

# **Generative Design: a new stage in the design process**

**Rita Margarida Serra Fernandes**

Thesis to obtain the Master of Science Degree in

**Architecture**

## **Examination Committee**

Chairperson: Prof. Dr. Ana Cristina dos Santos Tostões

Supervisor: Prof. Dr. António Paulo Teles de Menezes Correia Leitão

Member of the Committee: Prof. Dr. Francisco Manuel Caldeira Pinto Teixeira Bastos

**June 2013**



## ABSTRACT

Design processes are characterized by change. Unfortunately, CAD tools are currently being used just as a more efficient version of the traditional paper-based approach, an approach that does not help the designers at handling change, particularly for the exploration of different solutions or to adapt the design to evolving requirements.

Recently, new approaches have been introduced in the design process, which are better tailored for handling change. Generative Design is one of them and can be defined as the creation of shapes determined by algorithms.

This dissertation argues for one main point: integrating Generative Design as a new stage in the design process dramatically simplifies the handling of changes. In particular, we propose an algorithmic approach to design that overcomes the limitations of the traditional approach for using CAD tools.

Our approach requires the formalization of the design intents, for which we use programming. This approach has an initial cost. However, we argue that this cost is highly rewarding when the designers need to handle change, by allowing them to explore many different alternatives, quickly and effortlessly.

To evaluate the proposed approach we developed a three-dimensional model of a case study, a large and complex building, using a purely programming-based representation.

**Keywords:** Design process; Change; Generative Design; Algorithmic approach; Programming.



## RESUMO

Projetar é um processo caracterizado por mudança. Infelizmente, as ferramentas CAD estão atualmente a ser utilizadas apenas como uma versão mais eficiente da abordagem baseada em papel, uma abordagem que não ajuda os arquitetos a lidar com a mudança, em particular na exploração de diferentes soluções de projeto ou na adaptação à evolução dos seus requisitos.

Recentemente, novas abordagens têm sido introduzidas nos processos de projeto, as quais são mais aptas a lidar com a mudança. O Desenho Generativo é uma delas e pode ser definida como a criação de formas determinada por algoritmos.

Esta dissertação defende que a integração de Desenho Generativo como uma nova fase no processo de projeto simplifica drasticamente a incorporação de mudanças. Em particular, é proposta uma abordagem algorítmica para o processo de projeto que supera as limitações da abordagem tradicional no uso das ferramentas CAD.

Esta nova abordagem requer a formalização das intenções do projeto, para a qual utilizámos programação. O uso desta abordagem tem um custo inicial. Contudo, argumenta-se que este custo é altamente recompensado quando o arquiteto necessita de lidar com a mudança, permitindo-lhe explorar muitas alternativas diferentes, de forma rápida e sem esforço.

Para avaliar a abordagem proposta foi desenvolvido um modelo tridimensional de um caso de estudo, um edifício grande e complexo, usando uma representação puramente baseada em programação.

**Palavras-chave:** Processo de projeto; Mudança; Desenho Generativo; Abordagem algorítmica; Programação.



## THESIS STATEMENT

Design processes are characterized by change from the very beginning. The change may arise from, e.g., uncertain design intents or detail's and requirements' growth. Design tools must embrace change but the traditional use of these tools requires too much time and effort to modify models or even treat them as disposable when changes are needed.

Generative Design, in particular, an algorithmic approach, is an efficient, rigorous, controllable, and flexible tool that allows, with reduced time and effort, the production of several different models to explore design variations addressing different types of change.

This dissertation argues for one main point: integrating Generative Design as a new stage in the design process dramatically simplifies the handling of changes.





## Acknowledgments

A special acknowledgment to Professor António Leitão for his supportive guidance, for his inexhaustible availability, dedication and patience, for his enthusiasm and for everything he taught me during the development of this work but also for introducing me such an interesting and useful approach to design as Generative Design.

To Luís Santos for the enthusiasm he shown and for the relevant comments and advices.

To Filipa Tomé for her availability, advices and friendship.

To Mónica and Ana for their concern, encouragement, abetment and friendship.

To Diogo for all the friendship, tolerance, patience and indispensable support.

To my friends of IST, in particular to Pedro Snow, Margarida Pires and Maria João Tato, for the encouragement, advices and friendship, for making me believe that I can, and for the great times we shared along the entire course and we continue to share.

To my parents for their understanding, concern and support during the development of this work.

## Agradecimentos

Um especial agradecimento ao Professor António Leitão pela verdadeira orientação, pela disponibilidade, dedicação e paciência inesgotáveis, pelo entusiasmo, por tudo o que me ensinou durante a realização deste trabalho mas também por me ter apresentado uma abordagem ao projecto tão interessante e útil como o Desenho Generativo.

Ao Luís Santos pelo entusiasmo demonstrado e pelos relevantes comentários e conselhos.

À Filipa Tomé pela sua disponibilidade, conselhos e amizade.

À Mónica e à Ana pela preocupação, encorajamento, cumplicidade e amizade.

Ao Diogo por toda a amizade, tolerância, paciência e imprescindível apoio.

Aos meus amigos do IST, em particular ao Pedro Snow, à Margarida Pires e à Maria João Tato, pelo encorajamento, conselhos e amizade, por me fazerem acreditar que eu consigo, e pelos óptimos momentos que partilhámos ao longo de todo o curso e que continuamos a partilhar.

Aos meus pais pela sua compreensão, preocupação e apoio durante toda a realização deste trabalho.



# CONTENTS

<b>ABSTRACT .....</b>	<b>III</b>
<b>RESUMO .....</b>	<b>V</b>
<b>THESIS STATEMENT.....</b>	<b>VII</b>
<b>CONTENTS .....</b>	<b>XI</b>
<b>LIST OF FIGURES .....</b>	<b>XV</b>
<b>LIST OF TABLES .....</b>	<b>XIX</b>
<b>ABBREVIATIONS.....</b>	<b>XXI</b>
<b>GLOSSARY OF TERMS.....</b>	<b>XXI</b>
<b>INTRODUCTION .....</b>	<b>1</b>
OBJECTIVES .....	3
METHODOLOGY .....	3
STRUCTURE .....	4
<b>PART I : BACKGROUND .....</b>	<b>7</b>
<b>1 GENERATIVE DESIGN .....</b>	<b>9</b>
1.1 Beginnings and maturity of Generative Systems in Architecture .....	9
1.2 Generative Systems .....	13
1.2.1 Algorithmic systems.....	13
1.2.2 Parametric systems .....	16
1.2.3 Shape Grammars .....	18
1.2.4 Lindenmayer Systems .....	20
1.2.5 Cellular Automata .....	22
1.2.6 Evolutionary Systems and Genetic Algorithms .....	23

<b>2 DESIGN TOOLS .....</b>	<b>25</b>
<b>3 APPROACHES TO MODELING: TRADITIONAL vs ALGORITHMIC.....</b>	<b>29</b>
3.1 Traditional approach .....	29
3.2 Algorithmic-based approach .....	30
<b>4 GENERATIVE DESIGN PRACTICE .....</b>	<b>33</b>
4.1 Strategies within architectural practice .....	33
4.2 Case study: The National Swimming Centre, the Water Cube .....	37
<b>PART II : AN ALGORITHMIC APPROACH TO DESIGN .....</b>	<b>35</b>
<b>5 INTRODUCTION.....</b>	<b>37</b>
<b>6 MODELING STRATEGY .....</b>	<b>39</b>
6.1 Modeling strategy .....	39
6.2 Application to the case study .....	40
<b>7 DEFINITION OF THE MODEL .....</b>	<b>47</b>
7.1 Phase 1 – Underlying Geometry.....	47
7.1.1 Design surfaces.....	47
7.1.2 Solids.....	48
7.2 Phase 2 – Main Elements.....	49
7.2.1 Vertical walls and glass facades.....	49
7.2.1.1 Vertical walls.....	50
7.2.1.2 Glass facades.....	51
7.2.2 Inner bent wall .....	51
7.2.3 Slabs (of the middle floors, cover and ground floor) and fencing panels .....	51
7.2.3.1 Middle floor slabs .....	52
7.2.3.2 Cover Slab .....	52
7.2.3.3 Ground floor slab .....	53
7.2.3.4 Fencing panels.....	54
7.3 Phase 3 – Openings and frames .....	55

7.3.1 Frames of the lateral facades .....	55
7.3.2 Windows of the front and back facades .....	60
7.3.3 Windows of the inner bent wall .....	62
7.4 Phase 4 – Detail Elements .....	64
7.4.1 Glass facades - Suspended cable-net facades and revolving doors .....	65
7.4.2 Fencing posts .....	67
<b>8 ANALYSIS OF THE MODELING PROCESS .....</b>	<b>71</b>
8.1 General considerations .....	71
8.1.1 Decomposition .....	71
8.1.2 Legibility .....	71
8.1.3 Goal of the model .....	72
8.2 Parameters .....	73
8.2.1 Choice and number .....	73
8.2.2 Testing the model .....	74
8.3 Reuse of procedures .....	75
8.3.1 Decomposition .....	75
8.3.2 Abstraction .....	75
8.4 Methodologies .....	75
8.4.1 Boolean operations .....	75
8.4.2 Interpolations .....	76
8.5 Control .....	76
<b>9 EVALUATION .....</b>	<b>79</b>
9.1 Evaluation of the ability of the algorithmic approach to respond to the evolving requirements: comparison with the traditional approach .....	80
9.2 Conclusions .....	90
<b>CONCLUSIONS AND FUTURE WORK .....</b>	<b>93</b>
CONCLUSIONS .....	93
FUTURE WORK .....	95

CONTRIBUTIONS ..... 96

**BIBLIOGRAPHY ..... 97**

## LIST OF FIGURES

<b>Figure 1.1</b> Skeletons with construction lines as starting points for plan compositions, from Durand's <i>Précis</i> (source: Mitchell, 2001). .....	10
<b>Figure 1.2</b> Hierarchic process of top-down substitution, from Durand's <i>Précis</i> (source: Mitchell, 2001) .....	11
<b>Figure 1.3</b> Example of Durand's building vocabulary (source: Celani, 2002). .....	11
<b>Figure 1.4</b> Module of the Dom-Ino (source: Frampton, 2003). .....	12
<b>Figure 1.5</b> Top: The Vila Malcontenta as drawn by Palladio. Down: The final plan as generated from the shape grammar (source: Stiny et al. 1978).....	12
<b>Figure 1.6</b> The 2002 Serpentine Gallery (source: <a href="http://www.serpentinegallery.org/architecture/">http://www.serpentinegallery.org/architecture/</a> ).....	16
<b>Figure 1.7</b> Schematic representation of the application of the algorithm (source: Deuling, 2001).....	16
<b>Figure 1.8</b> International Terminal at Waterloo Station (source: <a href="http://grimshaw-architects.com/">http://grimshaw-architects.com/</a> ).....	18
<b>Figure 1.9</b> Left: Dimensionally different but identically configured arches. Right: Schemas with the parametric definition of the scaling factor for the truss geometry (source: Kolarevik, 2003).....	18
<b>Figure 2.1</b> Ivan Sutherland's Sketchpad system (source: Mark et al., 2008). .....	25
<b>Figure 4.1</b> Detail of the Smithsonian Courtyard Enclosure designed by FP with SMG as consultant (source: <a href="http://www.fosterandpartners.com">http://www.fosterandpartners.com</a> ). .....	34
<b>Figure 4.2</b> The Smithsonian Courtyard Enclosure designed by FP with SMG as consultant (source: <a href="http://www.fosterandpartners.com">http://www.fosterandpartners.com</a> ). .....	34
<b>Figure 4.3</b> The forthcoming White Magnolia Plaza consisting of three towers and smaller scale mixed-used buildings. The design was developed by SOM with Blackbox (source: <a href="http://www.som.com">www.som.com</a> ). .....	35
<b>Figure 4.4</b> The parametric model developed by Blackbox to control the exterior surface geometry and slabs heights for the White Magnolia's main tower (source: <a href="http://www.som.com">www.som.com</a> ). .....	35
<b>Figure 4.5</b> The 2005 Serpentine Pavilion designed by Álvaro Siza Vieira and Eduardo Souto de Moura in collaboration with ARUP (source: <a href="http://www.serpentinegallery.org/architecture.html">http://www.serpentinegallery.org/architecture.html</a> ). .....	36

<b>Figure 4.6</b> Example of Burry's Work in Sagrada Família, Barcelona. Detail of the lateral nave window based on Gaudi's original 1:10 scale plaster models that were reconfigured geometrically with precision and transformed into digital models (source: <a href="http://sagradafamilia.sial.rmit.edu.au/">http://sagradafamilia.sial.rmit.edu.au/</a> ). .....	36
<b>Figure 4.7</b> Night view of the Water Cube (source: <a href="http://90.146.8.18/en/archives/festival_archive/festival_catalogs/festival_artikel.asp?iProjectID=8672">http://90.146.8.18/en/archives/festival_archive/festival_catalogs/festival_artikel.asp?iProjectID=8672</a> ). .....	37
<b>Figure 4.8</b> "Weaire-Phelan foam" (source: Carfrae, 2007). .....	37
<b>Figure 4.9</b> Schematic illustration of the geometry formation (source: Carfrae, 2007). .....	38
<b>Figure 4.10</b> The physical model used in the competition (source: Carfrae, 2007). .....	38
<b>Figure 4.11</b> The structure of the Water Cube under construction (source: Carfrae, 2007). .....	39
<b>Figure 4.12</b> Detail of the structural model (Carfrae, 2007). .....	39
<b>Figure 6.1</b> Photo-realistic images of the Market Hall (source: <a href="http://www.architizer.com">http://www.architizer.com</a> ). .....	46
<b>Figure 6.2</b> Photo-realistic image of the Market Hall (source: <a href="http://www.mvrdv.nl">http://www.mvrdv.nl</a> ). .....	47
<b>Figure 6.3</b> Phases of the design process and hierarchy of the model. ....	48
<b>Figure 6.4</b> Top: Photo-realistic image (source: <a href="http://www.mvrdv.nl">http://www.mvrdv.nl</a> ). Middle: design surfaces. Down: bent solid and void solid. ....	49
<b>Figure 6.5</b> Second phase of the model. ....	49
<b>Figure 6.6</b> Scheme with the main elements of the model. ....	49
<b>Figure 6.7</b> Photo-realistic image (source: <a href="http://www.mvrdv.nl">http://www.mvrdv.nl</a> ). .....	50
<b>Figure 6.8</b> Third phase of the model. ....	50
<b>Figure 6.9</b> The facade of the building and the fencing posts. Detail of a photo-realistic image (source: <a href="http://www.mvrdv.nl">http://www.mvrdv.nl</a> ). .....	51
<b>Figure 6.10</b> Detail of the last phase of the model. ....	51
<b>Figure 6.11</b> Last phase of the model. ....	51
<b>Figure 7.1</b> Aboveground of the building. Middle: Bent solid. Down: Void solid. ....	53
<b>Figure 7.2</b> Design surfaces: outer and inner. Top: planar points. Middle: planar curves. Down: design surfaces. ....	54
<b>Figure 7.3</b> Method to define the Bent Solid. Top: Bent Solid. Middle: Design surfaces. Down: Planar surfaces. ....	54



<b>Figure 7.4</b> Method to define the Void Solid. Top: Void Solid. Middle: Inner design surface. Down: Planar surfaces.....	55
<b>Figure 7.5</b> Main elements of the model. ....	55
<b>Figure 7.6</b> Example of the production of vertical elements (vertical walls) with the Bent Solid as input. ....	56
<b>Figure 7.7</b> Vertical Walls.....	56
<b>Figure 7.8</b> Glass facades.....	57
<b>Figure 7.9</b> Inner bent wall. ....	57
<b>Figure 7.10</b> Partial plan of the building. Parameters of the vertical walls, glass facades and inner bent wall.....	57
<b>Figure 7.12</b> Example of the production of horizontal elements (slabs) with the Bent Solid as input....	58
<b>Figure 7.13</b> Middle floor slabs. ....	58
<b>Figure 7.14</b> Cover Slab.....	59
<b>Figure 7.15</b> Cover slab: conditions to obtain a continuous slab. ....	59
<b>Figure 7.17</b> Ground floor slab.....	59
<b>Figure 7.18</b> Fencing panels. ....	60
<b>Figure 7.19</b> Parameters of the slabs and fencing panels. ....	60
<b>Figure 7.20</b> Phase 3 of the model: representation of all the windows.....	61
<b>Figure 7.21</b> Frames of the lateral facades.....	62
<b>Figure 7.22</b> Lateral frames: method to calculate the abscissas of the anchor points. ....	63
<b>Figure 7.23</b> Partial representation of the lateral frames. ....	63
<b>Figure 7.24</b> Three examples of modules. ....	64
<b>Figure 7.25</b> Application of the modules. ....	64
<b>Figure 7.28</b> Types of composed frames.....	65
<b>Figure 7.29</b> Method to find the abscissas and the parameters. ....	66
<b>Figure 7.30</b> Method to compute the abscissas of the anchor points.....	67

<b>Figure 7.31</b> Method to compute the abscissas of all the windows. ....	67
<b>Figure 7.32</b> Representation of the windows of the inner bent wall. ....	68
<b>Figure 7.33</b> Method to found one point belonging to a surface. ....	69
<b>Figure 7.34</b> Computation of the vectors. ....	69
<b>Figure 7.35</b> Top: Oriented box. Down: Frame. ....	70
<b>Figure 7.36</b> Top: box being subtracted to the inner bent wall. Down: placement of the frame. ....	70
<b>Figure 7.37</b> Phase 4 of the model: Detail of the glass facades and inclusion of the fencing posts. ....	70
<b>Figure 7.38</b> Left: Glass facade in the second phase of the model. Right: Glass facade detailed. ....	71
<b>Figure 7.39</b> Steps for detailing the glass facade. ....	72
<b>Figure 7.40</b> Logic of the grid. ....	72
<b>Figure 7.41</b> Parameters of the glass attachments. ....	73
<b>Figure 7.42</b> Parameters of the revolving doors. ....	73
<b>Figure 7.43</b> Left: Fencing glass in the second phase of the model. Right: Inclusion of the posts. ....	74
<b>Figure 7.44</b> Partial plan (one balcony): Positions of the posts by controlling their distances along the longitudinal axis of the building. ....	75
<b>Figure 7.45</b> Method to define the solids around the shape of the building. ....	75
<b>Figure 7.46</b> Methodology to shape posts with identical sections. ....	76
<b>Figure 7.47</b> Method to shape the posts for the different floors. ....	76
<b>Figure 9.1</b> MVRDV's Market Hall as generated from our program. ....	88
<b>Figure 9.2</b> Building generated by changing the shape of the building. ....	88
<b>Figure 9.3</b> Building generated by changing the shape of the building by using different cross section curves along its length. ....	88
<b>Figure 9.4</b> Left: Building generated after changing the positions of the slabs. Right: Market Hall as generated from our program. ....	89
<b>Figure 9.6</b> Building generated by introducing several changes: variant 1. ....	90
<b>Figure 9.7</b> Buildings generated by introducing several changes: variant 2 and 3. ....	90

**Figure 9.8** Changing the section of the posts. Top: rectangular. Middle: circular. Bottom: I-beam profile. .... 92

**Figure 9.9** Changing the shape of the windows of the front and back facades. Top: rectangular. Bottom: elliptical. .... 93

**Figure 9.10** Changing the order of the frames of the lateral frames. Top: Frames following the sequence of the Market Hall. Down: Frames with a different order. .... 94

**Figure 9.11** Change in the formalization of the shape of the building: two parallelepiped shapes. .... 95

## LIST OF TABLES

**Table 6.1** Instances of the model, three examples. Left: Market Hall. Middle and Right: Variations. .. 52

**Table 9.1** Changes in the elements of the building just by changing its shape. .... 87

**Table 9.2** Changes in the elements of the building just by changing the position of the middle slabs (the symbol \* refers to elements that may not change by using a traditional approach). .... 89



## ABBREVIATIONS

**BIM** – Building Information Model

**CAD** – Computer Aided Design

**CAM** – Computer Aided Manufacturing

**GD** – Generative Design

**GS** – Generative System

**NURBS** – Non Uniform Rational Basis Spline

## GLOSSARY OF TERMS

**Generative Design** – Process through which various potential design solutions can be created determined by algorithms. The practice of using Generative Systems to mediate the design process.

**Generative System** – A system that generates options for design problems.

**Traditional approach for using CAD tools** – Approach in which CAD tools are used to represent or conceive a design based on abstract models produced with explicit modeling operations.

**Algorithmic system / Algorithmic approach to design** – Approach we propose to be integrated in the design process. The use of a Generative Algorithmic System that can easily handle change. Being controllable it is possible to generate several different variations of the same design.

**Formalization of the design intents** – To give a structure, defined enough, to the design intents so that they can be translated into algorithms.

**Program** – Formal representation of a design. An algorithm written in a way that the computer understands, i.e. a programming language, with specific and rigorous instructions that tells the computer what specific steps to perform.

**Programming** – The act of translating algorithms into a programming language so that they can be performed by the computer.









# INTRODUCTION

Design problems are ill-structured (Simon, 1973). In architecture, design involves a response to a problem which is often not clear at the outset and therefore the design process involves developing an understanding of the problem (Hudson, 2010). Moreover, the designed artifact operates in natural and social worlds, and they both introduce constraints (Gero, 1990). These constraints can be imposed by external conditions, e.g., site, climate, cost, laws, or by the concerns of the designer who establishes his own goals. Even with these countless constraints each design problem is characterized by a large range of possible solutions. This means that design also implies a process of selection amongst solutions, considering many tradeoffs. In this context, the design activity can be seen as an evolutionary process (Alfaris, 2009) that evolves from abstract ideas to more complex and concrete solutions, attending to the constraints and embodying new design requirements. This process advances and backtracks as many times as necessary in order to achieve a good solution to the design problem.

CAD tools always had an essential role in the design activities, contributing to increase their efficiency. Their constant evolution affects and profoundly changes the design processes and the architectural thinking.

At the beginning, digital tools served as documentation tools allowing to ensure accuracy and consistency in drawings, the expedite production of the desired number of identical documents and to edit them easily without having to erase and redraw parts or entire drawings manually. These tools evolved with the introduction of parametric features which enable the propagation of changes. Parametric tools started to highlight the logic of regeneration instead of redraw which was a significant advance regarding the efficiency of CAD processes. Three-dimensional modeling tools appeared parallel to the development of parametric software marking a departure from the bi-dimensional representation and, progressively, the exploration of complex forms became possible. The introduction of programming languages started decades ago but only recently their popularity increased in the field of architecture, which triggered the interest for algorithmic approaches to design, such as Generative Design (GD).

Despite this evolution, the traditional use of CAD tools, in which the computer operates as a more efficient and rigorous tool than the past paper-based approach, is still the most widespread in today's practice. Unfortunately, when changes are needed, it requires too much time and effort to modify models or even treat them as disposable. In fact, in the traditional use of CAD tools, just one solution is represented and the exploration of different solutions requires manual changes to the model. Thus, to incorporate those changes, the model is either manually edited, or it must be thrown away and a new model is developed. The effort and time required in both cases limit the amount of changes that designers are willing to make to their models and, in general, preclude the use of these models in the

knowledge development stages, which are prone to change. As a result, the traditional design loop is limited by the few solutions that are generated, among which one design solution is selected (El-Khaldi, 2007).

Design tools, in order to truly support and accomplish the needs of the design process, must embrace change. GD is defined by the creation of shapes determined by algorithms. When these algorithms are translated in a programming language and defined parametrically, they automatically embrace change. Thus, an algorithmic approach to design overcomes the limitations of the traditional approach. One of the main benefits of the algorithmic approach lays in the possibility of fast and effortless generation of a wide range of solutions, exploring different design approaches and implementing the continuously changing and evolving constraints that characterize the design process. The greater the number of possible solutions that can be evaluated, the more informed and supported are the decisions and the selection process. Due to its flexibility to support the introduction of changes, it is possible to have an evaluate-critique-modify loop where the algorithms store the logic of different design phases, addressing their different requirements. Moreover, the simple act of defining an algorithm forces a consistent and systematic design approach that promotes the prioritization of requirements and the clarification of design intents, which may benefit decision-making strategies. In advanced design stages, Generative Systems (GS) can be used to assist performance analyses and optimization, automatic extraction of documentation, mass customization strategies and direct fabrication through Computer Numerical Controlled (CNC) machines.

The use of algorithms is not a novelty in the architecture field and can be traced back to a time when computers did not exist. However, most of the early definitions and use of algorithms was focused on prescribing and externalizing author's design processes or goals in order to achieve a design with certain attributes. The introduction and use of programming languages, allowed the development of algorithms that are being used as part of the design process, to produce unique designs and achieve variation of the same design. GSs provide a synergy between the creativity and intuition of the designer and the capabilities of the computer, which allows handling a design solution space that could easily exceed human and time limitations.

In this context, it becomes essential to make architects aware of the advantages of this new approach and also prepare them for the potential drawbacks. In this dissertation we argue that GD should be introduced as a new and parallel phase in the design process, and as a complementary tool to sketches, physical models, prototypes or others. In particular, we propose an algorithmic approach to design that is rigorous, controllable and flexible, but that forces architects to translate intents into algorithms. It requires a formal description of the underlying design logic, extending the role of the designer from user to tool builder. Thus, it requires an obvious initial investment. Algorithmic thinking, mathematical thinking, abstract thinking and programming skills are indispensable roles for architects who want to embrace GD in their design processes. This new stage can initially be perceived as a large and superfluous effort but, as we will show, it is highly rewarding by the advantages that arise from its use. Similarly to the transition that happened with the revolutionary idea of replacing manual drawing with digital drawing, the transition from the traditional use of CAD tools to GD requires an investment, but an investment with a large return.

## OBJECTIVES

The main aim of this dissertation is to explore and evaluate the potential of GD as an auxiliary tool integrated in the design process. In particular, we explore and evaluate an algorithmic approach to design where the algorithms are translated into a programming language and defined parametrically.

To this end, we developed a three-dimensional model of a case study using a purely programming-based representation. The approach is evaluated on the case study of the forthcoming MVRDV's Market Hall building, currently under construction in Rotterdam, The Netherlands. Market Hall was chosen because it is a large and complex building that includes:

- a non-conventional overall shape;
- identical elements arranged according to composition rules;
- elements which have a geometry that is dependent on the overall shape of the building.

With such scenario it is possible to foresee that a change to the overall shape, however small it may be, will have repercussions in the, e.g., geometry, location, number, and so forth, of almost all the elements.

In general, all designs are prone to change because designers do not just build their first idea. GD supports change but it also requires an initial investment. Thus, it is important to balance this investment with the benefits we extract from it. As a result, the implementation of the case study has four main goals:

1. To evaluate the ability of the algorithmic approach to respond to the evolving requirements of the design process;
2. To evaluate the costs and benefits of the algorithmic approach, particularly when compared to traditional approaches;
3. To evaluate the ability of the algorithmic approach to be integrated in the early stages of the design process, when decisions are highly uncertain;
4. To evaluate the limits of the programming-based representation.

## METHODOLOGY

The methodology we followed to achieve the proposed aims consists in three main phases: (1) literature review, (2) choice and implementation of the case study (3) analyses, evaluation and conclusions.

The first phase, the literature review, consisted in a review of bibliography focused on digital tools for design, approaches to modeling, and emerging paradigms, particularly Generative Design. The research allowed the understanding of the underlying concepts of GD and GSs, the contextualization of these systems in the architectural practice over the years and lastly, the current state of the practice.

The second phase started with the choice of the case study, after a research on projects with characteristics that would enable the evaluation of the proposed approach. The design process we used to construct the model was based on an analysis of images, renderings, drawings, textual descriptions, and a drawing set provided by MVRDV office. We simulated the implementation of the model while the design process was underway so we needed an abstraction of the final project. The first step was to analyze the project and establish a chronological hierarchy simulating the order of concerns and priorities during the design process. After this analysis, we formalized the design ideas of the different elements, we defined algorithms, and we implemented them in a programming language, following those hierarchic relationships. This process was under constant evolution and evaluation. During the modeling process the expertise concerning the methods and ways to solve problems evolves thus, in some cases, the same element was defined following different approaches according to a critique and modify loop. These different versions were additionally developed for other reasons: (1) to overcome unexpected behaviors of the selected CAD, (2) to increase the detail of the elements, (3) to improve program performance, (4) to improve the legibility of the code, and (5) to increase the abstraction of the code. Different versions were stored and relevant considerations were documented.

The third phase started with the review of the collected material. The overall modeling process and the different solutions were analyzed in order to document tasks and strategies that can be used as lessons to inform the development of similar models. Then, we evaluated our approach and compared it to the original objectives. Lastly, based on the evaluation, we presented the conclusions of the work.

## **STRUCTURE**

This dissertation is divided into two main parts: “Background” and “An Algorithmic Approach to Design”. The Introduction, Conclusions and Bibliography are added to these two parts.

The “**Background**” is composed of four chapters:

### **1. Generative Design:**

In this chapter GD is defined, it is presented the beginnings of GSs in architecture and their maturity and, finally, the main characteristics of several GSs with practical examples, namely: Algorithmic Systems, Parametric Systems, Shape Grammars, L-systems, Cellular Automata, and Evolutionary Systems and Genetic Algorithms.

## **2. Design Tools:**

In this chapter it is briefly outlined the evolution of design tools until the fairly recent adoption and popularity of programming languages in architecture.

## **3. Approaches to Modeling: Traditional vs Algorithmic:**

In the second part of this work the traditional approach to modeling is confronted with the algorithmic approach. The role of the designer and the effort needed to handle change in models developed with these two significantly different approaches are characterized and evaluated.

## **4. Generative Design Practice:**

In the last section of this part we characterize the strategies that are being used to apply GD in actual practice. We also included a case study of a well-known and documented project, the Water Cube, which is an example of an integrated approach in which GD was used from the competition stage to detailed design.

The second part of the dissertation, “**An Algorithmic Approach to Design**”, is composed of five chapters:

## **5. Introduction:**

In this chapter it is reintroduced the motivation of this dissertation in order to contextualize the second part of this work.

## **6. Modeling process:**

In this chapter it is presented: (1) the modeling strategy followed to construct the model used to evaluate the algorithmic approach to design, (2) the application of that strategy to the chosen case study, (3) a brief description of the main operations and strategies adopted, and (4) examples of some instances of the developed model.

## **7. Definition of the model:**

A more detailed description of the main operations and strategies presented in the chapter 6 are explained in this chapter and some additional strategies considered relevant are included.

## **8. Analyses of the modeling process:**

In this chapter are summarized the results of the modeling approach when applied to the implementation of the case study in order to provide lessons that can be used to inform the development of similar models.

## **9. Evaluation:**

The evaluation of the algorithmic approach to design is presented in this chapter, in particular the evaluation of its ability to respond to the evolving requirements comparing with the traditional approach to modeling. The relevance of using this approach since the early stages of the design process and the limits of the programming-based representation are also discussed.

In the remainder of this dissertation we elaborate all these topics in order to allow a comprehensive overview of what is GD, provide strategies to use it and conclude about its relevance in the architectural practice.

## **PART I : BACKGROUND**

The invention of paper, of the movable printing process, of scale drawings and, finally, of perspective drawings caused changes in the architectural profession. The Information Technology revolution started with the advent of computers, in the 1950s, was the beginning of a profound reformulation of the architectural practice. Since then, many changes occurred in the way architects approach, create, think, represent, communicate, edit, evaluate and manage designs, but also in the way they construct them.

