2026 European Conference on Computing in Construction
Corfu, Greece
July 12-15, 2026

BETWEEN ARCHITECTURAL DESIGNERS AND DEVELOPERS: EMPIRICAL DESIGN RESEARCH FOR COMPUTATIONAL DESIGN SUPPORT

Elien Vissers-Similon[1] and Theodoros Dounas[2]
[1]University of Antwerp, Belgium
[2]Heriot-Watt University, United Kingdom

Abstract

This paper addresses the gap between architectural designers and computational design support developers by proposing a Lakatosian framework grounded in empirical design research. Combining Peirce's reasoning modes with Schön's reflective practitioner, we extend Pauwels et al.'s (2013) framework by revising the hypotheses on design thinking and tool adoption. We derive three guidelines: computational design support as aides of reasoning modes, as artefacts for experimenting, and as collaborative reasoning agents. For each guideline, we formulate a primary empirical research question. We also propose two empirical instruments to help answer those questions.

Introduction

The architectural design process continuously becomes more complex due to an increase of collaborators, stakeholders, regulations, and client demands. This in turn leads to a growth of computational design support tools development. However, the adoption rate of such tools is rather low in architectural design practice. This is especially true for small and medium-sized architectural design firms, who do not tend to set the narrative on emergent technologies but do represent most European firms. (The Architects' Council of Europe, 2024) Surveys show that parametric modelling tools have particularly low adoption rates for small and medium-sized firms (Stals et al., 2018) despite having access to user-friendly tools such as Grasshopper since 2007. (Tedeschi, 2010) Other surveys show that even building information modelling (BIM) software faces low adoption rates in such firms (Burgess et al., 2026) despite BIM's proven value for the AEC sector. The common denominator for these surveys is that they capture the *designers' perspective* on these computational design support tools. The barriers and limitations observed in the surveys unveil that architectural designers and developers are lost in translation. On one hand, designers assume that computational design support restricts their creativity, they expect expensive implementation difficulties and fail to see sufficient added value for their designerly processes. On the other hand, developers approach design processes as homogeneous processes rather than heterogeneous thinking processes and fail to understand layered decision-making processes within a design task.

To bridge this gap, this paper argues for more empirical design research into architectural design processes, to develop computational design support that is compatible with design thinking. By critically extending Pauwels et al.'s (2013) theoretical framework into a Lakatosian research programme, we establish practical guidelines for design support developers and formulate relevant empirical research questions.

Our argument is grounded in an ongoing empirical research programme studying the adoption of diffusion models in small and medium-sized architectural practice (Vissers-Similon et al., 2026) and educational settings (Vissers-Similon and Dounas, 2026A), which provides situated evidence of how reasoning modes, design artefacts and design features interact in real design processes.

Specifically, we address four research questions:

1. Which Lakatosian framework on computational design support can bridge the gap between architectural designers and developers?
2. Which resulting guidelines are relevant for developers?
3. Which resulting empirical design research questions are relevant for future research on computational design support?
4. Which instruments can we use to pursue these research questions?

Traditionally, computational design support is either a means of representation, a tool, or both. We consider computational design support to be a tool for decision-making within design processes.

State-of-the-art review

Design thinking and computational design support

The relationship between design and technology has been theorised through multiple paradigms. Simon's (1969) *Sciences of the Artificial* first established design as a legitimate scientific inquiry. Schön's (1983) *Reflective Practitioner* introduced a paradigm shift by challenging the technical-rational model, arguing that designers engage in 'reflection-in-action': a process fundamentally different from technical knowledge application. This catalysed subsequent research by Dorst (2011) and Cross (2011) establishing "design thinking" as a distinct mode of cognition operating through iterative cycles of problem-framing and solution-finding. Contemporary

understanding of design thinking draws significantly on Peirce's (1931) triadic framework: abduction as generating hypotheses, deduction as deriving consequences, and induction as testing through observation. Dorst (2011) demonstrates how design problems involve 'frame creation': the abductive leap of simultaneously determining problem and solution approach. Kolko (2010) shows designers rely heavily on abduction during design synthesis, indicating that reasoning mode importance shifts during a design process. Garbuio and Lin (2021) further distinguish explanatory from innovative abduction, mapping abductive activities to problem search frames, hypothesis generation, and hypothesis evaluation.

