ALGORITHMIC PATTERNS FOR FAÇADE DESIGN

Merging Design Exploration, Optimization and Rationalization

ABSTRACT

Recently, building envelopes have been exhibiting complex shapes and patterns, a trend supported by current digital technologies. Likewise, the design exploration of these envelopes has been combined with analysis and optimization processes, with the aim of achieving better performing solutions. Nevertheless, the exploration of architectural skins still poses some limitation to architects when using new design approaches, such as algorithmic design, simulation, and optimization strategies, as they require specialized expertise. Furthermore, the integration of design analysis and optimization processes in the algorithmic design workflow has shortcomings, such as the fragmentation resulting from the use of multiples models and tools, a process that is error-prone, time-consuming, and hard-working.

In this paper, we discuss a framework developed to ease the mathematical description of algorithmic façade designs, which intends to integrate the most used analysis and optimization processes in a continuous design exploration workflow. To this end, a classification of façades was developed, based on computationally relevant categorical dimensions, that integrates a set of fundamental algorithms and strategies for each dimension. The result is a library of algorithms available in different programming languages, and a set of guidelines that help architects select and combine the most useful algorithms for a given façade design. The algorithmic structure of the developed solutions is flexible, hence, suitable to connect with the chosen analysis and optimization processes, allowing their inclusion in a continuous workflow.

KEYWORDS

design processes; parametric workflows; generative design; design optimization

INTRODUCTION

Architecture has always explored the latest technological advances in terms of building technology, as well as architectural production and representation. Nowadays, the new digital tools play a relevant role in the conception, analysis, and production of architecture. Computation-based design approaches promoted design exploration and the emergence of complex shapes and patterns, which would be difficult and costly to produce without them (Moneo 2001). Within the architectural domain, these approaches had, and still have, great application in the development of building' façades due to the aesthetical and environmental relevance of the latter ones, since these mediate the inside and the outside of a building. Computational approaches also potentiate the design optimization of buildings (Alfaris & Merello 2008). Using optimization algorithms and one or more performance criteria, it is possible to automatically explore a substantial design space in the search for solutions with an improved performance. Finally, computer-aided manufacture enabled the fabrication of such solutions by allowing the conversion of virtual models into physical ones (Gibson et al. 2010), which has been changing the relation between conception and production (Kolarevic 2003).

Despite these advances, computation-based approaches are not yet widespread, mainly due to their complexity and the specialized knowledge required, especially the algorithmic approaches, which require architects to learn programming. Furthermore, it still takes time for the designer to gain enough experience to deal with more advanced algorithmic problems. This means the designer often ends up spending more time with the scripting task than with the design exploration task, thus limiting his creativity. The challenge, then, is to make such approaches, including algorithmic design, design optimization, and digital fabrication, accessible to architects. The aim of this work is to explore the application of these approaches in the design process of a building, more specifically, in the development of its envelope. The focus on the envelope results from the specialized knowledge and work that is usually required for its development, mainly when the goal is meeting several requirements, namely, aesthetical, structural, and environmental ones.

To minimize the existing limitations when using algorithmic approaches, a computational framework is proposed, aiming to reduce the initial time investment required for the scripting task, specifically in the production of building skins. This architectural-oriented framework systematizes and structures the algorithmic generation of façade solutions, embracing both the variety of different scenarios and the different analysis and optimization strategies that are now part of the design process. The framework

includes a library of predefined algorithms based on previous design experiences and known façade design strategies, assisting architects in the exploration of a façade solution by suggesting the most suitable strategies and algorithms for each specific situation. Therefore, it decreases the time needed to explore the desired idea, while promoting a better and more appropriate use of the different algorithmic strategies; an important milestone for the subsequent analysis and optimization stages.

BACKGROUND

ALGORITHMIC DESIGN

Algorithmic Design (AD) is a design approach that describes a design through a set of rules and algorithms (Alfaris & Merello 2008), allowing the designer to transcend "the factory-set limitations of current 3D software" (Terzidis 2004). Nevertheless, to take advantage of AD, architects need to learn new skills, such as programming, which is far from trivial. To promote the integration of AD in architecture, some programming languages (PL) were developed to facilitate the learning process to those without previous programming experience. Some PLs are textual (TPL), namely *Processing, Racket,* and *Python*, but the most popular ones are visual (VPL), such as *GenerativeComponents, Dynamo,* and *Grasshopper*, which allow the creation of AD programs through the manipulation of graphical elements instead of using text (Janssen 2014). Their popularity derives from (1) their intuitiveness, a crucial asset for beginners as they require less abstraction; (2) their ability to produce results more easily, without almost having to learn programming (Zboinska 2015); (3) the integration of important features to aid architects with the programming task, namely traceability (i.e. the relationship between parts of the program and those of the generated model), important for the comprehension, maintenance, and debugging of AD programs (Leitão et al. 2014), and (4) the real-time feedback, e.g., the use of sliders to represent parameters is relevant for immediately understanding the effects of changes in the programs.

