DrAFT: an Algorithmic Framework for Facade Design

Inês Caetano[1], António Leitão[2]
[1,2]INESC-ID/Instituto Superior Técnico
[1]ines.caetano@tecnico.ulisboa.pt
[2]antonio.menezes.leitao@ist.utl.pt

Architecture has always followed the times and their innovations and, currently, an architecture based on digital technologies has been emerging and has increasingly explored architectural facades. In this paper we use DrAFT, a computational framework for the generation and exploration of facade designs, to explore a set of different examples of building skins. DrAFT includes a classification of facades that helps in the identification of algorithms that best suits each design intent. After combining the algorithms provided by this framework, the designer can more easily explore the solution space of the intended design.

Keywords: *Generative design, facade design, DrAFT framework, Rosetta*

INTRODUCTION

Nowadays, the architectural facade is characterized by complex shapes and patterns, mainly, due to the use of new design tools (Pell 2010) which promote further design exploration. The development of Generative Design tools, particularly, the use of algorithmic approaches, have had an important role in the generation of these contemporary skins because they simplified the design of complex and intricate architectural surfaces, which would not be viable to produce manually. In addition, they also increase the design efficiency and their evolution has been changing, not only the design process, but also the architectural thinking (Kolarevic 2003).

Unfortunately, algorithmic approaches do not make facade design trivial. On the contrary, they require the rigorous specification of all algorithmic steps, a task that requires specialized knowledge and that, in many cases, can be quite complex.

In this paper, we propose a computational framework designed to simplify the algorithmic specification of facade designs. In practical terms, DrAFT - Draft Algorithmic Facade Tool - promotes the exploration of facade designs and simplifies the adaptation of the generated models to the ever-changing design process conditions. Here, we present a collection of examples developed using this framework, thereby demonstrating its usability and flexibility in facade design, and also showing other possible applications.

ALGORITHMIC APPROACHES TO DESIGN

Generative design (GD) is a computer-based approach to design that creates shapes through algorithms (Terzidis 2003). Algorithmic design is a process that explores complex forms from simple and iterative methods/rules while preserving specified qualities (Meredith 2008). For this, architects produce an intermediate algorithmic-based description of a design rather than its shape (Leitão 2013).

Parametric Design is a specific GD approach that

generates different instances of a design, where each instance represents a particular set of values for the design parameters (Barrios 2005), allowing the designer to freely explore a large solution space of the design briefing/program. Therefore, this allows architects to continuously evaluate several solutions, which would be difficult to do with traditional design methods.

In spite of their advantages, algorithmic-based design methods require a disciplined approach which, in many cases, is difficult to follow. It is important, then, to develop strategies that help designers implement these methods. In this paper, we address this problem and we contribute to the state-of-the-art by proposing a strategy for the development of algorithmic-based solutions for the generation of facades.

DRAFT FRAMEWORK

DrAFT is a computational framework created to help designers in the algorithmic description of facade designs, during the design exploration stage. It is based on a classification of facades composed by different categorical dimensions that we considered computationally relevant (Caetano et al. 2015). In practical terms, the designer combines the main characteristics of the idealized design with the categorical dimensions which, in turn, guide him in the selection of the most appropriate algorithms. It is noteworthy that this guiding process is not intended to replace the role of the designer, but to significantly reduce his programming effort and, therefore, improve his design workflow.

Thereafter, the selected algorithms are combined using functional operators, also known as higher-order functions (HOF), i.e. functions that receive other functions as arguments and/or compute other functions as results (Leitão 2014). The combination of the algorithms produces the corresponding facade design model which can be quickly modified as many times as needed, allowing the designer to more easily explore the solution space of his design. Therefore, this allows designers to adapt their designs to the ever-changing design process conditions and this process can be repeated as many times as needed, promoting continuous improvement in the design exploration.

Framework Structure

Our framework takes into account several stages that typically occur in facade design, thereby dealing with the different characteristics of the:

1. Facade's surface, including its shape;
2. Design units, that together create the whole facade pattern;
3. Distribution of the units;
4. Articulation between the previous parts, i.e. surface and units.

For each one, there is at least one categorical dimension in charge of producing the matching facade characteristic, which corresponds to a set of related computational functions.

