (e.g., Grasshopper) was used first, it was rated favourably in terms of accessibility, usability and satisfaction. However, if participants had already used a different interface, their subsequent ratings for the visual programming tool tended to be slightly lower, especially regarding ease of use, comprehension and satisfaction.

By contrast, participants' experience of the form-based interface showed a mild increase in comprehension over repeated usage, but ease of use, usability, and satisfaction tended to decrease.

The command-line interface exhibited an opposite trend: while initially perceived as more difficult, it scored progressively higher in subsequent surveys as participants became familiar with the underlying logic. This progression highlights the significant role of prior exposure in shaping user attitudes and underscores the influence of temporal bias in perception.

Overall perception of the interfaces

On average, participants rated both the visual programming and command-line interfaces similarly in terms of satisfaction, with slightly lower scores attributed to the form-based interface.

The visual programming interface was perceived as more approachable, while the command-line interface was seen as more efficient. The form-based interface, although familiar to many, was viewed as more limited in supporting exploratory workflows.

These impressions reflect how interface design mediates both usability and creative freedom.

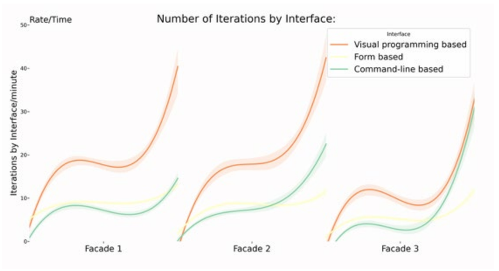

Figure 1
Number of iterations by Interface

Characterizing interaction patterns across interfaces

Data from the automated log system showed clear differences in how frequently users modified parameters under each interface. Participants using the visual programming interface averaged 17.6 iterations per minute, compared to 7.7 (form) and 8.1 (command-line). These differences were especially early on; however, by the third exercise, iteration behaviour across interfaces began to converge as can be seen on figure 1 and figure 2. This suggests that once participants understood the underlying façade-generation logic, the specific interface mattered less for basic parameter changes.

On average, participants in the command-line environment switched among different parameters at a slightly higher rate (4 per minute) than in the form (3.6) and visual programming (3.6) interfaces, reflecting increased comfort with text once the logic became familiar.

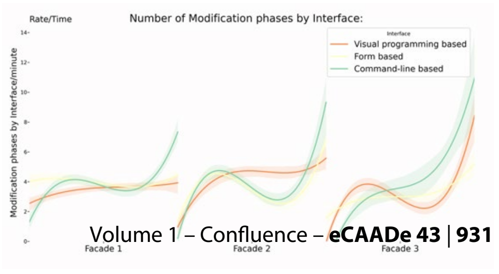

Figure 2
Number of Modification phases by Interface

CONTEXT FAMILIARITY IMPACTED BEHAVIOR

In addition to evaluating overall user interactions, we examined whether users' profiles influenced how they interact with an existing CD tool. This allowed us to assess how distinct user backgrounds shaped behaviours or performance.

Influence of preexisting Computational Design experience

Survey responses at the experiment's conclusion suggested that participants with a Computational Designer profile found the visual programming interface more approachable than other interfaces. Contrary to initial expectations, however, these users reported slightly more difficulty grasping the logic behind the façade-generation algorithm in the visual programming interface, rating their comprehension the lowest among all groups. One possible explanation is that user profiles with CD experience are accustomed to scripting their own logic from scratch; encountering a pre-built logic with its own parameters can introduce unfamiliar or unintuitive "black box" structures.

In terms of behavioural statistics (time taken, average rate, and number of values of the key indicators), CD profiles took slightly longer to complete the exercises on average (11.9 minutes) than BIM (10.7) and 3D modeler (9.9) profiles. The total number of values of the performance indicators was similar across user profiles, leading to slightly lower average rates for computational designers.

Interestingly, satisfaction in achieving final design goals did not vary much between user profiles. This indicates that prior CD experience did not significantly help or hinder the use of a pre-existing novel CD tool.

Interaction logs and behavioural patterns

Analysis of the automated logs confirmed that interface type, user profile, and the iterative nature of the task all influenced user behaviour. Drawing on the three data indicators, we observed initial differences across users and interfaces that tended to converge by the experiment's end.

In line with participants' survey feedback, the visual programming interface prompted higher iteration rates early on, particularly among non-Computational designer profiles, who engaged in trial-and-error exploration. This provided a tangible way to "learn" the logic. Interestingly, more experienced Grasshopper users exhibited lower initial iteration rates, opting for more deliberate scanning and prediction of changes, rather than testing every parameter.