Unfortunately, VPLs still present several limitations, mainly when projects get more complex: (1) the lack of scalability - the program gets incomprehensible and, thus, difficult to change (Leitão et al. 2012b; Janssen 2014); (2) confining the user to the predefined modules and components available in each VPL, a limitation accentuated when the solutions being explored need more advanced features (Zboinska 2015); and, finally, (3) the loss of capacity to support real-time feedback when programs get more complex, due to the software's performance decrease (Leitão et al. 2014). To overcome these limitations when dealing with more intricate solutions, architects usually need to replace VPLs with TPLs (Leitão et al. 2012a), as these are more powerful than the former. Unfortunately, mastering a TPL takes time and, when the programming expertise is still limited, architects often spend more time with the scripting task than with design exploration. Moreover, when facing unanticipated changes in their design process, the parametric model sometimes does not contain the adequate parameters to modify the model in the desired way (Davis et al. 2011). As a result, the model's programming structure is typically redrawn, a time-consuming process described by Woodbury as "erase, edit, relate and repair" (2010).

ALGORITHMIC PATTERNS

To promote the integration of algorithmic approaches in architecture, the use of algorithmic design patterns is currently in development. Qian (2009) defined *pattern* as a generic solution to a well-defined problem, while Alexander et al. (1977) and Qian et al. (2008) described *design patterns* as known strategies or solutions that can be reused to solve design problems. The main goal for developing algorithmic patterns is taking advantage of previously used knowledge and solutions in the search of new ones. As a result, this avoids repeated reinvention, while reducing the development time of algorithmic solutions (Khwaja & Alshayeb 2013) and anticipating unexpected design changes during the design process (Woodbury et al. 2007). Some works based on this issue have been developed (Woodbury et al. 2007; Qian 2009; Yu & Gero 2015; Woodbury 2010; Chien et al. 2015; Hudson 2010). Regarding the field of façade design, Su & Chien (2016) developed a set of algorithmic patterns to support the development of architectural skins at an initial stage using *Grasshopper* for *Rhino 3D*. Nevertheless, most of these studies (I) simply focus on the modularization of VPLs definitions, therefore also suffering from their inherent shortcomings; (II) do not support the integration of other types of algorithms, such as optimization algorithms, neither connecting them to the multiple analysis and simulation tools; and (III) do not address the application of algorithmic patterns in the exploration of buildings' façades.

On the other hand, tools like *Paneling Tools* for *Rhino* and *Grasshopper*, *Lunch Box* to *Grasshopper*, and *ParaCloud Gem*, also attempt to solve the mentioned limitations by addressing and simplifying typical modelling procedures applied in façade design – point-grids generation, mapping of elements in different ways, application of *attractors* to control element sizes, rotation, etc. Unfortunately, (1) its use is entirely manual (i.e. by interacting directly with the software environment instead of using algorithms), promoting iterative user-driven processes that can be tiresome and error-prone; (2) they resort to VPLs and, once more, suffer

from the typical limitations they possess; and (3) they are mostly used for generic panelization, subdivision, and population of surfaces, thus, being unable to directly address relevant concepts in façade design, such as materiality or the tectonic relation between elements.

To sum up, architects are limited by the specificity of existing tools – to extend their capabilities, they need either to build from scratch the necessary functionalities or to use a mix of different tools that, most of the time, are not compatible. For this reason, it is important to systematize and structure the algorithmic generation of façade solutions in a process that considers the variety of different scenarios and the set of different analysis and optimization strategies. The aim of the DrAFT framework is to create a library of predefined algorithms based on the notion of design patterns – i.e., by integrating design processes and known façade design strategies – to support architects in the exploration of an intended façade solution by suggesting the most suitable strategies and algorithms for each specific situation. It therefore reduces the time needed to explore the desired idea, while promoting a better and more appropriate use of the different algorithmic strategies; an important milestone for the subsequent stages – analysis, optimization, and fabrication**.**

FAÇADE RATIONALIZATION FOR FABRICATION

The design freedom achieved by the latest technologies is limited by physical constraints usually linked to structural performance and fabrication. One of the current concerns of designers is the ease of manufacturing of the complex solutions being developed, a process named as *geometric optimization* (Mesnil et al. 2015), that includes *rationalization*, i.e., the adjustment of a geometrically complex design towards a feasible and affordable way of production. As an example, the fabrication of large-scale free-form structures can be quite challenging, since the designed surface needs to be rationalized through a process named *panelling* (Fu & Cohen-or 2010; Eigensatz, Deuss, et al. 2010; Son et al. 2017; Andrade et al. 2017; Eigensatz, Kilian, et al. 2010; Flöry & Pottmann 2010), which corresponds to the segmentation of the surface into simpler elements that can be manufactured at a reasonable cost, while preserving the design intent (Eigensatz, Kilian, et al. 2010).