The systems that are being used in today's practice are the result of more than fifty years of research, development and commercialization. In fact, many of the ideas that are now widely adopted or considered leading edge have a long history in research.

CAD and CAD/CAM are technologies that can assist the design process from the early stages until the construction. However, compared to other disciplines, architecture has been slow in taking full advantage of these technologies.

Generative Design is an example of a well-established research area that can provide new ways of thinking that can have a profound effect on architectural design practice. It requires breaking with the traditions of CAD, transforming the designer from "user" to "tool builder", in the sense that he constructs his own algorithms to achieve a result, by customizing the CADs functionalities.



# 1 GENERATIVE DESIGN

Generative Design can be described as a process through which various potential design solutions can be determined by algorithms. As highlighted by Lars Hesselgren, Director of KPF Research (*cit. in* Stocking, 2009) “Generative design is not about designing a building. It’s about designing the system that designs a building.”

Algorithmic systems are at the foundation of all Generative Systems. Stiny and Gips (1978a) defined an algorithm as an explicit statement of a sequence of operations needed to perform some task. Computation is the act of performing those operations. Computations do not necessarily need to be performed by a computer. In fact, the use of computation within the architecture field is not new or even recent and can be traced back to a time when computers did not exist. However, the use of modern computers allows the architect to overcome time limitations and to quickly experiment different design solutions. By combining the speed of modern computers with the creativity and intuition of architects we create a powerful synergy that allows us to achieve better designs.

## 1.1 Beginnings and maturity of Generative Systems in Architecture

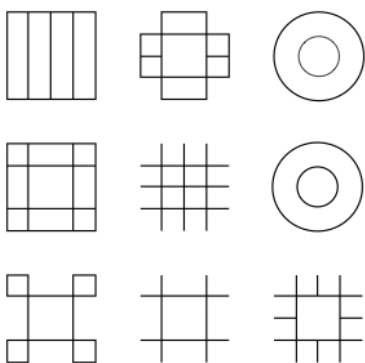
“Architecture, as a practical form of art, has been in need of computation - and computational aids - since ancient times” (Kalay, 2004: 63). Architects needed to calculate sizes, proportions, areas, volumes, and needed the tools to create geometric constructions and communicate them to the builders (Kalay, 2004).

Computation in architecture came to have special focus during the Renaissance by the influence of Vitruvius' work on the architects of that epoch. Written in the first century BC, Vitruvius's *De Architectura*, known as the *Ten Books on Architecture*, is the oldest recorded design method and known to be the first documented account of design rules. Vitruvius provided rules to describe different aspects of Roman design, including architecture, engineering, and city planning. For example, in his treatment of the temples and columns of the three Greek orders, he establishes the proportions

needed to achieve harmony while respecting the known constructive principles. Architects became interested in finding the most appropriate geometric relationships between the different elements of buildings, for that reason they trained in geometry and developed methods, or algorithms, that, with the help of the compass and straightedge, allowed them to compute these geometric relationships. Vitruvius's offered practices to guarantee good solutions and, since then, many architects and researchers started to develop theories, methods, and tools to make designs more predictable and the design process more tractable, teachable, and open to analyses and improvements (Kalay, 2004).

Andrea Palladio's main written legacy, *I quattro libri dell'architettura* (1570), became a reference for the neoclassical style, a manual of how one should design and construct public and private buildings. The book sets out rules of classical architecture, the conventions of composition and construction governing correct building practice, established by prescription and example (Stiny et al., 1978b). Conventions were based on assumptions such as bilateral symmetries and the use of certain proportions, and the design process was defined through a well-defined sequence of steps. The rationality of the process, with explicit instructions, allowed the production of algorithms that embedded Palladio's rules, especially of his Villas. Some examples of these studies are the parametric shape grammar of Stiny and Mitchell (Stiny et al., 1978b) and more recently, the work of Celani and Kubagawa (Celani et al., 2007).

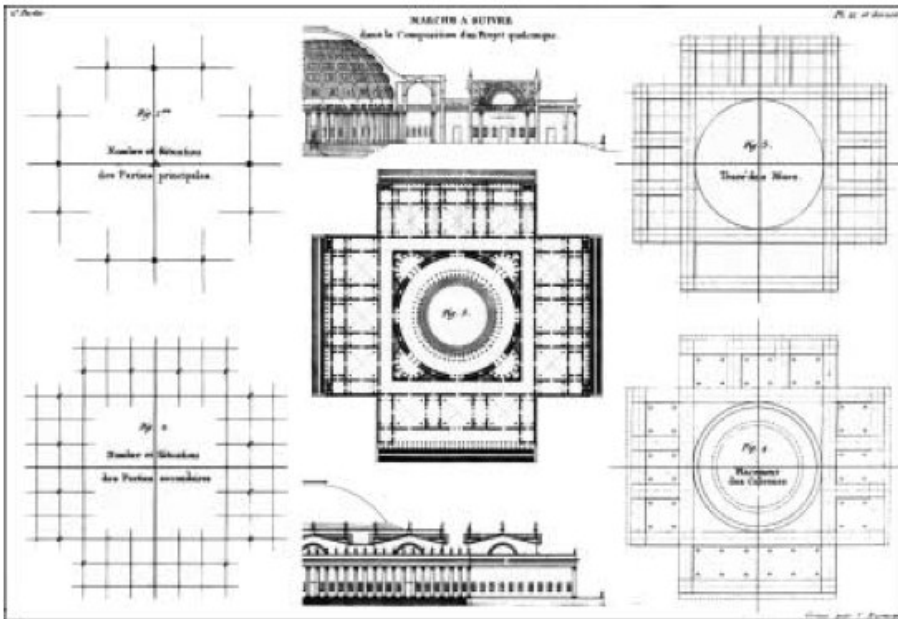
Renaissance architects saw form as a composition made from limited sets of elementary components, and the first truly systematic expression of this concept could be found in the texts of Jean Durand (Mitchell, 1973). Durand, and then Julien Guadet, both French architectural pedagogues, wrote highly influential texts that were used by generations of students: *Précis des Leçons d'Architecture* (Durand, 1802 - 1805), *Partie Graphique des Cours d'Architecture* (Durand, 1821) and *Éléments et Théories de l'Architecture* (Guadet, 1902).



**Figure 1.1** Skeletons with construction lines as starting points for plan compositions, from Durand's *Précis* (source: Mitchell, 2001).

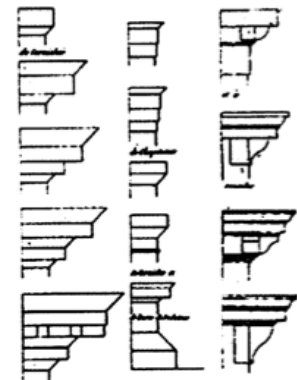
Durand and Guadet established and formalized a process that started with the production of an organizing skeleton based on abstract ordering devices such as grids and axes. Various combinations of these devices could be explored to produce the skeleton (Figure 1.1). Setting up the main axis and adding a grid with the secondary axis were the first steps of the process. This geometry was used to guide alternative ways to arrange the major rooms and circulation spaces, placing walls along the axes, and then columns between walls. The design would be developed by adding elements from an established vocabulary of elements such as stairs, doors, windows,

and other elements in the plan. After the plan was defined, the design would be finished by drawing sections and elevations. The process was developed hierarchically through a top-down substitution, from abstract devices to specific elements (Figure 1.2) (Mitchell, 2001).



**Figure 1.2** Hierarchic process of top-down substitution, from Durand's *Précis* (source: Mitchell, 2001).

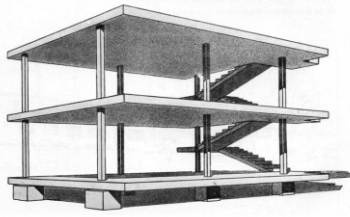
There was a well-defined vocabulary of discrete construction elements (Figure 1.3), there were rules expressed by allowable combinations of grids and axes, and it was possible to produce acceptable designs by arranging the elements of the vocabulary according to these rules (Mitchell, 2001).



**Figure 1.3** Example of Durand's building vocabulary (source: Celani, 2002).

This grammar is an example of a Generative System from the history of architecture that was not only influential for the students of architecture. Durand's idea of composing a building from discrete standard parts anticipated the industrial mass production. It is also possible to recognize the influence of his work on early and current CADs, that organize drawings in "layers", providing grids, construction lines, and "snap" operations for positioning graphic elements in relation to other elements, and, finally, in the libraries of pre-defined architectural objects, such as doors, windows or walls (Celani, 2002).

Le Corbusier, in 1914-1917, devised the Dom-ino constructive system. The Dom-ino is a system composed of concrete slabs, columns and foundations, that suggests a rational ordering of its elements and construction, as a way of endowing new buildings with certain formal attributes, both concrete (free facade and plan, pilotis, etc.) and abstract (economy of means, speed, accuracy and precision in construction). The relationships between elements are not only material but also concerned with dimensions and proportions, going beyond the mere functional arrangement of elements, proposing

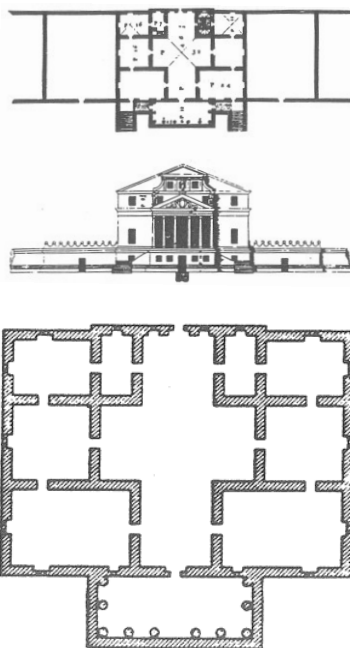


**Figure 1.4** Module of the Dom-ino (source: Frampton, 2003).

measures for the improvement of design and construction through prefabrication and standardization (Palermo, 2006). The module of this system (Figure 1.4) could be replicated in any direction. Some years later, in *Les 5 points d'une architecture nouvelle* (1926), Le Corbusier gave specific guidelines to achieve his "new architecture" - the pilotis elevating the mass of the ground; the free plan that is achieved through the separation of the load-bearing columns from walls; the free facade; the long horizontal sliding window; and the roof garden restoring the area of ground covered by the house (Frampton, 2003). Although these points are a syntax to achieve a design with certain attributes, they can be seen as goals that designers can follow to produce their own methods.

The maturity of Generative Systems in architecture happened after the beginning of the development of architectural oriented software in the middle of the twentieth century. The use of algorithms evolved from these approaches in which authors prescribed their processes, externalizing their design processes. The concept of reproduction was inherent to all the above examples in the form of procedures or rules that, when followed, would produce the intended designs. However, none of those processes explicitly discussed the idea of reproduction to achieve variation in designs (El-Khaldi, 2007).

One of the first systems written for variation in architecture was based on Shape Grammars with the purpose of generating Palladian Villas (1978b). Stiny and Mitchell developed a parametric shape grammar that generated not only the ground plans of Palladio's Villas but also the ground plans of Villas that did not exist (Figure 1.5). The authors' first attempt was "(...) made to recast parts of Palladio's architectural grammar in a modern, generative form" (Stiny et al., 1978b: 5). They studied the geometry of Palladio's Villa plans instead of other aspects of his architectural system, for example, the use of decorative motifs or the facades, because the systematization of the ground plan is the distinguishing feature of Palladio's Villas. In this study, Stiny and Mitchell found inconsistencies, and minor deviations between the constructed Villas and the drawings in Palladio's *I quattro libri dell'architettura*. Given that Stiny and Mitchell were implementing their grammar in a computer, they had to exercise their best judgment to eliminate the inconsistencies and rigorously define the rules in a computable way.



**Figure 1.5** Top: The Villa Malcontenta as drawn by Palladio. Down: The final plan as generated from the shape grammar (source: Stiny et al., 1978).

This work was followed by various authors. At this time, the authors were influenced by the possibilities of the evolving technology, namely the computer. Peter Eisenman (*cit. in Koder, 1994*) recognized (1) the power of using algorithms to produce results that the architect wouldn't know *a priori*,

and (2) that writing and correcting these algorithms would become one of the tasks of the design process. In the same period William Mitchell published *The Logic of Architecture* (1990) introducing into the design processes concepts such as algorithms, evolution, grammars, and logic. In 1993, Greg Lynn published *Architectural Curvilinearity: The Folded, the Pliant and the Supple*, one the first examples of the topological approach to design, characterized by a departure from the Euclidian geometry of discrete volumes manifested by continuous, highly curvilinear surfaces, in which he exclusively used NURBS represented by parametric functions to describe a range of possibilities. He also published *Animate Form* (1998) in which he uses animation software not as a medium of representation but of form generation (Kolarevik, 2003).

The introduction of new programming languages, in the late 90s, offered architects the possibility of writing their own generation tools. At this time, Generative Systems from other fields, namely biology and mathematics, began to be used by architects that finally could write their own algorithms.

The use of digital technology “(...) opened up new opportunities by allowing production of very complex forms that were, until recently, very difficult and expensive to design, produce or assemble using traditional technologies” (Kolarevik, 2003: 3). Although there is a great potential for producing designs that would be difficult to produce without the use of digital technology, the main advantage of Generative Design is the continuous exploration of a project, generating multiple variants, quickly and effortlessly. Generative Design should not be seen as a formal revolution, but as a revolution in the way of thinking a design. As a result, they can be applied to any style, allowing the development of unique projects within time and budget limitations.

## **1.2 Generative Systems**

In the next sections, some Generative Systems are briefly characterized, namely: Algorithmic systems, the basic system in all Generative Systems; Parametric systems, a special case of algorithmic systems; and then more specialized versions, such as Shape Grammars; L-systems; Cellular Automata; and Genetic Algorithms.

### **1.2.1 Algorithmic systems**

Algorithmic systems are the basic systems in all Generative Systems. Algorithms are sequences of instructions for solving a problem or achieving some end, written in a fixed vocabulary, that must be specified step-by-step

or in discrete steps, "(...) whose execution requires no insight, cleverness, intuition, intelligence or perspicuity, and that, sooner or later, comes to an end" (Berlinski, 1999: 35). Algorithms are sometimes known as methods, procedures, techniques, all the names conveying the idea of a set of instructions to solve a problem.

Algorithms can be expressed in different ways, according to the medium where they are described. They can be represented in different forms such as: code, graphics or verbal descriptions. In computers, algorithms can only deal with discrete units such as: numbers, alphabets or geometric elements. Algorithms are essential to the way computers process information, because a computer program is essentially an algorithm that tells the computer what specific steps to perform.

Boolos and Jeffrey (1999: 19) framed the concept of using algorithms, by humans or computers:

*"No human being can write fast enough, or long enough, or small enough to list all members of an enumerably infinite set by writing out their names, one after another, in some notation. But humans can do something equally useful, in the case of certain enumerably infinite sets: They can give explicit instructions for determining the  $n$ th member of the set, for arbitrary finite  $n$ . Such instructions are to be given quite explicitly, in a form in which they could be followed by a computing machine, or by a human who is capable of carrying out only very elementary operations on symbols."*

Such a symbiosis between human and computer is predicated on communication. That communication must be done in a common language that both understand and that allows to share information between human and computer. It is relatively easy to communicate information from computers to humans, who have the intelligence to understand different types of representation in which the computer is able to present the information such as textual, graphical or auditory messages (Kalay, 2004). The inverse, to communicate information from humans to computers, is difficult because computers lack the intelligence and the ability to interpret information, unless it is coded in a completely unambiguous way. Thus, to implement a concept in a computer the designer have to translate it into a formal language that the computer understands, a programming language (Leitão et al., 2012a).

Computers popularized and extended the notion of coding in architecture, the representation of algorithmic processes that express architectural concepts or solve architectural problems, by simplifying the implementation and computation of algorithmic processes (Leitão et al., 2012a).

Algorithmic systems for design require clear design intents expressed on discrete steps and units. They require components to generate data and components to interpret it, and the data can be of various types such as locations, shapes, shape properties, etc. Communication among components is possible through associations, constraints, and rules.

Associations are relationships that allow for triggering a series of changes by only changing the value of variables. For example, in the relationship  $x = 4 + y$ , if the value of  $x$  is changed the value of  $y$  must be updated and vice versa. Associations are useful to propagate changes throughout a model.

Constraints are conditions that must be satisfied. Although the word may suggest something restrictive, a constraint can be anything the architects wants the model, or building, to satisfy (Mark et al., 2008). Constraints can be sizes, positions, and spatial relationships between elements. An example of a constraint may be simply “this window is in the middle of this wall”, so that when the wall is changed, the window position is automatically adjusted. Constraints are useful to ensure certain behaviors of the model but if the model is over-constrained it may eliminate potential solutions or hinder the generation process.

A rule is a condition that, when satisfied, triggers an action. A typical form is an if-then condition-action. However, rules are not limited to if-then forms, they can also include other forms such as do-until, or for each-next. Building design rules requires designers to analyze and unpack relationships and dependencies within a design problem to be able to describe it via a set of clearly defined rules.

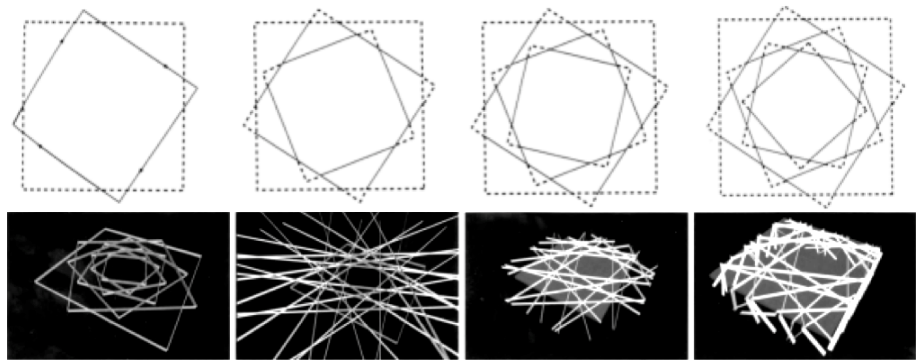
An algorithm may be either iterative or recursive. The process of iteration can describe a particular form of repetition with a variable state. Repetitive constructs are employed in these iterative algorithms, such as loops. Recursive algorithms are those that repeatedly refer to themselves until a specific condition is met. The strength of recursion lies in the ability to solve complex problems with simple solutions.

An algorithmic approach to design requires a rationalization process that forces designers to structure their thinking around relationships and sequences of tasks. Algorithmic systems are the most flexible and customizable of all generative systems. They do not impose a specific structure, or type of relationships. These systems only provide a working environment where the architect has the freedom to implement his own algorithm (El-Khaldi, 2007).



**Figure 1.6** The 2002 Serpentine Gallery (source: <http://www.serpentinegallery.org/architecture/>).

A built example based on an algorithmic approach to design is the pavilion for the 2002 Serpentine Gallery designed by Toyo Ito and Cecil Balmond with ARUP (Figure 1.6). The algorithm specifically designed for this building involves the rotation and scaling of a series of squares around a central axis (Figure 1.7). Each square is smaller and embedded in the previous one but rotated. By iteratively repeating this algorithm a pattern of squares is generated and, by extending the lines of the squares, a dense field of lines is defined creating numerous triangles and trapezoids. Then, these lines are folded over one box, defining the location of the structural members. The lines are materialized by choosing the size of the steel and the algorithm defines the specific areas that are panelized. In the end, the generated pattern is adapted in order to, e.g., provide entrances and create larger openings (Deuling, 2001). The result is a building in which there is no clear distinction between skin and structure and that, although looking complex and totally random, was generated by a well-defined algorithm.



**Figure 1.7** Schematic representation of the application of the algorithm (source: Deuling, 2001).

In the next sections we will describe more specific kinds of algorithmic systems that do impose particular approaches to the design problem.

### 1.2.2 Parametric systems

Parametric design or parametric modeling can be broadly defined as the description of a design problem using variables.

“Parametric modeling represents change” (Woodburry, 2010: 7). The idea is not new. In 1957, Patrick J. Hanratty created PRONTO, the first commercial software to provide parametric algorithms for translating data from computers to manufacturing machines and, in 1963, Ivan Sutherland developed Sketchpad and demonstrated the first graphical representation of parametrics. Sutherland’s “(...) invention of a representation that could adapt to changing context both created and foresaw one of the chief features of the computer-aided-design systems to come” (Woodburry, 2010: 7). Parametric design was championed in the 1980s and today many CAD applications offer the ability to establish relationships and use variables.



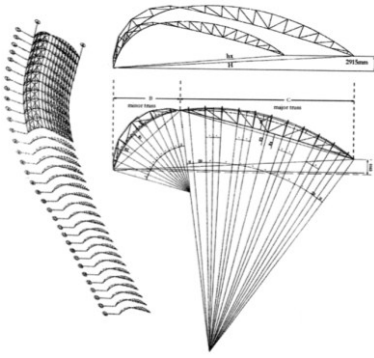
Parametric design software was developed principally for mechanical engineers, civil engineers, industrial designers and for the transport industry all of whom make good use of products that allow the 'whole' to be resolved into a large number of associated and adaptable parts (Burry et al., 1997). Mechanical engineers, for example, always had to optimize their designs based on performance analyses. Every time a design changed, no matter the dimension of that change, they needed to redraw their designs. Parametric systems allowed them to regenerate instead of redraw or edit their designs.

In parametric systems it is the parameters of a particular design that are declared, not an explicit shape. Parametric modeling is based on relationships between objects controlled by those variables. Establishing relationships between parts of a design allows them to change together in a coordinated way, thus defining an associative geometry. By assigning different values to the variables, alternative models can be easily created.

Parametric systems can be classified as a specific type of algorithmic systems. These systems are hierarchical algorithmic systems controlled by one-directional relationships. Dependencies only go one way, setting up the relationships between dependent and independent parts so that propagation of changes happens from independent to dependent parts.

Modern parametric software available commercially allows the definition of a parametric model in two main ways. One is to program the model using a specific textual programming language. The other, exemplified in tools such as Generative Components and Grasshopper, is to use visual programming languages to elaborate diagrams or graphs that represent the algorithm that will generate the parametric model. In general, these applications allow a limited but sufficient automation of the process to allow efficient exploration of parametric variations of forms (Celani et al., 2011). Compared to textual programming languages, these tools require a lower level of abstraction but "(...) long practice in using, programming and teaching parametric systems shows that, sooner or later, designers will need (or at least want) to write algorithms to make their intended designs" (Woodburry, 2010: 34). Moreover, it has been proven that tools such as Grasshopper, "(...) scale poorly with the complexity of the design task", so as the scripts grow it becomes difficult to understand and due to their poor abstraction mechanisms the users are forced to develop them with copy/paste of already defined fragments (Leitão et al., 2012b: 143).

One well-known example of the application of a parametric system is the International Terminal at Waterloo Station in London, designed by Nicholas



**Figure 1.9** Left: Dimensionally different but identically configured arches. Right: Schemas with the parametric definition of the scaling factor for the truss geometry (source: Kolarevik, 2003).

Grimshaw and Partners and built in 1993 (Figure 1.8). The building is a long curved train shed, with a variable span that gradually expands. Its curve and variable span were forced by site constraints. Its roof structure consists of a series of arches, dimensionally different but with an identical configuration that follow the imposed site curve. To solve these problems they created a parametric model, relating the size of the span and the curvature of individual arches, that encoded the rules of the whole family of arches, rather than modeling each arch separately (Figure 1.9) (Szalabaj, 2001). The parametric model was extended from the structural description of the arches to the cladding elements that connect them, i.e. to the entire building (Kolarevik, 2003). When a dimensional change was made in the parameters, it was propagated through the entire model.



**Figure 1.8** International Terminal at Waterloo Station (source: <http://grimshaw-architects.com/>).

### 1.2.3 Shape Grammars

Invented by Stiny and Gips in 1972, Shape Grammars (SG) laid the foundations for research into algorithmic design approaches in the context of design analysis and design synthesis (Alfaris, 2009).

A shape grammar is a set of shape rules that is applied in a step-by-step fashion to generate a set, or language, of designs. Shape grammars are spatial, rather than textual or symbolic, algorithms (Knight, 2000).

Each rule identifies a shape and replaces it by another shape. SGs operate by recursively applying these rules to a starting shape. The components of rules are shapes such as points, lines, planes or volumes. Rules consist of a left side, the initial shape, and a right side, the resulting shape. Designs are

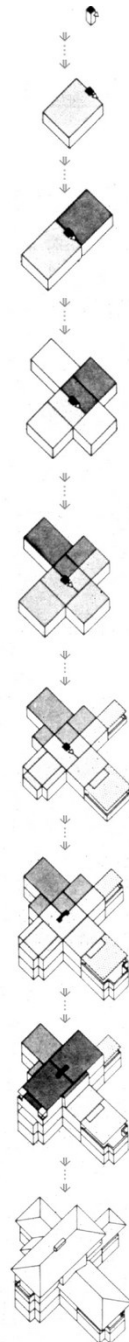
generated using shape operations such as addition and subtraction, and transformations such as mirroring, rotating or scale.

A key feature of these grammars is that they are built on the promise of mimicking the process of thinking of a designer based on the notion of recognition (Alfaris, 2009). The concept of recognition underlies shape grammars as these treat shapes as non-atomic entities that can be freely decomposed and recomposed (Knight, 2000). This decomposition is based on the notions of Embedding and Maximal Elements. Any shape is considered a maximal element that includes all possible embedded smaller elements with identical topology. For example, a rule to be applied to a line segment, surface or solid may be applied to all the embedded smaller line segments, surfaces and solids that are contained on the whole element. This allows for emergence of new shapes, that is, the ability to recognize shapes that were not explicitly defined in a grammar but emerge from any parts of the generated shapes (Duarte, 2001).

In general, there are several rules that can be applied to a given shape, and several different ways to apply them, which make SGs non-deterministic systems (Knight, 1999). Like other Generative Systems, SG can be parameterized. Parametric shape grammars compute designs with parameterized shapes such as its measures or points.

Applications of SG in architecture and arts comprise two complementary purposes, synthesis and analyses and a third one which is a combination of both. Synthesis grammars are developed to create original designs, that is, to create new design languages or styles from scratch. The use for analysis is focused on the identification of some existing design languages for the subsequent generation of variant solutions belonging to the same language.

Among the first grammars used for the analysis of architectural design languages one can highlight the Palladian Grammar by Stiny and Mitchell, from 1978, or the Prairie Houses of Frank Lloyd Wright by Koning and Eizenberg, from 1981 (Figure 1.10). Knight argued that, in practice, original languages were generated from past or existing ones. Realizing this, Knight, in 1981, proposed a method to develop new languages based on existing ones. First, a known style is analyzed and a grammar is inferred. Later, the rules of that grammar are transformed to become the basis for a new grammar and style. This approach can be used not only to characterize the historical evolution of known styles but also to create new styles based on the existent ones. Knight applied this model to analyze stylistic changes in architecture such as Frank Lloyd Wright's work or in De Stijl painting (Knight, 1999).



**Figure 1.10** SG for the Prairie Houses of Frank Lloyd Wright by Koning and Eizenberg (1981) (source: Knight, 2000).

More recently, an analytical SG was developed for the Malagueira houses designed by the architect Álvaro Siza Vieira (Duarte, 2001). This work proposes a process for providing mass-customized housing and suggests that the developed grammar provides a rigorous method for understanding and teaching Siza's design process.

In spite of the usefulness of SG for understanding design, the implementation of SG interpreters in computers has been extremely difficult. The main obstacle is that SGs are based on visual computations but computers are only capable of symbolic computation. Nevertheless, SGs are a rigorous method to understand and teach architectural styles and to help architects to formally express design intentions, forcing the designer to clarify ideas and approach the design in a hierarchical fashion, where stages and design priorities are expressed and formalized (Alfaris, 2009).

#### **1.2.4 Lindenmayer Systems**

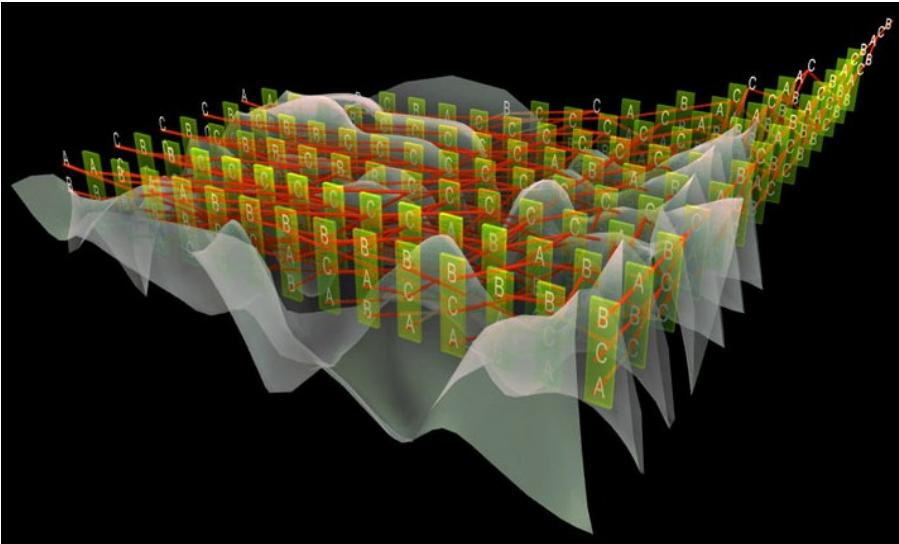
In 1968, the biologist Aristid Lindenmayer proposed a mathematical theory of plant development based on the central concept of string-rewriting. The algorithm can easily model simplified plants and simulate their growth process (Prusinkiewicz et al., 1996). These systems are known as Lindenmayer Systems or, for short, L-systems.

L-systems can be decomposed into a generative process and an interpretative process.

The main idea behind the generative process is string-rewriting. Each symbol of an initial string is replaced by another symbol or sequence of symbols, according to predefined rules that are applied in parallel. At each application of the replacing rules a new string is generated which is then subject to the same rules. This process is generally repeated for several generations.

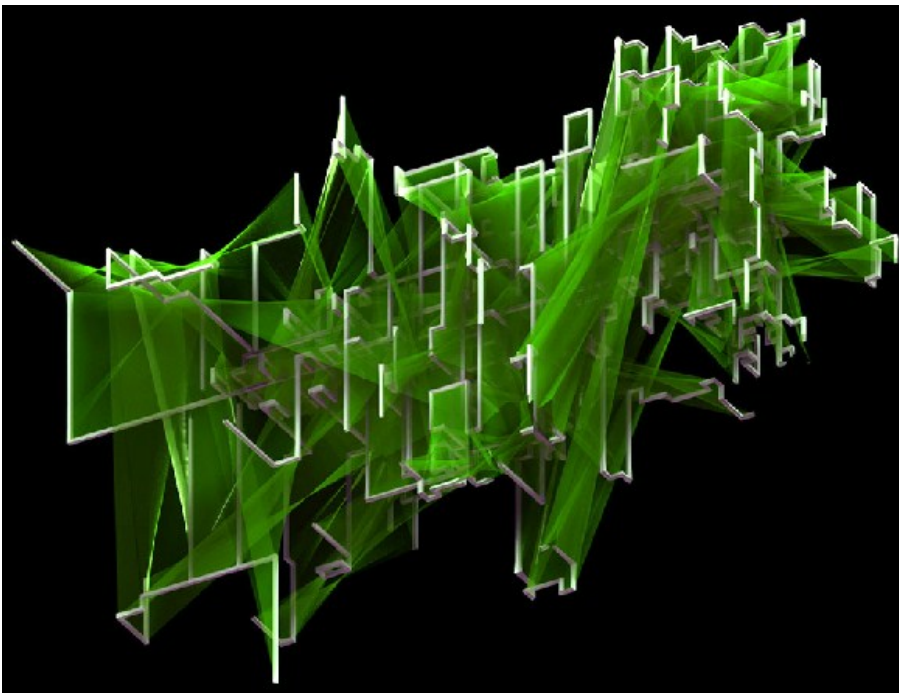
The interpretation process consists of reading the symbols of one or multiple generations of strings, and treat them as growth patterns or shapes. There are several methods to visualize strings, such as the mapping interpretation or the turtle interpretation.