The command-line interface allowed wide parameter ranges, sometimes resulting in extreme or invalid values (e.g., extremely large panel sizes) which could hinder the usage of the tool. CD user profiles completed the exercise in significantly less parameter changes despite taking a similar amount of time, leading to lower rates across the indicators.

In contrast, the form-based interface, which restricted parameter ranges, reduced the risk for invalid inputs but also limited novelty as can be seen on figure 3.

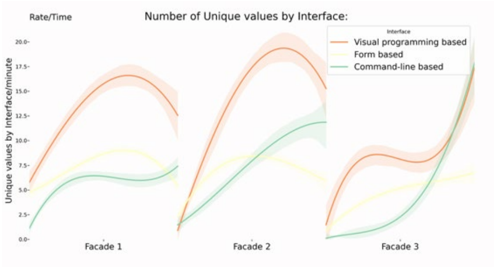

Figure 3:
Novelty rate by interface

Nonetheless, novices in the form-based setting often appreciated the "safety" of predefined boundaries, as it kept them from accidentally "breaking" the algorithm. Expert users, however, voiced frustration at not being able to exceed slider limits, reinforcing that interface constraints can affect creative exploration.

By the third and final exercise, the iteration rate, switch rate, and novelty rate tended to converge across all interfaces and all user profiles. Initially stark differences gave way to a more uniform pattern of interaction. This convergence implies that once participants understood the façade-generation logic, they relied less on interface-specific behaviours and more on consistent design tactics. As a result, overall design satisfaction scores also showed minimal variance by the end of the session, suggesting that prior expertise mattered most in the early learning phase, then diminished in influence.

Taken together, these patterns support the notion that context familiarity exerts its strongest effect at the beginning of computational design tasks, and that interface design can help unify user approaches once the underlying logic becomes clear.

Summary of observations on context familiarity

Taken together, our results indicate that prior familiarity with CD tools and techniques has a less pronounced impact on usage than one might assume. While users with advanced Grasshopper skills do approach an unknown logic somewhat differently—and may initially find it counterintuitive—the tool's transparency and parameter design appear to matter just as much. Over time, novices adapt quickly, levelling the field.

These insights highlight that an intuitive interface and well-structured parameters can mitigate initial skill disparities. They also suggest that, for broader industry uptake, focusing on the clarity of computational logics and interface design could be as critical as addressing users' coding backgrounds or software experience.

DISCUSSION

This section reflects on the outcomes of our exploratory experiment, situating our findings in the broader landscape of Computational Design (CD) adoption. We then consider the limitations inherent in this pilot study and propose directions for more extensive future research.

Summary of the Exploratory Findings

Our experiment, though limited in scale, offers preliminary insights into how non-expert users engage with pre-existing computational design logics. By merging automated interaction logs and participant feedback, we observed:

- **A distinct learning phase:** Participants, regardless of user profile, went through an initial trial-and-error stage aimed at deciphering the logic's parameters and behaviours. This phase entailed a higher iteration rate (6.78 seconds per iteration in the beginning vs 11.28 at the end) and more time spent probing parameter boundaries (300 seconds vs 169).
- **Preference for open-ended interfaces:** Tools that did not overly constrain parameter ranges (e.g., the visual programming and command-line approaches) were generally preferred, especially during early exploratory interactions. In contrast, a more form-based interface that restricted parameter values was perceived as limiting.
- **Transition from discovery to production use:** The visual programming interface encouraged experimentation and learning but sometimes felt less efficient once participants had a clear idea of their design goals. Meanwhile, the command-line approach, initially seen as less user-friendly, was often favoured later for more rapid, targeted inputs.
- **Minimal impact of prior expertise over time:** Although individuals with Grasshopper experience adapted somewhat faster in that

interface, differences in long-term satisfaction and outcomes between them and others were minor by the final exercises. Transparent parameters and intuitive interface design appear to mitigate initial skill gaps.

