MASS CUSTOMIZING THE RELATIONS OF DESIGN CONSTRAINTS FOR DESIGNER-BUILT COMPUTATIONAL MODELS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SELEN ERCAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE INTERNATIONAL JOINT DEGREE OF MASTER OF SCIENCE
IN
THE DEPARTMENT OF ARCHITECTURE
IN
COMPUTATIONAL DESIGN AND FABRICATION
TECHNOLOGIES IN ARCHITECTURE
BY
MIDDLE EAST TECHNICAL UNIVERSITY
DELFT UNIVERSITY OF TECHNOLOGY

SEPTEMBER 2010

Approval of the thesis:

MASS CUSTOMIZING THE RELATIONS OF DESIGN CONSTRAINTS FOR DESIGNER-BUILT COMPUTATIONAL MODELS

submitted by **SELEN ERCAN** in partial fulfillment of the requirements for the degree of **Master of Science in Architecture Department, Middle East Technical University, and Delft University of Technology** by,

Prof. Dr. Canan Özgen ______________
Dean, Graduate School of **Natural and Applied Sciences**

Assoc. Prof. Dr. Güven Arif Sargın ______________
Head of Department, **Architecture**

Assoc. Prof. Dr. Mine Özkar ______________
Supervisor, **Architecture Dept., METU**

Examining Committee Members:

Prof. Dr. Can Baykan ____________________
Architecture Dept., METU

Assoc. Prof. Dr. Mine Özkar ____________________
Architecture Dept., METU

Assoc. Prof. Dr. Rudi Stouffs ____________________
Building Technology Dept., TU Delft

Assist. Prof. Dr. Bige Tunçer ____________________
Building Technology Dept., TU Delft

Onur Yüce Gün, M.Sc. ____________________
Computational Design Specialist

Date: 13.09.2010

ABSTRACT

MASS CUSTOMIZING THE RELATIONS OF DESIGN CONSTRAINTS
FOR DESIGNER-BUILT COMPUTATIONAL MODELS

Ercan, Selen

International Joint M.S. Programme of Computational Design and Fabrication Technologies in Architecture, in the Department of Architecture
Supervisor: Assoc. Prof. Dr. Mine Özkar

September 2010, 120 pages

The starting motivation of this study is to develop an intuitively strong approach to addressing architectural design problems through computational models. Within the scope of the thesis, the complexity of an architectural design problem is modeled computationally by translating the design reasoning into parameters, constraints and the relations between these. Such a model can easily become deterministic and defy its purpose, if it is customized with pre-defined and unchangeable relations between the constraints.

This study acknowledges that the relations between design constraints are bound to change in architectural design problems, as exemplified in the graduation project of the author. As such, any computational design model should enable designers to modify the relations between constraints. The model should be open for modifications by the designer.

The findings of the research and the architectural design experiments in the showcase project suggest that this is possible if mass customized sequences of abstract, modifiable and reusable relations link the design constraints with each other in the model. Within the scope of this thesis, the designer actions are mass-customized sequences of relations that may be modified to fit the small design tasks of relating specific design constraints. They relate the constraints in sequence, and are mass customized in an abstract, modifiable and reusable manner. Within this study, they are encoded in Rhino Grasshopper definitions. As these mass customized relations are modifiable, they are seen as a remedy for enabling the designers to build models that meet individual and intuitive needs of the design problems that designers define.

Keywords: Design Constraints, Computational Design Model, Designer Actions, Mass Customization

ÖZ

TASARIMCI TARAFINDAN GELİŞTİRİLEN SAYISAL MODELLER İÇİN TASARIM KOŞULLARI İLIŞKİLERİNİN SERİ ÖZELLEŞTİRİLMESİ

Ercan, Selen

Mimarlıkta Sayısal Tasarım ve Üretim Teknolojileri Uluslararası Ortak
Yüksek Lisans Programı, Mimarlık EABD
Tez Yöneticisi: Doç. Dr. Mine Özkar

Eylül 2010, 120 sayfa

Bu çalışmanın başlangıç motivasyonu, mimari tasarim problemlerini sayısal modeller ile ele almak için sezgisel anlamda güçlü bir yaklaşım geliştirmektir. Bu tezin bağlamında, bir mimari tasarım probleminin karmaşıklığı, o problemin tasarım düşüncesinin değişkenler (parameters), koşullar (constraints) ve bunların ilişkilerine dönüşümü ile sayısal olarak modellenebilir. Böyle bir model, eğer önceden tanımlı ve değiştirlemez koşul ilişkileri ile özelleştirilirse, kolayca amacından sapıp determinist bir yapıya bürünebilir.