Framework Goal

Note that the goal of our framework is not to provide functions and algorithms to cover an entire range of facade designs, and neither to limit the facades that can be produced. It is rather to reduce the programming effort of the architects at the early stages of design, while speeding up the development of facades using an algorithmic approach. DrAFT allows the reuse of algorithms that are already developed and that are typically needed in the exploration of new designs of facades. This means that, not only can architects generate new designs just by using the selected algorithms, they can also combine these algorithms with some additional scripting when needed. This often happens when the idealized design is highly detailed and personalized, requiring a more specific algorithmic description. In these cases, the algorithms developed can then be incorporated in the framework, thereby further improving the matching process of subsequent facade designs.

CASE STUDIES COMPILATION

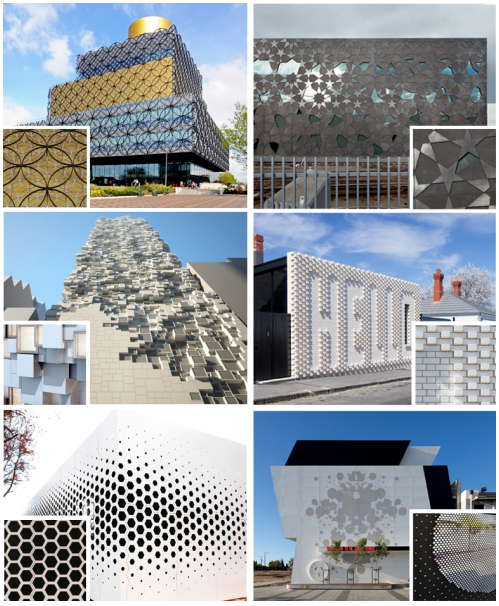


Figure 1
The selected case studies. First row: Library of Birmingham, in Birmingham, and the Yardmasters Building, in Melbourne; Second Row: Sheung Wan Hotel, in Hong Kong, and Hello House, in Melbourne: Last row: Formstelle Building, in Töging am Inn, and Precinct Energy Project, in Melbourne.

Similarly to Shape Grammars, the framework here presented can be considered original or analytical (Pupo et al. 2007). In the original case, the goal is to generate a completely new facade, while in the analytical case we look for an algorithmic interpretation of an already existing facade. In this section, we analyse and generate a set of existing facades to demonstrate the analytical capabilities of the framework. Initially, the selected facades were analysed and classified using the DrAFT classification. Figure 1 synthetises the selected facades.

The next stage was combining the functions provided by the classifications. As a result, this composition of functions generates the matching facade model of each project, which we will develop bellow. In some cases, it was possible to reproduce the facade models simply using the set of algorithms available. However, more specific designs required the development of additional algorithms to complement the provided ones.

Library of Birmingham

We will start with the Library of Birmingham by Mecanoo, visible in Figure 2. This building has a straight facade composed by several overlapped rings of two different sizes and colours, black and gold, which results in a unique pattern. First, we analysed this facade design and, then, we obtained the most appropriate algorithms through DrAFT classification. In practical terms, we selected a set of algorithms that:

1. Produced the flower shaped units (Figure 2-A);
2. Distributed the units in a regular-grid (Figure 2-B);
3. Created two layers of units with different sizes (Figure 2-C);
4. Overlapped both layers, thus creating the final pattern (Figure 2-D);

Lastly, we combined these algorithms using different functional operators and HOFs. Figure 2 shows an instance of the obtained model.

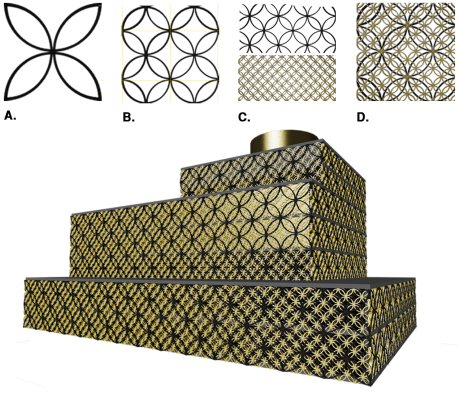


Figure 2
Library of Birmingham. The set of algorithms used to produce the model: A. Units shape; B. Units Distribution; C. Two layers of different sizes and colours; D. Layered facade articulation.