Limitations of this pilot study

Despite these promising observations, several factors limit the scope and transferability of our findings:

- **Small sample size**: The group of fourteen engineer-architects is not representative of the broader population. A larger and more diverse sample would help validate and deepen these initial observations.
- **Single design task:** We focused on one façade-generation algorithm, which—while illustrative—may not capture the diverse complexities of real-world projects. Extending the scope to more varied tasks could yield a more nuanced understanding of user interaction.
- **Short interaction window:** Participants tested each interface for a limited duration, and self-guided usage outside a controlled lab environment may reveal different patterns of adoption, appropriation, and skill development.
- **Protocol constraints:** This exploratory protocol relied on both automated logs and user surveys, which allowed us to map broad trends but lacks the depth to address some subjective or contextual user experiences. Nevertheless, the automatic logging approach proved particularly effective in revealing behavioural dynamics often missing from purely qualitative feedback.

Implications and next steps

Though preliminary, our results underscore the importance of interface design, parameter transparency, and thoughtful support for non-expert users in adopting computational design methods. This is especially pertinent given that behavioural data demonstrated significant variance overall, at times in contradiction with the qualitative survey. They also suggest that, while prior skill is advantageous, the right interface design can rapidly bring novices up to speed, particularly when it facilitates early understanding of underlying computational logic.

To build on these findings, future research could adopt the following avenues:

- **Longitudinal and In-Depth Studies:** Observing practitioners over extended periods, or incorporating comprehensive user interviews and real-time observational methods, may help capture how attitudes and behaviours evolve from initial discovery to day-to-day professional use.
- **Leveraging Web Analytics Approaches:** Tools like Google Analytics have long tracked user journeys, engagement, and conversion in web development. Adapting similar metrics to computational design interfaces—e.g., measuring which parameters users adjust most, how often they return to specific functionalities, or when they abandon tasks—could provide deeper insights into user behaviour and help inform platform enhancements. This analytics-driven perspective could also enable more sophisticated development strategies, targeting improvements for both new and experienced users.

Overall, this work demonstrates the viability of monitoring and analysing non-expert interactions with CD tools to uncover barriers and opportunities for broader adoption. By viewing these results as a pilot, we can adapt and refine the research protocols, ultimately aiming for more robust conclusions and evidence-based strategies to support computational design's growth in the AECO industry.

Our behavioural analysis based on automated data capture highlights subtle disconnects with the qualitative surveys and highlights distinct learning and usage phases in the use an unfamiliar tool across different interfaces.

REFERENCES

Ataman, C. and Dino, İ. G. (2021) 'Performative design processes in architectural practices in Turkey: architects' perception' *Architectural Engineering and Design Management*, 18(5), pp. 690–704. https://doi.org/10.1080/17452007.2021.1995315

Bernstein, P. (2018) *Architecture | Design | Data: Practice Competency in the Era of Computation*. Berlin, Boston: Birkhäuser. https://doi.org/10.1515/9783035610444

Bernstein, P. (2022) *Machine Learning: Architecture in the Age of Artificial Intelligence*. London: RIBA Publishing. https://doi.org/10.4324/9781003297192

de Boissieu, A. (2020): 'Super-utilisateurs ou super-spécialistes ? Cartographie des catalyseurs de la transformation numérique en agence d'architecture' *Cahiers De La Recherche Architecturale Urbaine Et Paysagère*, 9(10). https://doi.org/10.4000/craup.5551.

de Boissieu, A. (2022). 'Introduction to Computational Design: Subsets, Challenges in Practice and Emerging Roles' in Bolpagni, M., Gavina, R., Ribeiro, D. (eds) *Industry 4.0 for the Built Environment. Structural Integrity*, vol 20. Springer, Cham. https://doi.org/10.1007/978-3-030-82430-3_3

Caetano, I., Santos, L., and Leitão, A. (2020). 'Computational design in architecture: Defining parametric, generative, and algorithmic design' *Frontiers of Architectural Research, 9*(2), pp 287-300. https://doi.org/10.1016/j.foar.2019.12.008

Calixte, X. (2021). 'Les outils dans l'activité collective médiatisée en conception : traçabilité des usages au sein du processus de conception architecturale' (Doctoral thesis, ULiège) Available at: https://orbi.uliege.be/handle/2268/260874

Carpo, M. (2017) *The Second Digital Turn: Design Beyond Intelligence*. Cambridge, MA: The MIT Press. https://10.7551/mitpress/9976.001.0001

Chaillou, S. (2022) *Artificial Intelligence and Architecture: From Research to Practice.* Birkhäuser. ISBN: 9783035624007.

