Introduction to Computational Design: Subsets, Challenges in Practice and Emerging Roles

Aurelie de Boissieu

Abstract Computation-based approaches have been flourishing in the construction industry for the past years. From experimental practices to mainstream production, the usage of digital tools tends to be diverse and versatile. This is especially true for computational design (CD) which encompasses multiple practices, transforming the future of the industry and its stakeholders. Through the ever-increasing speed and capacity of computers, computation enables dealing with geometries and tasks which were traditionally either too time consuming or too challenging to be accomplished by human alone. However, CD is not just automating existing traditional processes or tedious tasks; it is about shifting the way we think and design. To better understand how to unlock the opportunities of CD, this chapter discusses the following: 1—the main subsets of CD, called parametric, generative and algorithmic design; 2—presents CD's different toolsets and their evolutions and finally 3—interrogates how CD is integrated in practice, with its emerging roles and skillsets.

Keywords Computational design · Parametric design · Generative design · Algorithmic design · Data driven · Optimization · Interoperability · Superuser · Platformization

1 Introduction

Computation-based approaches have been flourishing in the construction industry for the past years. From experimental practices to mainstream production, the usage of digital tools tends to be diverse and versatile. This is especially true for computational design (CD) which encompasses multiple practices transforming the future of the AECO[1] industry and its stakeholders.

[1] Architecture Engineering Construction and Operation.

A. de Boissieu (✉)
Liège University, allée de la Découverte 9, 4000 Liège, Belgium
e-mail: Aurelie.deboissieu@uliege.be

M. Bolpagni et al. (eds.), *Industry 4.0 for the Built Environment*,
Structural Integrity 20, https://doi.org/10.1007/978-3-030-82430-3_3

Research and experimentation in computation for architectural and urban design emerged early on in the 1960s, with works such as those of Yvan Sutherland, Yona Freidman or Nicholas Negroponte [1–3]. But digital tools dedicated to the construction industry only made their way in the everyday life of the office later on, especially with the democratization of CAD[2] in the late 80s. It was then followed by many other technologies, such as 3D modeling, rendering, document management systems or digital fabrication. Today, technical innovations continue at high speed.

1.1 Computer-Aided, Automated, Digitalized or Computational

CD practices may be diverse, but they share a common foundation: They rely on the power of computation as well as on computational thinking. Through the ever-increasing speed and capacity of computers, computation enables dealing with geometries and tasks which were traditionally either too time consuming or too challenging to be accomplished sustainably by human alone [4]. However, CD is not just automating existing traditional processes or tedious tasks; it is about shifting the way we think and design [2, 5, 6].

In the 60s, Negroponte highlighted the distinction between "computer aided," which is about traditional processes supported by computation tools, from "computerized" or "digitalized," which is about new processes enabled by computers [3]. Terzidis goes one step further with the term "computational," making the distinction between the uses of the technique (the computers, as in computer-aided or digitalized) and the uses of the thinking paradigm (the computation, as in computational design) [2, 7, 8] (Fig. 1).

This distinction emphasizes that employing computational design requires first and foremost a change in mind-set, rather than a simple application of digital tools to an existing design or process. In other words, "*Algorithms are used, not to enhance architectural designs, but rather to conceive them*" [7].

1.2 Computational Thinking

At the foundation of CD is the power of computation, which relies on algorithms. To put it simply, an algorithm is no more than a sequence of instructions[3] [1, 2, 6].

[2] CAD: Computer-aided design systems such as AutoCAD (Autodesk) or Microstation (Bentley Systems).

[3] To be more specific, Menges and Alquist define an algorithm as "*a finite sequence of explicit, elementary instructions described in an exact, complete yet general manner*" [2, p. 13].

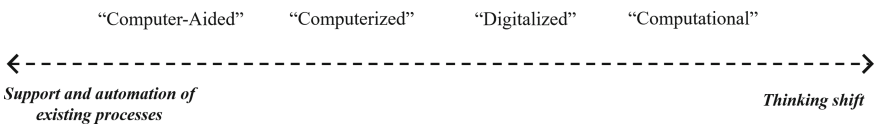


Fig. 1 From computer-aided to computational

For example, a cooking recipe can be considered as an algorithm: "take the ingredients A and B, mix them following rule C and then split the result according to rule D" is a sequence of well-ordered instructions. From this simplistic definition, further subtlety and complexity can be explored. In the context of CD practices, algorithmic technologies and thinking are used to define and manipulate both the flow of data and complex geometries of projects. Designers focus on defining a series of instructions, rules and relationships that gives them better control of the design, enabling the analysis, modification or generation of a wide range of data and geometries. These significantly powerful capabilities open up new fields of possibilities for the AECO industry and its stakeholders [6, 9–11].

To better understand how to unlock the opportunities of CD, this chapter discusses the main subsets of CD, called parametric, generative and algorithmic design, presents CD's different toolsets and their evolutions and finally interrogates how CD is integrated in practice, with its emerging roles and skillsets.

This chapter aims to give a beginner in the field some pointers on the fundamentals of CD practices. Chapter "Advanced Applications in Computational Design" gives further contemporary case studies of these practices.

2 Defining Computational Design Subsets: Parametric, Generative or Algorithmic

Menges and Ahlquist stated that the "*understanding of how systems operate is fundamental to computational design,*" allowing for a very open definition of CD. To enable a more specific characterization, three key subsets of CD can be identified: parametric, generative and algorithmic. These subsets are useful to contextualize the different CD capabilities, although their definitions are sometimes controversial in both industry and academic literature [12].

2.1 Parametric Design

There is no clear consensus among specialists on a definition of parametric design per se [12–17], but recurring characteristics can be identified.

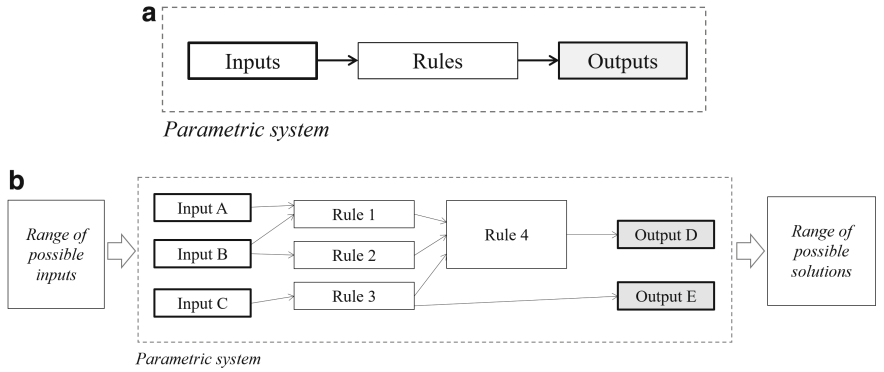


Fig. 2 **a** Inputs, rules and outputs in a parametric system: main dependencies **b** Inputs, rules and outputs in a parametric system: complex systems through simple rules and parameters combinations

Parametric design relies on the design and modeling of parametric systems. A parametric system is an association of clear rules and constraints in a specific order. Inputs can be fed into the system, and rules and constraints are then solved by propagating these inputs through every rule that depends on them, allowing the production of consistent outputs (see Fig. 2a).