However, most of these works only focus on simple patterning, i.e. triangular, quadrangular, and hexagonal meshes, whereas more complex geometric patterns are not considered, even though they have been applied to several architectural façades. It is therefore important to also consider the complex types of geometric patterning by providing an algorithmic framework of coherent mathematical definitions to facilitate the exploration and subsequent rationalization of these patterns.

METHOD

DRAFT FRAMEWORK

Architectural practice is highly dependent on the specific circumstances of the design brief and, thus, it is unlikely that the exact same approach can be used in a different project. Nevertheless, modular programming techniques allow designers to adapt and reuse ideas in different projects. This implies that the systematic application of these techniques reduces, at least partially, the initial investment required, as they would not have to rewrite all the algorithms from scratch every time they started a new design, decreasing the time and effort spent with the programming task. Furthermore, the use of design patterns would also benefit from the integration of different analysis and optimization processes in a continuous workflow, in which the traditionally detached design stages – design exploration, analysis and optimization – are now merged into a single process. As a result, the obtained solutions (1) are less exposed to errors, (2) respond more easily to the creative needs of the architect, and (3) achieve better performing behaviors.

Draft Algorithmic Façades Tool (DrAFT) is a framework based on a classification of façades according to different dimensions where, for each one, several algorithms were developed, covering a wide variety of design solutions (Caetano et al. 2015; Caetano & Leitão 2016). The classification is based on four main categorical dimensions that are further subdivided to deal with the specific details of each design stage: A) *Façade Geometry* defines the façade's surface geometry; B) *Elements* generates the façade's elements; C) *Distribution* makes the elements distribution along the façade's surface; and D) *Articulation* gives a final appearance to the façade by relating the previous dimensions. The DrAFT framework guides the designer in the identification of the most suitable algorithms for his design intent, promoting further design exploration and easier adaptation to the ever-changing design process conditions.

With the aim of empowering the architects' design exploration process, the DrAFT framework also provides a growing set of random optimization algorithms, including, e.g., Monte Carlo and Latin Hypercube sampling methods, to help users improve their design solutions. These depend on a fitness function and on the specification of the permitted variations, allowing the

search for the closest solution to that criterion, while simultaneously maintaining the initial design concept imposed by the architect. These algorithms were designed to be combined with the algorithms responsible for the geometric exploration, which are adapted during this stage, following a continuous integration of optimization processes throughout the design exploration workflow. Moreover, they allow a direct and automated connection with the analysis tools required for each type of analysis.

Finally, as the DrAFT framework gives a high level of flexibility to its user in terms of geometric exploration, it is important to ensure the obtained solutions are then possible to manufacture in terms of its assembly process and cost of fabrication. The DrAFT framework takes this into consideration by providing algorithms that allow the designer to control the maximum number of different elements that composes his design solution. This is done by discretizing a given parameter of the façade solution, e.g., a length, into *n* different values, where *n* can be specified by the designer, producing as a result *n* different sets of identical façade elements, thus enabling him to make a trade-off between the production cost and the original design concept. The potential of DrAFT in exploring a complex design solution is explained in the next section.

CASE STUDY APPLICATION

In this section, the DrAFT framework is used to develop a conceptual case study: the envelope of a rectangular building with a geometrical pattern that varies parametrically according to aesthetical and performance criteria. Initially, both the building's skin shape and the geometrical pattern were developed by resorting to a set of algorithms of the DrAFT framework. The façade exploration process was based on a conceptual idea of creating an envelope composed of several parametric diamond-shaped elements of different apertures (Figure 1). The size variation of the apertures should satisfy the need for natural light, while simultaneously matching the aesthetical criteria for the overall design, which aimed at creating a parabola curve shaped effect along the longest façades.