Sheung Wan Hotel

Sheung Wan Hotel (Figure 3) by Thomas Heatherwick is our next example. As the previous example, we also started with a design analysis of this facade to then classify it.

This facade is straight and it is composed by sev-

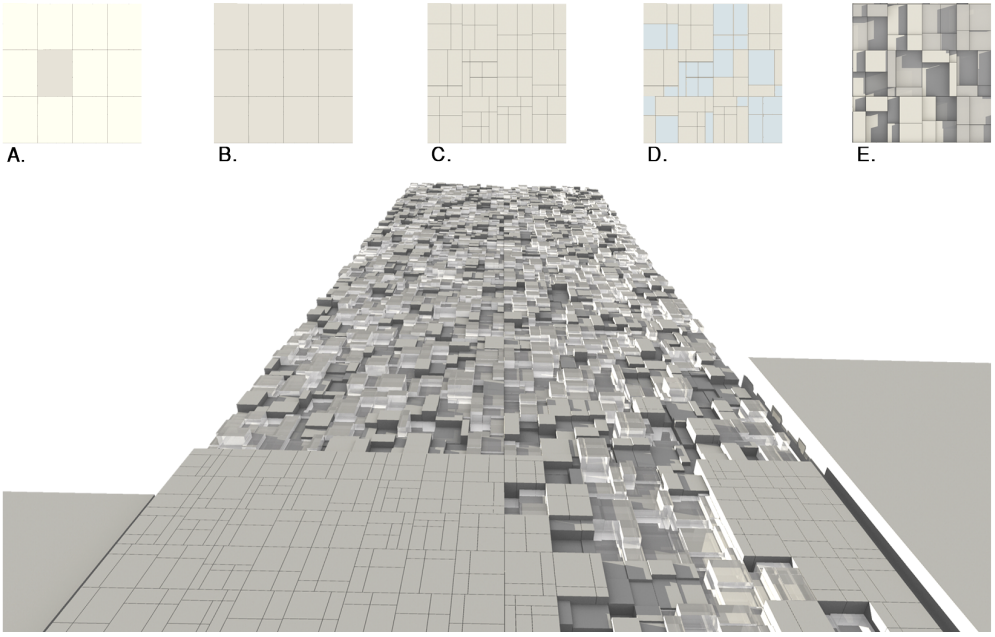


Figure 3
Sheung Wan Hotel model. A. Unit shape; B. Unit distribution in a regular-grid; C. Unit subdivision into smaller rectangular and squared units; D. Random application of materials, metal or glass; E. Random depth size.

eral rectangular units (Figure 3-A). These units are then mapped into a regular-grid (Figure 3-B) and they are further subdivided into smaller units with a shape that randomly varies between a squared and rectangular geometry (Figure 3-C). Lastly, the units depth and assigned material vary randomly from a certain height value of the building's facade (Figure 3-D and E). Figure 3 synthetizes the combination process of the algorithms and shows an instance of the generated model.

Hello House

The following example is the Hello House facade by OFF! Architecture, visible in Figure 4, which is composed by several stacked white bricks placed in two different positions: along the facade's surface or perpendicular to it.

Firstly, we considered the set of two bricks as the pattern unit, i.e. an horizontal and a perpendicular brick (Figure 4-A). Secondly, we distributed them into an alternated grid. Finally, to create the HELLO effect of the facade we used an image with this word to control the placing of the perpendicular bricks. If the bricks were coincident with the area of the word HELLO, they were aligned with the facade axis. Otherwise, they were placed so as to protrude.

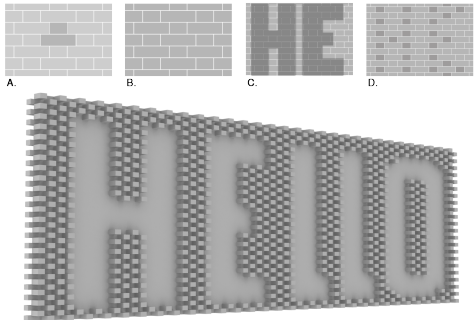


Figure 4
Hello House model produced using DrAFT framework. A. Pattern unit; B. Unit distribution; C. Picture to control the bricks positioning; D. Final pictorial effect.