The range of possible inputs (the parameters), the rules (which can be mathematic, geometric, etc.) and the relationships between them constitute the parametric system (Fig. 2b). For the same parametric system, different inputs will generate different outputs. The relationships between outputs and rules are explicitly defined by the users, prior to execution [15]. These relationships create constraints and dependencies between the different elements of the system. The propagations of data between dependencies in parametric systems are always unidirectional, so there are no loops possible in a parametric model, also referred as acyclic. Despite this limitation, complex systems can be defined through simple rules and parameters, associated with clear relationships (Fig. 2b).

In parametric design, the designer works on two levels: the definition of the parametric system and the exploration of meaningful results produced by the system, called instances [14]. These two levels interlace, and the designer goes through multiple iterations and explorations between system and instances (Fig. 3). Creativity and design are needed at both these levels.

London's Waterloo Station designed by Grimshaw in the early 90s is an interesting legacy example of parametric design. The highly constrained site and program led to a need for major variability in the width of the station. Indeed, the span of the roof continuously changes from one end of the station to the other, getting wider toward the entrance, allowing more space for train platforms (Fig. 4a). In this changing geometry, the logic of the structural truss remains consistent. This was solved using a parametric design of the truss system (Fig. 4b).

The truss geometry is expressed in terms of geometrical relationships and constraints, forming two asymmetrical arcs (Fig. 4b). With the site boundary lines

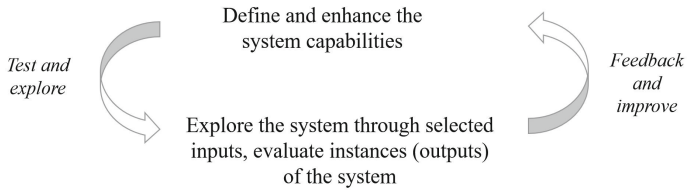


Fig. 3 From designing a parametric system to exploring instances and back

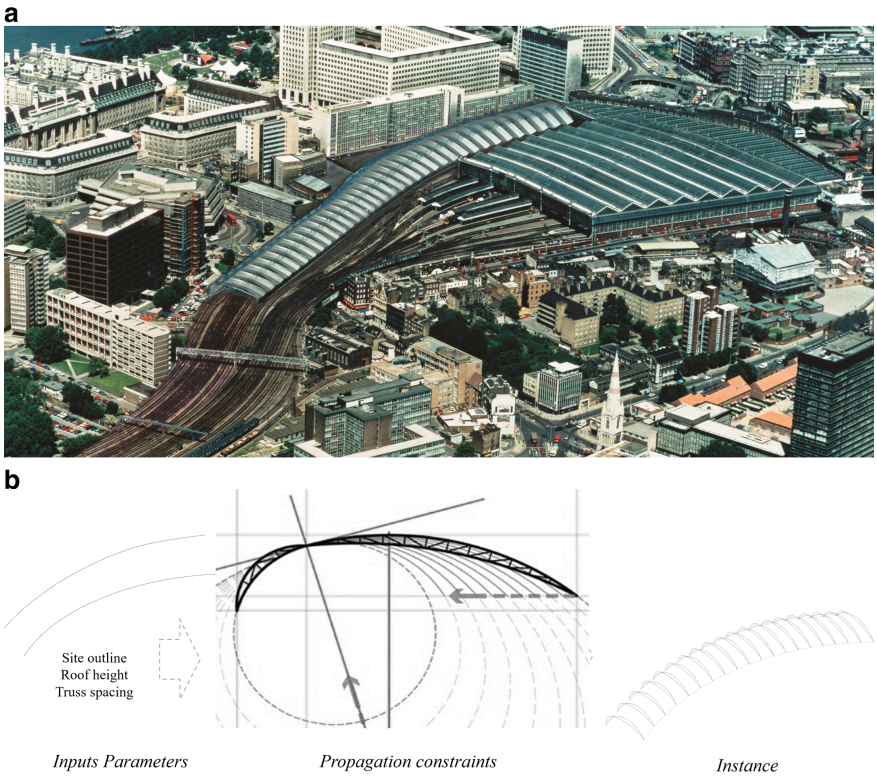


Fig. 4 **a** Waterloo Station: external view of a very dense site in central London, the roof span gets wider toward the station pedestrian entrance (*Credits* Grimshaw) **b** Waterloo Station's truss geometry (Credits for the geometry diagram: Shane Burger)

as input, the parametric design enables a fast and easy exploration and definition of the roof and its trusses (Fig. 5a) [18, 19]. It also allows the control of the geometry and its explorations, enabling a rationalized and ruled fabrication of the trusses [20] (see Fig. 5b).

We can observe here some of the main characteristics of parametric design as discussed in the literature: The design relies on the definition of geometric

relationships between specific inputs, allowing for the exploration of a wide range of options while maintaining relevance for the project. This shows the shift from designing a specific geometry to designing the system which controls the geometry, as well as the construction workflow. Designing fabrication and construction relevant workflows is a recurring theme in CD practices in general, not just in parametric design. Some further pointers on fabrication, materiality and workflows can be found in the works of Menges, Garber, Peters and De Kestellier among others [21–25].

Because parametric system's constraints dependencies are unidirectional, they are relatively easy to apprehend and anticipate. It makes parametric design an easier CD practice to access for beginners, but also a more constraint one because not all design intents can be approached through acyclic algorithms. Generative design however presents more flexibility in its approach but also more complexity.

2.2 Generative Design

As in parametric design, generative design is a rule-based approach capable of generating a wide range of outputs. But the algorithmic approaches used in generative design can be more diverse than in parametric design, which is constrained to be acyclical. Generative design allows designers to define problem spaces through sets of well-ordered rules and instructions, and ranges of possible inputs clearly defined in domains. From these definitions, generative systems will create ranges of outputs, also called the solution space.

The solution space (the range of outputs) is ranked according to a specific set of targeted criteria, also known as objectives. The solution space can then be explored or searched,[4] in order to optimize these selected objectives [12, 26, 27] (Fig. 6). These optional optimizations can balance multiple objectives and enabling the systems to support the designer in better understanding the effects and intricacies of competing goals of the generative process [28–31].