When considering "computational design thinking", Oxman's (2017) *Thinking Difference* establishes parametric thinking as a distinct form enabling fundamentally different modes of design cognition. Terzidis (2006) extends this by distinguishing parametric from algorithmic thinking. These frameworks reveal a consistent pattern: new computational capabilities enable new forms of design thinking, not merely more efficient existing processes.

Alexander et al.'s (1977) *A Pattern Language* focused on "design knowledge" rather than design thinking, proposing codification through patterns. This work followed an unexpected trajectory: initially developed for architecture, pattern languages were adopted by software engineering (Gamma et al., 1994) and are still relevant for software developers today. From software engineering, pattern languages circled back to influence computational architectural design, at first through Stiny and Gips's (1972) shape grammars and their extensions (Stouffs and Krishnamurti, 1996) and through many applications that have followed since. This crosspollination reveals both the portability and context specificity of design knowledge.

We note that, despite the increasing theoretical understanding of design thinking, it is the more tangible and specific design knowledge that generally shapes the development of new computational design support tools.

Empirical evidence from protocol analysis and practice studies

The above claims on design thinking, that reasoning modes shift and tools shape cognition, are not merely theoretical. Goldschmidt's (2014) linkography method demonstrates that design thinking operates through networks of 'design moves' with identifiable critical moments. Her work shows measurable patterns of fore-linking (divergent thinking) and back-linking (convergent thinking) that fluctuate over design sessions, providing quantitative evidence for dynamic reasoning mode dominance. Gero and Kannengiesser's (2014) situated Function- Behaviour-Structure (FBS) framework has generated extensive empirical evidence through protocol studies comparing design cognition across tools and media, showing statistically significant differences when using hand sketching versus parametric modelling. Suwa and Tversky (1997) identified four cognitive levels (physical, perceptual, functional and conceptual) showing expert architects make significantly more functional inferences from sketches than students. Suwa et al. (1998) extended this, identifying distinct design phases with different cognitive profiles. Dorst and Dijkhuis (1995) empirically compared Simonian and Schönian descriptions of the same protocol, finding reflection-in-action better preserves the relationship between process and content. Cramer-Petersen et al. (2019) demonstrate that abductive-deductive patterns dominate early-stage ideation. Critically, recent empirical work by the authors' research programme provides situated evidence from contemporary practice. Vissers-Similon et al. (2026) conducted a year-long living lab with a small Flemish architectural design practice, studying the adoption of diffusion models across complete design processes. The analysis represents design data on three encapsulating levels: *design features* (such as style, texture, geometry, spatial relationships, feeling, lighting, details); *design artefacts* (such as floorplans, sketches, mood boards, 3D models, generated images) as sequences representing design processes; and *design thinking* (specifically Peirce's reasoning modes) representing reasoning between artefacts. The results show that: (a) reasoning mode dominance shifts measurably across design stages, with abductive reasoning dominant during presentation sketch creation and annotation/discussion moments, deductive reasoning during cost estimation and mood board assembly, and inductive reasoning during 3D modelling and floorplan iteration; (b) despite diffusion model adoption, design decisions were predominantly situated in analogue moments, through manual annotation and discussion, and coincide with abductive reasoning; (c) generated artefacts function as effective design artefacts in Schön's sense, 'talking back' to the designer. Short empirical experiments in education (Vissers-Similon and Dounas, 2026A) also corroborate this, showing students make design decisions based on 'moments of interpretation' that are textual, visual or spatial in nature.

Information systems and reasoning mode alignment

Pauwels et al. (2013) provide the key synthesis linking computational design support to design thinking through Peirce's reasoning modes. Their framework connects cognitive reasoning modes to practical requirements of digital tools, arguing effective support must align with natural reasoning patterns. They propose three application development approaches: computational design support as surrogates for reasoning modes, as tools for experimenting, and as autonomous reasoning agents. They then argue that traditional information systems typically target only one reasoning mode at a time, which may explain limited adoption.

Research gaps

The state-of-the-art review establishes crucial principles but also reveals critical gaps. First, computational design support is most often based on design *knowledge* rather than design *thinking*. Second, design thinking operates through reasoning modes that shift in dominance, as demonstrated by empirical evidence from both protocol

and practice studies. Third, Pauwels et al.'s (2013) framework treats the reasoning cycle as static rather than dynamic and does not accommodate design artefacts as catalysts for reflection-in-action. Fourth, no formalised representation maps Peirce's reasoning modes onto concrete design activities in a way that is both empirically grounded and machine-readable. These gaps motivate a revised Lakatosian framework and accompanying empirical instruments.