Bu çalışma, yazarın mezuniyet projesinde de örneklendiği gibi, mimari tasarım problemlerinde tasarım koşulları ilişkilerinin değişmeye mecbur olduğunu kabul eder. Bu nedenle, tasarımcılar herhangi bir sayısal tasarım modeli ile, tasarım koşulları arasındaki ilişkileri değiştirebilmelidirler. Model, tasarımcının değişikliklerine açık olmalıdır.

Yapılan araştırmalar ve vitrin projede deneyimlenen mimari tasarım, bunun, seri özelleştirilmiş, soyut, değiştirilebilir, ve yeniden kullanılabilir ilişki sıralarının modelde tasarım koşullarını birbirine bağlaması ile mümkün olduğunu ortaya koymaktadır. Bu tezin kapsamında, tasarımcı eylemleri, belli tasarım koşullarını birbirine ilişkilendiren küçük tasarım görevlerini yerine getirmek üzere degiştirilip ayarlanabilen seri özelleştirilmiş ilişki sıralarıdır. Tasarımcı eylemleri, koşulları bir sıra ile ilişkilendirir ve soyut, değiştirilebilir, ve yeniden kullanılabilir bir şekilde seri özelleştirilmişlerdir. Bu çalışmada, ilişki sıraları Rhino Grasshopper tanımları içine yazılmıştır. Bu seri özelleştirilmiş ilişkiler değiştirilip ayarlanabilir oldukları için, tasarımcıları, kendi tanımladıkları tasarım problemlerinin sezgisel ve bireysel ihtiyaçlarını karşılayan modellleri geliştirebilir kılmak için çözüm olarak sunulmaktadırlar.

Anahtar Kelimeler: Tasarım Koşulları, Sayısal Tasarım Modeli, Tasarımcı Eylemleri, Seri Özelleştirme

ACKNOWLEDGMENTS

The author wishes to express her deepest gratitude to her supervisor Assoc. Prof. Dr. Mine Özkar for her encouraging criticisms, deepest insights and for her meaningful guidance and advice in integrating design and research.

The author would also like to thank her jury members Prof. Dr. Can Baykan, Assoc. Prof. Dr. Rudi Stouffs, Assist. Prof. Dr. Bige Tunçer and Onur Yüce Gün (M.Sc.) for their valuable critics, suggestions and comments.

The author would also like to acknowledge her studio tutors, for the graduation project in TU Delft: Ir. Huib Plomp for his leadership throughout the design process, and his endless patience and valuable comments regarding design reasoning; Ir. Sander Mulders, for his equally valuable encouragement and especially for his technical assistance regarding computational support; and Assist. Prof. Dr. Bige Tunçer, for her enlightening suggestions for computational/theoretical methods and her encouraging comments and critiques in the graduation project.

I am also grateful to my friends, especially Asli Deniz Özakar and Ekin Damdere, for their continuous support and for sharing all of my excitement and problems. I am grateful to Rafael Molina Planelles for his architectural eye, stimulating support in all terms. Finally, I am grateful to my family for their endless support and belief in me throughout this process.