In the early 2000s, EZCT developed a chair[5] based on generative design strategies [32]. Based on a matrix of voxels with the overall dimensions of a chair (the "definition domain" in Fig. 7b), the chair material is being added or removed at each iteration of the algorithm (Figs. 6 and 7b). The performance of the chairs produced is being evaluated at each iteration (Fig. 7c) with regard to structural stability in a multiple load strategy (Fig. 7a), as well as with regard to their resulting weight and volume (Fig. 7c). The balance between these two conflicting objectives (a light chair versus a strong chair) is driven by the algorithm. The chairs with the

[4] About the difference between explore or search: "*Search is a process for locating values of variables in a defined state space while exploration is a process for producing state spaces*" [27].

[5] EZCT Architecture and Design Research (with Hatem Hamda and Marc Schoenauer) Studies on Optimization: Computational chair design using genetic algorithms, 2004, Chair Model "T1-M" after 860 generations (86,000 structural evaluations).

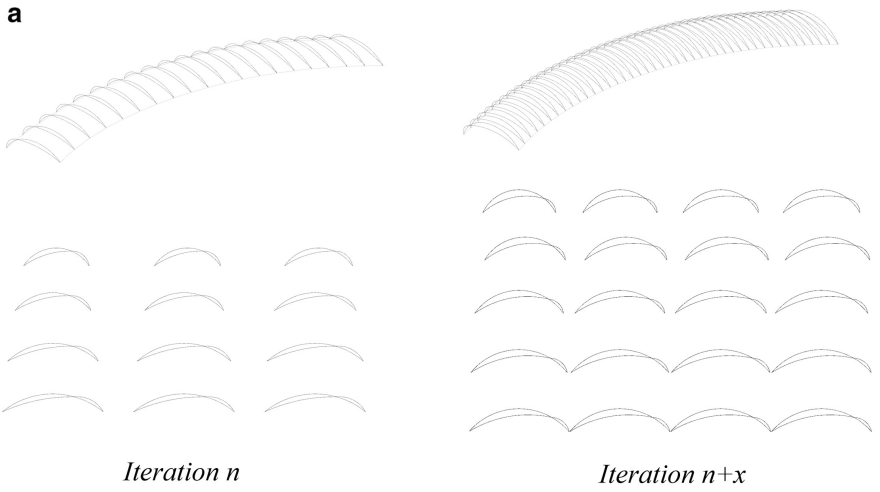


Fig. 5 **a** Example of possible iterations of the parametric system **b** Waterloo Station, internal view (*Credits* Grimshaw)

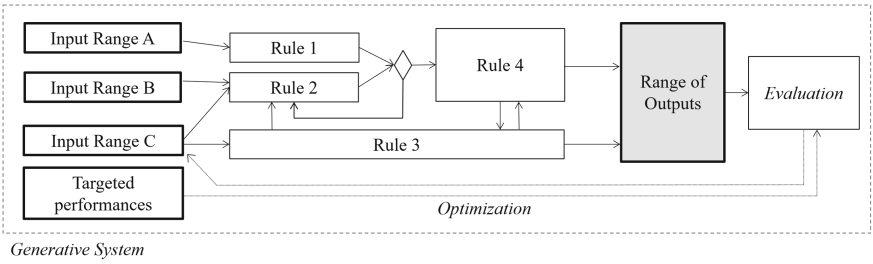


Fig. 6 Generative system principles with embedded optimization process

best performance are selected at each iteration and input back in the system to improve the next iteration, through what is called a genetic algorithm [32]. These multiple feedbacks loops allow an optimization of the overall chair generation. A wide range of chairs is being generated and optimized from one iteration to another, constituting the solution space of the design (Fig. 7b). From this solution space, instances can be selected according to the desired performances, to be then fabricated (Fig. 7d).

This EZCT project leverages generative design through a genetic algorithm, but multiple generative design approaches exist. We cannot name them all, but some useful references can be found through the following terms: shape grammars, cellular automata, L-systems or swarm systems [12].

While the example shown here is a relatively small-scale project, generative design is also widely used for architectural and urban design. Platforms such as Testfit.io, SpaceMakerAI or ArchiStar are relying on generative design to provide automated architectural layout options as well as feasibility studies at urban scale [33–35]. These platforms are discussed further in Sect. 3.2.

Furthermore, Anderson demonstrates how generative design can provide efficient desk layouts for office spaces [27]. In particular, it shows how it can successfully address specific design requirements (such as required adjacencies, clearances requirements for desks and doors, maximum and minimum density) as well as the irregularities of existing spaces (irregularly shaped offices, difficult edge conditions, obstacles in the space). When analyzing the performance of the algorithm, the authors observe that the layouts generated achieve a 97% match rate with the performance of what architects previously designed for existing offices of the same company [27].

In the end, generative design can be considered as more autonomous than parametric design, allowing complex explorations where multiple objectives and requirements are difficult to predict for humans without computation support.

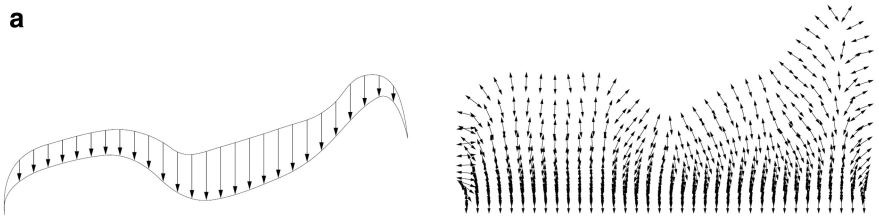
2.3 Algorithmic Design

The algorithmic design approach is widely quoted in the CD literature, although its scope often overlaps with parametric design and generative design.

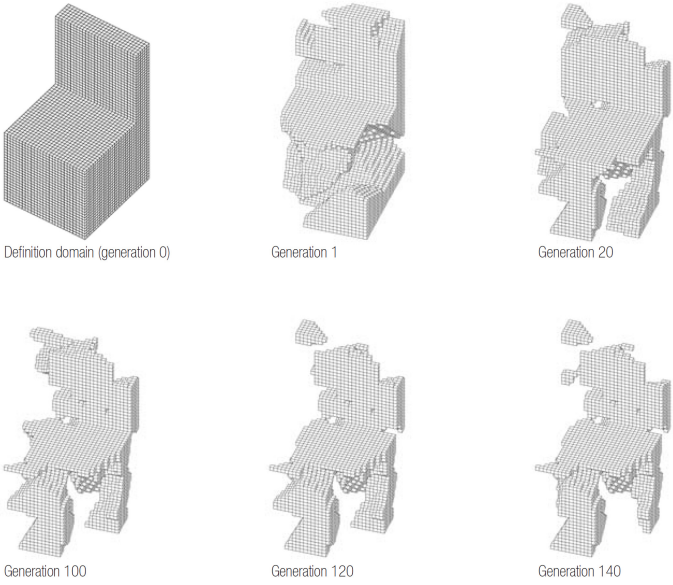
Algorithmic design approaches seem to especially leverage computational methods and thinking, beyond the use of computer as formal tools. Terzidis highlights for example how algorithmic design allows the designer to "*go beyond the common and predictable*" [7, 36]. In the end, algorithmic design allows more freedom than parametric design or generative design in the balance between intention and emergence [2, 36].