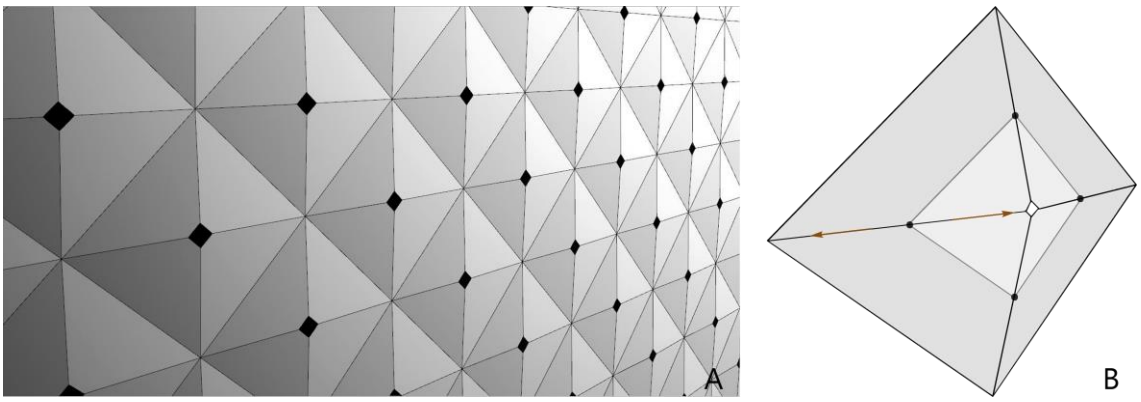

Figure 1: A – a conceptual model of the diamond-shaped façade elements; B – a conceptual representation of a single façade element, in which the grey line represents its deformation variation.

In practical terms, two algorithms were combined to create the diamond-shaped grid: one from the *Façade Geometry* dimension to shape the surface, and the other from the *Distribution* to produce a grid of points placed in a chess distribution – see Figure 2.A. Then, each grid cell was developed by combining some algorithms from the dimension *Elements*, which were responsible for (1) shaping the units in a diamond-shaped pyramid geometry (Figure 2.B), and simultaneously (2) applying a deformation to this shape (Figure 2.C) – i.e., the opening and closing of the pyramidal shape. The parametric variation of each cell depends on the distance between the cell and the attractor curve – Figure 2.D shows the initial geometrical pattern (without deformation), whereas Figure 2.E presents the deformation created by the attractor curve. The use of DrAFT algorithms allowed to iteratively adjust (1) each cell geometric shape, and (2) the applied deformation (i.e. the intensity of the attractor's effect, the geometrical limits of the deformation, etc.) almost instantaneously and without extra effort.

The following stage focused on the optimization of the obtained geometrical pattern, i.e. the search for a solution that, based on the user's design intent and the aesthetical requirements that came from it, was improved in terms of its performance regarding the quality of the building's natural lighting.

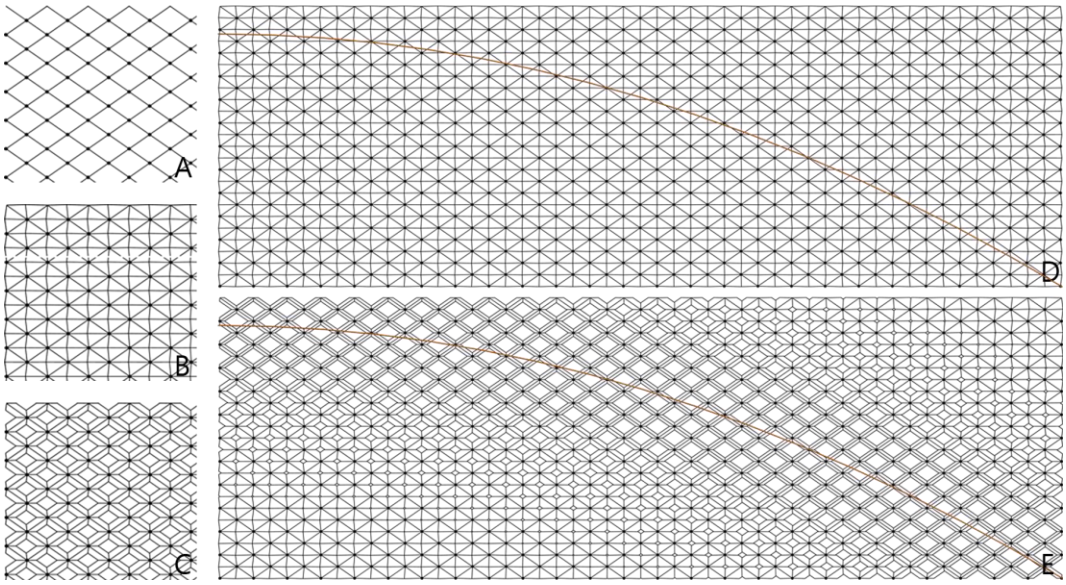

Figure 2: A conceptual representation of the building envelope pattern: A – creation of the diamond-shape grid; B – the distribution of the pyramidal elements; C – deformation of the pyramidal elements (creation of apertures); D – the overall pattern without deformation and the placement of the attractor curve; E – control of the elements' deformation according to their distance to the attractor curve.