TABLE OF CONTENTS

ABSTRACT..........iv

ÖZ..........vi

ACKNOWLEDGMENTS..........viii

TABLE OF CONTENTS..........ix

LIST OF FIGURES..........xii

CHAPTER

1. INTRODUCTION: WHAT TO MASS CUSTOMIZE AND WHY TO MASS CUSTOMIZE – THE SWITCHING ROLE OF THE DESIGNER..........1

    1.1 Mass Customization Overtaking Customization..........3

    1.2 Externalizing Individual Design Reasoning..........6

    1.3 Building an Intuitive Computational Design Model: The Efficient Translation of Design Reasoning..........7

2. THE NATURE OF PROGRAMMING IN ARCHITECTURE AND THE NATURE OF PROGRAMMING IN COMPUTATION..........10

    2.1 The Nature of Programming in Architecture..........10

    2.2 The Nature of Programming in Computation..........13

    2.3 Why Shall We Design with Changeable Entities?..........14

3. TRANSLATION OF DESIGN REASONING INTO RELATIONS BETWEEN THE DESIGN CONSTRAINTS: MODELING THE ARCHITECTURAL DESIGN SPACE..........17

    3.1 The Need to Externalize Design Reasoning in the Design Space..........17

    3.2 Externalizing Design Reasoning in Architectural Design..........26

3.2.1 Visual Representation..................................28

3.2.2 Numerical Representation.............................30

3.2.3 Translating Design Reasoning into the Relations between Design Constraints for Developing a Computational Design Model............33

4. DIFFERENT TYPES OF COMPUTATIONAL DESIGN MODELS: A CLASSIFICATION FOR THE ROLE OF THE DESIGNER..49

5. PROGRAMMING THE DESIGNER ACTIONS: MASS CUSTOMIZING SEQUENCES OF RELATIONS AND PROCESSING PARAMETERS IN A SPECIFIC ARCHITECTURAL DESIGN PROBLEM...64

5.1 Content and the Context of the Architectural Design Problem...68

5.2 Problem Space (Defining the Design Parameters)......74

5.3 Solution Space (Relating the Design Constraints with Each Other)……...80

5.3.1 Relating Number of Units to Water Management System……......................................82

5.3.2 Relating the Position of the Project Plot to Water Management System……..............................85

5.3.3 Relating the Position of the Project Plot to Noise levels and the Number of People Traveling……..90

5.3.4 Relating the Structure of the Surface Covering the Units to the Growth Pattern of the Agricultural Products……..95

5.3.5 Relating the Radius of the Openings in the Surface Covering the Units to the Number of Hours of Sunlight……..104

6. THE PROPOSED REMEDY IN MASS CUSTOMIZING PARAMETRIC AND CONSTRAINT BASED REALTIONS.........107

6.1 A Brief Summary and the Discussion Path……..........107

6.2 Mass-Customized Sequences of Relations Overtaking Pre-defined Relations in Parametric and Constraint-Based Design Models……..109

6.3 Conclusions and Contributions…………………….........112

6.4 Assessment of Limitations as a Further Study……....113

REFERENCES...116

APPENDICES

A. GRASSHOPPER DEFINITION OF THE GRADUATION PROJECT...120

LIST OF FIGURES

FIGURES

Fig 3.1. Two fabrications from the exhibition (FAB)BOTS in DHUB, curator: Marta Male-Alemany. Works by the master level students of “Machinic Control 1.0” design studio of AA and the “Digital Tectonics RS3” design studio of IAAC. These images can help to illustrate the tectonic properties of the “growth pattern of the agricultural product”...22

Fig 3.2. The example from Kilian’s study: Cone based approximation of a double curved surface using only developable surfaces. (Taken from: “Design Innovation through Constraint Modeling”, Digital Design: *The Quest for New Paradigms, 23rd eCAADe Conference Proceedings*, 2005, p.675, by Axel Kilian)..24

Fig 3.3. Numerical representation (taken from “The Future Information Modeling and the End of Theory: Less is Limited, More is Different”, by Cynthia Ottchen, *AD issue on: “Closing the Gap”* (Guest Ed.) Richard Garber. 79(2), Wiley & Sons, March-April 2009, pp.22-27.....................32

Fig 3.4. Planar adjacency graph (Taken from “Automated facilities layout: past, present and future,” *Automation in Construction*, vol.9, 2000, p.199, by Robin S. Liggett)..44

Fig 3.5. Relations of the Rectangle Algebra, RA (Taken from: “Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements,” by Can Baykan in *E-Activities in Design and Design Education*. ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız Paris: Europia Productions, 2003, p.140)...............................46