The mapping interpretation visualizes simultaneously multiple generations of string using different mapping methods. Symbols can be mapped for example to vertices of a surface or to objects attributes such as scaling, rotation or location, giving rise to different results such as, respectively, surfaces or individual objects (Figure 1.11) (Hansmeyer, 2003).



**Figure 1.11** Surface obtained by mapping letters from different generations to a surface (source: Hansmeyer, 2003).

Visualization through turtle interpretation can be in two-dimensional or three-dimensional space and is based on an object, a fictitious turtle, that follows instructions associated to symbols (Figure 1.12). A given symbol can be interpreted as, for example, “Move forward and draw a line”, “Move forward without drawing a line” or “Start a branch” (Prusinkiewicz et al., 1996). This mechanism works by interpreting a single generation of string individually and not simultaneously as the mapping process explained above.



**Figure 1.12** Turtle graphic visualizing generations of strings as paths which in the end are left. The turtle moves with an angle rotation of  $90^\circ$  and in the three-dimensional space (source: Hensmeyer, 2003).

To increase the modeling power of L-systems, symbols can be augmented with parameters whose values are used by the rules both for determining their own applicability and also to compute the new values that the replacement symbols will have for their parameters. These systems are

known as Parametric L-systems and they are different from basic L-systems in that the parameters can encode information to be used in the interpretation phase. For example, in visualization through turtle interpretation a parameter can be treated as the length that the turtle will move or as a rotation angle.

L-systems are considered a powerful tool for designers (Fasoulaki, 2008: 11) particularly, because of their ability to produce highly complex designs from extremely small inputs (Hansmeyer, 2003).

### **1.2.5 Cellular Automata**

Cellular Automata (CA) are generative systems, summarily described as self-reproducing systems by Von Neumann, one of the first scientists to consider such a model in the late 1940s.

A cellular automaton is a collection of cells organized in orthogonal grids, each with a finite number of states that are usually denoted as colors or numbers. These cells change their state according to update rules, applied to all cells of the grid in discrete time units. The update of each cell depends both on the current state of the cell and on the current states of its neighboring cells. CA are sequential systems where the behavior of each cell depends on the behavior of its neighbors (Alfaris, 2009).

There are a variety of hypotheses regarding the type of grid on which a cellular automaton is computed, it can be, for example, a line, a square or a triangle. The simplest type of CA is known as Elementary CA in which the "grid" is a line. This type of cellular automaton is composed of, at least, three cells, minimum condition for a neighborhood to exist, and each cell has two possible states, black or white, or 0 or 1, in a numerical representation. As a result, the application of rules depends only on the state of the cell itself, the state of the cell to its left, and the state of the cell to its right.

Cellular automata may show four types of behavior, defined based on the pattern of occurrence of certain behaviors over a defined time period: fixed point, periodic, chaotic or random. CA with fixed point behavior reaches a fixed state after a short period of time. Periodic behavior shows a repetitive pattern within a fixed period of time. A Chaotic, or Aperiodic, seems to repeat its behavior within different durations of time steps. Random does not seem to repeat its behavior (El-Khaldi, 2007).

Its applicability has been proven in areas such as mathematics, engineering, and behavioral sciences. In architecture, the application of CA with chaotic and random behaviors is limited because it is very hard to predict the

outcome (Alfaris, 2009). However, most of CA behavior is periodic and for this reason they are being used for pattern generation (El-Khalidi, 2007).

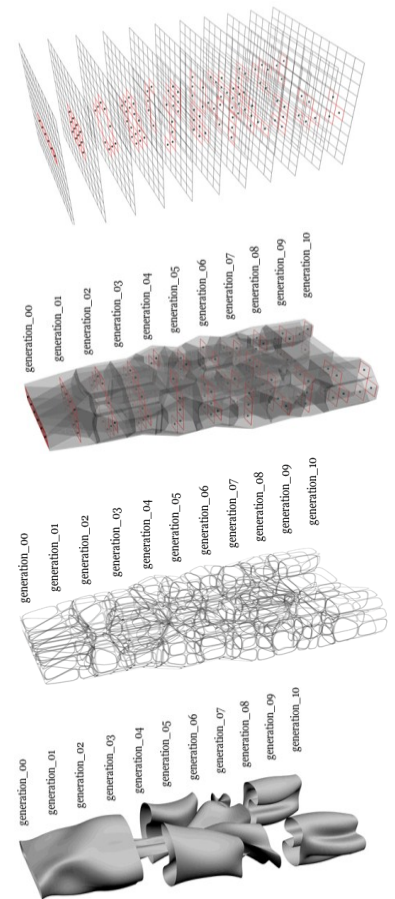
The experimental work of the architect Dimitris Gourdoukis is an example of the application of CA to architecture (Figure 1.13). He considered the different generations of CA as sections of an object. The centers of the active cells, the black, were used to generate a Voronoi diagram, itself a kind of Generative System. The edges of the Voronoi cells were used as the structural system and a smooth surface was used to define the enclosed space. The Voronoi diagram was used as a way of decomposing a space into regions where all regions are convex polygons. Each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other (Fasoulaki, 2008). The boundaries of the Voronoi diagram are the points that are equidistant to two generating points and the vertices are the points equidistant to three or more.

### 1.2.6 Evolutionary Systems and Genetic Algorithms

Evolutionary architecture “(...) proposes the model of nature as the generative force for architectural form” (Frazer, 1995: 9). Design is encoded in a set of chromosomes to which the rules of reproduction, gene crossover, and mutation are applied (Kolarevic, 2003). The key concept of these evolutionary models is the genetic algorithm (GA), a key element of biological evolution.

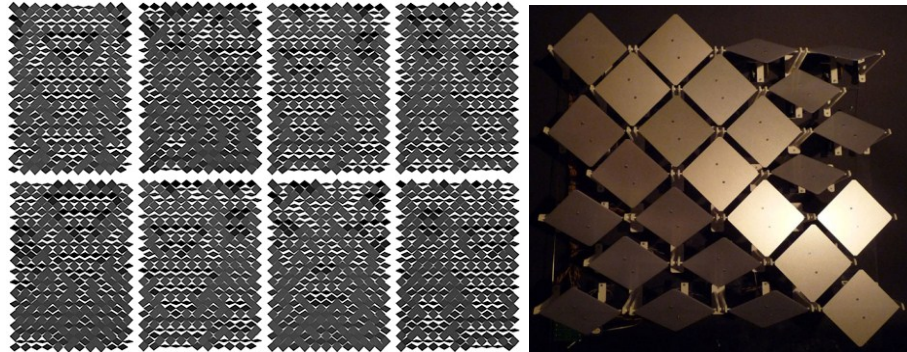
The various design parameters are encoded into a structure, similar to a string, and their values are changed during the generative process, generating a number of similar forms, “pseudo-organisms” (Kolarevic, 2003). Evolutionary concepts of selection and heredity are then applied to the generated forms and the best solutions of one generation are used to generate the next one. The “genes” that constitute the successful solutions are crossed over to generate new solutions, and variations are introduced by mutating some “genes” of the new solutions. The selection process is based on predefined evaluation criteria and fitness functions. This concept is fundamental since it is from this selection that the beneficial features are passed to new generations.

GAs can be used in architecture both as form-generation tools and as optimization tools. When used for optimization, the selection process treats as more apt those solutions that have better performance according to some criteria.

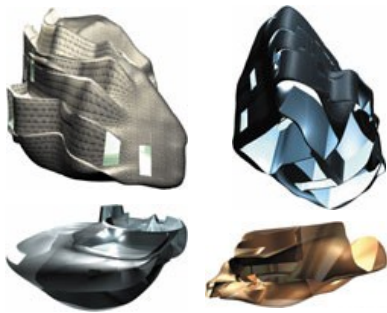


**Figure 1.13** The process applied by Dimitris Gourdoukis using CA (top) and Voronoi diagrams (third image) (source: <http://object-e-research.blogspot.pt/2007/10/growth-ca-voronoi.html>).

GAs are sometimes associated with other generative systems such as L-systems and Cellular Automata, taking advantage of their ability to model growth. This can be observed, for example, in a project of the Architect Marilena Skavara that explores Cellular Automata whose results are optimized using genetic algorithms (Figure 1.14).



**Figure 1.14** Left: Variants of an “evolutionary facade” that is optimized. Right: The selected facade (source: <http://www.microhappy.com>).



**Figure 1.15** Four hypotheses selected from a set of digitally obtained models (source: [www.archilab.org](http://www.archilab.org)).

Another example of the application of GAs in architecture is the project of the Atelier Kolatan and Mac Donald for mass customization of prefabricated housing, illustrated in Figure 1.15. The models represent four of the hypotheses selected from a set of digitally obtained models, where all variants were generated by the same genetic algorithm. This project explores the theme of the serial and organic compositeness in architectural and the ability of digital processes to create variability through different iterations.

As optimization tools, GAs were used to optimize one tower of Alvaro Siza’s School of Architecture at Oporto (Caldas et al., 2001). Natural lighting and overall environmental performance guided the generation of solutions. It was then possible to compare the existing facades with the facades achieved by the application of GAs and also to compare them with the best results obtained if the building had been built in another city.



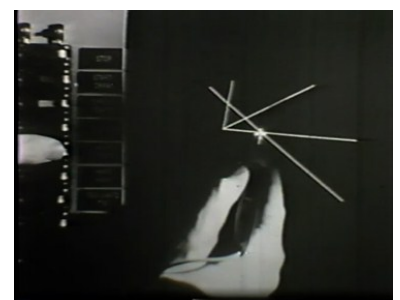
## 2 DESIGN TOOLS

The available design tools influence the design process, and they both have impact on the design. The invention and introduction of design tools along the history of architecture changed the ways of developing and externalizing design ideas and the evolution of the technology has been allowing new ways to manage and position design teams. The invention of paper, pencil, compass or the straightedge surely changed the architectural pattern. The same way, and already in the twentieth century, the advent of computers changed and is changing the processes of thinking and representing designs. The significant advance in CAD tools is allowing new design approaches and developing and constructing buildings that could not have been conceived decades ago.

The first impact of the advent of computers in buildings was to predict the behaviour of complex structures. The success of their application in the field of engineering triggered the interest for developing systems to aid architectural design problems. By the early 1960s there was sufficient interest to find if computers could create a design (Mitchell, 1989). And, in 1963, Sketchpad system, developed by Ivan Sutherland, demonstrated that computers could be used for the purposes of drafting and modeling as well as integrating the evolving design and analyses (Kalay, 2004).

The systems that followed Sketchpad were expensive, implemented in special environments and requiring especially skilled operators. The popular diffusion of CADs in architecture happened just in the late 70s, with the invention of the personal computer, cheaper and smaller than the previous. These tools, that became accessible for a larger community, were drafting systems. With the progressive advance of technology several companies started to develop software specifically tailored for the professional drafting purpose, like the well-known Autodesk, overcoming the limited capabilities of the first general-purpose systems. At this time, in the early 80s, the architectural community, from practice to schools, could use a CAD relatively easy (Mitchell, 1989).

In the late 80s, affordable CAD systems with three-dimensional modeling capabilities were introduced and with them the possibility of producing renderings. The role of the CAD tools was extended, they were no longer used exclusively in the drafting phases of the design process but also used



**Figure 2.1** Ivan Sutherland's Sketchpad system (source: Mark et al., 2008).

as a mean to conceive and communicate designs. These CAD tools deal with solids, polygons and later with NURBS. Three-dimensional modeling marked an important departure from the drafting systems, which were being used as an electronic version of the paper and just as a graphic tool. These systems freed designers from the conventional and bi-dimensional projections for helping designers to visualize models of their designs and communicate them realistically. In particular, the introduction of NURBS allowed the exploration of new forms and the visualization and manipulation of complex geometries that by hand or using bi-dimensional drawings would be laborious, time consuming and difficult to conceive.

At the same time of the appearance of 3D-modeling capabilities on CAD tools, parametric tools were introduced. Their aim is to allow the exploration of different solutions by establishing relationships between parts and by defining variable parameters creating associative geometries, which was a significant advance regarding the efficiency of CAD processes. In fact, these systems are based on systems developed much years before, particularly on PRONTO (1957) and Sketchpad (1963). Similarly to the introduction of 3D-modeling capabilities, these tools marked an important event by allowing the designers to regenerate their designs instead of remaking or editing the previous.

With the development of parametric software appeared other kind of tools, the Building Information Modeling (BIM). These tools, besides the geometry, store many other attributes of a design and can convey much information about several disciplines. Databases, e.g. with quantities or descriptions of elements, are combined with the model. These tools, in addition to the capabilities of the drafting and modelling systems, facilitate the management of the design and construction processes. As the parametric tools, BIM has a long history in research whose basis can be traced back to the software BDS (Building Description System), developed by Eastman in 1974.

Besides the abovementioned CAD tools, analyses software are being used since the 70s. These allow several types of simulation such as energetic, acoustic or structural behaviour. With them, architects are able to predict the behaviour of their future designs accurately and, with the information provided by the software, to change and optimize them.

Over the years, CAD tools have progressively introduced programming languages, which triggered the interest of designers in exploring new approaches to design and, particularly, in using Generative Design. By developing and implementing their own algorithms, designers are no longer locked to the standard CADs' functionalities and are able to customize them and develop tools specifically tailored to their needs.

We construct algorithms in everyday life, although they may not be rigorously defined or intentionally defined as so. To embrace Generative Design there is the need to formalize intents in order to define, rigorously, algorithms to be executed by a computer. The clear definition and formalization of the algorithms guiding a design, in a language that the computer understands, causes a huge difference in the benefits we can get from a CAD system. In the next chapter we characterize this algorithmic approach and we compare it with the traditional one, particularly, in what regards the effort to handle the changes that always arise during the design process.



## 3 APPROACHES TO MODELING: TRADITIONAL vs ALGORITHMIC

In this section we characterize the role of the designer and we evaluate the effort needed to handle change in models developed using two significantly different approaches: the traditional one, based on explicit modeling operations, and a modern one, based on the implementation of algorithms.

### 3.1 Traditional approach

In the traditional approach, that still governs today's practice, the designer interacts with two or three-dimensional graphic representations that mainly support the production of communication elements such as drawings or renderings. These CAD tools provide operations that allow the designer to easily create, manipulate, edit, and relate the parts of a model but their use is mainly descriptive: the tool is used as a replacement of the paper-based media and conventional physical models.

Using this approach, it is possible to handle complex designs as long as the design logic is clearly understood prior to the modeling tasks. The designer must previously examine the project to define which parts are to be represented and to determine the sequence of operations needed to create these parts. This modeling logic is sometimes visible in the auxiliary geometric elements present in the model or in the layers that organize the model but, in most cases, it is "(...) rarely recoverable and difficult to decipher" (Park et al., 2010: 363). As a result, even slightly different design options often require inordinate amounts of work, as very little modeling logic can be reused.

Moreover, handling changes is also difficult. In general, the more complex the model, the more difficult it is to adjust the model to some required change, as changing just one part may require manual adjustments to many other parts. As a result, changes in advanced phases of the design frequently require significant amounts of time and effort. This happens because the design logic was not described in a manner that is

understandable by the CAD tool and, thus, cannot be used to automatically regenerate the affected elements.

Modern CAD tools are tailored for designs containing repetitive elements. The tool usually provides operations that simplify the generation of such elements, for example, by allowing the designer to copy one element and paste it in several other places. Unfortunately, these operations become useless when the design does not contain such repetitive elements or when it contains elements that are identical except for some minor variation among them. In this last case, the best the designer can do is to copy a similar part, change it, and then paste it in the right place, repeating this process to all the variants, a slow, manual, and laborious task that requires high levels of accuracy. Any error might force the designer to repeat significant fractions of the work.

Another problem of this approach is the fact that it makes it very difficult for the designer to model shapes that are not available in the pre-defined geometric elements and operations provided by the tool. This means that the final design might be influenced by the limitations of the tool. In short, although these tools facilitate the production and visualization of complex geometries, they also “(...) limit exploration and effectively restrict design” (Woodburry, 2010: 23).

### **3.2 Algorithmic-based approach**

Contrary to the traditional approach, the algorithmic-based approach describes the design as a program written in a formal language (textual, visual, or both). This program can then be executed by a computer, generating a visible model as result. In order to write this program, the designer must know how to decompose the design logic and must select the appropriate programming elements to represent the fundamental parts as well as the relations between these parts.

Today many CAD tools offer programming environments: AutoLISP and RhinoScript are two examples of programming languages, both textual, that can be used with AutoCAD and Rhinoceros, respectively, and Grasshopper is a visual programming language that can be used also with Rhinoceros.

Using an algorithmic-based approach, the designer interacts with the computational mechanism that generates the digital representation. Obviously, conventional CAD tools also perform computations to generate

the digital representation; the difference is that in the traditional approach they are hidden from the user that interacts directly with the CAD.

Algorithmic systems are the core of all Generative Systems that drive the generation of designs. Algorithmic thinking requires the description of the design through a step-by-step process in which the ideas and intents have to be clearly defined, the relationships between elements are explicit, the requirements prioritized and the control mechanisms developed. This requires an obvious formalization effort.

A designer largely describes designs rather than processes, and usually representations are imprecise, relying on the human capacity to interpret them appropriately, a capability that a computer does not have: one misplaced character means that an algorithm will not work (Woodburry, 2010). Due to the large initial effort needed to formalize design, writing this algorithm might not be a trivial task and it might even be less cost-effective than the traditional approach. However, when it becomes necessary to incorporate changes in the design, this initial effort is quickly recovered. Depending on the structure of the program, a change might be as simple as updating the value of a parameter, or as complex as updating parts of the program. In all cases, however, the modified program is then used to regenerate the model. This is possible because the design is explicitly represented in the program, and changes in the design are translated to changes in the program. The quick model regeneration also allows the exploration of much larger solution spaces, which is advantageous for decision-making.

Just the attempt to construct an algorithm for a given process, that has necessarily to be defined unambiguously and deterministically without leaving anything to the imagination of the computing agent, provides an excellent means of exploring the process in all its aspects and features (Stiny et al., 1978). The initial effort needed to implement such a process at an initial stage may contribute for a consistent and systematic design approach that supports the understanding of relationships and clarifies intents, requirements and goals supporting the decision-making activity. The use of Generative Systems at conceptual stages can be seen as the root of an integrated approach to design that in further stages can assist the integration of other disciplines, including performance analysis, optimization, the automatic production of drawings, mass customization strategies and construction through “file-to-factory” (Kolarevik, 2003) processes and fabrication using Computer Numerically Controlled (CNC) machines.

In spite of all its advantages, there is one drawback in the use of a purely programming-based representation: if manual changes are made to the generated model, these changes will be lost when the program is re-executed and the model regenerated. The most obvious solution to this drawback is simply to forbid any manual changes to the model and always force its implementation in the program. Another possibility is to allow manual changes to the model, e.g., for studying different solutions, that are later incorporated in the program to synchronize it with the model.



## 4 GENERATIVE DESIGN PRACTICE

### 4.1 Strategies within architectural practice

Nowadays, the architectural practice uses technology to store, manipulate, and communicate designs, effectively connecting and coordinating design team members even when they are geographically separated. Designs, being “(...) descriptions of things that don't yet exist” (Mitchell, 2004: IX), may be stored in different media such as a paper, digital format or even in a designer's head. However, for larger and complex projects, the processing of the information, from the idea to the execution, requires division of labor among design teams. Conceiving, developing, evaluating and documenting designs became a specialized type of work (Mitchell, 2004).

Digital technologies have had a large impact on the professional practice and, thus, new skills are needed in the offices. Some decades ago the architects' role was extended to begin working with digital drawings, which was surely a revolutionary idea. Similarly, Generative Design (GD) is causing a revolution, extending the role of the architect to also become a programmer. GD requires algorithmic thinking, mathematical thinking, abstract thinking, and programming skills.

One of the motivations for using GD is to overcome the limitations of commercial CAD tools so that it becomes possible to effectively and efficiently tackle the specificities of each design. Thus, an increasing number of designers are using programming to specialize digital tools tailored to the needed tasks. Although many architects may become familiarized with algorithmic and mathematical thinking, programming is a specialized technical skill. In spite of the increasing interest in this area, the ability to program is still limited to a small group of architects and design teams (Santos et al., 2012).

Nowadays, programming is part of the architecture curricula in a large number of universities. However, although some architects may specialize in programming, the majority will only master it up to a certain level. This means that for very complex design tasks, the programming effort may be so large that it might not pay off to handle it to architects (Santos et al., 2012). In these cases, just like it happens with projects that involve engineers,

topographers, or other specialized members, the effective use of programming may require the presence of software engineers or, at the very least, of architects specialized in programming.

Today, collaborative teams are already established in some architectural and engineering practices. Another option, to avoid overload architects, is to use consultancy groups or independent consultants.

A consultancy group is the most commonly model for putting specialized skills into practice, and is adopted by both architectural and engineering offices (Hudson, 2012). Specialist Modelling Group (SMG) and BlackBox are two examples of architecture groups already using a GD approach and, within engineering groups, it is possible to highlight ARUP's Advanced Geometry Unit.



**Figure 4.1** Detail of the Smithsonian Courtyard Enclosure designed by FP with SMG as consultant (source: <http://www.fosterandpartners.com>).

The Specialist Modelling Group (SMG) is an internal research and design consultancy group within Foster + Partners (FP), established in 1998, and led by Hugh Whitehead. The specialized group is involved in project workflow, digital techniques, and the creation of customized CAD tools. Its group members work with project teams and are involved from concept design to fabrication. They are professionals originally trained as architects but that have also expertise in complex geometry, environmental simulation, parametric design, computer programming and rapid prototyping (Peters, 2007). Their designer mindset allows them to clearly understand the design brief and its requirements and, thus, to provide better support to design teams (Santos et al., 2012). Their work is not concerned with proposing form but searching for ways of describing it (Peters, 2007).



**Figure 4.2** The Smithsonian Courtyard Enclosure designed by FP with SMG as consultant (source: <http://www.fosterandpartners.com>).

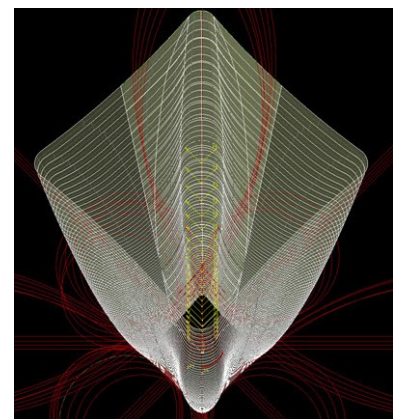
Blackbox group is a team of architects, designers and other experts that work within SOM (Skidmore, Owings & Merrill). This architecture firm claims to be one of the first to recognize the transformative power of computation and its relevance to the practice. For SOM, computers have supplanted most of the manual traditions of practice. They use methods in which the performance of the building guides the form-making but also for exploration of the power of computation as a creative design tool with ongoing research into parametric relationships. Algorithmic design processes using custom scripts are used to generate endless variations of formal studies in rapid succession, allowing designers to quickly study the effects of associating various aspects of form to various input criteria. This enables new types of collaborative communication between SOM architects, engineers, and outside disciplines. "Expertise from such disparate worlds as computer science, biology, chemistry, digital art, economics, and social science are now finding highly relevant seats at the SOM design table, resulting in algorithmic design processes that take the design teams into previously inaccessible regions of the architectural design space" (SOM).

SOM's Blackbox group is primarily concerned with research and development. Their focus is to make tools to improve the preliminary design process using skills in parametric modelling, geometry, scripting and analysis software. They are working with parametric and algorithmic processes to generate new approaches to designs. Their skills in parametric modelling and scripting are focused mainly in two areas: (1) the search for "optimal" solutions and, (2) the search for "novel". When a commercial application does not have the required tools for a certain design process, they take profit of their programming expertise and design their own tools, providing designers with valuable real-time feedback about their design choices, allowing them to respond from a more informed position while also helping to build a clear understanding of structural, solar, or light effects. The conduit by which SOM's parametric designers relate to the wider construction team is the digital model, combining different software, customizing and integrating a range of technologies from conception to construction.

Within ARUP, the Advanced Geometry Unit (AGU) is an example of a multidisciplinary team that mix architects, engineers and computer scientists, with mathematics, physics and programming skills, acting as internal and external consultants. Their primary role is to research complex structural geometry to support architectural visions and solutions (Hudson, 2012), to find new or unconventional solutions by exploring different strategies such as algorithms, Generative Systems, and non-linear structures (Santos et al., 2012).



**Figure 4.3** The forthcoming White Magnolia Plaza consisting of three towers and smaller scale mixed-used buildings. The design was developed by SOM with Blackbox (source: www.som.com).



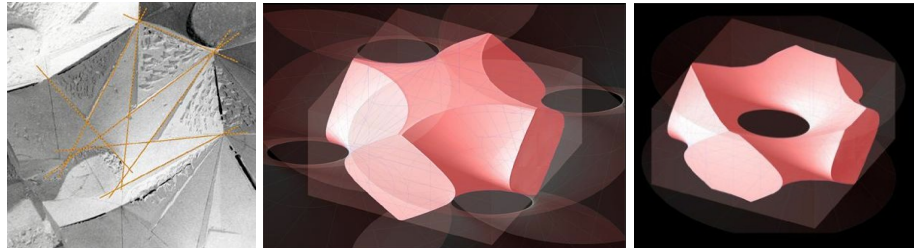
**Figure 4.4** The parametric model developed by Blackbox to control the exterior surface geometry and slabs heights for the White Magnolia's main tower (source: www.som.com).



**Figure 4.5** The 2005 Serpentine Pavilion designed by Álvaro Siza Vieira and Eduardo Souto de Moura in collaboration with ARUP (source: <http://www.serpentinegallery.org/architecture.html>).

When some tasks require custom software, AGU tries to extend existing CAD tools and analysis software with additional components and, when that is not possible, they develop new tools. AGU main focus is not to develop new technology but to reuse software as much as possible, with the objective of accelerating the design process (Santos et al., 2012). Projects have demonstrated working processes focused on capture and development of rule based systems using scripts. Their role includes the production of 3D models, interaction with analyses software and the production of detailed information for the manufacture of parts. The group's primary concern is to find solutions, rather than interpret and implement problems proposed by architects (Hudson, 2012).

Mark Burry is an example of an independent consultant that has been working on the continued construction of the unfinished design of Sagrada Família in Barcelona. Burry has been involved in parameterizing the geometric methods of Antonio Gaudi. The models are used to find solutions, by exploring and adjusting parameters, to find configurations that fit partially completed elements (Hudson, 2012). The model is then used to produce information to drive Computer Numerically Controlled (CNC) machines for fabrication.



**Figure 4.6** Example of Burry's Work in Sagrada Família, Barcelona. Detail of the lateral nave window based on Gaudi's original 1:10 scale plaster models that were reconfigured geometrically with precision and transformed into digital models (source: <http://sagradafamilia.sial.mit.edu.au/>).

## 4.2 Case study: The National Swimming Centre, the Water Cube

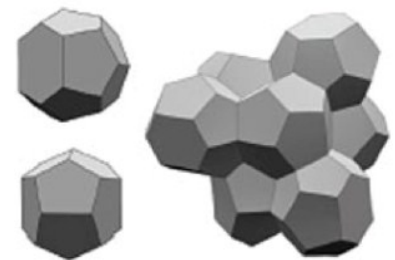


**Figure 4.7** Night view of the Water Cube (source: [http://90.146.8.18/en/archives/festival\\_archive/festival\\_catalogs/festival\\_artikel.asp?iProjectID=8672](http://90.146.8.18/en/archives/festival_archive/festival_catalogs/festival_artikel.asp?iProjectID=8672)).

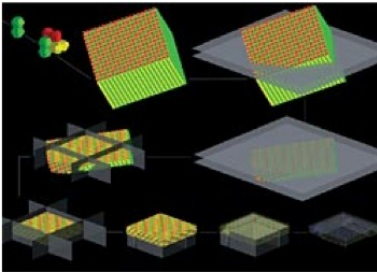
The National Swimming Centre, known as Water Cube, is the product of a competition entry by PTW Architects, China State Construction and Engineering Corporation (CSCEC), from Beijing, and ARUP. As previously referred, ARUP is a group where Generative Design already plays a crucial role. Next, we describe the development of the project Water Cube, when and how Generative Design was integrated in the design process as well as the various benefits it provided, both in terms of time and cost compared to a traditional approach to modeling.

The first challenge of the project was to decide the form of the structure and the look of the cladding. The team investigated different ideas and found the solution in Professor Denis Weaire and his assistant Dr Robert Phelan theory: the “Weaire-Phelan foam”, that derives from the structure of water bubbles in the state of aggregation found in foam. This solution divides space into cells of equal size (volume) with the least surface between them and without leaving any empty space. This foam captivated the ARUP’s designers and structural engineers due to its geometry be highly repetitive, regular and buildable, composed of repetitive units, but, at the same time, with a totally random appearance when viewed at an arbitrary angle (Eastman et al., 2008). This solution allowed them to construct a building in which the structure, the facade, and the roof that define the architectural space are a unique and continuous element.

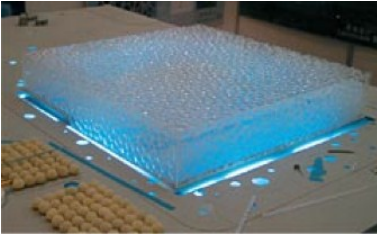
The approach for constructing the building, which led the process of geometry formation, started with the visualization of an infinite array of foam oriented in a particular way, creating a prism with a random appearance. Then, they carved out a block with the same external size of the stadium and with its basic shape, a box, by slicing that prism with two horizontal planes



**Figure 4.8** “Weaire-Phelan foam” (source: Carfrae, 2007).



**Figure 4.9** Schematic illustration of the geometry formation (source: Carfrae, 2007).



**Figure 4.10** The physical model used in the competition (source: Carfrae, 2007).

and four vertical planes. The major internal volumes were removed from the interior of that box and by converting the model to a wire frame, the geometry of the structure was defined (Carfrae, 2007). This structure contains more than 22000 steel beams and 12000 nodes which weigh tons which would then present a challenge for its optimization.

The project was developed in two main stages with different foci and concerns: competition and design development.

During the competition, because of the obvious time limitations due to deadlines, the team developed a large portion of the design but the main issue was the development of a method to produce a 3D model and the drawings for the presentation. They applied a scripting-based representation to model a wire-frame in order to provide the 3D model of the structure. The competition rules called for a physical model which was used to convey the project to the judges, so the team decided to build an accurate model composed of all the tubes, nodes and thousands of different cladding panels (Carfrae, 2007). The model of this complex structure was created by exporting the 3D model to a stereolithography (STL) file which was used to make a rapid prototype model.