DESIGN OPTIMIZATION – OPTIMIZATION ALGORITHMS

In this section, one of the optimization algorithms available within DrAFT framework – the Monte Carlo optimization algorithm (Shapiro 2003) – is used to optimize the design solution developed regarding its natural lightning levels. The goal was to improve these levels by geometrically controlling the façade pattern. To simplify the process, only one of the design constraints imposed could be parametrically changed during this process, which, in this case, was the intensity of the deformation caused by the attractor curve, and only one fitness criterion was intended to be achieved – a predetermined lightning level. The optimization algorithm used received as inputs (1) the fitness function, (2) the range of values accepted by the design parameter, and (3) the number of iterations (a higher number of iteration increases the probability of achieving a better solution as it evaluates a larger design space). After the prescribed number of iterations, the algorithm returns the value of the design parameter that corresponds to the best solution found so far, which can then be used to generate the final design. Figure 3 shows an instance of the case study 3D model with its corresponding lightning analysis results.

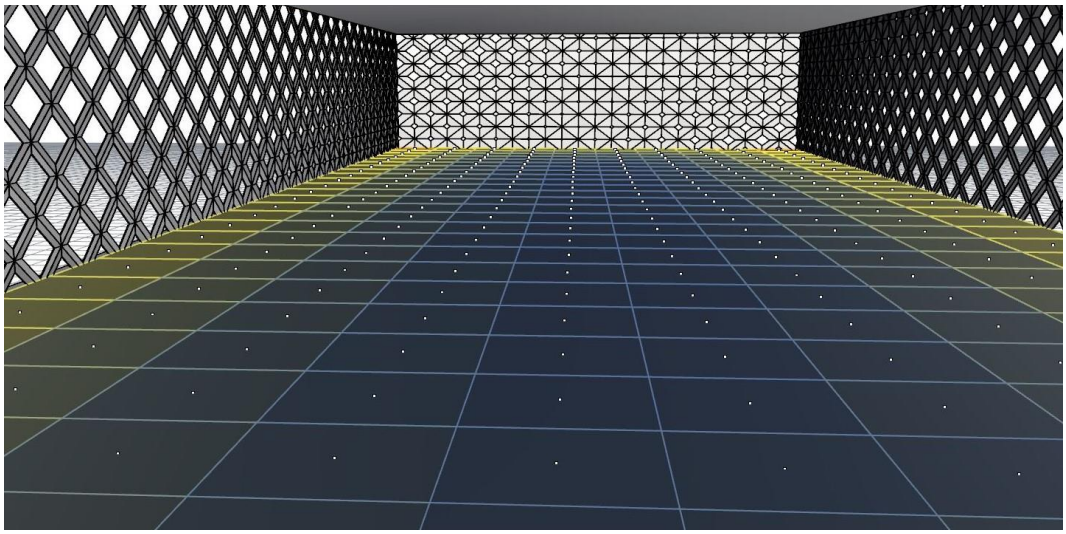

Figure 3: A 3D model view of the lightning analysis process.

DESIGN RATIONALIZATION FOR FABRICATION

One of the consequences of the use of algorithmic design methods is the expensive mass-customization they typically entail. In our case study, that is reflected in the large number of different pieces of the obtained façade design. To overcome this problem, the rationalization algorithm available in the DrAFT framework was applied to reduce this number. The initial solution was composed by 2424 different pieces, in a total of 6144, and, as the manufacturing process of these elements required the creation of molds, its fabrication was clearly unsustainable in terms of cost and waste. Using the rationalization algorithm allowed us to significantly reduce the number of different pieces, enlarging the number of identical pieces on each set. Table 1 organizes the values obtained in the different rationalization tests, which includes the number of different types of pieces, the *total of sets*, and the number of times each type is used, the *sets of pieces*. Figure 4 shows each solution's corresponding façade model. After several iterations, the solution that best matched the original idea, and that, simultaneously, had an acceptable manufacturing cost, was the one of discretization N = 10**.**

N		∞	100	60	20	10	5
Set of pieces	A	432	478	478	478	728	1208
	B	1560	1578	1602	1684	1806	2558
	C	2	70	116	254	502	1054
	D	2	36	108	250	488	1324
	E	2	44	108	246	546	
	F	2	52	120	256	578	
	G	2	52	122	254	640	
	H	4	46	118	260	856	
	I	4	60	140	268		
	J	4	42	134	274		
	K	2	54	154	300		
	...	…	…	...	...		
Total of Sets		2424	69	42	15	8	4

Table 1: A table with the number of different sets of pieces and the number of times each set is used. Note the case of N = 5, the façade pattern only uses 4 typologies of sets, meaning that the number of times each type of set appears is by far greater than the other examples.