Fig 3.6. Types of distance between two rectangles (Taken from: “Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements,” by Can Baykan in *E-Activities in Design and Design Education*. ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız Paris: Europia Productions, 2003, p.141)...............................46

Fig 3.7. Non-trivial, trivial, internal and external holes, and a kitchen layout violating the access requirement (Taken from: “Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements,” by Can Baykan in *E-Activities in Design and Design*

Education. ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız Paris: Europia Productions, 2003, p.142)..46

Fig 3.8 An exemplar image of a pattern definition from the design patterns study of Ramesh Krishnamurti and Tsung-Hsien Wang (http://www.andrew.cmu.edu/org/tsunghsw-design/) inspired by the design patterns study of Robert Woodbury (http://www.designpatterns.ca/)…………………………………………......48

Fig 4.1 Different models that use different translations (for design reasoning) into the components that compose them. (a) The translation that FloP uses: This matrix has to be formed to define the rectangle relations between every two elements of the design problem until every element has a defined relation with the rest. (b) The translation that is used in Stadsbalkon: “designtoproduction developed a software based on artificial-life methods, which optimized the exact position, inclination and strength of some 150 irregularly placed columns according to functional and constructional requirements, and which could be utilized by the architects to generate alternative design solutions.” (Taken from: http://www.designtoproduction.ch/content/view/7/27/) (c) The translation that is used in the design research of adaptable structures: The modules that are interpreted as embodying the recursive functions used for the parametric modeling of adaptable structures. (d) Various design parameters explored via constraints exemplified below. Design reasoning is partly translated into them in the particular models in consideration. The models that belong to the first group are bound to work with predefined parameters, constraints and their relations...........................50

Fig 4.2. (a) Designers should be able to operate on the translation of design reasoning into the parametric and constraint based relations. They should be free to decide which constraint should relate to which other constraint. (b) For parametric and constraint based models, the reasoning is translated into the relations defined for design parameters (c) that are explored by design constraints. (d) If designers are not able to operate on the translation of design reasoning into the parametric and constraint based relations, the model might become a deterministic one................52

Fig 4.3. Image showing the situation for the first group. The model belongs to ‘Stadsbalkon’ developed by Fabian Scheurer. Stadsbalkon: “designtoproduction developed a software based on artificial-life methods, which optimized the exact position, inclination and strength of some 150 irregularly placed columns according to functional and constructional requirements, and which could be utilized by the architects to generate alternative design solutions." (Taken from: http://www.designtoproduction.ch/content/view/7/27/)............................53

Fig 4.4. In FloP, constructing the automated design model is managed by defining all possible rectangle relations in terms of their constraints. This matrix has to be formed to define the relations between every two elements of the design problem until every element has a defined relation with the rest. For design generation with FloP see Fig5 in 2.2.2 (Taken from: "Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements", by Can baykan in *E-Activities in Design and Design Education*. ed. Bige Tunçer, Șaban Suat Özsarıyıldız, Sevil Sarıyıldız, Paris: Europia Productions 2003, p.143).......................54

Fig 4.5. The modules that are interpreted as embodying the recursive functions used for the parametric modeling of adaptable structures in the design research of M. Turrin, A. Killian, R. Stouffs, and S. Sarıyıldız (Taken from: "Digital Design Exploration of Structural Morphologies Integrating Adaptable Modules: A design process based on parametric modeling", *Proceedings of the 13th International Conference on Computer Aided Architectural Design Futures*, 2009, pp. 800-814)........57

Fig 4.6. (a) Full Grasshopper definition developed for the model of the graduation project. Different colors refer to different functions that belong to different scales. They embody constraints belonging to different scales. (b) Some parameters explored via constraints are exemplified. These are only one part of the parameters that the design elements are translated into. Author can modify both the parameters and the design elements. The definition is capable of changes. It responds to the changes with a new model because the functions are encoded with abstract, modifiable and reusable relations that relate the design constraints following a sequence...59