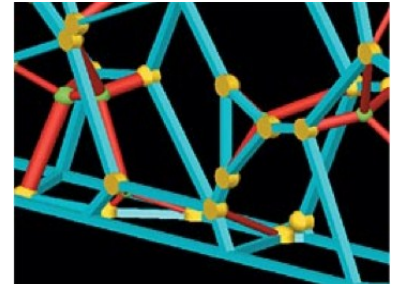
During the competition the team was focused on the model and didn't have enough time to analyze the structure. Thus, in the beginning of the design development, they faced a difficult question: would their concept work? (Carfrae, 2007). To prove that ARUP's geometry worked, CSCEC used a script implementing the Weire and Phelan solution, consequently ARUP could progress with their design and scripts. They used their wire-frame model in order to analyze and optimize the structure and developed scripts to export the analyzed and optimized model to several types of files containing geometric and structural data and also drawing files to allow communication with ARUP's partners. The scripts were also used to create a complete and accurate 3D model which allowed them to create the model in different representations such as surfaces, solids or structural elements as needed. The detailed drawings for construction and schedules were produced automatically from the 3D model. In the end, they had created a system that would take less than an week to generate the model and all the drawings even when a major change to the building size or shape was necessary (Carfrae, 2007).

Regarding the fabrication and construction, ARUP proposed prefabrication, idea that was rejected by the client. Thus, a process that in other countries would be done by applying CNC machinery to shorten the fabrication time was here done manually.

As previously referred, the structural design was one of the greatest challenges in the project. The Water Cube is a complex building composed of countless structural members. Thus, analyzing the structure and make the necessary changes, as many times as needed, would be an extremely accurate process as well as very time consuming. To overcome these issues, ARUP developed a script to automatically select the member sizes through an optimization process. The optimization code was written using a genetic algorithm and the alternative solutions were evaluated in Strand7 Software. This algorithm determined the size of all the beams which had to meet different law requirements at different points on each beam for almost two hundred of load combinations scenarios, which resulted in almost 272 million of design constraints (Eastman et al., 2008). The algorithm iteratively checked the entire structure by allowing the team to test different design configurations and receive feedback within 25 minutes. The possibility of linking this script to the analytical software allowed ARUP to propagate a change in any member of the structure to all the related elements. Thus, it allowed the creation of a complex structure that could be structurally optimized and which, compared with traditional approaches to modeling, saved millions of dollars on design costs and approximately a year and a half in the design and documentation (Stocking, 2009).



**Figure 4.11** The structure of the Water Cube under construction (source: Carfrae, 2007).



**Figure 4.12** Detail of the structural model (Carfrae, 2007).









**PART II : AN ALGORITHMIC APPROACH  
TO DESIGN**



## 5 INTRODUCTION

Design processes are characterized by change from the very beginning. The change may arise from, e.g., uncertain design intents or detail's and requirements' growth. The traditional use of CAD tools requires too much time and effort to modify models or even treat them as disposable when changes are needed.

Design tools must embrace change. Generative Design, in particular an algorithmic approach, is an efficient, rigorous, controllable and flexible tool. It allows, with reduced effort, the fast production of several different models to explore design variations addressing different types of change, thus it is aligned with the design process' needs.

This dissertation argues for one main point: integrating Generative Design as a new stage in the design process dramatically simplifies the handling of changes. In particular, we propose an algorithmic approach to design that overcomes the limitations of the traditional approach.

Our approach requires the formalization of the design intents. For this formalization we propose a programming-based representation that is defined parametrically.

Contrary to other GD approaches, such as Shape Grammars or L-Systems, our approach is, to a large extent, predictable, because, from the values of the parameters, one can predict the solutions that will be generated. Thus, the designer can rigorously specify his intended design, while allowing the parameters to express possible variations of that design. Many different designs can then be quickly generated by the computer using specific values for those parameters.

This symbiosis between the capabilities of the designer and the computer is predicated on communication. This communication requires a language that is understandable both by the computer and the designer, and that is the role of programming languages. In fact, our approach describes the design as a program written in a formal programming language.

To evaluate the proposed approach, we made an experiment: we formalized the design of a complex building while, at the same time, we planned for supporting as many changes as possible. To be more realistic, the case

study was focused on a building that is still under construction: MVRDV's Market Hall.

The design of the Market Hall is already finished, thus we simulated a model being implemented during the project development, starting from the concept that structured the design. This simulation required an abstraction from the final design: the aim was not to capture Market Hall's specific geometry but the main intents and priorities. This abstraction allowed us to generate several different models, including the actual Market Hall. The design process was simulated by analyzing different types of information about the project, namely textual descriptions, drawings, and photo-realistic images. After understanding the inherent generative logic it was required a translation into formal descriptions in order to implement it in the formal language.

Our approach may be also beneficial during the construction of the concept, as it allows the designers to very quickly produce several simple models to explore different concept alternatives. The design, and consequently the model, evolves from the chosen concept. Unfortunately, in the case of the Market Hall, there was no information about the previous concepts that were conceived. As a result, the simulation started from the final concept for the design.

The second part of this dissertation provides the modeling strategy and an overview of the complete model (chapter 6), a more detailed description of the strategies and tasks needed to define the model (chapter 7), an analysis of the modeling process and generative strategies, whose lessons might be useful to solve or implement similar problems (chapter 8) and the evaluation of the proposed approach (chapter 9).

## 6 MODELING STRATEGY

### 6.1 Modeling strategy

Architectural design is an activity that deals with the designer's self-imposed goals and constraints as well as with externally imposed constraints, e.g. site, costs, client' wishes, and so forth (Kalay, 2004). The design process commonly starts with an analysis of the externally imposed constraints, through which a designer defines a concept to structure the design. The modeling process we used followed the same strategy. Before starting the model construction, the site and exterior constraints are analyzed and the designers' process, concepts and intents are clarified. Different design processes, such as buildings designed primarily from the inside out or from the opposite direction, have consequences on the modeling process, particularly in the order of modeling operations.

The design process can usually be decomposed into different design phases, from conceptual to detailed design. Similarly, the design artifact can also be decomposed into its constituent elements which are developed on and along different design phases. Thus, our modeling process can also be decomposed according to two perspectives: (1) to match the design process decomposition in order to progressively increase the detail and definition of the model, and (2) to match the design artifact decomposition, dividing the model into the relevant elements that it represents. Based on the decomposition of the design process the model is characterized by a hierarchic structure with one-directional relationships between elements which detail increases along the model construction. Each element of the model, attending to the relationships with other elements, represents one design problem. Elements are tackled individually requiring a fragmented and iterative process in order to reduce the complexity of the tasks. To capture the underlying logic of each element we needed to: identify rules, identify constraints, identify and establish relationships with other elements, identify repeatable tasks, fix intents and define priorities, generalize, and consider the existence of exceptional cases.

In order to evaluate the ability of the proposed approach to handle change we gave special attention to the control of the model, namely to the choice of the parameters needed to define the different elements. The three-dimensional solution of each problem was achieved with the definition of

procedures, i.e. translating the formalization of the design, the algorithms, in a programming language.

## **6.2 Application to the case study**

Given that this formalization is an additional step in the design process, it is important to evaluate if the time spent in this step allows a significant reduction in the time and effort needed for handling changes, particularly in the more advanced phases of the project.

In order to evaluate our proposed approach to design, we chose a sufficiently complex design, the Market Hall building, and we executed the modeling process until we obtained a very detailed model of the building. We will now describe the application of the modeling process to this building.

The Market Hall can be divided into two parts: the aboveground part and the underground. The underground is dependent on the aboveground and is also constrained by the shape of the available land and by the structural needs. Being an early stage model, the basement was not modeled, as it can only be considered and developed later in the design process.

The Market Hall is located in a square without adjacent buildings, enabling a context-free modeling without imposing site, shape or dimension constraints. The building is characterized by an unconventional shape that creates a large hall, closed by two glass facades. This bent shape was the starting point for the design process, as it constraints both the building elements and the interior spaces. However, instead of capturing the exact geometry of the building, we captured the underlying ideas of the design, so that it would be possible to generate several different instances, including the Market Hall itself. To this end, we formalized and abstracted each design idea, and we implemented it in a programming language. This means that all shapes, relations, repetitions and variations, and rules identified in the Market Hall were formalized. At the same time, all these formalizations were generalized and parameterized to increase its applicability.

It is important to note that some of the rules that are present in the Market Hall have exceptions. These exceptions arise from the design of the interior rooms of the building. To give an example, we considered that there are balconies between all vertical walls of the building, and in all floors, except the ground floor. However, Market Hall has duplexes and, in those cases, there is just one balcony in the lower level. As the building was designed primarily from the outside these exceptions might be formalized later in the design process, with the definition of the interior of the building.





**Figure 6.1** Photo-realistic image of the Market Hall (source: <http://www.architizer.com>).



**Figure 6.2** Photo-realistic image of the Market Hall (source: <http://www.mvrdv.nl>).

When we analyze the building, it becomes obvious that there are strong dependencies between its elements. For example, the cover slab can only be represented after we decide on the overall shape of the building. As another example, we can consider the location of the windows that can only be decided after we know the location of the slabs and vertical walls. Similar dependencies can be established between many other elements. As a result, our design process must establish an order between the modeling of different elements.

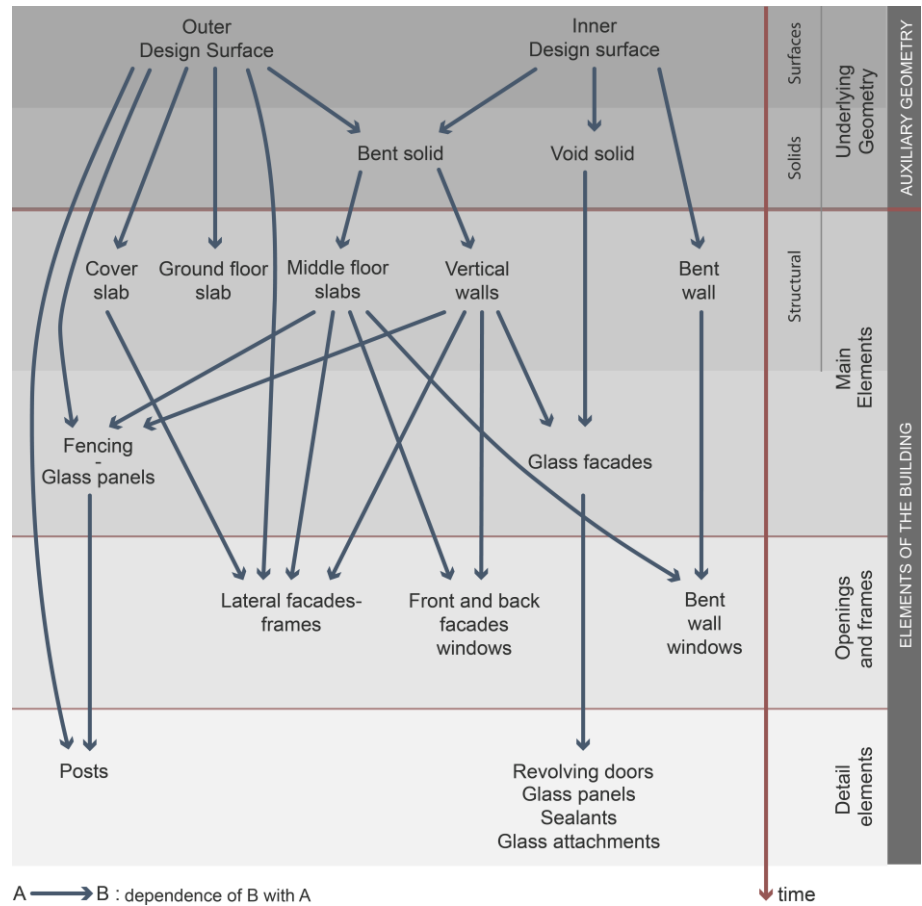
The design process we used to construct the model can be divided into four phases: (1) formalization of the shape of the building, (2) modeling of the main elements of the building, (3) definition of the openings and modeling of the frames, and lastly (4) the detail elements.

In the first phase, we defined the underlying geometry that captures the main concepts of the overall shape of the building. This geometry was then used as input to the remaining phases of the design process.

The decomposition of the process into four phases, each one increasing the detail of the model, led to the existence of dependencies between elements, allowing the automatic propagation of changes. In Figure 6.3 (page 48), we present the hierarchy of the model according to the four phases of the process, and the dependencies between elements. As can be seen, all the elements of the building depend on its overall shape, which we formalized in

the first phase of the modeling process. Additionally, it also shows two decompositions, of the modeling process that matches the one of the design process, in order to progressively increase the detail and definition of the model, and to match the one of the design artifact, decomposing the model into the relevant elements that it represents.

In the following pages we describe the main operations and strategies adopted to construct the model in accordance to the four phases we have just described.



**Figure 6.3** Phases of the design process and hierarchy of the model.

## Phase 1 – Underlying geometry

At the root of the process we formalize the underlying geometry of the overall shape of the building. The Market Hall is a longitudinal building that can be decomposed into two shapes: a bent shape and a shape that represents the hall that will be called “void”.

The underlying geometry can also be decomposed into surfaces and solids. The outer and inner design surfaces are the unconventional surfaces that dictate the shape of the building. They can be defined by lofting a set of planar splines. Then, the bent solid and the void solid are defined by joining the design surfaces with planar surfaces defined by their edges and by auxiliary line segments. Design surfaces guide the creation of solids, i.e. their shape controls the shapes of the solids, which together influence the elements of the building.

Although the original shape of the Market Hall is uniform along the length of the building, we decided to further generalize it by allowing the shape to change along that length. As we will see, this decision provides much greater control over the final shape of the building, while preserving the ability to faithfully reproduce the current shape of the Market Hall.

## Phase 2 – Main elements

The fundamental elements of the model were represented in the second phase. These elements are directly influenced by the overall shape of the building and their geometry is created from the underlying geometry defined in the first phase of the modeling process.

First, the “structural” elements were defined, such as: the vertical walls, the inner bent wall and the slabs. Lastly, the glass facades and the fencing panels were represented, already informed by the methods and parameters used to define the first set of elements.

The vertical walls, the middle slabs and the glass walls are defined by boxes intersecting the solids defined in the first phase. The inner bent wall is defined by assigning structural depth to the inner design surface, i.e., by offsetting it inwards and by creating a solid with the two surfaces. The cover slab is defined by intersecting a box with a bent solid defined by assigning structural depth to the outer design surface. The ground floor slab is defined by creating a planar surface, with the longitudinal edges of the bent solid and with two auxiliary line segments, and by extruding it downwards. Lastly, the fencing panels are defined by assigning structural depth to the outer design

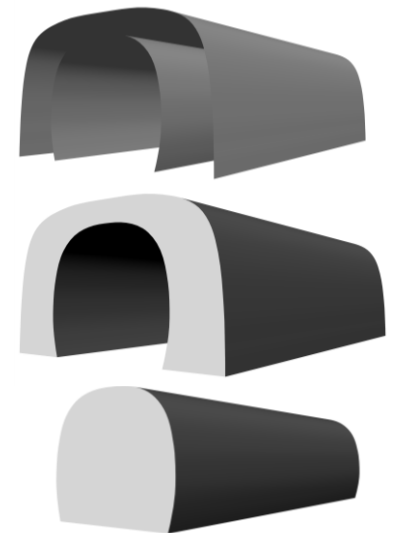


Figure 6.4 Top: Photo-realistic image (source: <http://www.mvrdv.nl>). Middle: design surfaces. Down: bent solid and void solid.

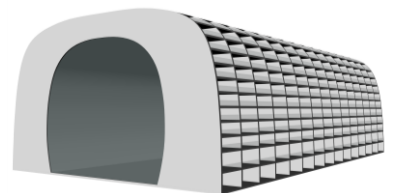


Figure 6.5 Second phase of the model.

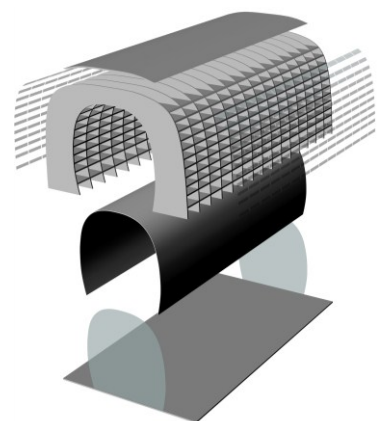


Figure 6.6 Scheme with the main elements of the model.

surface and by intersecting two sets of boxes with that solid, one horizontal with the height of the fences and one vertical with the lengths of the balconies.

### ***Phase 3 – Openings and frames***

The windows of the building were represented in the third phase. To this end, it was crucial to simultaneously analyze the interior and exterior walls of the building.

In the lateral facades, frames were added to the openings defined by the walls and slabs. Two main strategies were developed: (1) to find the anchor points of the frames ensuring that they are the boundary between the interior and the exterior of the building, and (2) to compose the facade, placing different types of frames on each floor.

The front and back facades and the inner bent wall were represented in the second phase of the model as continuous solids, thus, besides the addition of frames, their geometry was modified through the opening of the rough openings.

In both strategies, the main task was to find the anchor points to place the frames and to place a set of boxes, which when subtracted from the solids, represented the rough openings.

The front and back facades are the first and last vertical walls. To find the anchor points we needed to define two alignment curves for each facade and we needed a strategy to locate windows between these limits, with the exterior aesthetic as priority.

The inner bent wall is a longitudinal element, along the y-axis, that is in contact with the interior walls and the vertical “structural” walls. The priority of this phase of the design was to control the position of the interior walls, thus the position of the windows along the length of the building is user-defined, given their specific location on the y-axis. The anchor points, belonging to a bent surface, were found by intersecting vertical and horizontal surfaces with that bent surface. The windows were oriented according to the curvature of the wall by defining vectors, a normal vector to one surface of the wall and one tangent vector to a horizontal curve belonging to that surface, which are both calculated in those anchor points.

In the second phase of the modeling process, the fact that the building is bent did not influence the operations we used. In this phase, as shown above, we needed to consider the particularities arising from this specific



**Figure 6.7** Photo-realistic image (source: <http://www.mvrdv.nl>).



**Figure 6.8** Third phase of the model.

characteristic of the building. Considering a bent shape, uniform along the length of the building, the slabs of the building are not vertically aligned and, when the shape varies along its length they are curved segments. Thus, to calculate the location of the frames we needed to analyze the curved boundaries of the slabs, the bent boundaries of the front and back facades and consider the bent of the inner bent wall.

#### **Phase 4 – Detail elements**

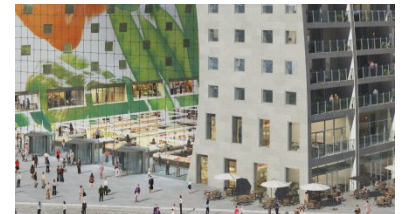
The last phase of the model was concerned with the inclusion of elements with relevance to the exterior aesthetic of the building, by further detailing some of the elements that were previously defined.

The glass facades were detailed by representing the main elements of a suspended cable-net facade, namely the glass panels, the sealants and the glass attachments. In the previous phases these facades were represented by two solids. In this phase, we replaced them with new elements that follow a panelization logic. To this end, we created a grid of parallelepipeds which we subtracted from the two solids defining the glass panels and which we intersected with the same solids to define the sealants. The glass attachments were added in the intersection points of the horizontal and vertical axis of the grid by selecting the points that were inside or in the boundary of a section of the facade. The revolving doors were also added to these facades, providing access between the hall of the building and the exterior, in user-defined locations along both facades.

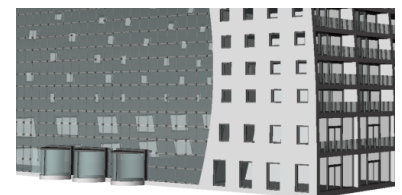
The modeling process ended with the representation of the fencing posts. These elements were defined according to the curvature of the building. To this end, we developed a methodology to orient their sections according to that curvature, the same of the glass panels of the fencing, and to dispose several elements along the surface. Continuous solids were defined around the overall shape of the building by sweeping these sections and, by subtracting a set of horizontal boxes with the height of the fences, the posts belonging to the different floors were defined.

#### **Instances of the model**

As was previously referred, instead of capturing the exact geometry of the Market Hall, we captured the underlying ideas of the design. The advantage of our approach is that it now becomes possible to generate an infinite number of different instances of that design, including one for the Market



**Figure 6.9** The facade of the building and the fencing posts. Detail of a photo-realistic image (source: <http://www.mvrdv.nl>).

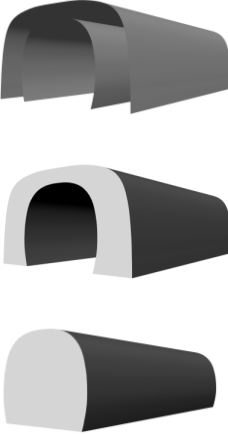

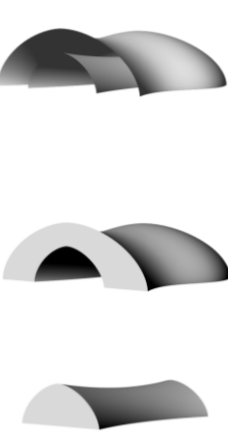
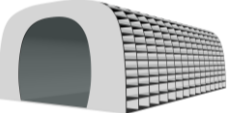
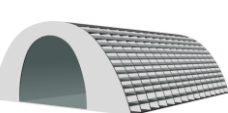
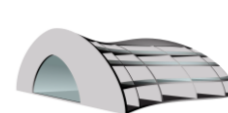

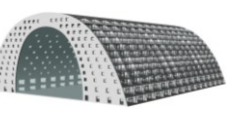


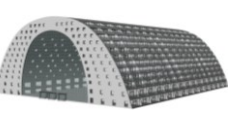



**Figure 6.10** Detail of the last phase of the model.



**Figure 6.11** Last phase of the model.

Hall itself. In Table 6.1, we present a few examples of possible variations that can be easily created by exploring the parameters of the developed model. Comparing the Market Hall with buildings A and B, it is also visible the effect of the formalization of the overall shape of the building that allows quite different shapes although still preserving some of the characteristics of the Market Hall. The model is highly adaptive, it allows the automatic propagation of changes between elements of the same model and along its different phases but also to reuse the underlying logic that we captured from the Market Hall to produce many variants.

PHASES	Market Hall	A	B
Phase 1 Underlying geometry			
Phase 2 Main elements			
Phase 3 Openings and frames			
Phase 4 Detail elements			

**Table 6.1** Instances of the model, three examples. Left: Market Hall. Middle and Right: Variations.

## 7 DEFINITION OF THE MODEL

In the last chapter we presented the modeling strategy we followed to construct the model, the hierarchy of the elements and their relationships, and a brief description of the main operations and strategies adopted. These were divided into four groups corresponding to the four phases of the design process, with different levels of detail.

In this section we present a more detailed description of the modeling tasks, including some additional strategies that we consider relevant. The description follows the four phases of the model: (1) Underlying geometry, (2) Main elements, (3) Openings and frames, and (4) Detail elements.

### 7.1 Phase 1 – Underlying Geometry

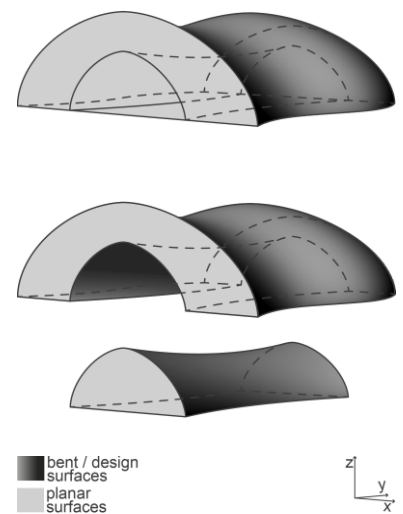
At the root of the modeling process we formalized the underlying geometry of the overall shape of the building. Its aboveground can be decomposed into two parts: a central void and a mass. The mass corresponds to the bent shape that is featured by the multi-functional program, conceptually developed to cover and involve the central void, a square serving as market during the day. This bent shape defines the shape of the central void. The building can be characterized by a longitudinal development, which we will consider to be the y-axis (Figure 7.1).

The underlying geometry we needed is composed of surfaces and solids, namely: design surfaces (the bent surfaces) and the Bent and Void solids. These solids are composed of design surfaces and planar surfaces as we illustrate in Figure 7.1. The methods we used to define the underlying geometry are described below.

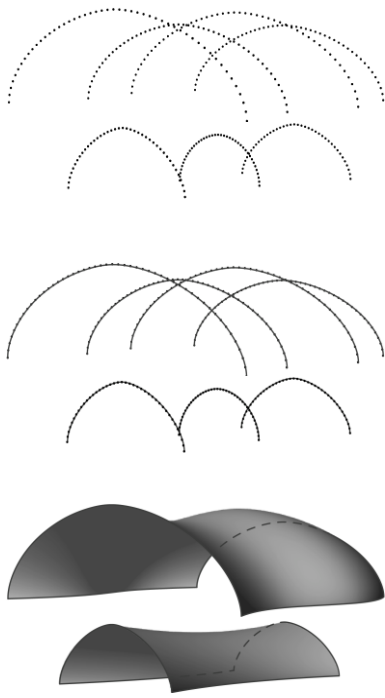
#### 7.1.1 Design surfaces

To formalize the overall shape of the building we needed two design surfaces: the inner and the outer design surfaces.

They can be defined by lofting a set of parametrically controlled planar curves lying in planes parallel to xz and along the y-axis.



**Figure 7.1** Aboveground of the building. Middle: Bent solid. Down: Void solid.



**Figure 7.2** Design surfaces: outer and inner. Top: planar points. Middle: planar curves. Down: design surfaces.

Although the original shape of the Market Hall is uniform along the length of the building, we decided to further generalize it by allowing the shape to change along that length, i.e., by using different curves to define the underlying geometry. This abstraction provides much greater control over the final shape of the building, while preserving the ability to faithfully reproduce the current shape of the Market Hall.

For each curve with a different design we defined a procedure that computes points belonging to the curve, so that each curve can be produced by tracing a spline through these points. These procedures may contain points manually defined or computed by the parametric definition of well-known mathematical curves. For example, the outer design surface we illustrate in Figure 7.2 was defined by using four curves with two different designs and we used the mathematical definition of a super ellipse to define both.

The parameters we use to shape a design surface are: a list with procedures and their corresponding parameters, a list with ordinates to locate the curves along the y-axis.

**7.1.2 Solids**

We created the solids by joining surfaces, which are defined separately. Two solids were defined to formalize the overall shape of the building: (1) the “Bent Solid”, and (2) the “Void Solid”. These solids are composed of design surfaces and planar surfaces, whose shape is determined by the design surfaces.

*1. Bent Solid*

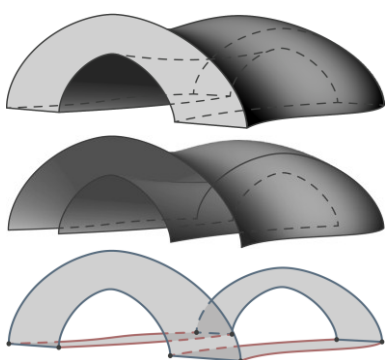
The Bent solid is composed of two bent surfaces (the outer and inner design surfaces) and four planar surfaces.

The outer and inner design surfaces are defined by lofting two sets of curves, as we explained above. As we illustrate in Figure 7.3, their edges together with four auxiliary line segments are used to define the geometry of the planar surfaces.

The parameters to shape this solid are just the ones needed to define the two design surfaces.

*2. Void Solid*

The Void solid is composed of four surfaces, the inner design surface and three planar surfaces.

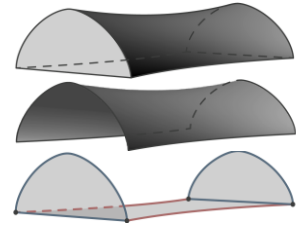


**Figure 7.3** Method to define the Bent Solid. Top: Bent Solid. Middle: Design surfaces. Down: Planar surfaces.



The inner design surface is defined by lofting a set of curves. As we illustrate in Figure 7.4, its edges together with two auxiliary line segments are used to define the geometry of the planar surfaces.

The parameters to shape this solid are just the ones needed to define one design surface.

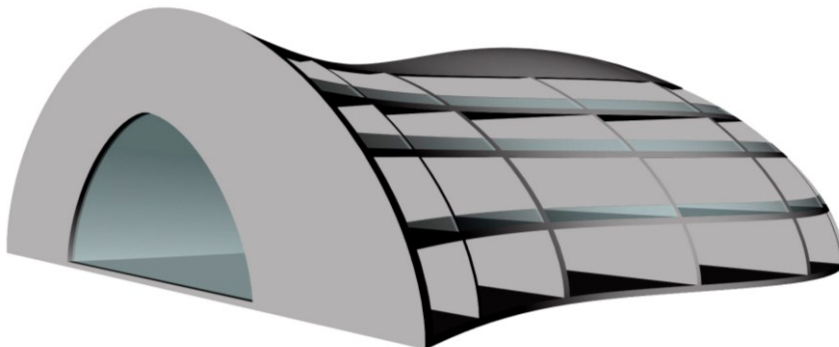


**Figure 7.4** Method to define the Void Solid. Top: Void Solid. Middle: Inner design surface. Down: Planar surfaces.

## 7.2 Phase 2 – Main Elements

The main elements of the model are: the vertical walls, the glass facades, the slabs, the inner bent wall, and the fencing panels of the balconies (Figure 7.5). The geometry of all these elements depends directly on the overall shape of the building, thus it is created from the underlying geometry defined to formalize it in the first phase of the modeling process.

In further stages of the modeling process the geometry of the main elements or their parameters are used to increase the detail of the model, to define further elements or to manipulate their geometry to add detail.



**Figure 7.5** Main elements of the model.

### 7.2.1 Vertical walls and glass facades

The vertical walls and the glass facades are disposed along the y-axis and they are both solids contained in the solids of the underlying geometry.

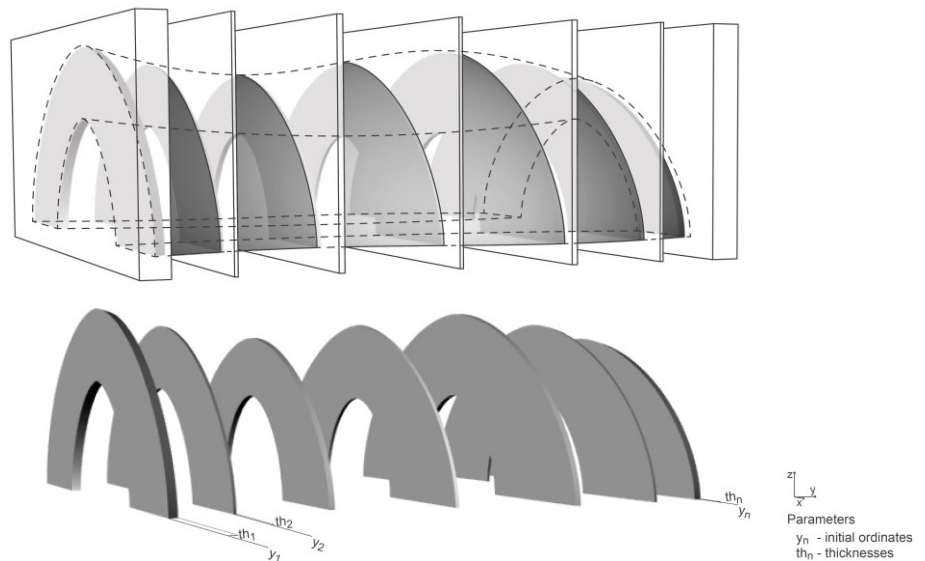
Initially, we defined the vertical walls and the glass facades with different procedures according to the established sequence for the definition of the elements. In a posterior phase of review, their similarity and the recurrent need for similar procedures along the modeling process encouraged the development of a common procedure by abstracting those previously defined. Thus, we defined a procedure, called “Vertical Elements”, which is reused in the definition of several elements that are contained in other elements and are disposed along the y-axis.