Methodology

This paper employs a dual methodological approach. Its primary method is *critical conceptual analysis*: we systematically extend an existing theoretical framework through gap identification and hypothesis revision, following Lakatos's (1970) methodology of scientific research programmes. Its secondary method draws on *design science research* (Hevner et al., 2004): we treat the revised framework and its accompanying empirical instruments as design artefacts, subject to evaluation against utility criteria. The complete methodological approach is visualised in Figure 1.

Selection and analysis of base framework

We select Pauwels et al.'s (2013) framework because it explicitly links computational design support to *design thinking* through Peirce's reasoning modes, rather than to design knowledge. Alternatives were considered: Hevner et al.'s (2004) design science framework focuses on information system artefacts rather than design cognition; Oxman's (2017) parametric thinking framework addresses a specific tool typology; Gero and Kannengiesser's (2014) Function- Behaviour-Structure ontology provides an empirical coding scheme but not a normative framework for tool development.

Gap identification

The base framework rests on two major hypotheses, regarding design thinking and tool adoption. We interrogate each hypothesis against two evidence sources: (a) established protocol analysis studies showing reasoning mode shifts (Goldschmidt, 2014; Suwa et al., 1998; Cramer-Petersen et al., 2019) and (b) situated empirical evidence from the living lab study (Vissers-Similon et al., 2026) mapping reasoning mode transitions across complete design processes in a real architectural practice. A hypothesis is revised when it cannot accommodate documented variability (criterion of flexibility) or when it treats support as replacement rather than augmentation (criterion of compatibility with reflective practice).

A Lakatosian framework for design thinking support

Lakatos (1970) introduced a structure for scientific research programmes distinguishing between a 'hard core' of fundamental commitments and a 'protective belt' of auxiliary hypotheses subject to modification. Applied to our case, this suggests computational design support should be developed as design research programmes: maintaining core commitments while allowing implementations to evolve based on empirical evidence. To accommodate empirical modification of the auxiliary hypotheses in the protective belt, our Lakatosian framework includes practical guidelines for developers of computational design support, and theoretical research questions to facilitate empirical design research.

Design science component: from framework to empirical instrument

Following Hevner et al.'s (2004) design science framework, which itself adopts Lakatos's structure, we propose an empirical instrument as a design artefact: a structured protocol-analysis coding instrument mapping Peirce's reasoning modes onto observable design activities. This instrument operationalises the revised framework and can be evaluated through the empirical studies we propose. Its design draws on Gero and Kannengiesser's (2014) Function- Behaviour-Structure coding scheme, Goldschmidt's (2014) linkography, and Vila-Henninger et al.'s (2022) abductive coding methodology, calibrated against Vissers-Similon et al.'s (2026) three encapsulated levels: design features, design artefacts and design thinking. The empirical instrument provides a formal structure into which findings can be encoded, closing the loop between the empirical observation of design thinking and machine-readable applications of design knowledge in computational design support.