Precinct Energy Project

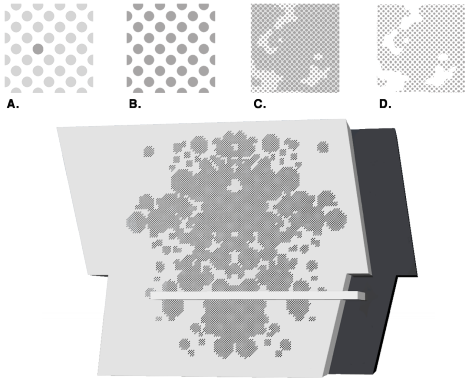


Figure 5
Precinct Energy Project model. A. Definition of the unit; B. Unit distribution in an alternated-grid. C. Pictorial effect with an image; D. Final effect, which was then subtracted from the facades' surface.

As in the previous example, Precinct Energy Project (Figure 5) by PHTR Architects has a pictorial facade that produces a design similar to an inkblot. However, in this case the inkblot effect is created by the existence or absence of perforations.

In practical terms, this example has a straight and perforated facade, wherein the perforations have a circular shape. We used cylinders to produce these holes, which were placed according to an image representing the inkblot effect that we wanted to produce. In the end, the cylinders were subtracted from the facade surface, thus creating the perforated surface visible in Figure 5.

Note that the process that controlled the placement or not of a cylinder at each position was similar to the one described in the previous example. Figure 5 synthetizes the whole generation process of this example, and also the set of algorithms that were used.

Formstelle / Campus Netzwerk

The facade of the Formstelle building, visible in Figure 6, was designed by Format Elf Architekten and is characterized by a perforated surface with hexagonal holes. These were produced using hexagonal prisms, which were then distributed along the facade in an alternated-grid, thereby creating a pattern similar to a honeycomb (Figure 6-A and B).

Also note that the size of the perforations varies continuously along the facade, thus reaching its maximum at the center and its minimum at the ends (Figure 6-C). To produce this effect, the size transformation of these prisms was controlled by an attractor, which is a point or a set of points that act like virtual magnets. In this case, we used a set of points creating a horizontal line in the middle of the facade and, then, the size of each prism was calculated according to the distance between its location and the nearest attractor point.

Finally, the hexagonal prisms were subtracted from the facade surfaces in order to produce the perforated surface effect (Figure 6-D). Figure 6 shows an instance of the obtained model.

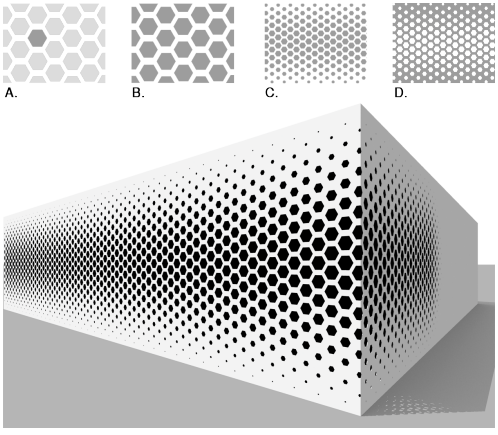


Figure 6
Formstelle building. A. Pattern unit; B. Unit distribution in an alternated-grid; C. Units size transformation; D. Facade with a perforated articulation.

Yardmasters Building

Our last example, the Yardmasters Building by McBride Charles Ryan, visible in Figure 7, was also generated using the algorithms provided by its classification. To generate this example we had to produce a facade with a regular shape, a layer with the Islamic pattern and also some window openings with an irregular shape. In practical terms, the pattern was produced by the repetition of a unit (Figure 7-A) distributed in an alternated-grid (Figure 7-B). The obtained pattern was then used to shape the windows