## 1. Vertical Elements

A set of boxes is positioned along the y-axis. We created the geometry of the vertical elements by intersecting these boxes with one or several solids.

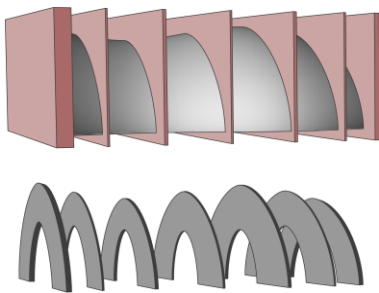
The parameters we use to shape the vertical elements are: the solid/s that contains the vertical elements and two lists, one with their positions and another with their thicknesses (Figure 7.6).

This procedure is reused in the definition of several elements with the appropriate solids to be used as input.



**Figure 7.6** Example of the production of vertical elements (vertical walls) with the Bent Solid as input.

### 7.2.1.1 Vertical walls



**Figure 7.7** Vertical Walls.

The vertical walls are defined by reusing the procedure “Vertical Elements” with the Bent Solid as input (Figure 7.7).

The walls of the Market Hall are equidistant. Accordingly, we developed an auxiliary procedure to calculate the positions of the walls, required by “Vertical Elements”, given a distance between them. However, the use of this parameter constrained these positions, so in the end, the parameter to locate these walls was kept equal to that of the “Vertical Elements” to allow the exploration of a larger solution space and to not over-constrain the model.

Besides this parameter, the parameters of the Bent Solid and the thicknesses of the walls are required (Figure 7.10, page 57).

### 7.2.1.2 Glass facades

The glass facades are defined by reusing the procedure “Vertical Elements” with the Void Solid as input (Figure 7.8).

To locate these elements, a list with their position could have been used but, instead, we constrained the position of these facades to the center of two vertical walls that are also the front and back facades of the building.

The parameters to shape the glass facades are the ones of the Void Solid, the parameters of the walls and the thickness of the glass facade (Figure 7.10).

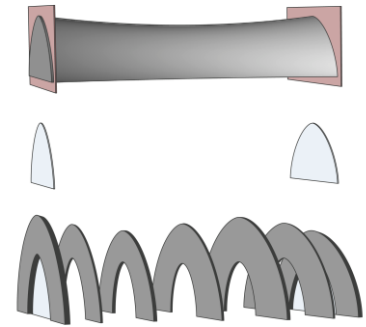


Figure 7.8 Glass facades.

### 7.2.2 Inner bent wall

The inner bent wall is the boundary between the hall of the building and its interior. We defined it by offsetting outwards the inner design surface, according to the thickness of the wall, and by creating a solid with the two surfaces (Figure 7.9).



Figure 7.9 Inner bent wall.

The parameters to shape the inner bent wall are the ones needed to define the inner design surface and the thickness of the wall (Figure 7.10).

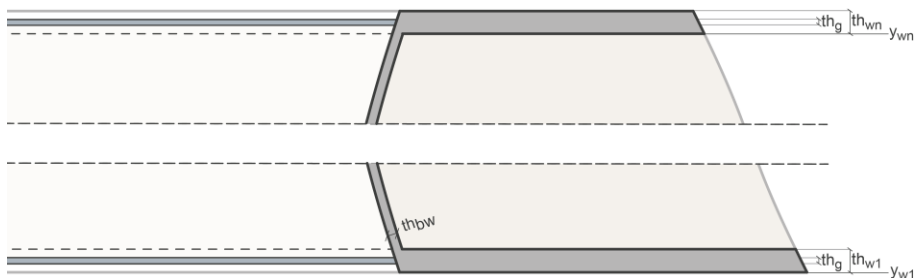


Figure 7.10 Partial plan of the building. Parameters of the vertical walls, glass facades and inner bent wall.

Parameters  
 Vertical walls (w):  
 $y_{ws}$  - initial ordinates  
 $th_{ws}$  - thicknesses  
 Glass (g) facades:  
 $th_g$  - thickness  
 Inner bent wall (bw):  
 $th_{bw}$  - thickness

### 7.2.3 Slabs (of the middle floors, cover and ground floor) and fencing panels

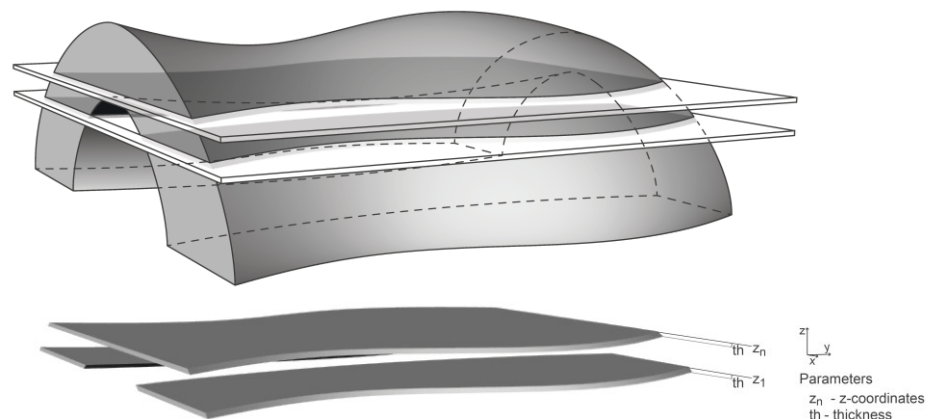
Initially, we defined the slabs and the fencing panels with different procedures, as we did for the vertical walls and glass facades. Due to the recurrent need of horizontal elements, i.e. solids contained in other solids and disposed along the height of the building, we developed a common procedure by abstracting those previously defined. Thus, we defined a procedure, called “Horizontal Elements”, which is reused in the definition of several elements.

### 1. Horizontal Elements

A set of boxes is positioned along the height of the building. By intersecting those boxes with one or several solids, the geometry of the horizontal elements is defined.

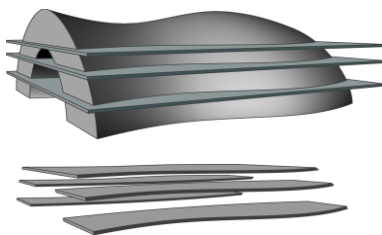
The parameters we use to shape the horizontal elements are: a solid/s in which the horizontal elements are contained, a list with the z-coordinates of each element and a thickness, which was considered identical on each invocation of the procedure (Figure 7.12).

This procedure is reused in the definition of several elements according to the appropriate solids to be used as input.



**Figure 7.12** Example of the production of horizontal elements (slabs) with the Bent Solid as input.

#### 7.2.3.1 Middle floor slabs



**Figure 7.13** Middle floor slabs.

The geometry but also the number of slabs depends on the geometry of the overall shape of the building as it can have two slabs, at right and left, or one continuous slab.

These slabs are defined by reusing the procedure “Horizontal Elements” with the Bent Solid as input (Figure 7.13).

To locate these slabs we kept the parameter required in “Horizontal Elements”, a list with the z-coordinates of the elements, as the Market Hall has different floor to ceiling heights along its height.

Besides this parameter, the parameter of the Bent Solid and the thickness of the middle slabs are required (Figure 7.19, page 60).

#### 7.2.3.2 Cover Slab

The cover slab is a bent solid, thus the upper floor has variable interior heights.

Its geometry is influenced by the outer design surface, thus, we first defined a bent solid by offsetting inwards that design surface, according to the thickness of the cover slab, and by creating a solid with the two surfaces. To create the cover slab we applied the procedure “Horizontal Elements”, i.e., by intersecting one box with that bent solid.

The parameters we use to define the cover slab are illustrated in Figure 7.19, page 60.

The shape of the building, defined by using different section curves along its length, creates a solid with several bents. Thus, to obtain a continuous cover slab along the length of the building it is crucial to control simultaneously its overall shape and the parameters we use to create the slabs, as we illustrate in Figure 7.15. On one hand, if these conditions are not verified, we obtain results that are different from the Market Hall, e.g. the creation of a last floor that is just partially covered; on the other hand these results may even suggest a different solution for the slab.

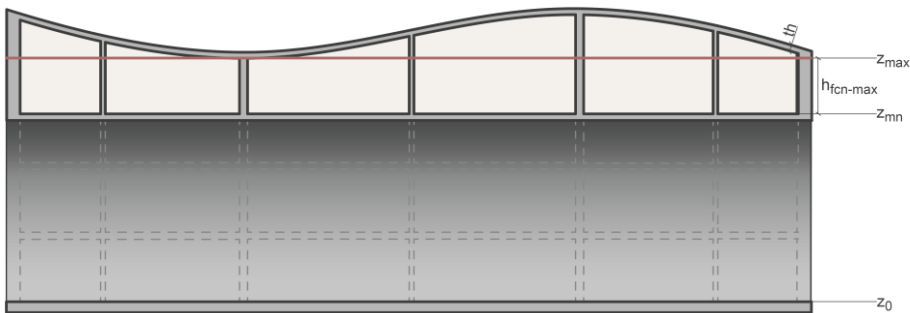


Figure 7.15 Cover slab: conditions to obtain a continuous slab.

### 7.2.3.3 Ground floor slab

The ground floor slab is the boundary between the basement and the aboveground part of the building. The underlying geometry formalized the aboveground, thus we did not defined this slab by intersecting boxes as we did for the remaining elements.

To create it, we first defined a planar surface with the longitudinal edges of the outer design surface and with two auxiliary line segments. By extruding that surface downwards, according to the thickness of the slab, the ground floor slab is defined (Figure 7.17).

The parameters to shape this element are the ones needed to define the outer design surface and the thickness of the slab (Figure 7.19, page 60).

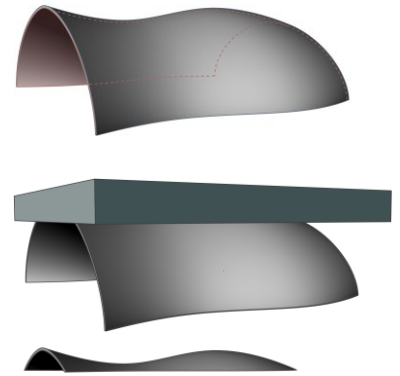


Figure 7.14 Cover Slab.

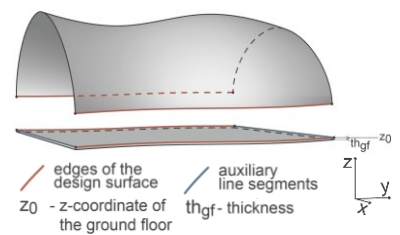
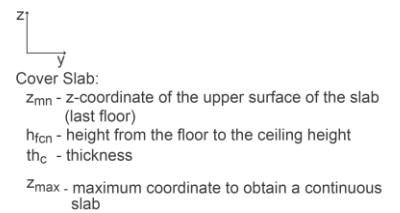


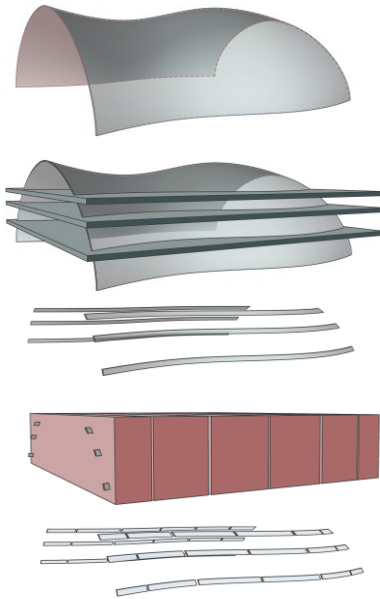
Figure 7.17 Ground floor slab.

### 7.2.3.4 Fencing panels

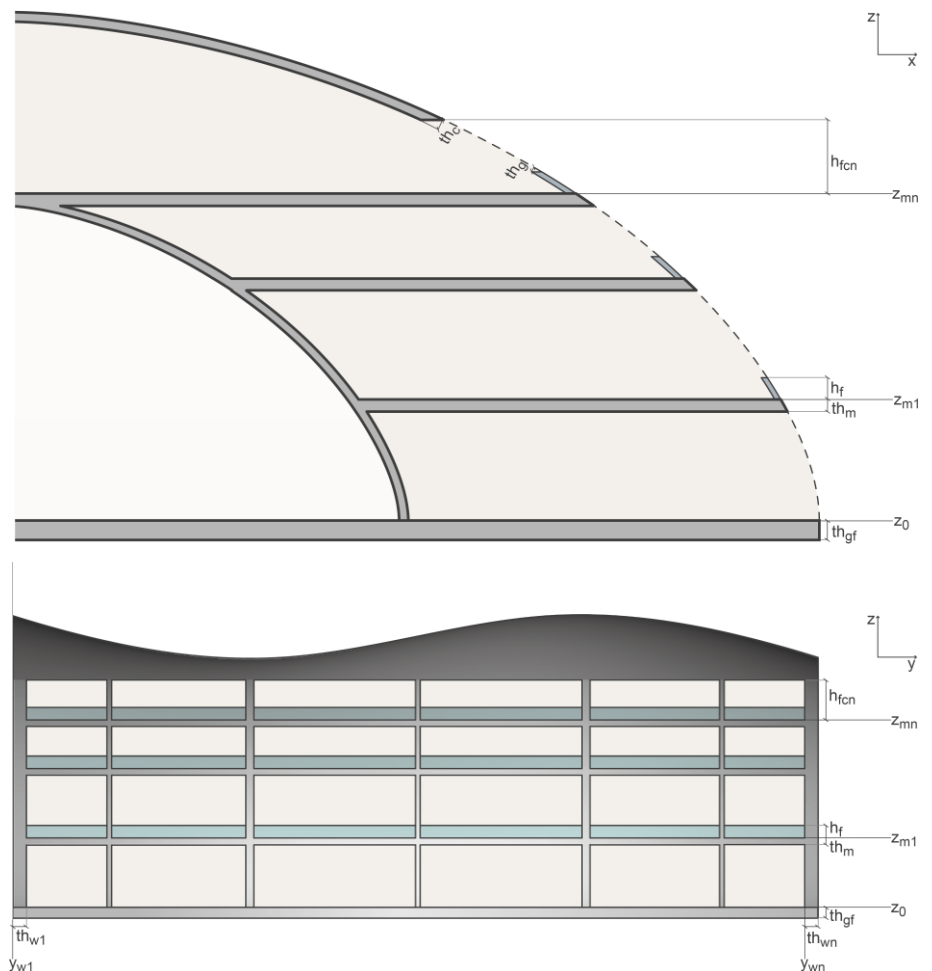
The fences of the balconies are composed of glass panels with posts, without top and bottom rails.

As the shape of the glass panels follows the shape of the outer design surface we defined a bent solid with the thickness of the glass by offsetting inwards that surface and by creating a solid with the two surfaces. Then, we defined one glass panel for each floor, by reusing the "Horizontal elements" procedure with that bent solid as input, i.e., by intersecting a set of horizontal boxes with the fencing height with that solid. One glass panel for each balcony is defined by reusing the "Vertical Elements" procedure with the continuous glass panels of each floor as input, i.e., by intersecting a set of vertical boxes with the width of balconies (Figure 7.18).

To shape the glass panels we needed the parameters we used to define the outer design surface and the remaining are illustrated in Figure 7.19.



**Figure 7.18** Fencing panels.



**Parameters**

**Middle (m) slabs:**

$z_m$  - z-coordinate of the upper surface  
 $th_m$  - thickness

**Cover Slab (c):**

$z_{mn} - z_m$  of the last floor  
 $h_{fc}$  - height from the floor to the ceiling height  
 $th_c$  - thickness

**Ground floor (gf) Slab:**

$th_{gf}$  - thickness

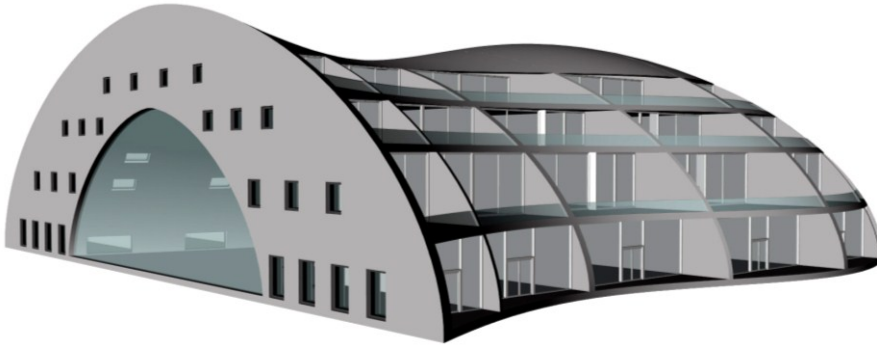
**Fencing glass (g) panels:**

$h_f$  - fencing height  
 $th_g$  - thickness of the glass  
 $y_{ws}$  - initial ordinates of the vertical walls  
 $th_{ws}$  - thicknesses of the vertical walls

**Figure 7.19** Parameters of the slabs and fencing panels.

### 7.3 Phase 3 – Openings and frames

The building has openings to provide views or accesses at the lateral facades, at the front and back facades and at the inner bent wall (Figure 7.20).



**Figure 7.20** Phase 3 of the model: representation of all the windows.

The frames of the lateral facades are needed to close openings defined by the slabs and the vertical walls. In the remaining elements we needed to define rules to open rough openings once the front and back facades and the bent wall were defined in the second phase of the model as continuous solids. The glass facades have also doors to provide the access between the exterior and the hall of the building, this is explained at section 7.4.1.

Before elaborating a strategy to represent the frames, it was crucial to analyze simultaneously the interior walls of the building and its exterior in order to understand the priorities that guided the design process. Although the interior spaces will not be developed in this model, we still need to know if the positions of the windows constraints the position of the interior walls or the opposite.

The three methodologies were applied only to the right half of the building, as we noticed that the windows are symmetric across an  $yz$  plane. This constraint allowed us the use of mirror operations or the calculus of symmetric points, which is more time- and cost-effective, both in the implementation and in the execution of the program.

In the following three sections we explain the methodologies, associated tasks, and considerations about each strategy.

#### **7.3.1 Frames of the lateral facades**

The frames of the lateral facades are the boundary between the interior rooms and the balconies. The frames are added to the openings defined by

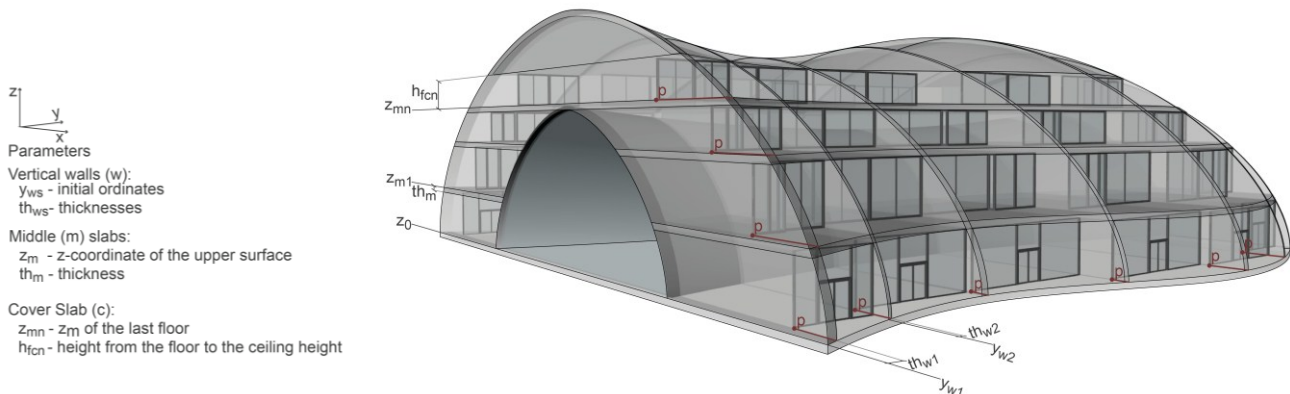
the walls and slabs of the building. The frames of the Market Hall are composed of different styles of frames together and are disposed along the length of the building.

The methodology we developed to represent these frames can be divided into two main tasks that are described below, namely: (1) to find the anchor points to place the frames, and (2) to shape the frames and to compose the facade placing their different types.

### 1. Anchor points

The position of the frames depends on the walls and slabs. Thus, the parameters we used to create those elements are used to compute the z-coordinates and y-coordinates of the anchor points of the frames as we can see in Figure 7.21.

The overall shape of the building, more specifically the outer design surface, constrains the shape of the walls and slabs and consequently the position of these windows, particularly their abscissas. In a bent shape, such as the one of Market Hall, the slabs of the building are not vertically aligned, thus the abscissas vary from floor to floor. Moreover, in a building defined by using different section curves along its length, the boundaries of the slabs are curved. Consequently, the abscissas of the frames vary not only from floor to floor but also from balcony to balcony on the same floor as we illustrate in Figure 7.21.



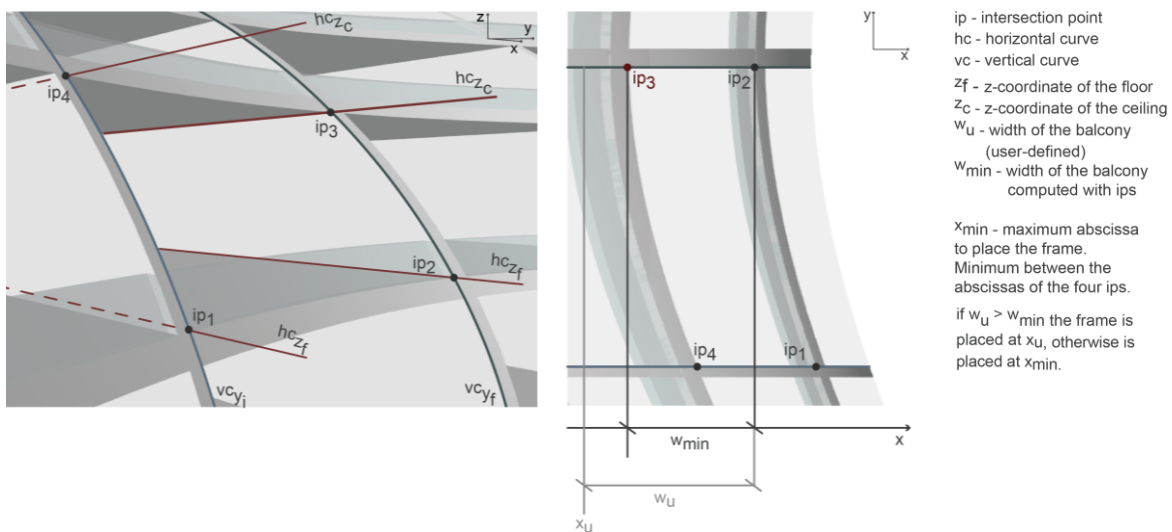
**Figure 7.21** Frames of the lateral facades.

To obtain these abscissas we developed a method that analyzes the abscissas of four points and we select the inwards abscissas in order to ensure the non-existence of any opening. For each balcony, we shaped two vertical surfaces, parallel to the plane xz, at the extreme ordinates in which the balconies are defined. Their intersection with the outer design surface defines two curves. Each of these curves is intersected with two horizontal line segments at the floor and ceiling heights, defining four intersection points (Figure 7.22, on the left). The abscissas of the intersection points are

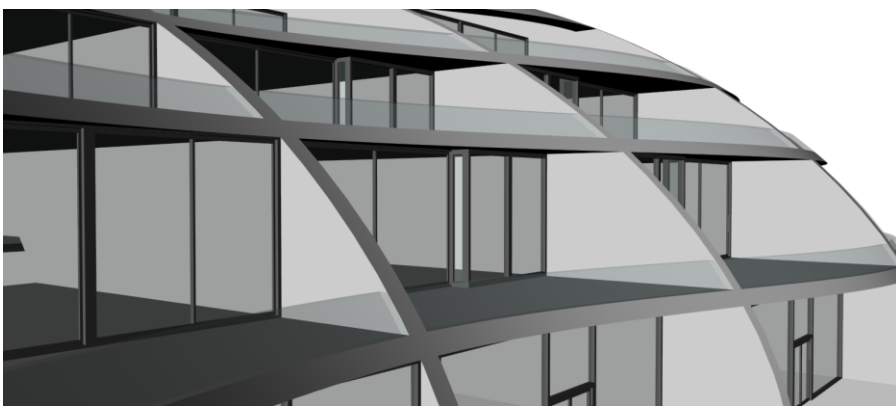


compared; the inwards abscissa ensures the non-existence of openings and consequently leads to the minimum possible width of the balcony.

To provide the control of the width of the balconies we developed a rule by adding a control parameter: the balconies' width. We calculate the width of the balcony by using the inwards abscissa and compare it with the user-defined width: if the assigned value is bigger than the minimum width we calculate a new abscissa, otherwise the abscissa we use to place the frame is the previously found by comparing the four intersection points (Figure 7.22, on the right). In Figure 7.23 we show an example in which the user-defined width was smaller than the width that ensured the non-existence of openings.



**Figure 7.22** Lateral frames: method to calculate the abscissas of the anchor points.



**Figure 7.23** Partial representation of the lateral frames.

## 2. Frames and composition of the facade

In Market Hall there are different styles of frames on each balcony which are adjacent, thus defining frames composed of different styles.

The modeling process of the composed frames was divided into three different steps: first, we defined the basic elements that are repeated in all

styles, then we defined the different styles by fixing relationships between the repeating elements and, lastly, we defined four types of composed frames by fixing relationships between styles.

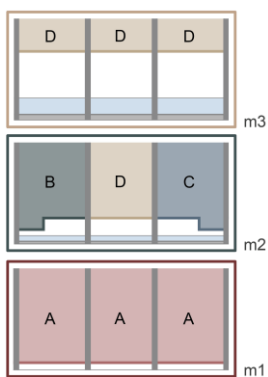
Each composed frame was defined according to an anchor point, the total length and the total height. The dimension of the frames depend on the distance between the walls and the slabs, thus we compute them using the parameters we used to define these elements. We defined the composed frames with the rightmost abscissa. This abscissa ensures, independently on the type of composed frame, the non-existence of any opening.

In Figure 7.28 (page 65) we illustrate the main parameters and the relationships we established between styles of frames, such as symmetries or proportions, to define four types of composed frames.

In the Market Hall, these different composed frames are arranged according to a sequence that is repeated horizontally and used in different floors.

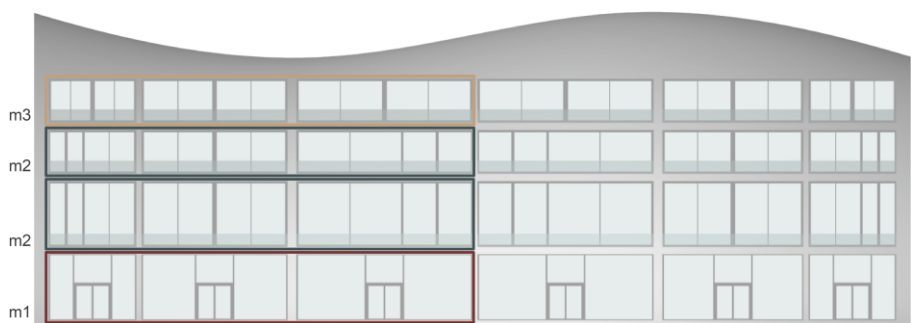
To compose the facades we defined sequences of types of frames in order to create modules. In Figure 7.24 we show three examples of modules that we captured and adapted from the Market Hall. In Figure 7.25, we illustrate the application of those modules in a building: the modules are repeated once, horizontally, and the module 2 is used in two different floors. The method we developed also allows to compose a facade in which there is no horizontal neither vertical repetition of types of composed frames.

Lastly, we developed a procedure to compute the parameters of the frames and apply them to the linear sequences of composed frames, one for each floor.

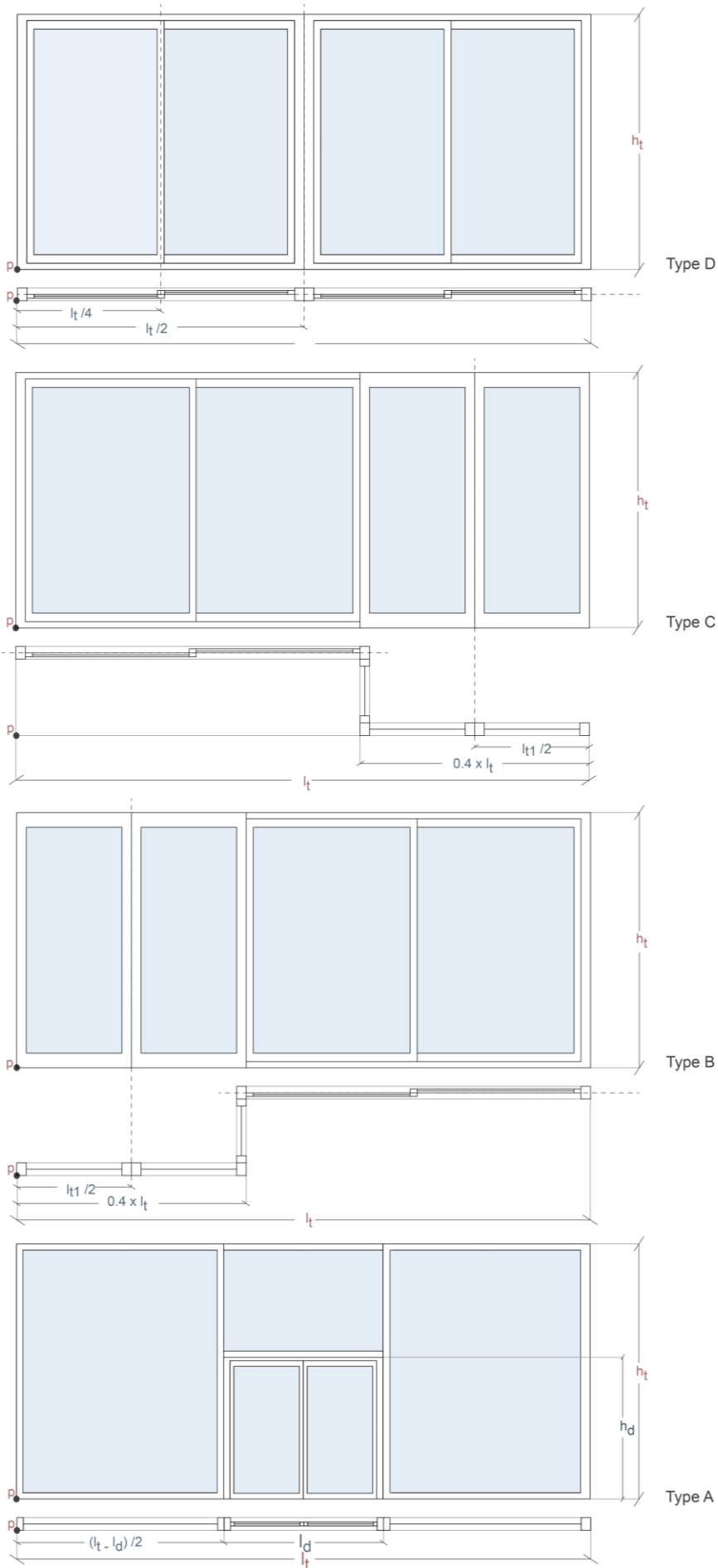


m - module  
A, B, C, D - types of composed frames

**Figure 7.24** Three examples of modules.



**Figure 7.25** Application of the modules.



$h_t$  - total height of the frame  
 $l_t$  - total length of the frame  
 $h_d$  - height of the door  
 $l_d$  - length of the door

**Figure 7.28** Types of composed frames.

### 7.3.2 Windows of the front and back facades

The front and back facades of the building are simply vertical walls with windows on each floor. By analyzing the elevations of the Market Hall we concluded that the exterior aesthetic of the building was prioritized.