To some extent, the computational chair from EZCT (Fig. 7) can be considered a relevant algorithmic design case as well as generative design one. Indeed, its design strategy is widely based on emergence more than on intention, leading to some unexpected instances of the chair (Fig. 7b). Meanwhile, the work from Anderson

a



b



c

CHAIR	VOXELS	DIMENSIONS (PxLxH)	VOLUME	WEIGHT 1 *	WEIGHT 2 *	STABILITY
Test1-860	4714	52 cm x 54 cm x 100 cm	0.0377 m3	18.85 kg	22.62 kg	FINE
Test2-700	4420	54 cm x 50 cm x 96 cm	0.0353 m3	17.65 kg	21.18 kg	FINE
Test3-840	4789	52 cm x 40 cm x 100 cm	0.0383 m3	19.15 kg	22.98 kg	LOW
Test4-640	4456	60 cm x 46 cm x 98 cm	0.0356 m3	17.8 kg	21.36 kg	FINE
Test5-620	2953	54 cm x 44 cm x 94 cm	0.0236 m3	11.8 kg	14.16 kg	LOW
Test6-680	4412	50 cm x 60 cm x 96 cm	0.0331 m3	16.55 kg	19.86 kg	FINE
Test7-680	4158	56 cm x 54 cm x 92 cm	0.0332 m3	16.6 kg	19.92 kg	LOW
Test8-700	2885	60 cm x 46 cm x 92 cm	0.0230 m3	11.5 kg	13.8 kg	LOW
Test9-720	2334	50 cm x 36 cm x 92 cm	0.0186 m3	9.3 kg	11.16 kg	LOW
Test10-760	3920	50 cm x 44 cm x 92 cm	0.0186 m3	15.65 kg	18.78 kg	LOW

Fig. 7 **a** EZCT preliminary structural studies for a structural chair, a multiple loads strategy (*Credits* Philippe Morel) **b** EZCT computational chair, extracts from the solution space geometries (*Credits* Philippe Morel) **c** EZCT computational chair, extracts from the solution space evaluations (*Credits* Philippe Morel) **d** Chair model "T1-M 860" (fabricated instance) (*Credits* Philippe Morel)

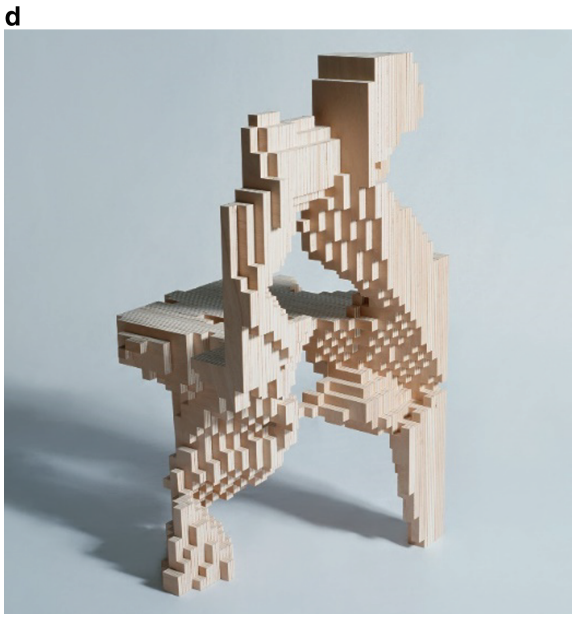


Fig. 7 (continued)

[27], discussed previously in Sect. 2.2, shows a generative design of desk layouts where the outputs are widely expected.

The characterization of a design approach in one subset or another is not always straightforward, and some overlap can be noticed.

In this section, we discussed two particular characteristics of CD: the thinking shift toward computation (vertical axis in Fig. 8) as well as the aim toward emergence, where outputs are non-predictable (horizontal axis in Fig. 8). These characteristics distinguish CD from CAD, as well as from designing using existing parametric objects or automating well-defined workflows, where outputs can be anticipated. They also allow the relative positioning of parametric, generative and algorithmic design to one another, toward a practice of computational thinking and an aim for emergence.

3 Computational Design Toolsets: From Visual Programming to Platformization, a Journey Toward Getting CD into Everyday Practice

In the following section of this introduction to computational design, we interrogate the technical characteristics of CD toolsets, their evolution and adoption in professional practice. The trajectories of these toolsets for the past few years tell us how CD made it into everyday practice and how it might evolve with new emerging

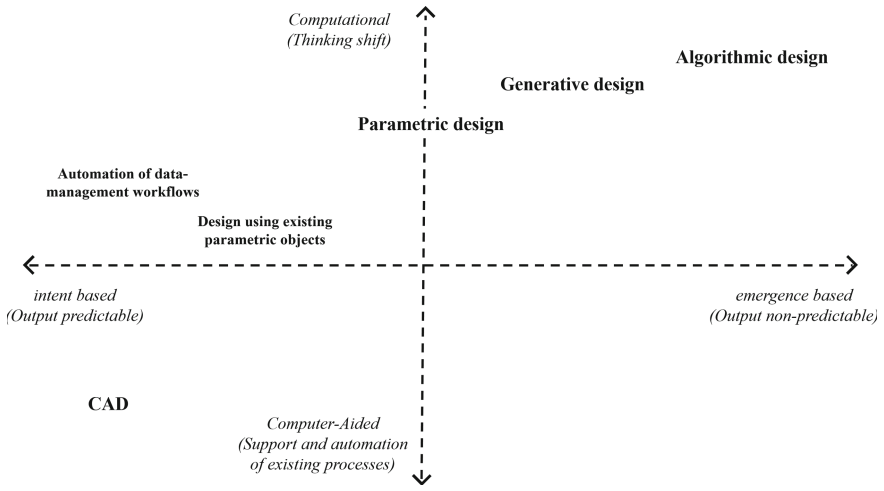


Fig. 8 Parametric, generative and algorithmic design: toward practices of computational thinking and emergence

platforms. Implementing new technologies in practice implies significant costs, with a large part being the development of specific and demanding expertise. This creates an issue with CD accessibility and contributes to adoption in practice being fragmented. This is why the evolution of toolsets into becoming more accessible to beginners is significant for the deployment of CD, and the rise of visual programming tools in the industry has especially been a game changer.