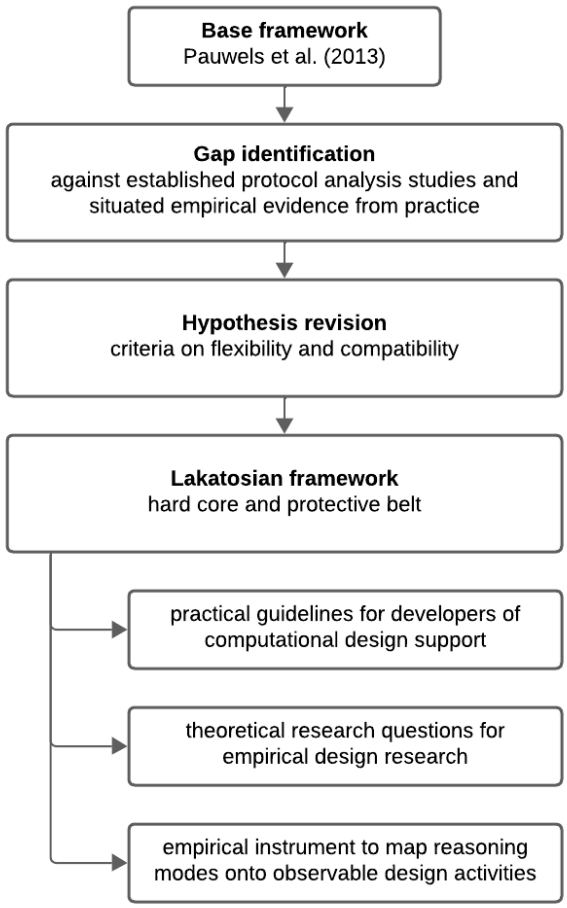

Figure 1: Methodological flowchart

Revised hypothesis on design thinking

The original hypothesis from the base framework states: "Creative designing rests on a cyclic combination of abductive, deductive and inductive reasoning processes." (Pauwels et al., 2013) We support Peirce's reasoning as foundation, because reasoning and decision-making is at the core of designing. However, the original hypothesis leaves no flexibility for varying importance of reasoning modes along a design process.

Protocol analysis and empirical evidence support this: Goldschmidt's (2014) linkography shows fore-linking and back-linking patterns shift measurably. Cramer-Petersen et al. (2019) demonstrate abductive-deductive patterns dominate early ideation. Vissers-Similon et al.'s (2026) living lab provides direct situated evidence: abductive reasoning dominates during presentation sketch creation and manual annotation of generated images; deductive reasoning during cost estimation and mood board assembly; inductive reasoning during 3D modelling and floorplan iteration. Design decisions, the moments that actually advance the project, coincide predominantly with abductive reasoning, conducted through manual annotation and collegial discussion, even when diffusion models are actively adopted.

The living lab thus confirms Schön's (1983) reflecting-in-action with contemporary evidence: generated artefacts from diffusion models succeed 'just as much as traditional artefacts' at 'talking back' to the designer.

We rephrase reasoning modes within the context of design thinking: abduction yields a plausible conclusion based on established design artefacts and background knowledge; deduction deducts a logical conclusion from established artefacts; induction infers a general theory from observations of established artefacts. This leads to an amended hypothesis: "Design thinking rests on a combination of abductive, deductive and inductive reasoning modes. At certain moments during a design process, design thinking adapts so that some reasoning modes are more dominant than others."

Revised hypothesis on tool adoption

The original hypothesis from the base framework states: "Traditional information systems typically target only one of these reasoning processes at a time, which might explain their limited applicability and usefulness." (Pauwels et al., 2013) We question whether targeting a single mode is inherently problematic. What truly matters is understanding which modes are dominant at which stages. The living lab illustrates this: diffusion models function effectively when targeting the right moment, but their outputs require human abductive interpretation to yield design decisions.

The amended hypothesis is thus: "Design support systems typically target only one of the reasoning modes at a time, without understanding which reasoning mode is dominant during the specific design task at hand, which might explain their limited applicability and usefulness."

Guidelines for computational design support

Based on the revised hypotheses, we propose three non-exclusive practical guidelines for developers of computational design support.

Guideline 1: Aides of reasoning modes

Developers should focus on tools that aide reasoning, targeting the right mode for the task at hand. Vissers-Similon et al.'s (2026) living lab shows that during sketch design, when abductive reasoning dominates, designers create hand drawn presentation sketches over printed 3D model views. A mode-aware tool could detect this abductive phase and offer precedent images as prompts, rather than imposing deductive constraint-checking. However, developers should be cautious of the epistemic mismatch in how humans and LLMs produce judgement. (Quattrociocchi et al., 2025)

Guideline 2: Artefacts for experimenting

Developers should focus on tools providing experimental artefacts. The critical question is whether tool outputs function as design artefacts in Schön's (1983) sense: objects provoking reflection-in-action. The living lab provides direct evidence: generated images from diffusion models functioned as effective Schönian artefacts when designers manually annotated them to extract design features (style, colour scheme, materials) and discussed annotations with colleagues. The artefact quality that mattered was *interpretive affordance* - the capacity to provoke functional inference, which Suwa and Tversky (1997) identified as the hallmark of effective design artefacts.