The position of the windows is constrained by the shape of the facades, which is constrained by their position and by the overall shape of the building.

The main strategy we developed to represent these windows is concerned with the definition of the anchor points that locate the frames and the boxes, which subtracted from the vertical walls define the rough openings.

#### 1. Anchor points

The positions of the frames are constrained by the position of the facades and once there are windows in different floors it is necessary to control their position along the height of the building.

We computed the ordinates of the anchor points with the parameters we used to define these facades and the thickness of the frames. We constrained them to be centered with the facades.

To compute their position along the height of the building we used the parameters of the slabs and a list with the heights from the floor to the base level of the windows, one value for each floor (Figure 7.29).

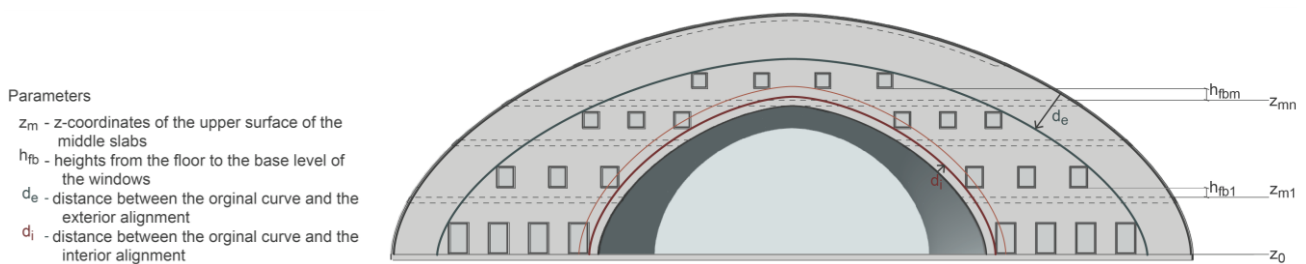


Figure 7.29 Method to find the abscissas and the parameters.

The abscissas of the anchor are directly constrained by the shape of the facades, in particular by the outer and inner design surfaces. The windows follow an alignment related with their bent edges. Due to the bent shape of the Market Hall, the slabs are not vertically aligned, thus the width of the facade varies from floor to floor as well as the position of the frames along this width.

To align the windows relatively to the shape of the facades we defined two alignment curves: by offsetting inwards the bent edge of the outer design surface defines the exterior alignment; by offsetting outwards the bent edge of the inner design surface defines the interior alignment (Figure 7.29).

To control the position of these offsets it is required the distance between the original and the offset curve. This control is relevant considering the aesthetic of the building but also to control their position relative to two other elements: the inner bent wall and the frames of the balconies.

To compute the abscissas we developed a method to find the extreme abscissas between which the windows are disposed on each floor and then computed the intermediate abscissas.

### 1.1 Algorithm to find the abscissas

For each floor two horizontal line segments are drawn, one at the height of the base and another at top of the window. The intersection of the two alignment curves with the two line segments origins four intersection points. The abscissas of the two points belonging to the same curve are compared, which is done for both curves: the maximum abscissa, i.e. the rightmost, of the two intersection points belonging to the interior alignment defines the initial abscissa; the minimum abscissa, i.e. the leftmost, of the points of the exterior alignment defines the final abscissa. The two fundamental abscissas represent the extreme limits between which the windows are disposed (Figure 7.30).

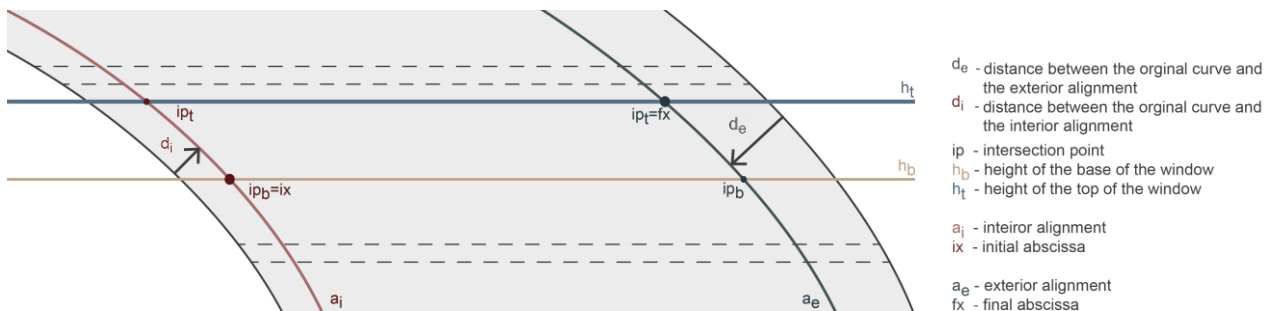


Figure 7.30 Method to compute the abscissas of the anchor points.

With the two extreme abscissas found we developed a procedure to calculate the middle abscissas for each floor in order to dispose several windows between the initial and final abscissas (Figure 7.31). The parameters we used to control the position of the windows are the number of windows and their width, one value for each floor.

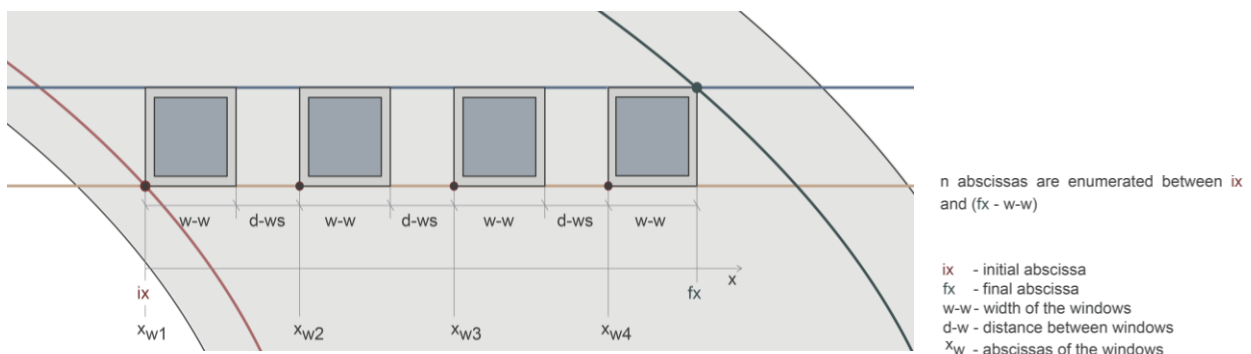
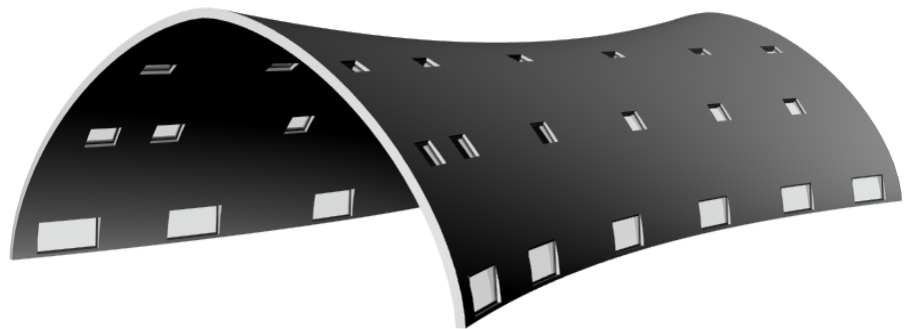


Figure 7.31 Method to compute the abscissas of all the windows.

Due to the bent shape of the building we had to deal with exceptional cases to find the extreme abscissas between which the windows are disposed. We implemented them for the most likely scenarios where the intersections between the horizontal segments and the alignment curves may not occur, namely in the upper and ground floors.

### **7.3.3 Windows of the inner bent wall**

The inner bent wall, boundary between the market and the interior of the building, is characterized with windows. In the second phase of the modeling process this wall was defined as a continuous solid, thus besides the addition of frames, its geometry was modified through the opening of the rough openings (Figure 7.32).



**Figure 7.32** Representation of the windows of the inner bent wall.

By analyzing the frames of the inner bent wall we identified several composition rules, such as symmetries on the windows of each floor. However, these rules are just apparent, the expected positions of the windows along the longitudinal axis of the wall is modified, even though little. The priority of this phase of the design was to control the exact position of the interior windows, thus their position along the y-axis is user-defined.

The direction and position of the windows depends on the curvature of the inner bent wall. In a building with a uniform shape along its length, such as the Market Hall, the windows belonging to the same floor have the same direction. However, in buildings defined with different sections along the y-axis we need to find the direction of each window.

The process we developed to represent these windows can be divided into three main tasks, namely: (1) to find one point at a bent surface for each window, (2) to calculate vectors at that point to orient each window according to the local curvature of the wall, and (3) to shape the boxes, to be subtracted from the wall, and the frames, both oriented according to those vectors.

## 1. Points in a surface

As we illustrate in Figure 7.33, a point, belonging to the interior surface of the bent wall, was found by intersecting a horizontal surface, parallel to the plane  $xy$ , and a vertical surface, parallel to the plane  $xz$ , with that bent surface.

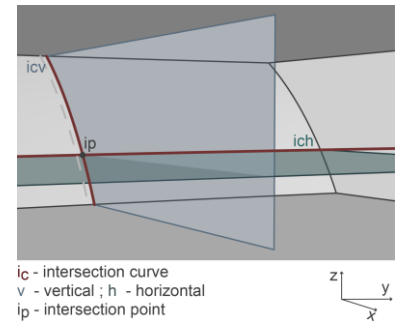
The vertical surface is shaped at the middle ordinate of the rough opening, the user-defined parameter to locate the windows. However the point in which the vectors are calculated influence the direction of the windows. Moreover, if the wall has different bents, the windows of the same floor will be horizontally misaligned. Thus, it is crucial to consider different  $z$ -coordinates for the horizontal surfaces that we use to obtain the points in order to choose the preferred alignment for the windows. We considered three options for the height of the points and analyzed their effect, namely: (1) the height of the base level of the windows leads to windows aligned at the base but the top heights variable, (2) the height of the middle of the window leads to windows aligned at the middle height but both, base and top heights variable, and (3) the height of the top level leads to windows aligned at the top but with their base heights variable.

To find the points, we choose the height of the base level of the rough opening, the option 1, for two main reasons: regarding the ground floor level we considered an advantage to have windows with the bases aligned and to control that height for cases in which the windows start at the ground level; we considered more relevant to control the height of the base level for security reasons in the upper floors. This parameter contributes also for consistency with the parameters we applied to define the windows of the front and back facades. For each floor, it is required a distance from the floor to the base level of the rough opening from which we compute the  $z$ -coordinates of the horizontal surfaces.

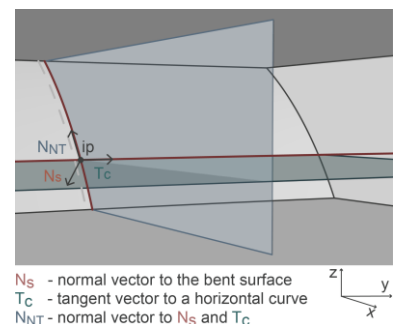
## 2. Vectors

To orient each window, three vectors are calculated at the previously found point, namely: a normal vector to the bent surface, a tangent vector to a horizontal curve belonging to that surface, and a normal vector to the previous two (Figure 7.34).

We obtained the first two vectors by using specific primitives of the library of Rhinoceros. With those two vectors we computed the third vector by applying the cross product.

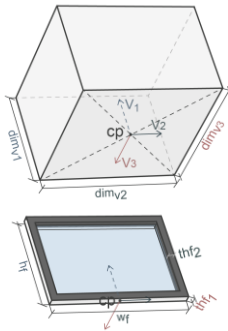


**Figure 7.33** Method to find one point belonging to a surface.



**Figure 7.34** Computation of the vectors.

### 3. Oriented boxes and frames

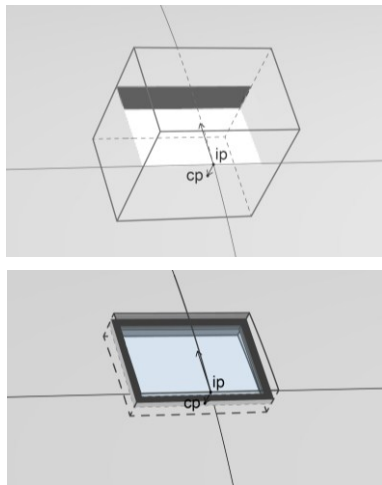


cp - central point  
v - vectors ; dim - dimension  
hf - height ; wf - width ; th - thicknesses

**Figure 7.35** Top: Oriented box. Down: Frame.

We then defined a procedure to create boxes oriented according to three vectors, three dimensions associated to them, and an anchor point as we illustrate in Figure 7.35, on the top. The frames were defined by subtracting two of those boxes and by placing one rectangular surface at the middle thickness of the frame. We defined them with the same parameters as the boxes and, additionally, with one more thickness (Figure 7.35, below).

The anchor point we use to place the frames is the used to calculate the vectors but displaced to the center of the thickness of the wall, according to the direction of the normal vector to the surface. This displacement ensures that the frames are placed at the middle thickness of the wall.



ip - intersection point  
cp - central point

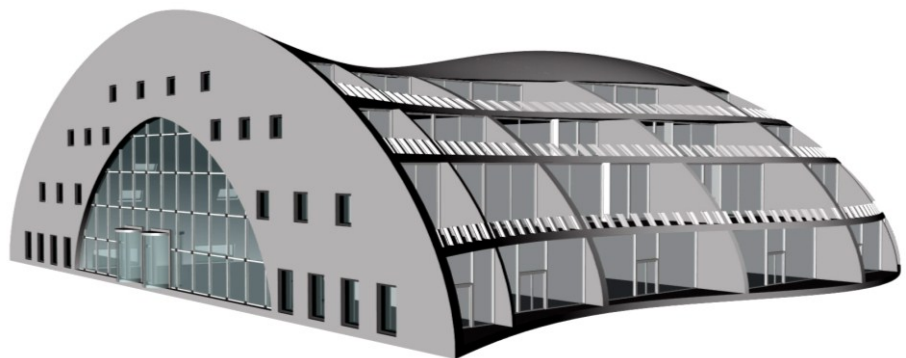
**Figure 7.36** Top: box being subtracted to the inner bent wall. Down: placement of the frame.

The geometry of the wall is modified by intersecting boxes with the inner bent wall and the frames are placed at the same location (Figure 7.36).

## 7.4 Phase 4 – Detail Elements

The last phase of the model is concerned with the inclusion of elements with relevance in the exterior aesthetic of the building, by further detailing some of the elements that were previously defined.

The glass facades were detailed by representing the main elements of a suspended cable-net facade and the revolving doors are also represented. The lateral facades of the building have balconies on all the floors above the ground and extend along the entire width between all the vertical walls, thus these elements have a big aesthetic impact.

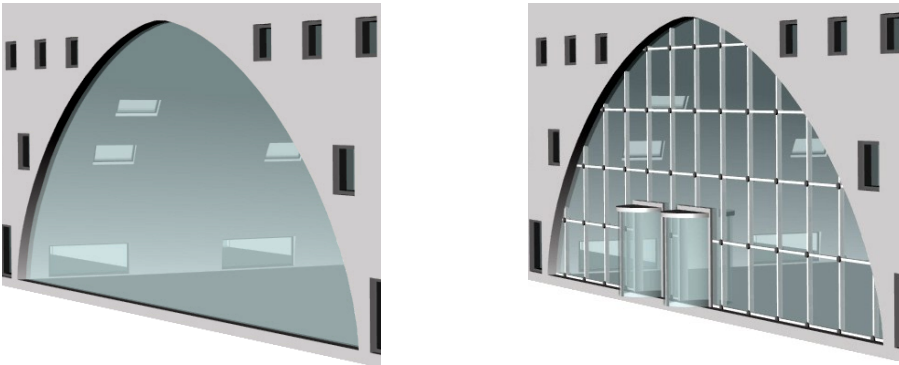


**Figure 7.37** Phase 4 of the model: Detail of the glass facades and inclusion of the fencing posts.



### 7.4.1 Glass facades - Suspended cable-net facades and revolving doors

As previously referred, the hall of the building is closed with glass facades and connected by doors to the neighborhood. In the second phase of the model these facades were only represented by two solids. With the advance of the design process and consequent increase in detail it is relevant to detail these facades and represent the doors (Figure 7.38).

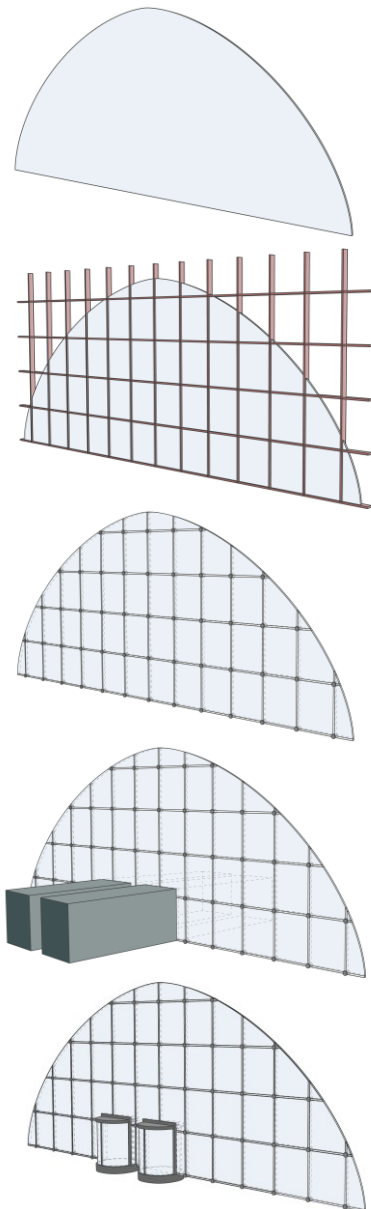


**Figure 7.38** Left: Glass facade in the second phase of the model. Right: Glass facade detailed.

The building hall is free of any interior partitions, thus visual transparency providing maximum visual permeability between the market and the neighborhood was the concept prioritized by MVRDV. The facade type, a suspended cable net glass facade, was chosen according to this aim and the entrances are provided by revolving doors. These facades are composed of a cable-net made up of horizontal and vertical cables held together at their intersections by clamps that serve as the point of attachment for the glass panels, which are supported in their corners by circular corner patch plates.

To detail these facades, we represented the elements with major aesthetic impact, namely: the glass panels, the sealants and the glass attachments. Concerning to the revolving doors we simplified their elements and represented just the exterior elements.

The boundaries of the glass panels follow the same alignment than the cable-net to provide minimum obstruction view. The glass has a maximum span to handle with loads which has also costs implications. This fact supported the choice of the key parameters to shape these elements: the dimensions of the glass panels and the distance between them, which is the thickness of the sealants. Due to the bent shape of the facades, there will be two types of glass panels: regular and irregular. The regular panels are parallelepipeds with equal dimensions, user-defined parameters, but concerning to the irregulars, they have one curved boundary, their shape



**Figure 7.39** Steps for detailing the glass facade.

and dimensions depends on the shape of the facade and on the regular panels.

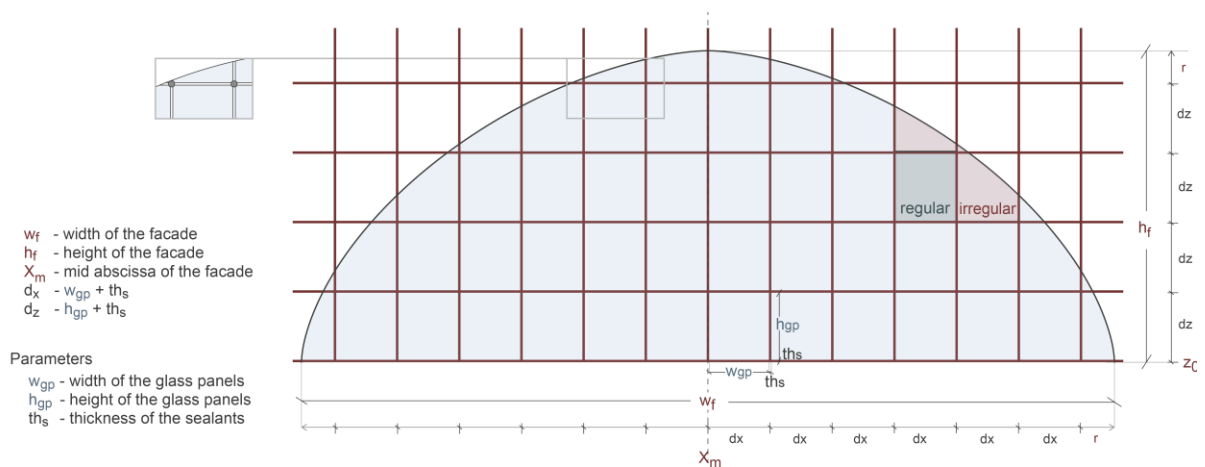
The process we developed to represent these elements (Figure 7.39) is based on the creation of a tridimensional grid composed of vertical and horizontal parallelepipeds. By subtracting the grid from one continuous glass facade, defined in the second phase of the modeling process, defines the glass panels and by intersecting the grid with the same facade defines the sealants. The glass attachments were added in the intersection points of the horizontal and vertical axis of the grid by selecting the points that were inside or in the boundary of a section of the facade.

Lastly, boxes with the dimensions of the revolving doors are subtracted from the glass panels, sealants and glass attachments to open the rough openings and then the revolving doors are placed in these locations.

Below, we present the considerations about the panelization logic (1) that guided the creation of the grid, and the method to shape the glass attachments (2) and the revolving doors (3).

### 1. Panelization logic

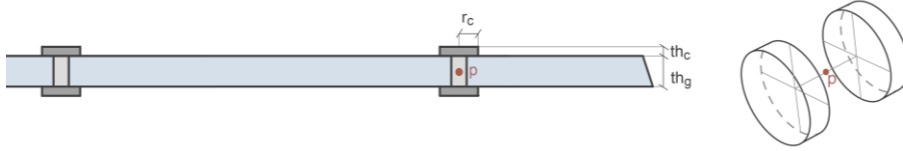
Two rules were captured from the facade composition of the Market Hall to create the grid, namely: regular panels are arranged from bottom to top and from a facade central axis to the right and left sides. The dimensions of the vertical and horizontal parallelepipeds were computed according to the maximum dimensions of the facade. In Figure 7.40 we illustrate the logic we developed based on the above mentioned rules.



**Figure 7.40** Logic of the grid.

## 2. Glass attachments

The glass attachments were simplified and are represented as a system of two cylinders, one on each side of the glass panels. We defined that system according to an anchor point, the intersection point of the axis of the grid, the dimensions of the cylinders and the thickness of the glass (Figure 7.41).



### Parameters

$p$  - anchor point

$th_g$  - thickness of the glass

$th_c$  - thickness of the attachments

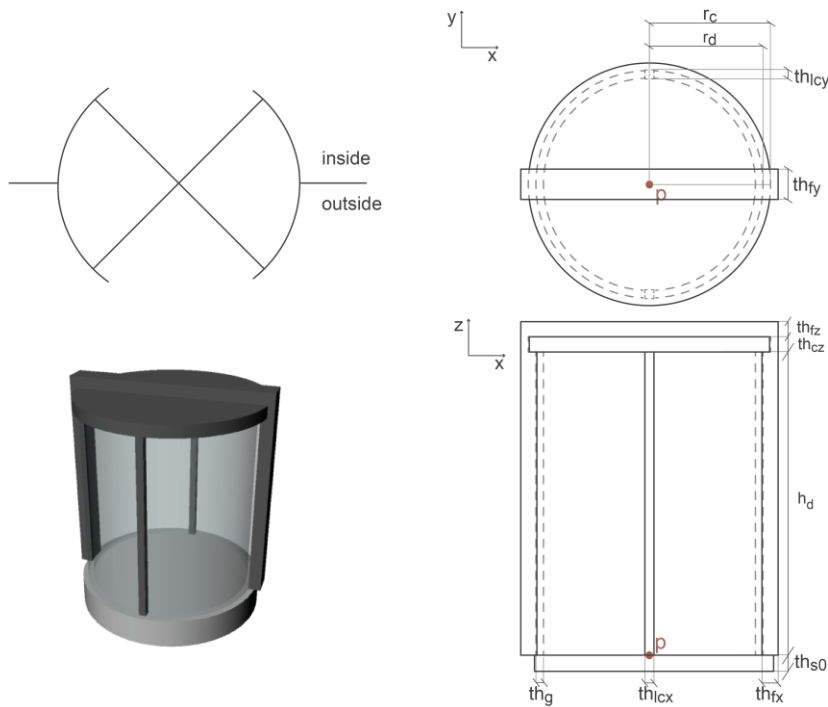
$r_c$  - radius of the attachments

Figure 7.41 Parameters of the glass attachments.

## 3. Revolving doors

Each revolving door is composed of several elements juxtaposed, namely: the cylindrical glasses, the lock columns, the cover and the frame. We also added a cylinder to be added to the ground floor slab of the building.

We established relationships between the different elements of these doors. All the elements were shaped according to the same anchor point placed at the center of the base, at the ground floor level. The parameters we used to shape these doors are illustrated in Figure 7.42. To locate them along the width of the facade the user defines the abscissas of the anchor points, the ordinates and z-coordinates are computed according to the parameters of the glass facade.



### Parameters

$p$  - anchor point

$h_d$  - height of the door

$r_d$  - width of the door (radius)

$th_{fx}$  - thickness of the frame

(x-axis)

$th_{fy}$  - thickness of the frame

(y-axis)

$th_{fz}$  - thickness of the frame

(z-axis)

$th_{cx}$  - thickness of the lock column

(x-axis)

$th_{cy}$  - thickness of the lock column

(y-axis)

$th_{cz}$  - thickness of the cover

(z-axis)

$r_c$  - radius of the cover

$th_g$  - thickness of the glass

$th_{s0}$  - thickness of the ground

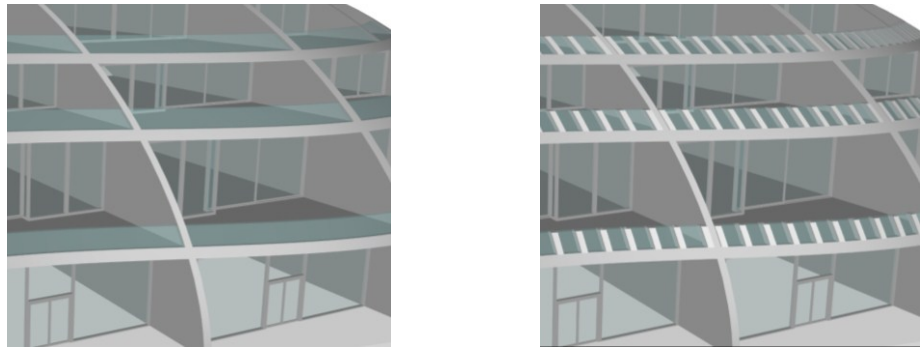
floor slab

Figure 7.42 Parameters of the revolving doors.

## 7.4.2 Fencing posts

The fences of the balconies are composed of glass panels with posts, without top and bottom rails. In the second stage of the model the fences were represented only by the glass panels, one for each balcony. At this stage, we included the posts (Figure 7.43).

The posts of the Market Hall follow the curvature of the overall shape of the building as the fencing panels.



**Figure 7.43** Left: Fencing glass in the second phase of the model. Right: Inclusion of the posts.

The strategy we used to include the posts can be divided into two main tasks, namely: (1) to find the location of the posts, and (2) to shape them in those locations.

### 1. Location of the posts

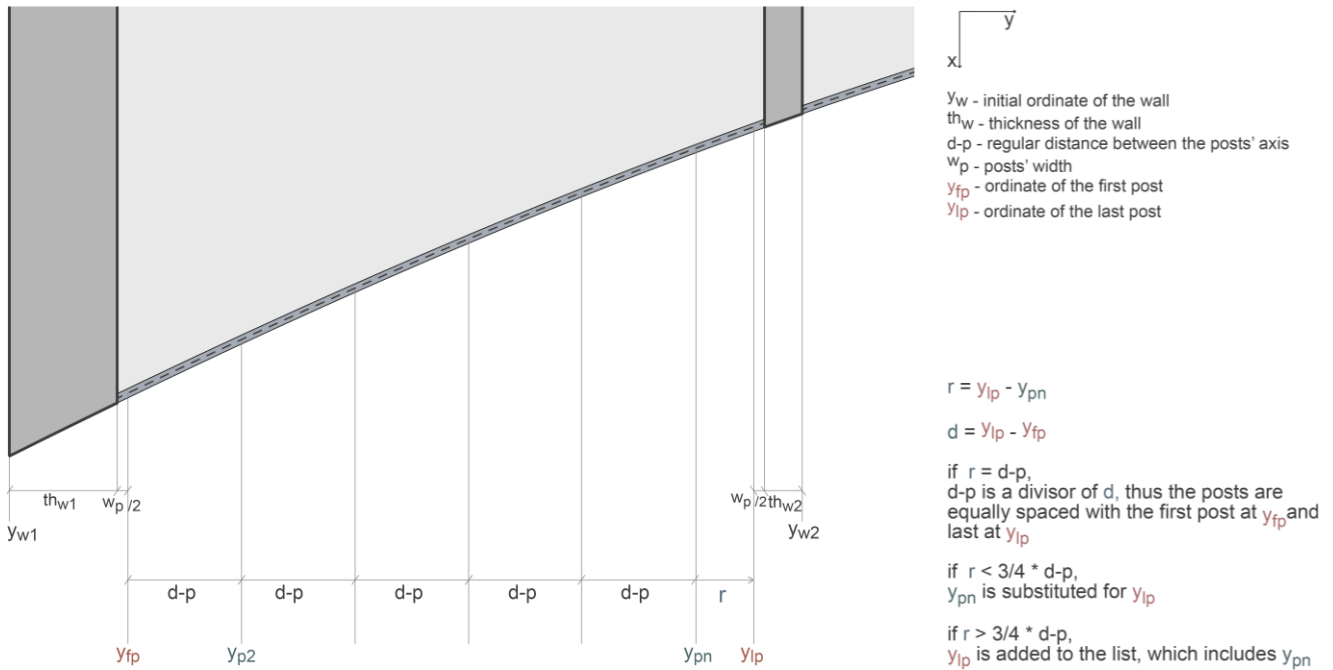
Buildings defined with equal section curves along their length, such as the Market Hall, have slabs with straight boundaries, thus their shape allows to control simultaneously the glass spans and the distance between posts

In buildings defined with different section curves along their length, i.e., with slabs with curved boundaries, by controlling the true distances between the posts, i.e., the glass spans, results in posts vertically misaligned, and by controlling the distances in the side view of the building results in different true distances but posts vertically aligned.