3.1 Programming Toolsets for Designers

Early experimentation of computation in architecture required a big deal of computer science knowledge and skills. The designer toolset first evolved through the appropriation of tools from other industries (such as aerospace or industrial design among others) and later to the development of tools dedicated to artists and to the construction industry.

Today, visual programming tools enable beginners in computation to get started much more easily. Visual programming allows users to describe parameters and rules visually, using a language of blocks, and to create relationships between these blocks using connecting wires (Fig. 9). The produced visual program is often called a script or a graph. Visual programming tools currently used in the AECO industry often come with already defined rules but can also be extended with custom functionality, which can require coding. Visual programming tools and their extensions through external libraries, plugins, etc., enable a wide range of CD practices, including the three discussed subsets: parametric design, generative

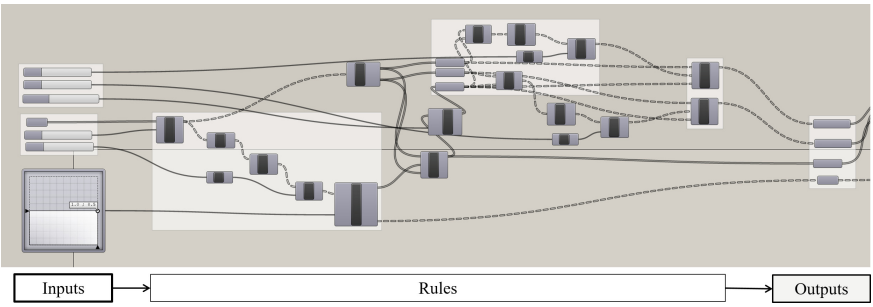


Fig. 9 Examples of a visual programming interface: Grasshopper (McNeel)

design and algorithmic design. As discussed in Sect. 2.1, parametric systems are a bit simpler to apprehend because they are unidirectional. They are also easy to implement in visual programming. Indeed, the loops and feedback needed in generative design or algorithmic design can be difficult to manipulate through wires, while textual programming is more adapted to their implementations.

Generative components, developed in the 90s by Bentley Systems [37], and Grasshopper, developed in the late 2000s by McNeel [38] (Fig. 9), were, and still are, very successful visual programming tools that have had a significant impact on the CD field of practice. In more recent years, Autodesk developed its own visual programming system called dynamo [39] as has Dassault System with XGen [40].

Some authoring tools such as Revit or ArchiCAD use embedded propagation systems to enable parametric objects and a reactive design environment. However, they rely widely on predefined objects, with limited sets of rules and constraints possible presets rules and constraints. While it is very helpful to structure data in models and provide an accessible user experience, it constrains the designer to rely on the existing library of rules and relationships. That is why, most of the time, they are not recognized as being flexible enough to enable CD without the use of programming or visual programming [41, 42].

3.2 Computational Design Platforms

With the increase of computationally demanding processes, it is getting more and more important to enable non-specialists to access CD toolsets. Tools are constantly improving user access to algorithms. For example, HumanUI [43, 44] proposes some toolsets to develop custom user interfaces in order to make complex CD developments accessible to non-specialists. Improvements are also made through library management systems, like dynamo player for example [39], allowing non-specialist users to easily run existing scripts developed by colleagues, with little or no visibility of the code itself.

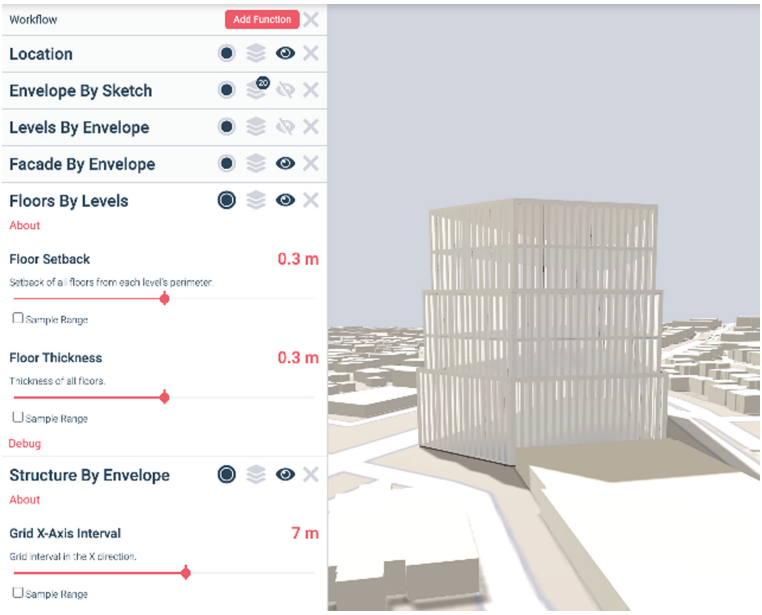


Fig. 10 Screenshot from the Hypar interface (2020), on the right the preview of the resulting instance, on the left the rules are ordered vertically, from top to bottom, with user-friendly inputs exploration capability

A new generation of CD tools is currently emerging to address these issues too, taking a relatively different direction. These are cloud-based platforms and propose to make the CD experience easy to access to a more general audience and not just specialists. Today, platforms like Hypar allow designers to make their scripts available with a user-friendly interface on a web page [45] (Fig. 10). Non-specialist users can access these web "toolkits" and explore CD processes in a relatively easy manner. Hypar is especially designed as a customizable platform allowing any algorithm to be implemented by specialists, while also allowing non-specialist users to access it.

Other platforms like SpaceMaker, ArchiStar or Testfit propose specialized cloud-based tools [33–35] to provide CD as a service. They develop powerful algorithms for various purposes (such as different building configurators, urban feasibility generation and evaluation, automated parking layouts) and make them available to non-specialists, whether they are architects or engineers, but also property developers, town planners, investors, etc. With these platforms, in just a few steps, any user can select a construction site and evaluate thousands of design options according to bespoke performance objectives, such as unit mix, expected density, compliance with local regulations or environmental factors. This could be soon leading the way to different business models for designers. But although these platforms are promising, they are still very new in the industry, and their adoption is just getting started.

3.3 Computational Design and Data

Furthermore, these CD platforms (especially ArchiStar and SpaceMaker) claim developing and embedding artificial intelligence (AI) in their generation and optimization algorithms [33, 34]. While embedding AI in CD algorithms is still relatively new in the industry, it seems to be developing at high speed. But AI requires very large and detailed data sets, which the industry does not fully have yet, even if CD practices focus on data is important.