EVALUATION

This paper was purposely structured to clearly mark the transition between design stages, with the aim of facilitating the understanding of how the combination and implementation of the different algorithms is done. In practical terms, the design stages are confined in a single two-way workflow. The use of an algorithmic approach allows the user to develop a single script that contains all the geometric information of a design together with algorithms to evaluate and optimize that same design, regarding different performing criteria. Also, it frees the designer from having to create 3D models in different formats to suit the different design and simulation tools, avoiding problems such as loss of information and error propagation, thus potentiating more accurate results in a shorter time.

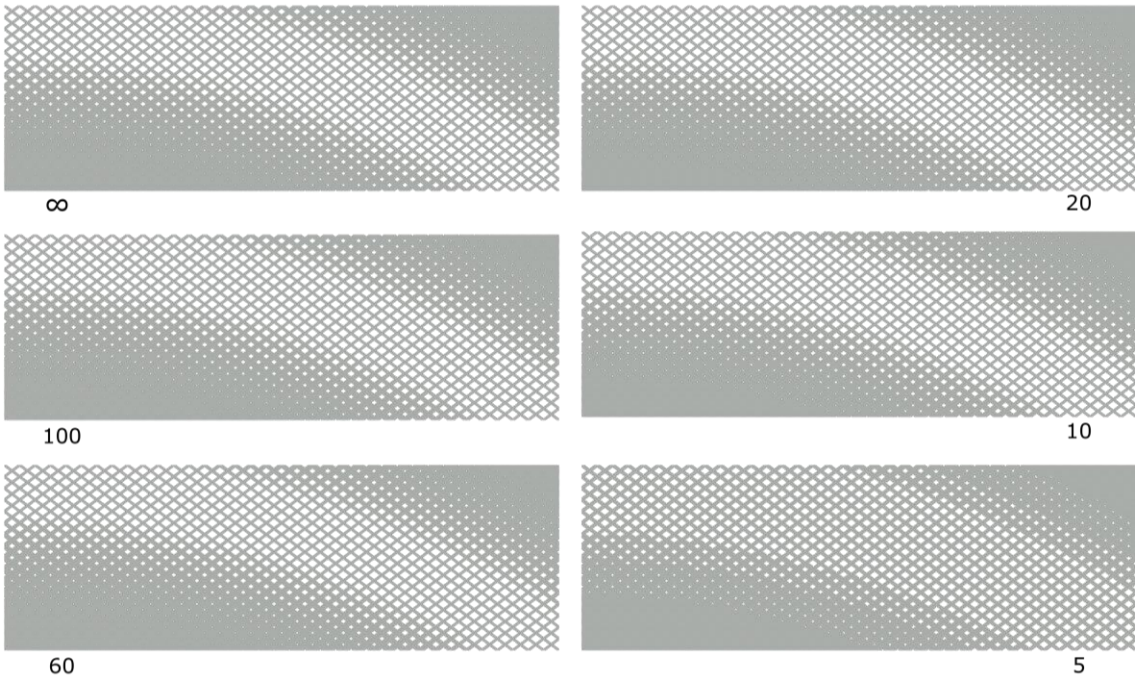

Figure 4: The façade models of the examples presented in Table 1.

Figure 5 is a conceptual representation of the design stages considered by the DrAFT framework. The *Design Exploration* stage starts with the initial idea, covering all subsequent processes until a design solution is achieved, which corresponds to the *Design Development* stage. The latter, therefore, embraces the stages explained in the previous sections: 1) the geometric exploration of the design solution, i.e. the *Surface Grid*, *Surface Mapping* and *Façade Cells* development stages; 2) the optimization of the solution regarding a certain fitness criterion, which encompasses the *Design Optimization* stage; and 3) the rationalization of the obtained solution for its subsequent fabrication, the *Design Rationalization* stage. All these five stages are considered in the DrAFT framework, which includes a set of different algorithms suitable for the corresponding design task. The combination of these algorithms therefore allows a (1) greater design flexibility, (2) the visualization of different design scenarios, and (3) the use of more specific and suitable design strategies. Moreover, this algorithmic combination constitutes a single program that is responsible for generating all the design exploration levels, discarding the need of having to create more than one program/model to address each design stage and, therefore, constituting a continuous design workflow.