Thus, we had to prioritize one of these controls. We prioritized the existence of posts vertically aligned in order to achieve the aesthetic regularity observed in the Market Hall, instead of controlling the glass spans. Thus, we developed a method to compute the positions of the posts based on their distances measured in the side view of the building and along the y-axis.

We computed the ordinates of the middle axis of the posts based on the rules we captured from the Market Hall which we illustrate in Figure 7.44. Its starting point is the creation of the outer design surface of the building, once

the posts follow that curvature. To the posts be centered with the glass panels we created an offset surface of that design surface at the middle thickness of the glass. Then, based on the parameters we used to represent the walls and with a user-defined distance between posts we computed the ordinates.



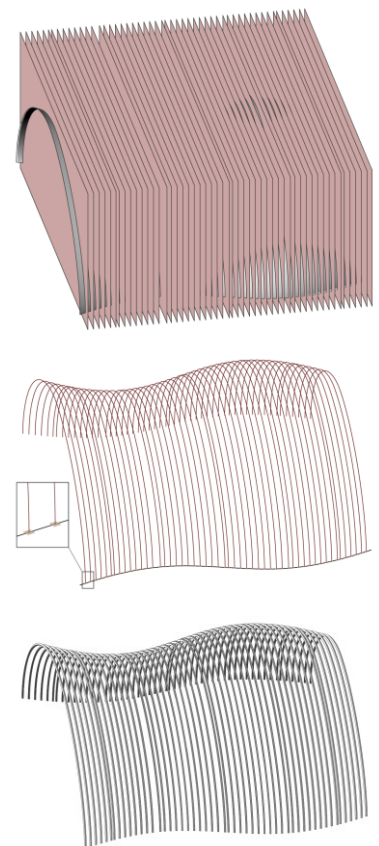
**Figure 7.44** Partial plan (one balcony): Positions of the posts by controlling their distances along the longitudinal axis of the building.

## 2. Method to shape the posts

The Market Hall has slabs with straight boundaries, thus it has posts with identical sections and identical shapes when viewed from the side view of the building.

However, in buildings with slabs with curved boundaries we must prioritize one of these aspects. Prioritizing the existence of identical shapes in the side view leads to the existence of posts with different sections and dependent on the overall shape of the building. Prioritizing the existence of an identical section leads to the existence of posts with different shapes when viewed from the side view of the building. We implemented both methodologies but we explain and illustrate the one that prioritizes the existence of an identical section because it allows us to control that section and, consequently, the exploration of a larger solution space by changing it.




First we defined solids around the shape of the building by sweeping sections and capping their extremities (Figure 7.45 and Figure 7.46). To this end, we defined vertical surfaces, parallel to the plane xz, at the ordinates of

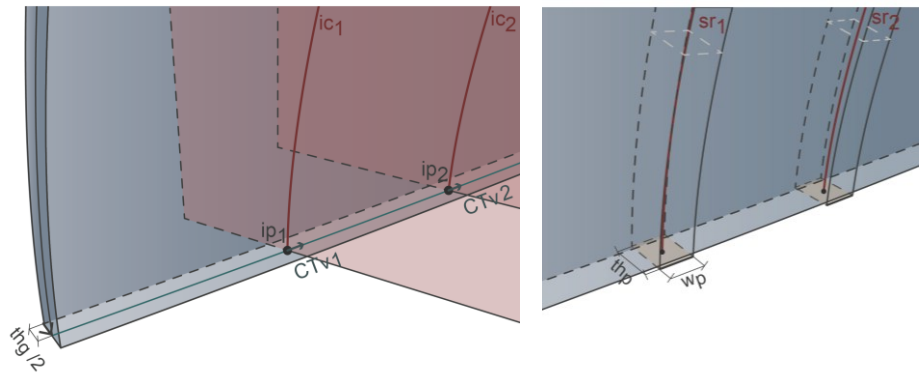


**Figure 7.45** Method to define the solids around the shape of the building.

the posts' axes, which intersected with an offset surface at the middle thickness of the glass defines curves, the sweep rails.

A procedure was defined to produce rectangular sections oriented according to a director vector and with two dimensions (width and thickness). The central point of each section, i.e. the anchor points, is the initial point of the rails. The direction vectors are calculated as the tangent vector of a point belonging to a curve. That curve is a longitudinal edge that is extracted from the offset surface. The sections are drawn at the central points with the direction of the tangent vectors, the width is attributed in the direction of the tangent vector and the thickness in the perpendicular direction.

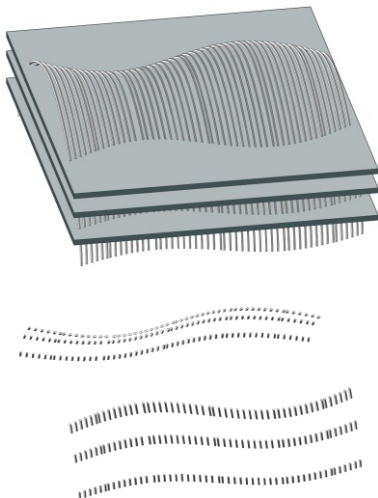
-   $ic = sr$  intersection curve/  
sweep rail
- $ip$  - intersection point
- $CTv$  - curve tangent vector
- $th_g$  - thickness of the glass
- $th_p$  - thickness of the posts
-  - section
-  - extracted edge, at the  
middle thickness of the  
glass



**Figure 7.46** Methodology to shape posts with identical sections.

The posts for the different floors are defined by reusing the “Horizontal Elements” procedure with the solids around the shape of the building as input, i.e. by intersecting a set of horizontal boxes with the height of the fencing (Figure 7.47).

The parameters to shape the posts are the parameters already needed to define the fencing panels, namely the parameters of the outer design surface, the thickness of the glass, the z-coordinates of the slabs, the ordinates and thicknesses of the walls, and the fencing height; and the parameters needed to define the positions and the section of the posts, namely, the distance between posts' axis and the dimensions of the sections.



**Figure 7.47** Method to shape the posts for the different floors.

# 8 ANALYSIS OF THE MODELING PROCESS

In this chapter, we summarize the results of our modeling approach when applied to the implementation of the case study. Although each different project might require different strategies, the analysis of our case study provides many relevant lessons that can be used to inform the development of similar models.

This analysis addresses the following themes: (1) general considerations about the construction of the model; (2) the choice and management of the parameters; (3) the reuse of procedures; (4) methodologies; and (5) strategies to control the model.

## 8.1 General considerations

### 8.1.1 *Decomposition*

The model was decomposed into different parts that were recursively decomposed into simpler elements. This involved breaking each task into smaller and more manageable subtasks. Instead of investing a lot of time implementing a complex procedure to define an element that, in the end, might not be what we were looking for, this decomposition allowed a continuous refinement of procedures until we obtained the best results. This strategy contributed for an easier and more understandable problem-solving process. Lastly, the implementation of small subtasks was also useful to allow reuse of the corresponding procedures, thus contributing to a clearer and smaller program.

### 8.1.2 *Legibility*

In order to address new requirements the implementation may need to be modified, which may be necessary at any time during the model construction. To facilitate these changes and also to facilitate model sharing between several users or developers, the legibility of the code is extremely important, particularly when the changes are made long after the initial implementation. To this end, the code must be commented, with special

attention to the exceptional cases, or to decisions that are not fully explicit in the code.

Another important lesson is that the effect of each parameter should be clear and naming conventions should be used that allow a reader to quickly infer the purpose of the parameter. Similar considerations should be made for the names assigned to the procedures, so that they reflect the result of their invocation. When several methods were defined to produce a similar result the factor that distinguishes them was incorporated in the name of the function. On the other hand, functions that produce a similar result but have different parameters were named in a way that incorporates the distinguishing parameters in the name.

Another significant aspect of the code legibility is the existence of a well-defined and well-known hierarchy between elements, as this proved to be very important to facilitate later changes.

### ***8.1.3 Goal of the model***

Models can be defined for several purposes, e.g. to explore different design options, to produce drawings or photo-realistic images, to serve as an interactive tool in presentations to clients, etc.

Each of these purposes may have different, contradictory, requirements, such as the time needed to obtain results and the detail of the model. If the goal of the model is to explore different solutions or to serve as an interactive tool, quick delivery of results is the most important requirement, whereas the production of communication elements might require a higher level of detail. Representation of elements with great detail increases both the time needed to develop the computational program, as well as the time needed to generate the model and, possibly, to render it. Thus, the aim of the model must be explicit since the beginning of the modelling process.

An example of this situation was experienced during the modeling of the fencing posts. After the definition of the strategy to position them it was easy to divide the fencing panels into different elements. However, although this did not add any important information or visual effect to the model, the production process was very time consuming so, to make a trade-off between detail and performance, we decided to not divide the glass panels in order to obtain faster results.

The methodologies we used to produce the elements of the model also have an important role when the model is to be used for several different purposes, as they allow us to obtain results with different levels of detail



depending on the goal of the model. The definition of the frames of the balconies is an example of this situation. The first task of this strategy was to find the anchor points to place each frame. To test this methodology, we used rectangular surfaces in the place of the frames in order to achieve faster results. After completing the process, we could quickly replace the surfaces by the frames. The use of this strategy was not motivated by the need for a model that served different purposes but it shows that the programming approach is flexible enough and allows easy adjustments to the detail of the model.

## **8.2 Parameters**

### ***8.2.1 Choice and number***

A procedure is a formal description of a process that produces the intended result. Different results may be obtained by providing different values to the parameters. The independent parameters must be defined according to what we want to investigate and explore. Thus, the choice of parameters is crucial for taking advantage of the model and for controlling its behavior.

The choice of parameters should be made after analyzing the context of each design problem. As an example, consider the problem of placing the fencing posts and the windows of the front and back facades. In this case, it is possible to identify a common task: to dispose several elements between two positions. Both placements needed the calculus of intermediate and extreme locations. However, to place the fencing posts we prioritized the control of a regular distance between elements, while for the windows, the parameters we chose were the number and width of the windows, to allow aesthetic control and to ensure equally spaced windows. Thus, although the task is similar, the parameters were chosen in order control different things.

A complex model, in which elements are defined based on a hierarchic structure, has long chains of dependency between elements. These structures are characterized by one-directional associations leading to the direct propagation of parameters. This has repercussions in the number of parameters but also in the auxiliary tasks needed to manipulate them within the definition of the procedures. Thus, the choice of the parameters should compare the user's effort to assign certain parameters with the effort needed for the programming task, and with the legibility and extensibility of the code. The choice of the parameters to locate the vertical walls is an example of this situation. One of the parameters that we initially considered was the distance between the walls, so that the user would not need to provide their

specific locations. This required the definition of auxiliary procedures to compute those locations, which were also required within the definition of other elements that have relationships with the positions of the vertical walls. The increased programming effort, the reduced code readability and the extension of the code was not compensated by the simplicity of use that it provides.

The choice of the parameters can also restrict the exploration of the model. The previous example of the vertical walls also reflects this situation. Although the Market Hall has equally spaced walls, using the distance between walls as unique parameter would severely constraint the positions of the walls, thus reducing the solution space. By allowing the location of the walls to be user-specified, a much large number of different models can be produced.

A formalization that allows many different variations of a model will necessarily have many parameters. Fortunately, there are several strategies that can be used to decrease the number of parameters without over-constraint the model. One is to establish rules between the parameters of different elements. This strategy was used in the modeling of the glass facades: their location was constrained to be centered within two walls, thus saving the addition of an additional parameter for an arbitrary location. Another strategy is to use global variables. As an example, consider the revolving doors of the building. These elements have their own parameters but, within models of the same scale, some of these parameters are rarely modified, as they depend almost exclusively on the manufacturer. In this case, it is preferable to store the values of the parameters as global variables, avoiding polluting the code with too many unnecessary parameters.

### ***8.2.2 Testing the model***

To facilitate the test of complete models, we created a different file for each model where the specific values of the parameters were stored as global variables. As previously mentioned, the definitions of different elements share parameters, thus, changing a global variable propagates its value to all the elements that require it. This strategy allowed the fast and easy modification of the parameters and avoided errors due to the possible inconsistency between the values of identical parameters in different elements.

## **8.3 Reuse of procedures**

### ***8.3.1 Decomposition***

During the model construction, we defined procedures in order to automate repeatable tasks according to the project needs such as: calculus automation, production of rectangular surfaces according to different directions, boxes with different inputs etc. Many of these procedures ended up becoming part of a generic library, that we continuously reused to produce geometry according to the specific parameters we used in the model. The procedures for “vertical elements” and “horizontal elements” are good examples. They were reused in the definition of several elements which confirmed the relevance of their definition. They contributed for a better legibility of the code and to decrease the number of procedures that the user would have to understand.

### ***8.3.2 Abstraction***

Each project, and consequently each model, has its own and specific requirements. However, some buildings might have identical elements or elements that have the same underlying logic which are then implemented with similar procedures. These procedures tend to become more evident in later phases of the modeling process when there is an effort to abstract the commonalities between different but similar procedures. The result is that groups of similar procedures are replaced by a common, more abstract and more parameterized definition that not only solves the same problems solved by all the procedures it replaces but can now also be used in other projects. Obviously, this abstraction process requires effort, particularly, to repeat the tests and debugging phases, which is always time-consuming and may not be cost-effective.

Even when the abstraction process is not cost-effective for a given project, the ideas can be saved to be used in future projects. By using the knowledge gained in previous projects, the programming task becomes progressively easier from model to model.

## **8.4 Methodologies**

### ***8.4.1 Boolean operations***

There is usually more than one way of getting a certain shape from a combination of Boolean operations. This is visible in the “Vertical Elements”

and “Horizontal Elements” procedures that, initially, were defined to produce a set of elements contained in a solid by subtracting a set of non-wanted parts from that solid. However, computing the non-wanted parts was more complex than providing the location of the wanted parts so we decided to simplify the procedure by replacing the subtraction on non-wanted parts with the intersection of the wanted parts. In both cases, the result is the same, but using intersections would have saved us time and contributed to clarify and simplify the algorithms. We concluded that these differences must be evaluated before implementing the functions in order to save time and effort. This kind of expertise makes the development of future models much more efficient.

#### **8.4.2 Interpolations**

Modeling processes usually resort to interpolations. In our case study, the construction of the model started with the definition of bent surfaces that result from the interpolation of section curves that result from the interpolation of points. Unfortunately, this process is not fully deterministic because it depends on the interpolation algorithms used by the CAD tools and, as a result, it is not possible to precisely know the final shape that results from the interpolation. However, we frequently need to know the overall dimensions of the resulting shape, as they are needed to establish reference points, symmetry axis, etc. In order to overcome this problem we developed a method based on the bounding box of the interpolated object, where all necessary dimensions are measured relative to this bounding box. This method was recurrently used along the modeling process, allowing to effectively solve the lack of control over the exact dimensions of the objects created through interpolations.

#### **8.5 Control**

The model that is produced depends on the parameters that were used. Some of these parameters correspond to locations in space, implemented as points of cross section curves. Any change to these points is propagated to the final shape, as it entails the production of different cross section curves, which entails the production of different shapes, etc. This method of shape control proved to be simple and intuitive, allowing us to easily predict the shape that will be created.

However, it is also possible to use a different control method that might be more practical for the end user or even for the stages where the model is being tested. This control method is based on allowing the user to draw the

cross section curves in the modeling environment of Rhinoceros, without forcing him to manually extract the points of the curves, which would be laborious and time-consuming. The user just has to interactively select the curves he wants to use and our program will directly use them to continue the model generation. This control method was used to produce the curves of the Market Hall that we copied from a drawing.

As previously referred the choice of the parameters is important to control the model behavior but they do not provide control inside the algorithms we use. The methodologies developed to shape an element have also great influence in the solution space that may be achieved with an algorithmic definition of a design. A good example of this situation was experienced during the modeling of the fencing posts. The first approach to shape these elements started with the definition of a bent solid with the outer design surface that was offset inwards according to the thickness of the posts. This solid was used as input to "Vertical Elements" procedure that, by intersecting a set of boxes produced solids around the building shape. Lastly, these solids were used as input to "Horizontal Elements" procedure that produced the fencing posts by intersecting a set of horizontal boxes with the fencing height. Comparing this approach, in which the geometry of the posts is dependent on the overall shape of the building, with a different approach, in which we defined the solids around the building shape by sweeping a section around curves, it is possible to conclude that, in fact the later approach allows a bigger solution space. The section is easily changed to allow the creation of different types of posts which was impossible using the first method. In fact, the first approach easily produced posts identical to those of the Market Hall, but restricted the future exploration of different solutions.



## 9 EVALUATION

The main aim of this dissertation is to explore and evaluate the potential of Generative Design as an auxiliary tool integrated in the design process. In particular, we explore and evaluate an algorithmic approach to design. Our proposal is to introduce a formalization step using a pure programming-based representation of the intended design. We claim that this step speeds up the handling of changes that always occur as the design evolves. Moreover, this formalized representation can be easily generalized for supporting a much larger solution space.

In order to evaluate our approach, we developed three-dimensional model using a purely programming-based representation of a case study based on MVRDV's Market Hall building. The implementation of the case study had four main goals:

1. To evaluate the ability of the algorithmic approach to respond to the evolving requirements of the design process;
2. To evaluate the costs and benefits of the algorithmic approach, particularly when compared to traditional approaches;
3. To evaluate the ability of the algorithmic approach to be integrated in the early stages of the design process, when decisions are highly uncertain;
4. To evaluate the limits of the programming-based representation.

The evaluation consisted in simulating several types of changes and measuring their impact, particularly, in the time and effort required for their implementation compared to the traditional approach. In the next sections we describe these simulations and we conclude about the relevance of using an algorithmic approach since the early stages of the design process and the limits of the programming-based representation.

## **9.1 Evaluation of the ability of the algorithmic approach to respond to the evolving requirements: comparison with the traditional approach**

The model was developed according to four phases, simulating the design process, from the formalization of the shape of the building until the modeling of detailed elements. The hierarchy between elements ensured that the shapes produced in each phase fit the shapes produced in the previous phases, allowing automatic change propagation.

The model was defined with 77 independent parameters, of which 32 are related to the revolving doors and frames, which we stored as global variables and were rarely changed during the entire process. Some of the parameters are, actually, lists of values. A list is a group of values that may consist of any data type, e.g., numbers, strings, procedures or even other lists. Thus, the number of parameters just reflects the number of single variables that we used, because, in fact, some of the parameters are lists of parameters, or even lists of lists of parameters that depend on the number of curves that are used to define the overall shape of building, on the number of vertical walls, on the number of floors, on the number of windows of the inner bent wall, and on the number of revolving doors. The solution space is not just a function of these parameters but depends also on the strategies we used to define the elements. In several cases, a parameter represents an actual procedure. For example, it is possible to change the shape of an element by using a different procedure as the value of the corresponding parameter. To evaluate the ability of the algorithmic approach to handle change, we tested five main scenarios. Starting from the complete Market Hall, we simulated the introduction of different changes, namely: (1) changing just the overall shape of the building, (2) changing just the dimensions or positions of the elements of the building, (3) simultaneously changing the overall shape of the building and the dimensions or positions of its elements, (4) changing the shape of some elements of the building, and (5) changing the order of elements of the building. Lastly (6), we also simulated the reuse of the program in other projects by changing the formalization of the overall shape of the building.

In the next pages we present and evaluate each of these scenarios.



### 9.1.1 Scenario 1: Changing just the overall shape of the building

A change in the overall shape of the building causes many different changes in its elements, namely in their geometry, position, direction, and number of subelements. These changes are summarized in Table 9.1.

Observing this table it is possible to conclude that the only elements that do not necessarily change in a coordinated way with the shape of the building are the revolving doors.

ELEMENTS		CHANGE			
		Geometry	Position	Direction	Number
Vertical walls		✓			
Inner bent wall		✓			
Slabs		✓			
Opening and frames	Lateral facades		✓		
	Front and back facades		✓		
	Inner bent wall		✓	✓	
Glass facades	Panels	✓			
	Sealants	✓			✓
	Glass attachments				✓
	Revolving doors				
Fencing	Panels	✓			
	Posts	✓	✓	✓	

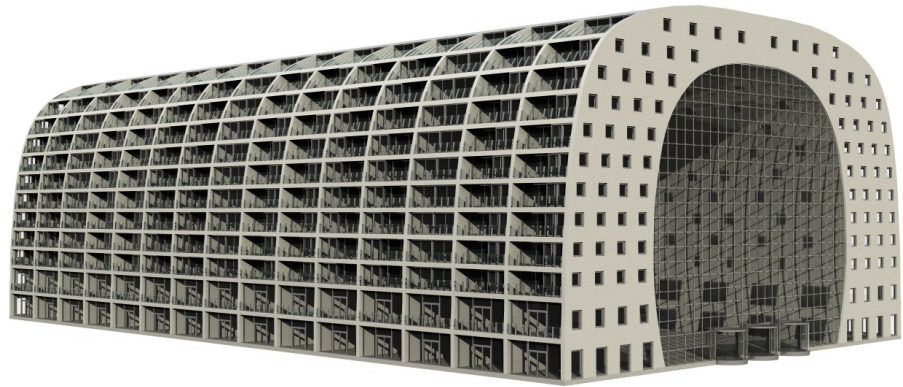
**Table 9.1** Changes in the elements of the building just by changing its shape.

The effect of this change is clearly visible in Figure 9.2. This building was generated after changing the parameters that control the shape of the building but maintaining a constant section along its length. In Figure 9.3 it is also possible to observe this effect, but now the building was generated with different sections along the longitudinal axis. The remaining parameters were kept identical to those used to generate the Market Hall (Figure 9.1).

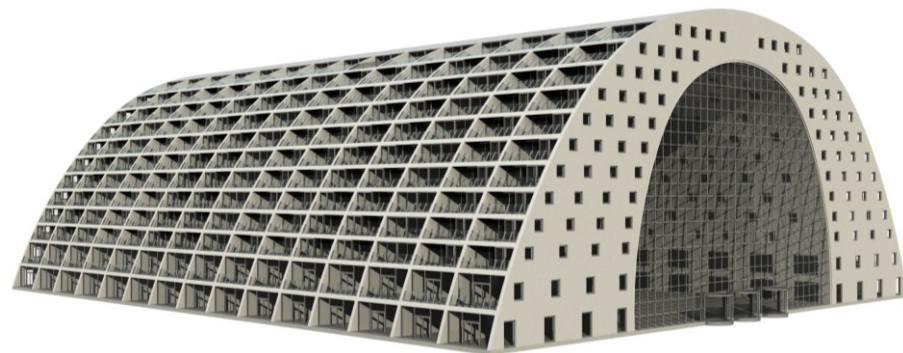
These results were produced by changing just two arguments to produce the building illustrated in Figure 9.2 and four arguments to produce the one illustrated in Figure 9.3, which we made in minutes.

This is the kind of change that it is very difficult to accommodate using a traditional approach to modeling. Some elements, whose geometry does not vary by introducing a change in the overall shape of the building, such as the shapes of the frames, could be reused from model to model but their positions and directions would have to be recalculated, except for the revolving doors. Thus, the majority of the elements would have to be remade. This process would certainly be very time-consuming and laborious

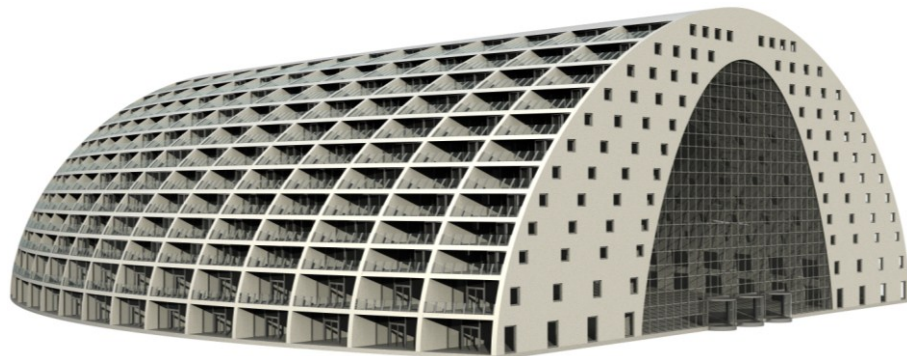
and, for each change in the shape of the building, the effort would have to be repeated.



**Figure 9.1** MVRDV's Market Hall as generated from our program.



**Figure 9.2** Building generated by changing the shape of the building.



**Figure 9.3** Building generated by changing the shape of the building by using different cross section curves along its length.

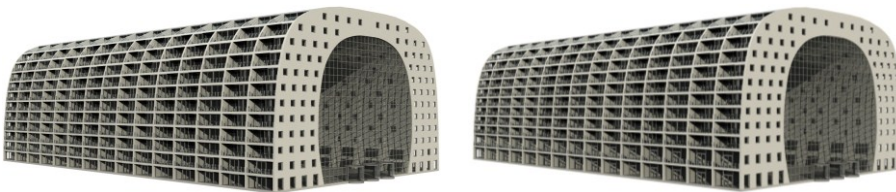
### **9.1.2 Scenario 2: Maintaining the shape of the building and changing the dimensions or positions of the elements**

Another possible change may require a change in the dimensions or positions of certain elements. To give a concrete example, any change in the position of the middle slabs would have repercussions in several other elements, as it is visible in Table 9.2.

ELEMENTS		CHANGE			
		Geometry	Position	Direction	Dimension
Middle slabs		✓			
Opening and Frames	Lateral facades		✓		✓
	Front and back facades		✓*		
	Inner bent wall		✓*	✓*	
Fencing	Panels	✓	✓		
	Posts	✓	✓	✓	

**Table 9.2** Changes in the elements of the building just by changing the position of the middle slabs (the symbol \* refers to elements that may not change by using a traditional approach).

The effect of this change is visible in Figure 9.4 (on the left). That building was generated by simply changing the parameter containing the z-coordinates of the middle floors, a change that was implemented in minutes. Observing that figure, and comparing with the Market Hall (Figure 9.4, on the right), it is clear that this change do not have much visual impact, however several elements of the building are affected by that change, however small it may be.



**Figure 9.4** Left: Building generated after changing the positions of the slabs. Right: Market Hall as generated from our program.

Considering a traditional approach the designer would have to remake the middle slabs and many other elements that need to be adjusted. It is important to note that the windows of the front and back facades and of the inner bent wall would not necessarily change. However, to maintain the relationship between the floor level and the height of the windows those changes would be required. This is the kind of situation in which the designer could avoid some changes in order to save time and effort, even though abdicating of the previously defined relationships between the positions of the elements.

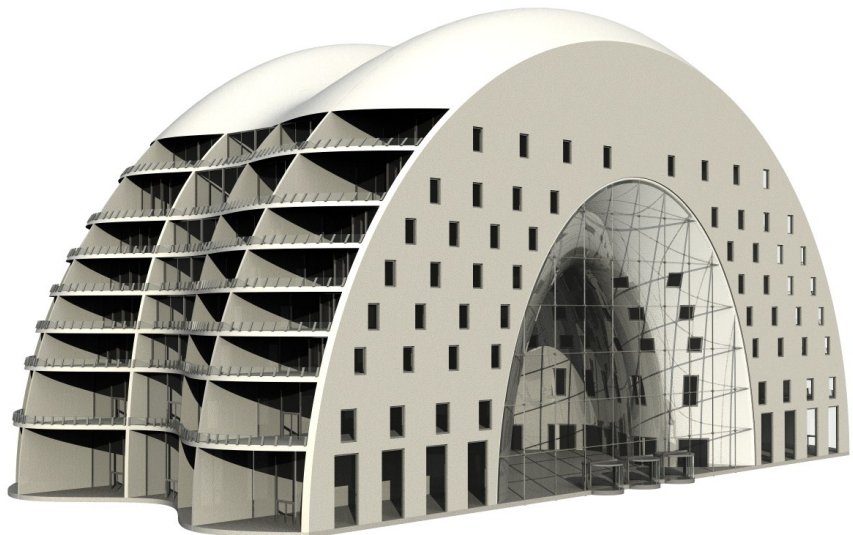
Given this example, we concluded that a change that might even seem simple to solve using a traditional approach, would continue to require an enormous effort to modify all affected elements, in fact, similar to the effort needed to produce the elements in the first place.

### **9.1.3 Scenario 3: Changing the shape of the building, its dimension and the positions and dimensions of its elements**

Another effect of the introduction of a change in the overall shape of the building is visible Figure 9.6 and Figure 9.7. In these cases, different buildings were generated by more drastically changing the section curves and by using different sections along the longitudinal axis. Besides the change of the shape of the building, other parameters were changed, e.g., the number and position of the floors and walls, the number of windows of the front and back facades, the number and positions of the revolving doors, and the dimensions of elements such as the windows or glass panels. The relationships between elements were kept and adapted to a different scale.

Similarly to the first and second scenarios, these results were produced by changing a small set of parameters, a task that was made in minutes. It should be clear that these changes are much more difficult to accommodate using a traditional approach than what it already was in the first two scenarios as they are, in fact, a combination of those two sets of changes.

Using a traditional approach, it would not be possible to reuse any element from one model to the next. The number of changes would be so large that, in fact, it would probably be easier to just start building the model from scratch, thus wasting the effort already spent in the previous model.