Indeed, as the adage says, "garbage in, garbage out" which means that if the inputs of an algorithm are not well controlled and well structured, the outputs are likely not to be up to the designer's expectations. CD practices are using and producing loads of data sets, data which can be geometric but also alphanumeric or semantic (e.g., as in the EZCT project in Fig. 7c).

3.4 Designing the Practice's Toolset

To address the current challenges of the AECO industry, designers should not be afraid of designing their own technological environment and especially to develop their tools when the current ones do not meet their needs. With the growing impact of algorithms and computation on architecture, running a practice with ill-suited tools is becoming increasingly risky. Developing tools and ecosystem of tools should now be a part of the designer's job [10, 25, 46].

The development of CD toolsets goes in parallel with the rise of communities of users and developers. These communities of professionals and amateurs alike share problems and advise each other, creating significant knowledge bases on forums, blogs, podcasts and videos. It is worth highlighting that supportive communities are a significant driver for both the CD practice and its adoption.

While the digital transformation of the construction industry is moving so many aspects of collaboration and ownership [47], it is quite surprising to observe that open-source software is still barely used in the AECO industry. Apart from a projects such as Speckle [48, 49] or Blender [50], most software used in the building industry is proprietary, sometimes creating governance issues [51, 52].

4 Computational Design in Practice: Roles and Skillsets

CD adoption in everyday work, as well as the evolutions of those practices and organizations, is a key topic to understand CD [10, 11]. While the evolution of practices, with its organization, roles and competencies, is well studied for building

information modeling[6] (BIM) practices [53–55], the same cannot be said about CD. After discussing what CD entails in Sect. 2, and its toolsets characteristics and evolutions in Sect. 3, in the following section we further discuss the human and organizational aspect of CD practices, especially around its related roles and skillsets.

4.1 *Existing and Emerging Roles in the Field of Computational Design*

Currently, CD practices are still very much "*niche*" and fragmented and cannot be yet described as business as usual for the AECO industry. We can observe that CD practices are very much led by individuals, either as a full specialization, or as integrated in wider scopes. These two polarities, specialization versus integration, are representative of the roles currently available to the CD savvy designer [56].

In integrated roles, most of the time, computational designers are designers with a particular expertise in computation and operate in the office in the same way as their colleagues. Their primary role is to work on projects, and if the project to which they are allocated does not lend itself to computational developments, they will potentially have to put their CD skills on hold. They often go from one project to another without much time to consolidate the development of methods or tools carried out. If they leave the practice, their knowledge often leaves with them. If they train their colleagues, it is often based on their own individual good will and free time, without systematic recognition by the practice management. Whitehead called them "architect plus" [Whitehead in 57].

When in dedicated roles, people get to specialize further on CD and software development in general, often referred to as "computational design specialists". Computational design specialists provide training for their colleagues and develop tools and assets (methods, scripts, etc.) for the office. They can work on projects, but mostly on contributions related to their CD expertise, allowing them to further extend their experience and skills. Large practices can have their own dedicated specialist teams. That is the case at Foster + Partners with the Specialist Modeling Group and the Applied Research and Development one [26, Whitehead in 57], at Zaha Hadid Architects with the Code Group, or at Grimshaw with the Design Technology Department [56]. Consultancy practices have been flourishing too, opening the way for different business models and practices, like with the experiences of Gehry Technologies or Case [11].

Both integrated and dedicated roles have their advantages and disadvantages, and today there is no one-size-fits-all solution. When establishing a CD resourcing strategy, the office's ambition, needs and investment capacity should all be carefully

[6] See Chapter "Building Information Modelling and Information Management".

considered. In particular, CD resourcing strategy should be defined in close relation to a wider digital transformation plan for the office.

Currently, specialized and integrated roles are cohabitating in practices, allowing a wide range of possible positions and integration strategies. Unfortunately, these blurred lines between roles also mean that computational designers are not always as well recognized as they could be, and their career paths and perspectives often lack visibility [10].

4.2 *Skillsets and Competencies: The Figure of the* Superuser

The figure of the *superuser* proposed by Deutsch in 2019 has been quite successful among computational designers. The *superuser* finally described and crystallized a complex and not very well-documented professional reality, in terms of skills, roles and career. It highlights the cultural and technical transformation underway and puts a name on a series of characteristics of new actors of the digital transformation, to finally recognize and make visible the complexity of their roles and skillsets. Anyone can identify as a *superuser*, whether they are project manager, architect, engineer or others, and, ultimately, whether they are specialists or not.

Superusers are experts in CD and digital design technologies in general and are defined by their skills rather than by their roles. While very strong technical knowledge is recognized in *superusers*, it is not what identifies them most. The competences of a *superuser* should comprise the following: firstly the ability for computational thinking, but also inter-personal skills, such as the ability to communicate, connect and collaborate [10, 57, 58], and an ability to conceptualize and structure strategies for a project or for an office [10]. The figure of the *superuser* therefore goes beyond the distinctions between CD subsets (parametric, algorithmic and generative), or between CD and other digital practices such as BIM.

Many fears and uncertainties exist regarding the transformation of the building industry and its jobs, especially with regard to automation [9, 59, 60]. But the increase in expertise and the evolution of means of production are not new phenomena for designers. For sociologist Olivier Chadouin, the architectural profession has always encountered new skills and new scopes and persists in time and through difficulties thanks to its great heterogeneity [61]. As such, the construction industry should not be afraid to embrace new skills such as computational ones. Nevertheless, integrating heterogeneous skillsets and roles within practice must be thoughtfully considered. Most practices are still struggling with digital transformation and have difficulties in identifying opportunities for the application of CD, as well as difficulties with recognition and career paths for the *superusers* [10].

5 Conclusions: The Futures of CD Practices

5.1 Computational Capabilities for Design: A New Paradigm

In this chapter, we have discussed how important it is to distinguish computational thinking from a computer-aided mindset. While computer-aided systems support making traditional existing processes faster, computational ones enable completely new processes and ways of thinking. Computational design (CD) is not just automating existing traditional processes or tedious tasks; it is about shifting the way we think and design.

CD systems can be grouped into three main categories. Parametric systems are propagation based, allowing the designer to organize rules, inputs and relationship in specific orders, and enabling the exploration of a wide ranges of solutions. Generative design systems are ruled-based too, but use more diverse algorithmic approaches and introduce goals. They enable the designer to define a problem space and the algorithms that will produce the solution space, which can then be widely explored or searched. Algorithmic design usually describes systems that fits neither in the parametric or generative categories. In the end, it is a broader CD subset, allowing more freedom in the balance between intention and emergence, enabling explorations of unpredictable outputs.