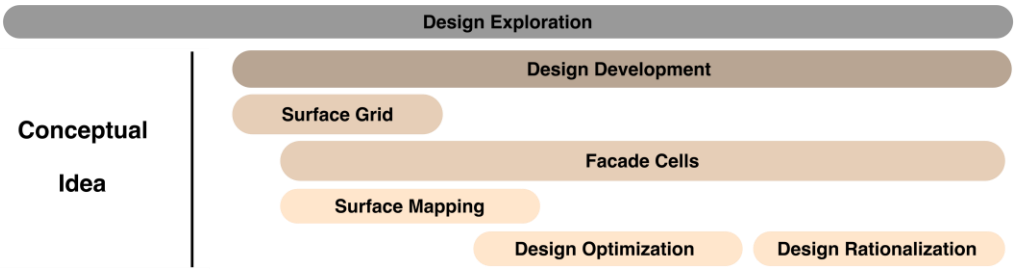

Figure 5: A temporal diagram representing the design stages embraced by the DrAFT framework.

CONCLUSION AND FUTURE WORK

Currently, we are witnessing an increased concern regarding environmental problems, in which architects play an important role. Architects are more aware of the negative impacts architectural practice has in the environment, and they are now focused on reverting this situation. A design paradigm in which design decisions are based on performance is emerging, named Performance-Based Design. This new design approach is based on the use of computational design methods, which support architects in the search for better performing solutions. Algorithmic approaches, analysis studies, and optimization processes are

catalyst for this design trend, but, in turn, require specialized knowledge and specific tools.

Unfortunately, architects are still facing several limitations during this design process: (1) the use of algorithmic approaches requires the mastering of programming languages, a skill for which most of the architects were not trained; (2) the transition between design stages usually means switching between digital tools, which is not only error-prone, but it also forces the production of different models to fit the requirements of different tools; and (3) the current variety of optimization algorithms requires architects not only to master a wide range of tools, but also to know which one is better for a certain design scenario, and also to learn how to correctly apply and interpret each of them.

To support architects in getting better results in viable time, a workflow based on an algorithmic approach must be designed to merge the main design stages – design exploration, analysis, and optimization – into a continuous and flexible process that, not only solves the problem of using multiple tools and digital environments, but also supports the design decision along the design process, actively informing the architect in the selection and application of the most suitable algorithmic patterns and strategies to a specific design scenario. The DrAFT framework presented in this paper was designed precisely for that purpose.

ACKNOWLEDGMENTS

This work was supported by national funds through *Fundação para a Ciência e a Tecnologia* (FCT) with reference UID/CEC/50021/2013, and by the PhD grant under contract of University of Lisbon (UL), Instituto Superior Técnico (IST) and the research unit *Investigação e Inovação em Engenharia Civil para a Sustentabilidade* (CERIS).

REFERENCES

Alexander, C. et al., 1977. *A Pattern Language: Towns, Buildings, Construction*, New York: Oxford University Press.

Alfaris, A. & Merello, R., 2008. The Generative Multi-Performance Design System. In *ACADIA 08 › Silicon + Skin › Biological Processes and Computation*. pp. 448–457. Available at: http://cumincad.scix.net/cgi-bin/works/Show?acadia08_448.

Andrade, D., Harada, M. & Shimada, K., 2017. Framework for automatic generation of facades on free-form surfaces. *Frontiers of Architectural Research*. Available at: http://dx.doi.org/10.1016/j.foar.2017.04.003.

Caetano, I. & Leitão, A., 2016. Exploring Buildings' Surface Patterns. In *Architecture In-Play International Conference*. Lisbon, Portugal.

Caetano, I., Santos, L. & Leitão, A., 2015. From Idea to Shape , From Algorithm to Design: Algorithmic-based Processes in Architecture. In *The next city - New technologies and the future of the built environment [16th International Conference CAAD Futures 2015*. São Paulo, Brazil, p. 483.

Chien, S., Su, H. & Huang, Y., 2015. PARADE: A pattern-based knowledge repository for parametric designs. In *Emerging Experience in Past, Present and Future of Digital Architecture, Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia*.

Davis, D., Burry, J. & Burry, M., 2011. Understanding visual scripts: Improving collaboration through modular programming. *International Journal of Architectural Computing*, 9(4), pp.361–376. Available at: http://multi-science.metapress.com/content/0063717qu0x17w36/.

Eigensatz, M., Deuss, M., et al., 2010. Case Studies in Cost-Optimized Paneling of Architectural Freeform Surfaces. In C. Ceccato et al., eds. *Advances in Architectural Geometry 2010*. Springer, pp. 47–72.

Eigensatz, M., Kilian, M., et al., 2010. Paneling Architectural Freeform Surfaces. *ACM Transactions on Graphics*, 29(4), pp.1–10.

Flöry, S. & Pottmann, H., 2010. Ruled Surfaces for Rationalization and Design in Architecture. In *ACADIA 2010: Life in:formation*. pp. 103–109.