Guideline 3: Collaborative reasoning agents

Developers should focus on collaborative rather than autonomous reasoning agents. Vissers-Similon et al.'s (2026) living lab highlights the necessary collaboration between human designers and diffusion models to make effective design decisions: diffusion models did not generate the project's final façade design, but the human designers did considerably speed up their decision-making process in collaboration with diffusion models.

Empirical research questions

The framework's flexibility depends on empirical evidence to inform the hypotheses. For each guideline, we define one primary research question, which empirical instruments help to answer – and in turn, those new empirical findings might adjust the protective belt's hypotheses. The following empirical design research questions align with the established guidelines:

1. **RQ-E1:** Which reasoning mode is dominant at which moment in the design process?
2. **RQ-E2:** Which types of artefacts facilitate design thinking at which moment in the design process?
3. **RQ-E3:** Which type of interaction facilitates design thinking at which moment in the design process?

Empirical instruments

We propose empirical instruments to help answer the empirical research questions above.

First, a structured coding instrument operating at three levels, which links findings from established protocol-analysis instruments to findings from architectural practice, at each of those three levels. At the micro level, Gero and Kannengiesser's (2014) Function- Behaviour-Structure coding scheme provides the theoretical, ontological base. To map these acts of design, we use "design features". At the meso level, we use abductive coding (Vila-Henninger et al., 2022), to map the "design artefacts" onto their design processes. At the macro level, we relate to Goldschmidt's (2014) linkography method to document how designers think, to map "design thinking". A complete methodology of mapping design features, artefacts and thinking on these three levels is described in Vissers-Similon et al.'s (2026) *Mapping Design Features, Artefacts, and Thinking: Developing a Method for Assessing the Impact of Diffusion Models' Implementation in Interior Design*. By mapping reasoning modes onto observable design activities, we create an empirically grounded instrument for protocol-analysis coding.

Second, for a machine-readable representation, which addresses the fourth and final gap from the "research gaps" section in this paper, we propose an ontology that can be navigated by collaborative reasoning agents. This ontology uses design features, artefacts and thinking as building blocks. The theoretical foundation of such an ontology is described in Vissers-Similon and Dounas' (2026B) *OntoCAAD: an empirically grounded Ontology for Collaborative Agentic Architectural Design*. The OntoCAAD ontology provides the conceptual infrastructure which collaborative reasoning agents could navigate: it formalises design knowledge into classes (Feature, Artefact, Actor, DesignDecision) and formalises design thinking through class interactions (reframe, reason, make, implement) that align with the epistemic pipelines of both human and machine reasoning. (Quattrociocchi et al., 2025) A collaborative reasoning agent built on this ontology could reframe a design problem by identifying relevant features, generate artefacts through artificial intelligence models, and support the designer's reasoning while leaving the actual design decision with the human. The ontology's Actor class distinguishes Person and Agent subclasses, allowing for a precisely calibrated distribution of agency and authorship.

Results: a Lakatosian framework for computational design support

The resulting framework is visualised in Figure 2.

Discussion: empirical research agenda

Regarding RQ-E1, Vissers-Similon et al.'s (2026) living lab provides preliminary answers based on one case study. The empirical mapping instrument can now be applied across additional design processes to test generalisability.

Regarding RQ-E2, Vissers-Similon et al.'s (2026) living lab provides preliminary answers on generated artefacts by diffusion models, which facilitate design thinking through annotation and discussion. OntoCAAD's Artefact class provides the structured vocabulary for further large-scale comparative analysis.

Regarding RQ-E3, Vissers-Similon and Dounas' (2026A) educational experiments show decisions occur at 'moments of interpretation', and OntoCAAD's interact() function and AgentRole enumeration provide categories for evaluating interaction modalities in further large-scale comparative analysis.