5.2 Toward a Platformization of CD

In its early development in the 60s, the coding knowledge needed in order to leverage computational thinking was a limitation to its adoption in the day-to-day practice. It is not the case anymore thanks to various toolsets developments. In particular, visual programming has been a game changer for the adoption of CD in the AECO industry. Some further developments, especially in cloud-based platform, seem to be next in line to transform the involvement of computation in practice. These platforms propose a new kind of CD service, enabling access to powerful CD algorithms to non-specialists.

5.3 CD and Data-Centric Practices: A Possible Reconciliation with BIM?

Data-centric approaches are on the rise [5, 6, 26] in the construction industry, especially with AI-related targets [6, 62]. Data quality is a major issue for all industries today. In data science, it is usually considered that the work consists of 80% of preparing the data and only 20% of exploiting it [63]. In the construction

industry, for most firms, the data is neither controlled nor unified and remains disparate and partial from one project to another. BIM practices arise as a powerful means of defining a base of practices aimed at ensuring the quality of data and its exchanges, making it feasible that the data produced can be used during and after the project as needed.

This makes it even more surprising to see the Chinese wall that seems to exist between CD and BIM. BIM can be regarded as focused on data-oriented collaborative practices. When we see the rise of interrogations around CD and data as well as CD and data exchange [26, 64, 65], the lack of interaction between the two fields can be surprising.

Several explanations could be discussed. One differentiating aspect is that while BIM is involved all along the project life cycle, the early design stages do not seem to be its strongest use case. If BIM is key in the expansion of architectural workflows [24], it is often criticized for missing the point of design by an often too strong focus on production means instead of design objective [66] and by having technological foundations too rigid to address the flexibility required by early design stages [6, 42].

For some authors, it is not that BIM and CD must be kept apart, but that they should merge as they evolve. That is the case for Aish and Bredella, for whom BIM must evolve toward CD in order to fully integrate with it; otherwise, it will remain locked in its technological premises, ill-suited to architectural design [42]. On the other hand, several authors interrogate BIM and CD through the angle of communication, interoperability and data-sharing issues [26, 49, 64, 67]. This opens very interesting fields of research and practice, opening the possibility of breaking the silos between BIM and CD in order to develop strong data-centric practices for the AECO industry.

5.4 The Importance of the Digital Transformation for CD Adoption

Despite the numerous opportunities and advantages of computation, the adoption of CD in everyday practice is still very niche. The general digital transformation of the AECO industry is slow, as is the adoption of CD. This greatly impacts the opportunities for designers to nurture their skills and get recognition for it. The recent success of the book *Superuser* from Deutsch [10] is a good indicator: Very little attention is given to the actors and the drivers of the AECO's digital transformation. Their skills, roles and career opportunities are neither clear nor easy. Further to solid technical and cultural knowledge on computation in general, interpersonal skills as well as leadership ones are greatly needed for these CD actors, whether they work in a CD dedicated team, or in design positions with CD exposure.

In the coming years, it is likely that the challenges the AECO industry faces will continue to multiply. We will need all the intelligence and means possible, and CD is enabling great possibilities. However, implementing CD requires the industry to go through a significant digital transformation, which is costly in terms of investments, and its factors for success or failure are complex and resource intensive. Confusion, anxiety, resistance, frustration: if poorly led, the effects of risky change management are harmful. Yet multiple companies rely on enthusiastic individuals to drive change, without fully supporting their efforts over time. If practices' leadership does not seriously engage in the process, then these computational designers will face many difficulties, causing frustration and disillusion. These computational skills take a long time to develop, and if they cannot flourish and struggle to have an impact in AECO as it is currently, they might leave the industry.

Resilience and adaptation are key in our world today, and the multiple capabilities CD enables can help current and future AECO stakeholders to not be victims of the upcoming industry's challenges, but rather be empowered actors bringing about change.

References

1. Picon, A.: Digital culture in architecture: an introduction for the design professions. Birkhauser Architecture (2010)
2. Menges, A., Ahlquist, S.: Computational Design Thinking. Wiley (2011)
3. Negroponte, N.: Towards a humanism through Machines. In: Menges, A., Ahlquist, S. (eds.) Computational Design Thinking. AD Readers (1969)
4. Denning, P.J., Tedre, M.: Computational Thinking. MIT Press (2019)
5. Bernstein, P.G.: Architecture, design, data : practice competency in the era of computation. In: Birkhauser Architecture (2018)
6. Carpo, M.: The second digital turn: design beyond intelligence. MIT Press (2017)
7. Terzidis, K.: Algorithmic design: a paradigm shift in architecture? In: Proceedings of the 22nd eCAADe Conference, pp. 201–207 (2004)
8. Terzidis, K.: Algorithmic Architecture. Architectural Press (2006)
9. Susskind, R.E., Susskind, D.: The future of the professions: how technology will transform the work of human experts. Oxford University Press (2015)
10. Deutsch, R.: Superusers: design technology specialists and the future of practice. Routledge (2019)
11. Shelden, D.: The disruptors: technology-driven architect-entrepreneurs. Wiley (2020)
12. Caetano, I., Santos, L., Leitão, A.: Computational design in architecture: defining parametric, generative, and algorithmic design. Front. Architec. Res. (2020)
13. Davis, D.: The future of architectural discourse. https://www.danieldavis.com/, https://www.danieldavis.com/the-future-of-architectural-discourse/ (2013). Last accessed 2021/01/31
14. Aish, R., Woodbury, R.: Multi-level interaction in parametric design. Lect. Notes Comput. Sci. **3638**, 151–162 (2005)
15. Janssen, P., Stouffs, R.: Types of parametric modelling. In: CAADRIA 2015—Proceedings of the 20th International Conference of Association for Computer-Aided Architectural Design Research in Asia, Emerging Experience in Past, Present and Future of Digital Architecture, pp. 157–166 (2015)
16. Woodbury, R.: Elements of parametric design. Routledge (2010)