Fu, C. & Cohen-or, D., 2010. K-set Tilable Surfaces. *ACM Transactions on Graphics*, 29(4), pp.1–6.

Gibson, I., Rosen, D. & Stucker, B., 2010. Development of Additive Manufacturing Technology. In *Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*. Springer, pp. 19–43.

Hudson, R., 2010. *Strategies for parametric design in architecture: An application of practice led research.* University of Bath.

Janssen, P., 2014. Visual Dataflow Modelling: Some Thoughts on Complexity. In E. M. Thompson, ed. *Fusion - Proceedings of the 32nd eCAADe Conference - Volume 2, Department of Architecture and Built Environment, Faculty of Engineering and Environment, Newcastle upon Tyne*. England, UK, pp. 305–314.

Khwaja, S. & Alshayeb, M., 2013. Towards design pattern definition language. *Software - Practice and Experience*, 43(7), pp.747–757.

Kolarevic, B. ed., 2003. *Architecture in the Digital Age: Design and Manufacturing*, New York: Spon Press.

Leitão, A., Lopes, J. & Santos, L., 2014. Illustrated Programming. In *Acadia 2014: Design Agency*. pp. 291–300.

Leitão, A., Santos, L. & Lopes, J., 2012a. Programming Languages For Generative Design: A Comparative Study. *International Journal of Architectural Computing*, 10(1), pp.139–162.

Leitão, A., Santos, L. & Lopes, J., 2012b. Programming Languages For Generative Design: Visual or Textual? In *RESPECTING FRAGILE PLACES [29th eCAADe Conference Proceedings], University of Ljubljana, Faculty of Architecture (Slovenia)*. Ljubljana, pp. 139–162.

Mesnil, R. et al., 2015. Isogonal moulding surfaces: A family of shapes for high node congruence in free-form structures. *Automation in Construction*, 59, pp.38–47. Available at: http://dx.doi.org/10.1016/j.autcon.2015.07.009.

Moneo, R., 2001. The Thing Called Architecture. In C. Davidson, ed. *Anything*. New York: Anyone Corporation, pp. 120–123.

Qian, Z.C., 2009. *Design Patterns: Augmenting Design Practice in Parametric CAD Systems*. SIMON FRASER UNIVERSITY.

Qian, Z.C., Chen, Y. V & Woodbury, R., 2008. Developing a simple repository to support authoring learning objects. *International Journal of Advanced Media and Communication*, 2(2), pp.154–173. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-44649122063&partnerID=40&md5=cd28a7b9ee48947891b4cf542ab39261.

Shapiro, A., 2003. Monte Carlo Sampling Methods. In *Handbooks in Operations Research and Management Science*. Elsevier, pp. 353–425. Available at: https://doi.org/10.1016/s0927-0507(03)10006-0.

Son, S. et al., 2017. Mathematical Algorithms of Patterns for Free-Form Panels. In *Proceedings of the 2nd World Congress on Civil, Structural, and Environmental Engineering (CSEE'17)*. pp. 1–8.

Su, H. & Chien, S., 2016. Revealing patterns: Using parametric design patterns in building façade design workflow. In *Living Systems and Micro-Utopias: Towards Continuous Designing, Proceedings of the 21st International Conference on Computer-Aided Architectural Design Research in Asia*. pp. 167–176.

Terzidis, K., 2004. Algorithmic Design : A Paradigm Shift in Architecture ? In *Architecture in the Network Society [22nd eCAADe Conference Proceedings / ISBN 0-9541183-2-4] Copenhagen (Denmark) 15-18 September 2004, pp. 201-207*. Warsaw, pp. 201–207.

Woodbury, R., 2010. *Elements of Parametric Design*, New York: Routledge.

Woodbury, R., Aish, R. & Kilian, A., 2007. Some Patterns for Parametric Modeling. *27th Annual Conference of the Association for Computer Aided Design in Architecture*, pp.222–229. Available at: http://moodle.ncku.edu.tw/file.php/2687/assignments/ParametricPatterns.pdf.

Yu, R. & Gero, J.S., 2015. An empirical foundation for design patterns in parametric design. In Y. Ikeda; et al., eds. *Proceedings of the 20th International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA 2015*. pp. 551–560. Available at: http://mason.gmu.edu/~jgero/publications/Progress/15YuGeroCAADRIA15.pdf.

Zboinska, M.A., 2015. Hybrid CAD/E platform supporting exploratory architectural design. *CAD Computer Aided Design*, 59, pp.64–84. Available at: http://dx.doi.org/10.1016/j.cad.2014.08.029.

View publication stats