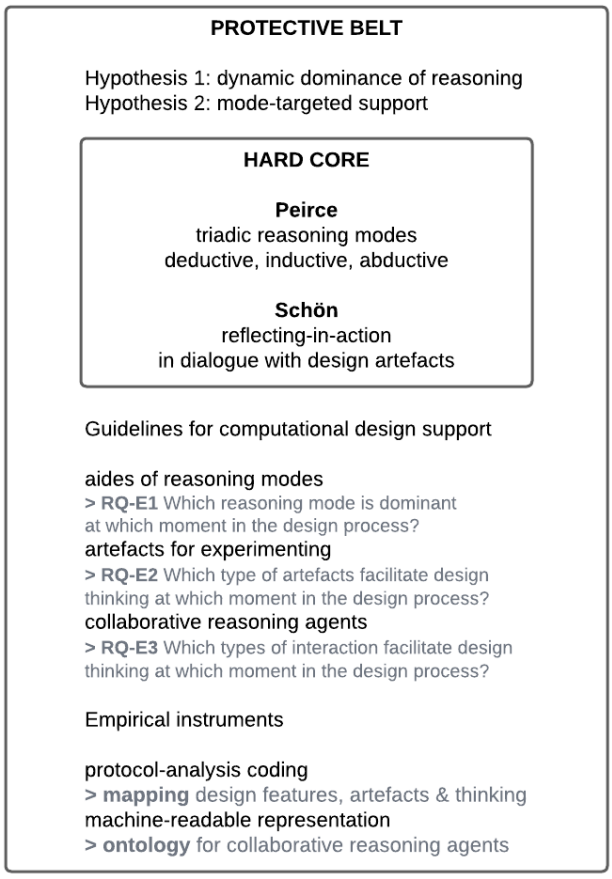

Figure 2: Lakatosian framework for computational design support

Validation roadmap

To provide substantiated answers to the empirical design research questions, and thus to solidify the Lakatosian framework for computational design support, we develop a research programme with the following roadmap.

Phase 1 (completed): establishing the framework. The educational experiments (Vissers-Similon and Dounas, 2026A) and year-long living lab (Vissers-Similon et al., 2026) provided foundational data to establish the Lakatosian framework.

Phase 2 (ongoing): initial calibration of the framework. The empirical mapping instrument as described in Vissers-Similon et al. (2026) will be applied to empirical observations of four additional living lab studies, with small and medium sized architectural firms. The empirical machine-readable instrument, OntoCAAD – as described

in Vissers-Similon and Dounas (2026B), will be built and tested to (i) finetune the ontology and (ii) define potential collaborative reasoning agents.

Phase 3: carrying out the research programme. The OntoCAAD ontology will be integrated in a digital platform, alongside prototypes of useful collaborative reasoning agents. This empirical infrastructure will then be used for testing design sessions across multiple small and medium sized architecture firms to answer RQ-E1, RQ-E2 and RQ-E3 for each collaborative reasoning agent.

Conclusions and contribution

This paper advocates for more empirical architectural design research to bridge the gap between designers and developers of computational design support. We propose a Lakatosian framework based on Peirce's triadic reasoning modes and Schön's reflection-in-action, to substantiate two hypotheses: on dynamic dominance of reasoning modes, and on mode-targeted support.

The framework produces three nonexclusive guidelines for computational design support and two empirical instruments to help achieve these guidelines. The guidelines are computational design support as aides of reasoning modes, artefacts for experimenting and collaborative reasoning agents. The empirical instruments are protocol-analysis coding through mapping, and machine-readable representation through an ontology.

The key contribution lies in recognising that the development of computational design support constitutes applied design research. Each tool instantiates hypotheses about design thinking and creates opportunities for empirical validation, with actual adoption in practice as the end goal. The framework's guidelines provide three primary empirical research questions, which can be answered with the help of two empirical instruments. First, the protocol-analysis coding instrument allows us to map reasoning modes onto observable design activities. Second, the machine-readable representation by the OntoCAAD ontology translates validated findings into structured data, bridging design cognition research and computational agent development.

Limitations

This paper is a conceptual extension grounded in, but not constituting, new empirical data collection. The protocol-analysis instrument is established but not yet calibrated. The ontology instrumented is described but not yet built or calibrated. The empirical findings which serve as the foundation for this research programme are all collected in the Flemish architectural design context; thus, generalisability remains to be demonstrated. The authors bring combined experience in architectural practice, computational design research and empirical design cognition studies, which informed gap identification but may introduce biases toward particular tool typologies.

Acknowledgments

The research on this Lakatosian framework, as well as the completed and ongoing phases of the validation roadmap, was done primarily by Elien Vissers-Similon, under the supervision of Theodoros Dounas. In this paper, Dounas contributed to the state-of-the-art review, research gaps, methodology, discussion and conclusions sections.

References

Alexander, C., Ishikawa, S. & Silverstein, M. 1977. A Pattern Language, New York, Oxford University Press.