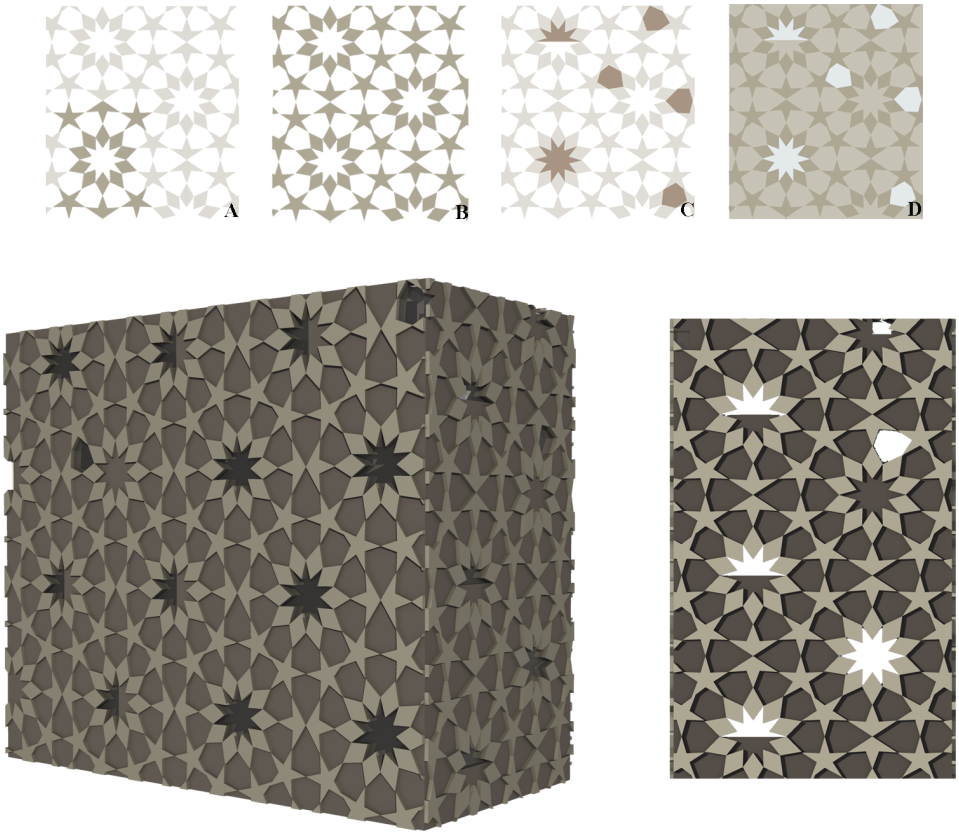


Figure 7
The Yardmasters Building model. A. Units geometry; B. Units distribution in an alternated-grid; C. Creation of the windows; D. Applied articulation, with the Islamic pattern on the facade's surface; Below: the Yardmasters' final model.

openings (Figure 7- C/D) and, finally, it was applied on the facade's surface. Nevertheless, we had to develop some additional scripting in order to produce the Islamic units, as they have a more specific shape.

Figure 7 summarizes the generation process of the Yardmasters building, including the definition of the surfaces and the pattern units.

In this example, although we had to develop the algorithms to describe the Islamic pattern geometry, all the other design parts were produced using the predefined functions. As a result, we can conclude that the DrAFT framework helps architects in the generation of a large variety of facades by reducing both the programming effort and the time spent.

OTHER APPLICATIONS

In this section, we present other possible applications of the DrAFT framework. We start by developing an original facade.

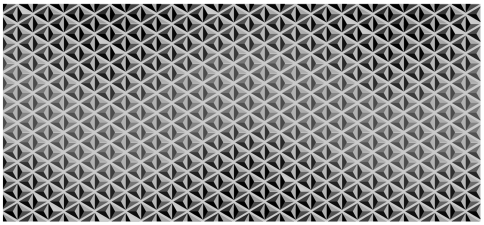
As an example, we will consider that we want a straight facade with pyramidal elements. To this end, we select the algorithms that respectively generate a (1) straight surface and (2) pyramidal elements. Let us also assume that we want the height of the pyra-

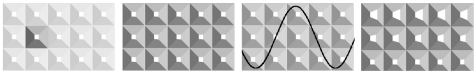
mids to vary according to the distance to a curve. Our framework also provides a function to produce this type of size variation. Therefore, the algorithms selected so far allow us to define the function that creates the units.