Fig 4.7. Spiraling used in an architectural design project (Taken From: *Pamphlet Architecture 27: Tooling*, by B. Aranda, C. Lasch, New York: Princeton Architectural Press, 2006, p.19)...61

Fig 4.8. (a) Packing which executes the task of bringing "stability through adjacency" (b) Square packing (c) Recipe for packing (Taken From: *Pamphlet Architecture 27: Tooling*, by B. Aranda, C. Lasch, New York: Princeton Architectural Press, 2006, p.26, 25, 23) 62

Fig 5.1. Schiphol Airport and the adjacent agricultural lands: These are the dominant land uses in the region, and are surrounded by a mobility network..66

Fig 5.2. One part of the mobility network of the Schiphol region. Different means of transportation are highlighted, including the means of connecting Haarlem Station to Schiphol Airport. An area of 7.1 km^2 has been chosen for the architectural design problem, which is indicated in red. The area is located on the southeast of Haarlem, southwest of Amsterdam and on the northwest of Schiphol Airport.............................67

Fig 5.3. (e) A project plot (indicated in green) measuring 100m x 100m is selected from the area, which is indicated in red (measuring 7.1 km^2) in Fig 5.2. (a) The curve indicating project plot can be seen in the Rhino scene. It is not yet parameterized. (b) The curve parameter, which is created to parameterize project plot, is indicated in orange. (c) Two units are generated when a curve is set for the parameter “project plot”. Number of units depends on the constraints such as those exploring noise levels, climatic conditions or wind, light, etc., for different locations of the project plot. (d) The curve parameter, which is created to parameterize project plot, is now indicated in green because a curve is set in the parameter..67

Fig 5.4. Randstad of the Netherlands within the European mobility network..69

Fig 5.5. Process of reclaiming land (a) the land is below sea level, (b) pumping stations drain the seawater and (c) a polder system is introduced, including agricultural fields and farmhouses.........................69

Fig 5.6. The polder system (Original image taken from: *Man-Made Lowlands: History of water management and land reclamation in the Netherlands,* by G. P. van de Ven (ed.) Utrecht, The Netherlands: Uitgeverij Matrijs, 2004)...70

Fig 5.7. Reclamation of the Haarlemmermeer Polder in 1852 (Original images taken from: *Architecture Annual 2002-2003: Delft University of Technology*, ed. Henco Bekkering, The Netherlands: 010 Publishers, 2004 and *Sea of Land: The polder as an Experimental Atlas of Dutch Landscape Architecture,* by W. Reh, C. Steenbergen, D. Aten. Wormerveer-Zaanstad, The Netherlands : Stichting Uitgeverij Noord-Holland, 2007)...70

Fig 5.8. The parking lot at Schiphol Airport (Capacity: 16,000 cars)..71

Fig 5.9. Image showing different species cultivated in the Netherlands..72

Fig 5.10. The parking lot of Schiphol Airport (top) is an important infrastructural element of the mobility network in the region, as is the agricultural land (bottom). Agricultural land is under threat from several different forces that are trying to find architectural expression in this region..73

Fig 5.11. A sketch externalizing the reasoning related to the dynamics of the site, taking into account Schiphol Airport. Although the reasoning has been put to paper, it has not yet been translated into the computational components of the model...74

Fig 5.12. Different scales indicate different groups of constraints. The sketches positioned below different groups of constraints indicated by different scales show the instances of design reasoning that are to be translated into the particular components under consideration...............75

Fig 5.13. The polder module at 1/10000 scale: The group of constraints belonging to the 1/10000 scale drives the design. One polder module covers an area of 2km x 3km for the Haarlemmermeer Polder…...........77

Fig 5.14. . The polder block (all of the darker gray area in Fig 5.13) at 1/1000 scale: The design is driven by the group of constraints belonging to scales 1/1000, 1/100 and 1/10, which are modeled in relation to each other. One polder module covers an area of 1km x 3km for the Haarlemmermeer Polder...77

Fig 5.15 The polder parcel (all of the blue area in Fig 5.14) at 1/1000 scale: The design is driven by the group of constraints belonging to scales 1/1000, 1/100 and 1/10, which are modeled in relation to each other. One polder module covers an area of 1km x 200m for the Haarlemmermeer Polder...78