Chen, K., Choo, T-S. and Norford, L. (2019) 'Enabling Algorithm-Assisted Architectural Design Exploration for Computational Design Novices' *Computer-Aided Design and Applications*, 16, pp. 269–288. https://doi.org/10.14733/cadaps.2019.269-288

Davis, D. (2014) 'Quantitatively Analysing Parametric Models' *International Journal of Architectural Computing*, 12(3), pp. 307–319. https://doi.org/10.1260/1478-0771.12.3.307

Denning, P. and Tedre, M. (2019) *Computational Thinking*. Cambridge, MA: The MIT Press. https://doi.org/10.7551/mitpress/11740.001.0001

Deutsch, R. (2019) *Superusers: Design Technology Specialists and the Future of Practice.* Routledge, Taylor & Francis Group. ISBN: 9781351138987.

Ercan, B. and Elias-Ozkan, S.T. (2015) 'Performance-based parametric design explorations: A method for generating appropriate building components' *Design Studies*, 38, pp. 33–53. https://doi.org/10.1016/j.destud.2015.01.001.

Ergün, O., Akin, S., Gursel Dino, I., and Surer, E. (2019). 'Architectural design in virtual reality and Mixed reality Environments: A Comparative analysis' *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 914–915. https://doi.org/10.1109/vr.2019.8798180.

Gardner, N. (2019) 'New Divisions of Digital Labour in Architecture' *Feminist Review*, 123(1), pp. 106–125. https://doi.org/10.1177/0141778919879766

Gardner, N. (2022) 'Digital Transformation and Organizational Learning: Situated Perspectives on Becoming Digital in Architectural Design Practice' *Frontiers in Built Environment*, 8. https://doi.org/10.3389/fbuil.2022.905455.

Hegazy, M., Yasufuku, K. and Abe, H. (2019) '[DC] Immersive Gamified Environments (IGE) as an Approach to Assess Subjective Qualities of Daylighting in Architectural Spaces' in *Proceedings of the 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1371–1372. https://doi.org/0.1109/VR.2019.8798086.

Heumann, A. and Davis, D. (2020) 'Humanizing Architectural Automation: A Case Study in Office Layouts', in Gengnagel, C., Baverel, O., Burry, J., Ramsgaard Thomsen, M. and Weinzierl, S. (eds) *Impact: Design with All Senses*. DMSB 2019. Cham: Springer. https://doi.org/10.1007/978-3-030-29829-6_51

Kwieciński, K. and Słyk, J. (2023) 'Interactive generative system supporting participatory house design' *Automation in Construction*, 145, p. 104665. https://doi.org/10.1016/j.autcon.2022.104665

Leach, N. (2021) *Architecture in the Age of Artificial Intelligence: An Introduction to AI for Architects*. Bloomsbury Publishing. ISBN: 9781350165519.

Meng, Z., Zahedi, A. and Petzold, F. (2020) 'Web-Based Communication Platform for Decision Making in Early Design Phases' in *Proceedings of the International Association for Automation and Robotics in Construction (IAARC)*, Kitakyushu, Japan. https://doi.org/10.22260/ISARC2020/0138

Menges, A, and Ahlquist, S. (2011) 'Computational Design Thinking', in Menges, A. and Ahlquist, S. (eds.) *Computational Design Thinking*. London: John Wiley & Sons. ISBN: 978-0470665701.

Raimbaud, P., Lou, R., Merienne, F., Danglade, F., Figueroa, P., and Hernandez, J. (2019) 'BIM-based Mixed Reality Application for Supervision of Construction' in *Proceedings of the 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1903–1907. https://doi.org/10.1109/VR.2019.8797784.

Silcock, D., Schnabel, M.A., Moleta, T. and Brown, A. (2021) 'Participatory AR – A Parametric Design Instrument' in *Proceedings of CAADRIA 2021*, pp. 295–304. https://doi.org/10.52842/conf.caadria.2021.2.295.

Stals, A. (2019) 'Emerging digital practices in architectural design in small offices - Perceptions and uses of parametric modeling.' (Doctoral Thesis, ULiège)

Beach, T.H., Hippolyte, J.-L., and Rezgui, Y. (2020) 'Towards the adoption of automated regulatory compliance checking in the built environment' *Automation in Construction*, 118, p. 103285. https://doi.org/10.1016/j.autcon.2020.103285

Sun, M., Sun, P., Dong, Y., and Lopez, J. (2021) 'Mass Production – Towards Multidimensional, Real-time Feedback in Early Stages of Urban Design Processes' in *Proceedings of CAADRIA 2021*, pp. 649–658. https://doi.org/10.52842/conf.caadria.2021.2.649.