We set that the units distribution is done in an alternated-grid. In practical terms, the function to distribute them is a higher-order function, which receives other functions as arguments:

1. The function that creates each unit, as it knows how the distribution is done but not the elements to distribute.
2. The function that describes the facade surface as it requires the set of points on which the distribution will be done.

Figure 8
An instance of the pattern produced using the set of algorithms above.





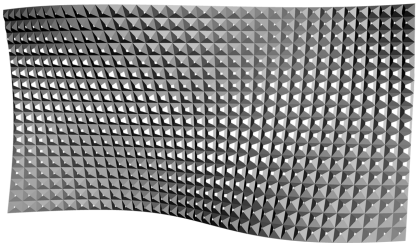


Figure 9
An instance of the pattern produce using the algorithms above, which now include units with a different shape (pyramidal) and a distribution in an regular-grid.

Lastly, we will define that the elements are applied on the facade's surface and the colour used is gray. The result is visible in Figure 8.

Now, we can simply vary this facade design by changing some of the input values, thereby producing different instances of the same design, or by altering some of the functions used, thus changing some design characteristics. Therefore, to modify the type of distribution and geometry of both units and facade (see Figure 9), we do not need to change the rest of the structure, i.e. the functions in charge of producing the other facade parts.

To complete the definition of the facade design, we can also optimize the generated models. So, imagining that we want to maximize the light passing through the previous example (Figure 9), we define as parameters:

1. the smoothness of the attractor-curve (in a range from 0.5 to 6.0);
2. the minimum pyramids openings (between 0.1 and 1.0);
3. the height of the pyramids (with a maximum of 0.5m).

A simple but effective optimization algorithm can then be implemented just by repeatedly sampling the parameter space, generating the corresponding facade, and computing the amount of light that it lets through, saving the values of the parameters that maximize that amount of light.

For better control of the optimization process we can fix some of the parameters, e.g., the smoothness, and optimize for the remaining parameter space. This allows us to obtain a model that combines a design that pleases us with values for the remaining parameters that are close to the optimal (see Figure 10).

PORTABILITY

The current implementation of the framework was done using the Rosetta IDE (Lopes and Leitão 2011). This has the significant advantage of making the framework portable across the different CAD and BIM tools supported by Rosetta, allowing us to produce identical models in Rhino, AutoCAD, SketchUp, Revit, and ArchiCAD. This means that the DrAFT framework is not restricted to a single CAD tool, as it happens with other similar frameworks, thus liberating the de-

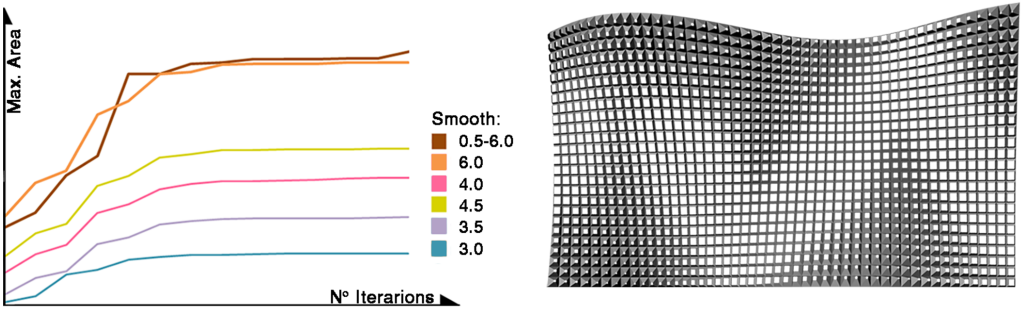


Figure 10
On the left: a graphic with the areas maximized for each smoothness value; On the right: the selected final model.

signer from the limitations of any specific software. Moreover, it allows the designer to easily change the CAD tool that he wants to use (Figure 11).

Additionally, Rosetta also promotes portability across the supported programming languages, allowing the exploration of the framework in different programming languages such as Autolisp, Phyton, Processing and Javascript. As a result, in order to use our framework, designers can choose the programming language that they are more familiarized with, without forcing them to learn a new language.