**Figure 9.6** Building generated by introducing several changes: variant 1.



**Figure 9.7** Buildings generated by introducing several changes: variant 2 and 3.

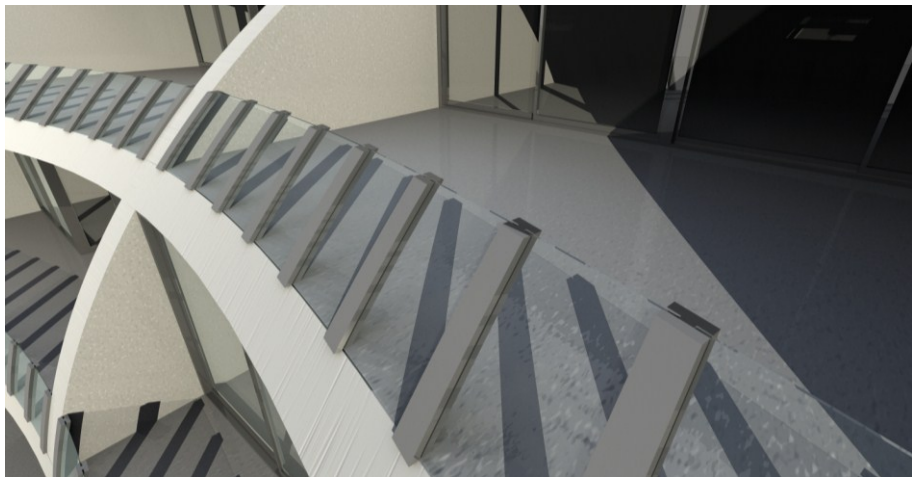
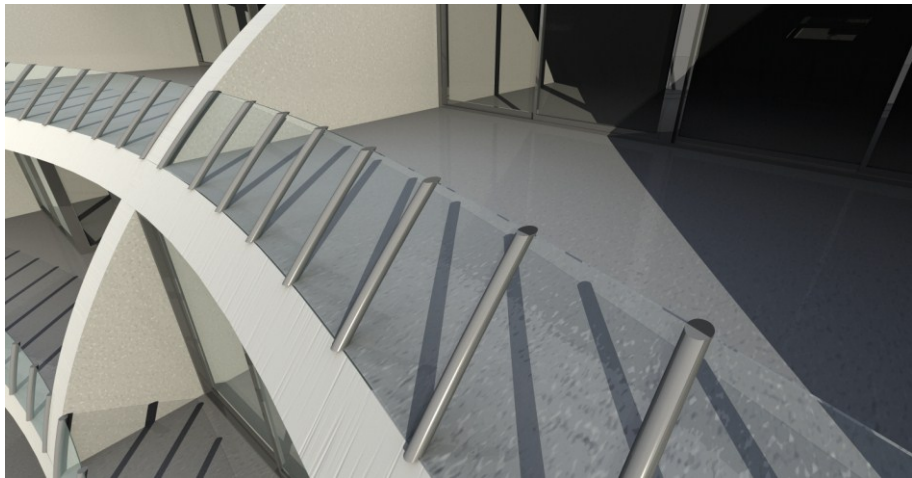
#### ***9.1.4 Scenario 4: Changing the shape of the elements of the building***

As it was already explained, for each design idea we formalized and abstracted the idea in order to implement the corresponding algorithms in a programming language. In this scenario we evaluate more drastic changes that imply modifications not only in the parameters but also in the developed algorithms. As before, in this scenario we test the resilience of the program to handle this kind of changes. As concrete cases, these changes were simulated for two different elements, namely, the windows of the front and back facades, and the fencing posts.

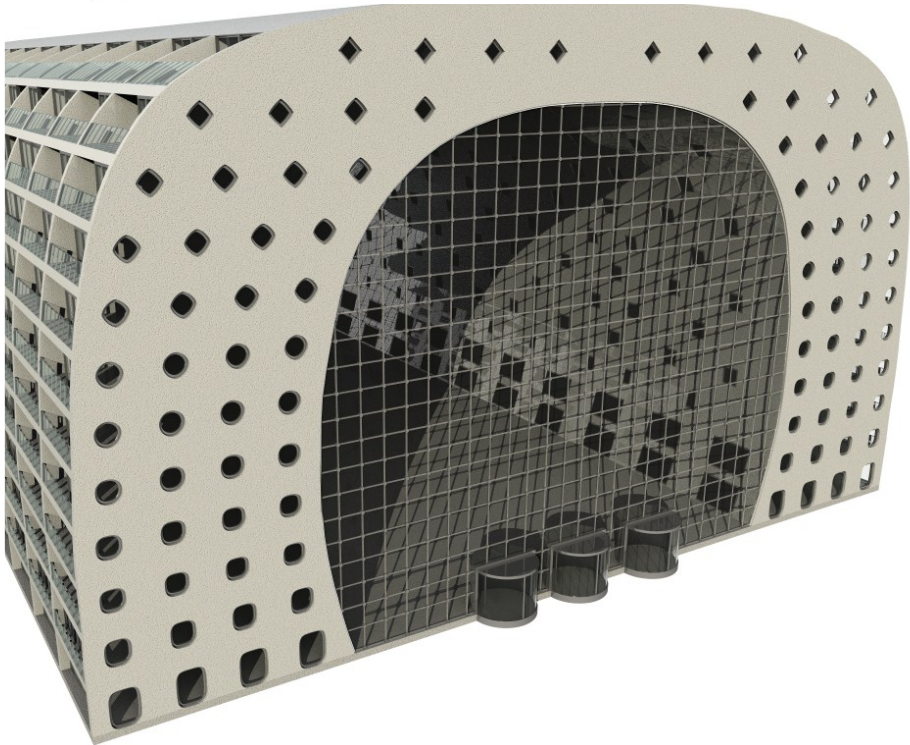
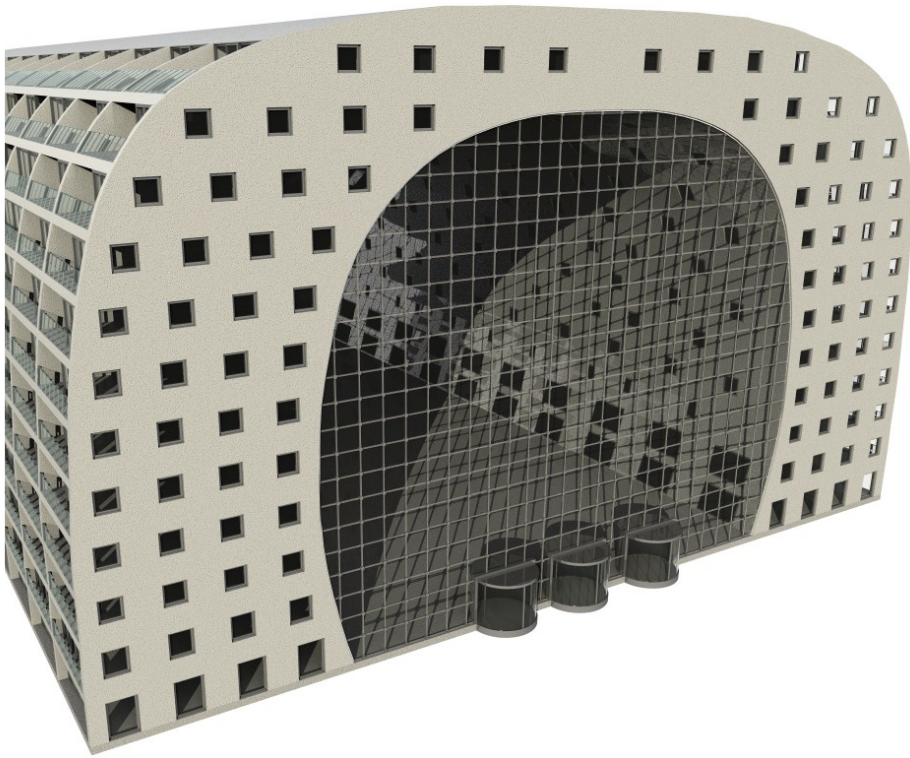
In both cases the underlying logic of the modelling process continued valid. The overall methodology we followed to define these elements consisted, briefly, of several geometric auxiliary operations to find the anchor points to position the elements. In the particular case of the fencing posts those anchor points were used to place a section which was used to perform a sweep with a rail, whereas in the windows those points were used to place solids which were subtracted from a vertical wall, defining rough openings where frames were placed.

The modification of both procedures was similar. The section of the posts (initially rectangular, to be identical to those of the Market Hall) and the shape of the solids and frames (boxes and rectangular frames, for the same reason) were abstracted. We generalized the procedures dealing with these specific shapes to accept an additional parameter for a procedure that produces the desired shape, e.g., a section, a solid, or a frame. Thus, if the user wants to test different shapes it is just required to define a procedure that produces the desired shape and provide it as the value of the procedural parameter. The modifications we made to the program were completed in less than an hour, which is a very small amount of time for a task that, in the traditional approach, requires changes to many elements. Moreover, in our approach, this generalization effort only needs to be done once, allowing us to then instantly test many different alternatives, while the traditional approach requires the entire repetition of the process and, consequently, of the effort.

In Figure 9.8 (page 92), we show the effect of changing the shape of the posts. On the top the posts were defined with a rectangular section, on the middle with a circular section, and on the bottom with an I-beam profile. The same effect is visible in Figure 9.9 (page 93) where the shape of the frames was changed from rectangular to elliptical.



**Figure 9.8** Changing the section of the posts. Top: rectangular. Middle: circular. Bottom: I-beam profile.



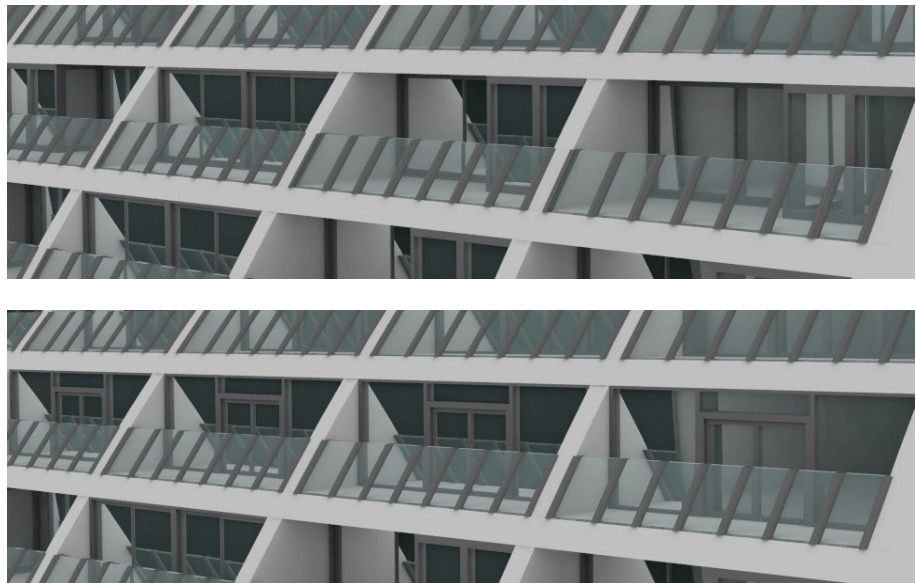
**Figure 9.9** Changing the shape of the windows of the front and back facades. Top: rectangular. Bottom: elliptical.

### **9.1.5 Scenario 5: Changing the order of elements of the building**

The Market Hall is characterized by composed frames in the lateral facades of the building. The different types of these frames follow a sequence on each floor. In this scenario we evaluate the costs of changing the order of the elements, without change anything else.

The developed procedure received in its arguments the sequence of procedures that produce the frames of each floor. To change the order of the types of frames we just changed the sequence of procedures. This is visible in Figure 9.10 and was made in seconds.

Neither the shape of the frames nor their position was modified. Thus, in a traditional approach, the designer could reuse the previous elements. The designer could extract their position and use "cut" or "copy" and "paste" to rearrange their order, which would solve the problem easily. However, in buildings with so many elements as the Market Hall, this task is certainly laborious, time consuming, and repetitive, which makes it very error-prone.



**Figure 9.10** Changing the order of the frames of the lateral frames. Top: Frames following the sequence of the Market Hall. Down: Frames with a different order.

### **9.1.6 Scenario 6: Changing the formalization of the shape of the building**

The more drastic experiment was made by simulating a change in the formalization of the building from a bent and longitudinal shape to a parallelepiped shape. In Figure 9.11 it is possible to observe the effect of this change.





**Figure 9.11** Change in the formalization of the shape of the building: two parallelepiped shapes.

During the modeling process and, particularly, in the review phase, we considered the use of the program for future design problems. Thus, we abstracted the procedures to receive abstract entities, e.g. solids, surfaces or curves, and just the last procedure that produces one element was particularized to receive the parameters that were used to formalize the overall shape of the Market Hall, that is to be applied to the underlying geometry that was defined in the first phase of the model. To obtain these results it was just required to define a set of procedures, the underlying geometry, to formalize the new shape. This auxiliary geometry was used as input in the procedures already defined, to which we did not made any change. The definition of this new set of procedures was made in approximately half an hour.

If an equivalent reuse is intended in a traditional approach to modeling, the designer might use some elements such as the frames of the lateral facades in other models when their overall dimensions are identical, but, in general, he would have to start the model from scratch.

It is obvious that some of the procedures that were previously developed could not be directly used for these new buildings as they were specially tailored to produce elements for a bent shape and, thus they would need to be modified. However, we verified that by decomposing the design problems into smaller problems and by abstracting and generalizing the corresponding procedures, we were able to reuse a significant number of them from model to model. In this experiment, as previously referred, the code was reviewed and some of the code fragments were re-written and re-tested, a task that requires time and effort. However, this time and effort is usually smaller than writing the procedures from scratch and much smaller than remake the entire project from scratch, as it usually happens in traditional approaches.

Another important advantage is that when some underlying logic, e.g. walls, slabs or frames, is repeated in several projects, it becomes possible to save a significant amount of time and effort by defining a library containing generalized implementations of the corresponding algorithms.

## 9.2 Conclusions

The six simulations discussed above show different types of change that may arise during the design process. As we showed, these changes were quickly implemented using our approach, even when they required modifications to our program.

Observing the results of scenarios 1 and 3, in which we simulated a change in the overall shape of the building and, additionally, in scenario 3, in the dimensions and positions of its elements, we concluded that the hierarchy we defined allowed us to quickly explore the solution space. In fact, the produced buildings are aesthetically different from Market Hall but their underlying logic is identical.

In scenario 2, in which we simulated just a change in the dimension or position of some elements, shows that, due to the relationships between the overall shape of the building and its elements, a change that might seem small and that do not have a large aesthetic impact still needs to be propagated through many elements. In our approach, this propagation is automatically done, but in the traditional approach this kind of changes may require as much time and effort as changing the shape of the building because it is necessary to change not only the specific elements where the changes were introduced but also all the elements that directly or indirectly depend on them.

In scenario 4, in which we were forced to modify the program, we proved that the proposed approach is sufficiently flexible not only to accommodate changes previously anticipated, but also to changes that were not planned and were only known introduce them later in the design process as they arise.

The change simulated in the scenario 5, in which we changed the order of the elements, is probably one of the simplest changes that may arise during the design process. Among all analysed scenarios, this was the change that would be easier to accommodate in a traditional approach. However, when the number of elements that must be updated gets larger, the amounts of effort and work-hours becomes considerable, whereas our proposed approach can implement the change in seconds.

The results of the scenario 6 showed us that an investment in the generalization of procedures functions pays off, as it makes them reusable in future projects. It also suggested us the development of libraries with general procedures that can be used from model to model. However, it is important to note that, in some cases, over-generalization can produce procedures that are too complex.

During the discussion of the scenarios, it was also clear that standard elements, i.e. elements whose geometry does not depend on the overall shape of the building, e.g. frames and revolving doors, can easily be used from model to model and are prime candidates to become part of a library of standard modelling procedures.

From the observation of the results of the evaluation, we can confirm our main thesis: the initial cost required to produce a program that formalizes a design is quickly recovered when it becomes necessary to incorporate changes in the design. We also concluded that our proposed approach can truly assist the designers in decision making activities by allowing the quick generation of multiple differentiated models, each exploring different design solutions. Finally, our approach has the benefit of forcing the formalization of design ideas, thus contributing for a clearer understanding of the design problems and of the design solutions.

The aim of MVRDV for the Market Hall was to design an impressive building, an icon to influence future projects that will be constructed in its surroundings, thus it is a unique building that will not be repeated. However, some projects call for designing several buildings with common characteristics but with some variants from building to building. It should be obvious that our approach, in addition to being aligned with the need to introduce changes along the modeling process, is also aligned with the need to produce several buildings that share the same generative logic. Thus, it allows mass-customization strategies, in which the effort required to produce one program is recovered by its reuse to model several buildings while still supporting handling changes that can be propagated to all different buildings.

This experiment was also made to determine the limits of our approach, that is, how far we can go by using an algorithmic approach represented with a purely programming approach to architectural design. Given the level of detail that was implemented in our models, we can now safely conclude that it is possible to go very far indeed.

Lastly, we could also conclude that a programming approach requires a more careful approach to design because any errors in the design quickly

show up as bugs in the formalization. This has the advantage of allowing early discovery of problems that are more costly to solve using traditional approaches, but delays the visualization of results.

# CONCLUSIONS AND FUTURE WORK

## CONCLUSIONS

Design processes are characterized by change. Unfortunately, CAD tools are currently being used just as a more efficient version of the traditional paper-based approach, an approach that does not help the designers in handling change, particularly for the exploration of different solutions or to adapt the design to evolving requirements.

Recently, new approaches have been introduced in the design process, which are better tailored for handling change. Generative Design is one of them and can be defined as the creation of shapes determined by algorithms.

Despite being a minority, well-established architectural offices are integrating Generative Design in their design processes to allow a more efficient approach to design. It has been playing an essential role that allows the exploration of variants of the same project and also its optimization with reduced costs.

This dissertation argues for one main point: integrating Generative Design as a new stage in the design process dramatically simplifies the handling of changes. In particular, we propose an algorithmic approach to design that overcomes the limitations of the traditional approach.

This proposal introduces a formalization step using a programming-based representation of the intended design. We claim that this step speeds up the handling of changes that always occur as the design evolves. To evaluate our proposal, we formalized the design of a building, the MVRDV's Market Hall.

We outlined a generic modeling strategy that includes the algorithmic formalization step that starts with the analysis of the externally imposed constraints and the designers' processes, concepts and intents. Then, we decomposed our modeling process according to two perspectives: (1) to match the design process decomposition, in order to progressively increase the detail and definition of the model, and (2) to match the design artefact decomposition, dividing the model into the relevant elements that it represents. As a result, the developed model is characterized by a hierarchic structure that reflects our modeling strategy.

Then, we applied this strategy to our case study, a complex building in which there are strong dependencies between its elements. The design process we used to construct the model was divided into four phases: (1) formalization of the shape of the building, (2) modeling of its main elements, (3) definition of the openings and modeling of the frames, and lastly (4) the detail elements. Instead of

capturing the exact geometry of the Market Hall, we captured the underlying ideas of the design for supporting a larger solution space.

To evaluate the algorithmic formalization, we simulated several types of changes and measured their impact, particularly, in the time and effort required for their implementation compared with the traditional use of CAD tools. These simulations allowed us to verify the relevance of using an algorithmic approach to design starting from the early stages of the design process, and also to explore the limits of the programming-based representation.

With this evaluation we proved that our approach is sufficiently flexible, not only to accommodate changes previously anticipated, but also changes that were not planned. This conclusion can be extended to the architectural practice in which changes arise, frequently without being anticipated. Although the additional step of design formalization requires an obvious initial investment, we believe that it is not only possible but actually cost-effective to use an algorithmic approach to design as a new stage in the design process. In fact, the initial cost, namely in time and effort, is quickly recovered when changes are needed.

Our generalization of the Market Hall design allowed us to also produce buildings that are different from the Market Hall. This was made possible due to the automatic propagation of changes between the elements of the building. Thus, an algorithmic approach to design proved to be aligned with the design process' needs by allowing not only an effortless introduction of changes but also the production of an infinite number of different models exploring different designs solutions, and, thus truly assisting designers in decision-making activities.

Our experiment proved also that this approach can be integrated right after the development of the concept that structures the design. This early integration allows designers to work with a powerful tool along the design process, from conceptual design until construction. In fact, our evaluation showed that an algorithmic approach to design allows mass-customization strategies, in which the effort required to produce one program is recovered by its use to model several similar buildings.

The development of our model allowed us to provide lessons to the development of similar models. For dramatically reducing the effort for handling change in the design process, we can highlight the importance of the definition of the parameters and the strategy we used to construct the model. This strategy can be summarized in four main steps: (1) to analyse the design and formalize its intentions, (2) to abstract and generalize these intentions, predicting the changes that can arise in the design process, (3) to decompose the definition of the elements in subtasks, and (4) to define these subtasks so that they operate with abstract entities. The abstraction and decomposition are of great importance for incorporating changes that were not previously anticipated.

In summary, an algorithmic approach can be integrated into the design process to make it more cost-effective at handling changes, giving the designer a competitive advantage. Our proposal do not excludes other approaches for the design process. It is an additional stage that does not replaces the creative work of the designer. Instead, it allows him to go farther in the exploration of different design solutions.

## **FUTURE WORK**

Future work should improve the evaluation of the algorithmic approach to design we proposed. Below, we present some topics that might help achieving that goal.

### **1. Improve the formalization of the Market Hall**

**1.1** We simulated the use of the proposed approach since the early phases of the design process, when intents are uncertain. The main aim was to generalize and simplify the solutions to increase its applicability, in order to develop a model capable of handling change. Thus, the future development of this work should focus on the following stages of the design process that requires more detailed and less generic solutions. Additionally, our case study, the Market Hall, was designed from the outside-in so, in the following phases, the interior of the building should be developed.

**1.2** During the modeling process we simplified the formalization of some parts of the building and we developed alternative ways to control the more complex scenarios. However, it is still relevant to implement the algorithms covering all possible scenarios. The implementation of these algorithms will be useful to evaluate their real complexity and measure the time and effort needed, in order to conclude if it is cost-effective to implement them in cases where there is no conclusive evidence about their future need.

**1.3** The model of the Market Hall we developed relies heavily upon Boolean operations. Unfortunately, the CAD that was chosen, Rhinoceros, presented great limitations in these operations, which forced us to spend large amounts of time developing workarounds that decrease the legibility of the code. Future work should improve this situation by isolating these workarounds from the Market Hall formalization.

**1.4** We proved that it is possible to reuse parts of the formalization from project to project. Thus, it might be useful to improve the algorithms we implemented in order to achieve the same results but in less time.

**1.5** We showed that our model, due to the methodologies we used, is flexible enough to be adaptable to produce representations with different levels of detail. However, it would be good to have a mechanism that would enable the designer to choose the level of detail without requiring manual changes to the program.

### **2. Develop new experiments**

**2.1** The model was developed and the proposed approach evaluated based on two simulations: (1) by simulating its implementation during the development of the project, and (2) by simulating several changes that would be needed during the design process.

These simulations were limited by the lack of information about the different phases of the design process as well as about the changes to the design that were effectively needed and made. Thus, in order to overcome these limitations, it will be useful to develop other experiments, in collaboration with the architects that developed the project or, even better, in real practice, by implementing the changes and testing the resilience of the formalization based on the actual and needed changes as they arise. Both will contribute for a better and more informed evaluation of the proposed approach.

**2.2** It would be also interesting to continue the comparison between the traditional approach and the algorithmic approach we propose. To this end, the two approaches should be evaluated simultaneously by implementing a real design and measuring the differences in models with the same level of detail.

### **3. General work**

**3.1** BIM software is becoming increasingly popular for their ability to combine the three-dimensional model with data from several disciplines associated to the design. It would be relevant to develop a mechanism capable of generating the same databases or to connect these models, developed exclusively using a programming-based representation, to one of the existing BIMs.

Briefly, future work can follow two different routes, continuing this work or developing new experiments. Both will contribute mainly for a better evaluation of the ability of the algorithmic approach to handling change and to develop methods to improve its application.

## **CONTRIBUTIONS**

This dissertation proves the relevance of Generative Design, in particular an algorithmic approach, as an auxiliary tool that supports decision-making activities in architectural design practice.

It proves its applicability in the context of complex design problems and the ability to use it starting from the conceptual phases until advanced phases of the design, including those where a high level of detail is required.

This approach is also useful to allow the exploration of different solutions, as the effort needed to define the initial model can be quickly recovered in the subsequent models, either from the same project, or from similar projects.

This dissertation also provides relevant lessons to the development of similar models.



## BIBLIOGRAPHY

- Alfaris, A. (2009) "Emergence through conflict: the Multi-Disciplinary Design System (MDDS)", PhD thesis, Massachusetts Institute of Technology.
- Berlinski, D. (1999) *The Advent of the Algorithm: The Idea that Rules the World*, New York: Harcourt.
- Boolos, G. and Jeffrey, R. (1999) *Computability and Logic*, London: Cambridge University Press.
- Burry, M. and Murray, Z. (1997) "Computer aided architectural design using parametric variation and associative geometry" in: *Challenges of the Future: ECAADE Conference Proceedings*, Vienna, Austria, pp. 257-266.
- Caldas, L. and Rocha, J. (2001) "A Generative Design System applied to Siza's School of Architecture at Oporto" in: *Proceedings of the Sixth Conference on Computer Aided Architectural Design Research in Asia*, Sidney, Australia, pp. 253-264.
- Carfrae, T. (2007) "Box of Bubbles", *Ingenia*, nº33, pp. 45-51.
- Celani, M. (2002) "Beyond analysis and representation in CAD: a new computational approach to design education", PhD thesis, Massachusetts Institute of Technology.
- Celani, M. and Kubagawa, B. (2007) "O Método Projectual de Andrea Palladio: uma implementação em VBA" in: *Proceedings of GRAPHICA 2007*, Curitiba, Brasil.
- Celani, M. and Vaz C. (2011) "Scripts em CAD e ambientes de programação visual para modelagem paramétrica: uma comparação do ponto de vista pedagógico" in: *Anais do V TIC 2011*, Salvador, Bahia, pp.1-13.
- Deuling, T. (2001) *Serpentine Pavillion – Case Study*, Available at: <http://www.collectivearchitects.eu/blog/77/serpentine-pavilion-casestudy&gt;%09%5Baccessed%095%09April%092013%5D.%20> [25/04/2013].
- Duarte, J. (2001) "Customizing mass housing : a discursive grammar for Siza's Malagueira houses", PhD thesis, Massachusetts Institute of Technology.
- Eastman, C., Teicholz, P., Sacks R. and Liston K. (2008) *BIM Handbook*, New Jersey: John Wiley & Sons, Inc..
- El-Khaldi, M. (2007) "Mapping boundaries of generative systems for design synthesis", SM thesis, Massachusetts Institute of Technology.

Fasoulaki, H. (2008) "Integrated Design: A Generative Multi-Performative Design Approach", SM thesis, Massachusetts Institute of Technology.

Frampton, K. (2003) *História crítica da arquitetura moderna*, São Paulo: Martins Fontes.

Frazer, J. (1995) *An Evolutionary Architecture*, London: Architectural Association Publications.

Gero, J. (1990) "Design Prototypes: A Knowledge Representation Schema for Design", *AI Magazine*, 11, nº4, pp. 26-36.

Hensmeyer, M. (2003) *L-Systems in Architecture*, Available at: <http://www.michael-hansmeyer.com/flash/l-systems.html> [25/04/2013].

Hudson, R. (2010) "Strategies for parametric design in architecture: an application of practice led research", PhD thesis, University of Bath.

Koder, S. (1994) "Interview with Peter Eisenmann", *Ars Electronica 1994*, Available at: [http://90.146.8.18/en/archives/festival\\_archive/festival\\_catalogs/festival\\_artikel.asp?iProjectID=8672](http://90.146.8.18/en/archives/festival_archive/festival_catalogs/festival_artikel.asp?iProjectID=8672) [25/04/2013].

Kalay, Y. (2004) *Architecture's New Media: Principles, Theories, and Methods of Computer-Aided Design*, Cambridge, Massachusetts: The MIT Press.

Kalay, Y. (2009) "The Impact of Information Technology on Architectural Design in the 21st Century" in: *Joining Languages, Cultures and Visions: CAAD Futures 2009, Proceedings of the 13th International CAAD Futures Conference*, Montréal, Canada, pp. 21-34.

Knight, T. (1999) *Applications in architectural design, and education and practice*, Available at: <http://www.shapegrammar.org/education.pdf> [25/04/2013].

Knight, T. (2000) *Shape Grammars in education and practice: history and prospects*, Available at: [http://www.mit.edu/~tknight/IJDC/frameset\\_abstract.htm](http://www.mit.edu/~tknight/IJDC/frameset_abstract.htm) [25/04/2013].

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

Leitão, A. and Santos, L. (2012a) "Programming Languages for Generative Design: A Comparative Study", *IJAC - International Journal of Architectural Computing*, 10, nº 1, pp.139-162.

Leitão, A. and Santos L. (2012b), "Programming Languages for Generative Design: Visual or Textual?" in: *Respecting Fragile Places - 29th eCAADe Conference Proceedings*, Ljubljana, Slovenia, pp. 549-557.

Lynn, G. (1998) *Animate Form*, New York: Princeton Architectural Press.

Mark, E., Goldschmidt, G. and Gross, M. (2008) "A Perspective on Computer Aided Design after Four Decades" in: *26th eCAADe Conference Proceedings*, Antwerp, Belgium, pp. 169-176.

Menges, A. (2006) "Instrumental Geometry", *Architectural Design*, 76, nº2, pp. 42-53.

- Mitchell, W. (1973) "Vitruvius Computatus" in: *Environmental Design Research, Proceedings of EDRA 4 Conference*, Stroudsbouurg, France, pp. 384-386.
- Mitchell, W. (1989) "Afterword: The Design Studio of the Future" in: *CAAD Futures Proceedings 1989*, Cambridge, Massachussets, USA, pp.479-494.
- Mitchell, W. (2001) "Vitruvius Redux: Formalized Design Synthesis in Architecture" in: Antonsson and Cagan (ed) *Formal Engineering Design Synthesis*, Cambridge: Cambridge University Press, pp. 1-19.
- Mitchell, W. (2004) "Foreword" in: Kalay (ed) *Architecture's New Media: Principles, Theories, and Methods of Computer-Aided Design*, Cambridge, Massachusetts: The MIT Press, pp. IX-XII.
- Oxman, R. (2006), "Theory and design in the first digital age", *Design Studies*, 27, n°3, pp. 229-265.
- Oxman, R. (2008) "Performance based Design: Current Practices and Research", *IJAC - International Journal of Architectural Computing*, 6, n° 1, pp. 1-17.
- Palermo, H. (2006) "O sistema dom-ino", SM thesis, Universidade Federal do Rio Grande do Sul.
- Park, K. and Holt, N. (2010) "Parametric Design Process of a Complex Building In Practice Using Programmed Code As Master Model", *IJAC - International Journal of Architectural Computing*, 8, n°3, pp. 359-376.
- Peters, B. (2007) "The Smithsonian Courtyard Enclosure" in: *Expanding Bodies: Art - Cities - Environment: Proceedings of the 27th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, Nova Scotia, Canada, pp. 74-83.
- Prusinkiewicz, P. and Lindenmayer, A. (1996) *The Algorithmic Beauty of Plants*, New York: Springer-Verlag.
- Santos, L. (2009), "Sistemas Generativos de Projecto: Integração de Ferramentas Digitais no Projecto de Arquitectura", SM thesis, Instituto Superior Técnico.
- Santos, L., Leitão, A. and Lopes, J. (2012), "Collaborative Digital Design: When the Architect meets the Software Engineer" in: *Digital Physicality - Proceedings of the 30th eCAADe Conference*, Prague, Czech Republic, pp. 87-96.
- Simon, H. (1973), "The Structure of Ill Structured Problems", *Artificial Intelligence*, 4, n°3, pp. 181-201.
- SOM – *Inside the Blackbox: SOM's Technological Trajectory*, Available at: <https://www.som.com/publication/inside-blackbox-soms-technological-trajectory> [25/04/2013].
- Stiny, G. and Gips J. (1978a) *Algorithmic Aesthetics: Computer Models for Criticism and Design in the Arts*, Berkeley: University of California Press.
- Stiny, G. and Mitchell, W. (1978b) "The Palladian Grammar", *Environment and Planning B*, 5, pp. 5-18.

Stocking, A. (2009) *Generative Design is Changing the Face of Architecture*, Available at: <http://www.cadalyst.com/cad/building-design/generative-design-is-changing-face-architecture-12948> [25/04/2013].

Szalapaj, P. (2001) *CAD Principles for Architectural Design*, Oxford : Architectural Press.

Woodburry, R. (2010) *Elements of Parametric Design*, New York: Routledge.