Burgess, M., Wilson, C. & Fan, Y. V. 2026. Adoption drivers and barriers of BIM in Europe. Energy and Buildings, 116953.

Cramer-Petersen, C. L., Christensen, B. T. & Ahmed-Kristensen, S. 2019. Empirically analysing design reasoning patterns. Design Studies, 60, 39–70.

Cross, N. 2011. Design Thinking, Understanding how designers think and work, Berg Publishers

Dorst, K. 2011. The core of 'design thinking' and its application. Design Studies, 32, 521–532.

Dorst, K. & Dijkhuis, J. 1995. Comparing paradigms for describing design activity. Design Studies, 16, 261–274.

Gamma, E. et al. 1994. Design Patterns, Elements of Reusable Object-Oriented Software, Pearson Publishers.

Garbuio, M. & Lin, N. 2021. Abductive reasoning in problem finding. Academy of Management Proceedings.

Gero, J. S. & Kannengiesser, U. 2014. The FBS Ontology of Design. In: An Anthology of Theories and Models of Design, Springer, 263–283.

Goldschmidt, G. 2014. Linkography: Unfolding the Design Process, MIT Press.

Hevner, A. R. et al. 2004. Design science in IS research. MIS Quarterly, 28, 75–105.

Kolko, J. 2010. Abductive thinking and sensemaking. Design Issues, 26, 15–28.

Lakatos, I. 1970. Falsification and the Methodology of Scientific Research Programmes. In: Criticism and the Growth of Knowledge, Cambridge University Press.

Lawson, B. 2005. Oracles, draughtsmen, and agents. Automation in Construction, 14, 383–391.

Oxman, R. 2017. Thinking difference, theories and models of parametric design thinking. Design Studies, 52, 4–39.

Pauwels, P., De Meyer, R. & Van Campenhout, J. 2013. Design thinking support: IS versus reasoning. Design Issues, 29, 42–59.

Peirce, C. S. 1931. Collected Papers, Harvard University Press.

Quattrociocchi, W. et al. 2025. Epistemological fault lines between human and AI. arXiv:2512.19466.

Schön, D. A. 1983. The Reflective Practitioner, London.

Simon, H. A. 1969. The Sciences of the Artificial, MIT Press.

Stals, A., JANCART, S. & ELSEN, C. 2018. Influence of parametric tools on complexity in everyday work of SMEs. Archnet-IJAR, 12, 206–227.

Stiny, G. & GIPS, J. 1972. Shape grammars. IFIP Congress, Ljubljana, 1460–1465.

Stouffs, R. & Krishnamurti, R. 1996. On a query language for weighted geometries, Third Canadian Conference on Computing in Civil and Building Engineering (eds. O. Moselhi, C. Bedard, S. Alkass), Canadian Society for Civil Engineering, Montreal, 783- 793.

Suwa, M. & Tversky, B. 1997. What do architects and students perceive in their design sketches? Design Studies, 18, 385–403.

Suwa, M., Purcell, T. & Gero, J. S. 1998. Macroscopic analysis of design processes. Design Studies, 19, 455–483.

Tedeschi, A. 2010. Interview with David Rutten. MixExperience.

Terzidis, K. 2006. Algorithmic Architecture, London.

The Architects' Council of Europe 2024. The Architectural Profession in Europe 2024.

Vila-Henninger, L. A. et al. 2022. Abductive coding. Sociological Methods & Research, 51, 968–1005.

Vissers-Similon, E., De Walsche, J. & Dounas, T. 2026. Mapping Design Features, Artefacts, and Thinking: Developing a Method for Assessing the Impact of Diffusion Models' Implementation in Interior Design. Journal of Interior Design.

Vissers-Similon, E. & Dounas, T. 2026a. The Human Designer in Times of Artificial Intelligence: Diffusion-Driven Architectural Design Explorations. In: Agkathidis, A., Hudert, M. & Medel-Vera, C. (eds.) Architecture in the AI Era for Research, Practice, and Pedagogy. Singapore: Springer Nature Singapore.

Vissers-Similon, E. & Dounas, T. 2026b. OntoCAAD: an empirically grounded Ontology for Collaborative Agentic Architectural Design. In: Proceedings of the 44th Conference on Education and Research in Computer Aided Architectural Design in Europe (eCAADe). 7-11 September 2026. Lübeck.