RELATED WORK

Some authors have already developed some work inspired by the wide variety of contemporary facades. Pell (2010), Moussavi (2006) and Velasco et al. (2015) tried to organize this variety of designs and each one created a classification of facades based on different concepts. Nevertheless, none of the previous classifications helps architects with the algorithmic description of new facade designs.

Su and Chien (2016) recognized the existence of some algorithmic patterns in facade designs. These similar algorithmic structures can be reused later to generate further designs, one of the basic concepts of our framework.

On the other hand, tools like the Paneling Tools plug-in for Rhino and Grasshopper, the Lunch Box add-on to Grasshopper, and ParaCloud Gem, a standalone toolkit that adds generative capabilities to any CAD system that supports **.obj**, **.stl**, **.collada**, and **.dxf** file formats, already attempt to solve the problems here described. All of them are capable of creating grids of points on a surface, mapping elements in different ways, applying attractors to control elements size, etc.

However, when these tools are manually used in an iterative user-driven process, they can be tiresome and error-prone. In addition, when they are used in an Application Programming Interface (API) or as plug-ins to a domain-specific programming language, such as Grasshopper, a certain level of automation is obtained, however, the designer is always bound to the specific functionalities provided by the tool, thus limiting its agency in exploring different combinations of operations and extending the capabilities of the tool's pre-defined operations. Besides that, these tools are more used for generic panelization, subdivision, and population of surfaces thus, although they have been used to generate complex facade patterns, they are not fully architectural-oriented which means that they do not directly address relevant concepts in facade design such as materiality or the tectonic relation between the facade elements.

Dynamo for Revit and Grasshopper for Rhino also allow users to implement the functionalities proposed in this paper. However, the freedom allowed by these tools becomes difficult to manage in complex facades (Leitão et al. 2012). In these cases, a more structured and systematic approach like the

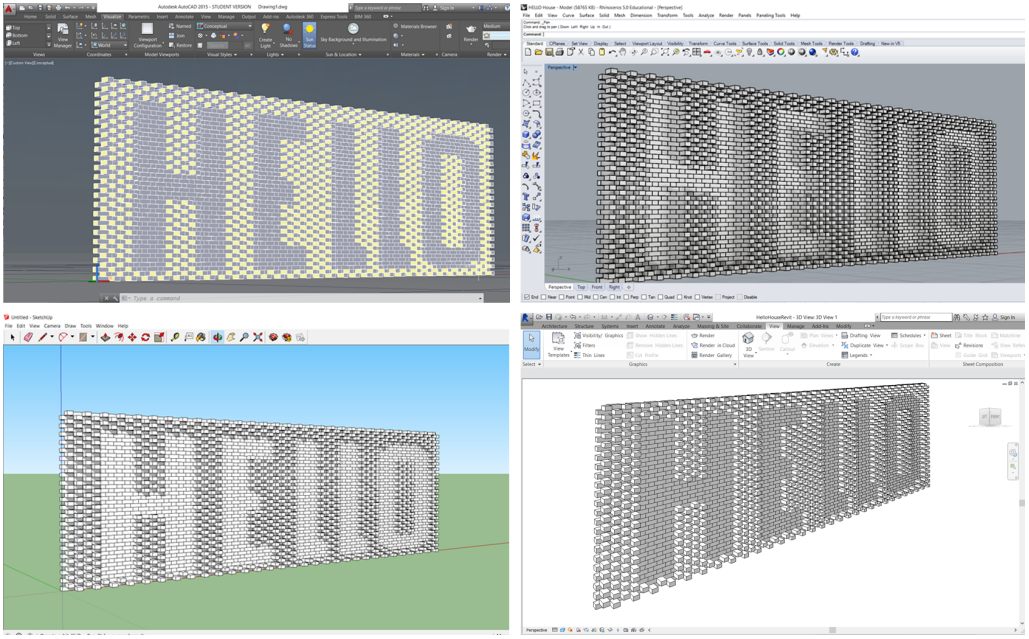


Figure 11
Re-execution of the same model in different backends (AutoCAD, Rhino5, SketchUP and Revit).

one we propose is more manageable.