17. Schumacher, P.: Parametricism: a new global style for architecture and urban design. Archit. Des. **79**, 14–23 (2009)
18. Burger, S.: Natural and intuitive. http://shaneburger.com/2011/08/designing-design/ (2011). Last accessed 2021/01/31
19. Carlos, C., Richard, N.: Beyond modelling: avant-garde computer techniques in residential buildings. Jornadas Investig. en Constr. (2005)
20. Kolarevic, B.: Architecture in the Digital Age: Design and Manufacturing. Spon Press (2003)
21. Menges, A.: Integral formation and materialisation, Computational form and material gestalt. In: Computational Design Thinking. Wiley (2008)
22. Migayrou, F.: Architecture non standard. Centre Pompidou ed (2003)
23. Krüger, S., Borsato, M.: Developing knowledge on digital manufacturing to digital twin: a bibliometric and systemic analysis. Procedia Manuf. **38**, 1174–1180 (2019)
24. Garber, R.: Workflows: Expanding architecture's territory in the design and delivery of building. Wiley (2017)
25. Peters, B., De Kestelier, X.: Computation works: the building of algorithmic thought. Wiley (2013)
26. Thomsen, M. R., Tamke, M.: Design transactions. UCL Press (2020).
27. Anderson, C., Bailey, C., Heumann, A., Davis, D.: Augmented space planning: using procedural generation to automate desk layouts. Int. J. Archit. Comput. **16**, 164–177 (2018)
28. Pottman, H., Asperl, A., Hofer, M., Kilian, A., Bentley, D.: Architectural geometry. Bentley Institute Press (2007)
29. Burry, J., Burry, M.: The new mathematics of architecture. Thames & Hudson (2010)
30. Leach, N.: Digital Cities. Wiley (2009)
31. DeLanda, M.: The limits of urban simulation. In: Digital Cities. Wiley (2009)
32. Morel, P., Agid, F., Feringa, J.: Studies on optimization : computational chair design using genetic algorithms. http://transnatural.org/wp-content/uploads/2011/09/EZCT_Booklet-Screen.pdf (2004). Last accessed 2021/01/31
33. SpaceMakerAI.: Early stage planning. Re-imagined. https://www.spacemakerai.com/ (2020). Last accessed 2021/01/31
34. ArchiStar.: ArchiStar property insights. https://archistar.ai/ (2020). Last accessed 2021/01/31
35. TestFit.: TestFit: the world's most powerful building configurator. https://testfit.io/ (2020). Last accessed 2021/01/31
36. Terzidis, K.: Algorithmic form. In: Computational Design Thinking (2003)
37. GenerativeComponents.: An overview of GenerativeComponents. https://communities.bentley.com/products/products_generativecomponents/w/generative_components_community_wiki (2003). Last accessed 2021/01/31
38. Grasshopper.: Grasshopper, algorithmic modeling for Rhino. https://www.grasshopper3d.com/ (2007). Last accessed 2021/01/31
39. DynamoBIM.: Open source graphical programming for design. https://dynamobim.org/learn/ (2016). Last accessed 2021/01/31
40. XGenerativeDesign.: Generative design engineering. https://ifwe.3ds.com/media/generative-design-engineering (2018). Last accessed 2021/01/31
41. Davis, D.: Modelled on software engineering: flexible parametric models in the practice of architecture. RMIT University (2013)
42. Aish, R., Bredella, N.: The evolution of architectural computing: from building modelling to design computation. arq Archit. Res. Q. **21**, 65–73 (2017)
43. Berg, N.: NBBJ releases human UI to bring parametric modeling to the masses. Architect, Journal of the American Institute of Architects (2016)
44. Heumann, A.: Human UI. Github https://github.com/andrewheumann/humanui (2016). Last accessed 2021/01/31
45. Hypar.: Hypar live. https://www.youtube.com/watch?v=VAFXAcwXNDU (2020). Last accessed 2021/01/31

46. Davis, D.: Design ecosystems: customising the architectural design environment with software plug-ins. https://www.danieldavis.com/design-ecosystems-customising-the-architectural-design-environment-with-software-plug-ins/ (2013). Last accessed 2021/01/31
47. Fok, W., Picon, A.: Digital property: open-source architecture. Wiley (2016)
48. Speckle.: Introduction to Speckle. https://github.com/speckleworks (2020). Last accessed 2021/01/31
49. Stefanescu, D.: Alternative means of digital design communication. In: Sheil, B., Thomsen, M.R., Tamke, M., Hanna, S. (eds.) Design Transactions. UCL Press (2020)
50. Blender Foundation.: Blender.org. https://www.blender.org/foundation/ (2002). Last accessed 2021/01/31
51. Davis, D.: Architects versus autodesk. The magazine of the American Institute of Architects (2020)
52. Collective. An open letter that reflects customer perspectives on Autodesk in 2020 (2020)
53. Succar, B., Kassem, M.: Macro BIM adoption: conceptual structures. Autom. Constr. **57**, 64–79 (2015)
54. Hochscheid, E., Halin, G.: A framework for studying the factors that influence the BIM adoption process. In: 36th CIB W78 2019 Conference ICT in Design, Construction and Management in Architecture, Engineering, Construction and Operations, pp. 275–285 (2019)
55. Ahmed, A.L., Kawalek, J.P., Kassem, M.: A comprehensive identification and categorisation of drivers, factors, and determinants for BIM adoption: a systematic literature review. Comput. Civ. Eng. **2017**, 220–227 (2017)
56. de Boissieu, A.: Super-utilisateurs ou super-spécialistes ? Cartographie des catalyseurs de la transformation numérique en agence d'architecture. Les Cahiers de la Recherche Architecturale urbaine et paysagère **10** (2020)
57. de Boissieu, A.: Modélisation paramétrique en conception architecturale: caractérisation des opérations cognitives de conception pour une pédagogie. Universite Paris-Est (2013)
58. Davies, K., McMeel, D., Wilkinson, S.: Soft skills requirements in a BIM project team. In: Proceedings of the 32nd International Conference of CIB W78 (2015)
59. McAfee, A., Kestenbaum, D.: Experts debate: will computers edge people out of entire careers?. NPR (2015)
60. Davis, D.: Why architects can't be automated. Architect Magazine (2015)
61. Chadoin, O.: Etre architecte, les vertus de l'indétermination. Presses Universitaire Limoges (2013)
62. Miller, N., Stasiuk, D.: Negotiating structured building information data. In: Sheil, B., Thomsen, M.R., Tamke, M., Hanna, S. (eds.) Design Transactions. UCL Press (2020)
63. Anderson, C.: Creating a Data-Driven Organization. O'Reilly (2015)
64. Poinet, P.: Enhancing collaborative practices in architecture, engineering and construction through multi-scalar modelling methodologies. Aarhus School of Architecture (2020)
65. Miller, N.: [make]SHIFT: Information exchange and collaborative design workflows. In: ACADIA, pp. 139–144 (2010)
66. Fano, D., Davis, D.: New models of building: the business of technology. In: Shelden, D. (ed.) The Disruptors. Wiley (2020)
67. Boeykens, S.: Bridging building information modeling and parametric design. In: eWork and eBusiness in Architecture, Engineering and Construction—Proceedings of European Conference on Product and Process Modelling, ECPPM 2012, pp. 453–458 (2012)