In summary, with these tools the architect is limited by their non-domain specificity or, in order to extend their capabilities, he needs to build from scratch the necessary functionalities or use a mix of different tools that most of the time are not compatible. Our work extends the state-of-the-art by systematizing and structuring, in an architectural-oriented framework, the parametric generation of a wide range of facade typologies, and by operationalizing it resorting to a simple algorithmic approach that uses and combines different functions that directly implement facade design concepts.

CONCLUSION

The exploration of architectural facades is not new. However, by resorting to recent digital technologies, architects can once again focus on facade design, promoting the exploration of complex patterns and geometries. In this paper we showed how the DrAFT framework can help designers generate different facade designs. The current implementation was done using the Rosetta IDE (Lopes and Leitão 2011), allowing the exploration of facade designs in different programming languages and the generation of the corresponding models in different CAD and BIM applications.

The framework uses a classification of facades that guides the selection of the appropriate algorithms for each type of facade design. The algorithms might then be used directly, or might be combined using functional operators, promoting a systematic exploration of designs which ultimately aims to a higher productivity by improving the time spent in scripting tasks, and adding flexibility to the designers' workflow. Due to the simplicity of the functional composition, this framework accommodates the ever-changing nature of a design process by fa-

cilitating the test of several designs, or instantiations of the same idea, in any design stage. In this paper we also demonstrated the framework's flexibility by exploring a set of existing facades.

In the near future, we plan to expand DrAFT, covering a wider range of facades. To make this framework more usable, we are particularly interested in conducting a field study of its application, to identify weaknesses of the proposed processes and opportunities for extensions.

ACKNOWLEDGEMENTS

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

REFERENCES

Barrios, C 2005 'Transformations on Parametric Design Models: A Case Study on the Sagrada Familia Columns', *Proceedings of CAAD Futures 2005*, Vienna University of Technology, Austria

Caetano, I, Santos, L and Leitão, A 2015 'From Idea to Shape, from Algorithm to Design: A Framework for the Generation of Contemporary Facades', *The Next City - 16th CAADFutures, CCIS 527, Springer*, São Paulo, Brazil

Kolarevic, B (eds) 2003, *Architecture in the Digital Age - Designing and Manufacturing*, Spon Press, London, U.K

Leitão, A 2013 'Teaching Computer Science for Architecture', *Future Traditions - Proceedings of the 1st ecaade Regional*, FAUP Porto, Portugal

Leitão, A 2014 'Improving Generative Design by Combining Abstract Geometry and Higher-Order Programming', *Proceedings of the 19th CAADRIA.*, Kyoto, Japan

Leitão, A, Santos, L and Lopes, J 2012, 'Programming Languages for Generative Design: A Comparative Study', *International Journal of Architectural Computing*, 10(1), pp. 139-162

Lopes, J and Leitão, A 2011 'Portable Generative Design for CAD Applications', *ACADIA 11: Integration Through Computation - Proceedings of the 31st ACADIA*, Calgary, Canada

Meredith, M 2008, 'Intro', in Sakamoto, T and Ferré, A (eds) 2008, *From Control to Design: Parametric/Algorithmic Architecture*, Actar

Moussavi, F 2006, *The Function of Ornament*, Actar

Pell, B 2010, *The Articulate Surface - Ornament and technology in Contemporary Architecture*, Birkhauser

Pupo, R, Pinheiro, E, Mendes, G, Kowaltowski, D and Celani, G 2007 'A Design Teaching Method Using Shape Grammars', *Proceedings of the 7th International Conference Graphics Engineering for Arts and Design*, Curitiba, Brasil

Su, H and Chien, S 2016 'Revealing Patterns: using Parametric Design Patterns in Building Facade Design workflow', *Proceedings of the 21st CAADRIA*, Melbourne, Australia

Terzidis, K 2003, *Expressive Form: A Conceptual APproach to Computational Design*, Spon Press, New York

Velasco, R, Brakke, A and Chavarro, D 2015 'Dynamic Façades and Computation: Towards an Inclusive Categorization of High Performance Kinetic Façade Systems', *CAADFutures 2015, CCIS 527, Springer*, São Paulo, Brazil