Fig 5.16. The project plot (one of the red squares in Fig 5.15) at 1/100 and 1/10 scales: The design is driven by the group of constraints belonging to scales 1/1000, 1/100 and 1/10, which are modeled in relation to each other. The project plot covers an area of 100m x 100m for every different position in which it is placed. Every time the place of the project plot is changed, an area of 100m x100m is dealt with for the changes in the values of parameters...78

Fig 5.17. This image explains the model developed by the author for the graduation project in which design parameters are explored via constraints. The various constraints belonging to different groups are shown in different colors. The relations formed between the constraints that belong to the same or to different groups are shown with lines and arrows..80

Fig 5.18. This sketch shows the constraints that are used to ensure a continuation of the existing rules of the water management system when draining one project plot, and to support author’s scenario (in such actions as changing the position of the project plot)...............................81

Fig 5.19. The “number of units” in the project plot determines the weight of the construction, which then determines the amount of water to be drained. If 6 units are to be constructed on the project plot, its physical reflection on the overall design is that, corresponding to the function developed and encoded in the Grasshopper definition, (b) the existing ditch line is offset to a specific value of width – that is 1.2m – according to the increasing weight of the construction to be made on the plot. Existing ditch line and the offset line are indicated in red and in green, respectively. (a) Existing ditch line is indicated in green. If 2 units are to be constructed on the project plot, then the width of the ditch is calculated accordingly. The effect of the change in the “number of units” on the “width of ditches” can be seen. (c) The value of width is calculated by multiplying offset value (2m) by the “number of units” and by dividing by 10, which results in 0.6. Offset value, and the “number of units” are two of the design elements that are parameterized in the function. Both of the slider and the panel are indicated in green. Offset value is parameterized with a number slider in the function, by which the author sets different floating points...83

Fig 5.20. Image showing the primary and secondary drainage canals (in blue on the left) and ditches (in blue on the right) of the selected site of 7.1 km^2, shown in red in Fig 5.2...84

Fig 5.21. Design reasoning regarding the dynamics of the water management system, externalized in a physical model..........................84

Fig 5.22 The Grasshopper model of a re-forming of the ditches after different changes in position of the project plot can be seen. In the created definition, the ditches are re-formed in each case 100m from the borders of the project plot (this interval is slightly larger in these examples owing to the larger size of the project plot used). (b) The initial position of the project plot is shown in yellow hatching. Each of the above examples show different changes in position, being: (f) moving the plot along the xy plane or (e) rotating the plot in a counter clockwise direction or (d) moving the rotated plot again along the xy plane or (c) rotating the relocated plot in a clockwise direction. (a) The scale of the design model can be seen at the top..85

Fig 5.23. The difference between optimizing the grid (a) to 100m and (b) to 50m can be seen. (c) The panel, which is created to parameterize the value of optimizing the grid, can be seen in green. The value is set to 100m. (d) The panel, which is created to parameterize the value of optimizing the grid, can be seen in green. The value is set to 50m.........86

Fig 5.24. Significant parameterizations at 1/1000 scale can be seen: (a) “Selected site” in the Schiphol region, (b) the optimizing of the grid of the “selected site”, (c) and the “position of project plot”. “Position of project plot” is related to the other constraints at 1/100 scale. It links the two scales. In 1/100 scale it is linked (d) to the circle packing function, (e) to the function that calculates the effect of the noise level, (f) to the function that calculates the “width of ditches”, and (g) to the function that calculates the building density in the plot. (h) It is also related to other parameters in the same scale that is 1/1000. In all, the particular curve that has been set into the “project plot” parameter, is also set into the “curve” input parameters’ of the various components of these functions...88

Fig 5.25. (b) The Grasshopper model of the plan in the event of a change in the “position of project plot”. The project plot can also be seen in Fig 5.15. The function encoded in the definition calculates the “number of units” every time the position of the project plot changes - yet the position change is explored not by moving the project plot to various locations in the site, but by changing the values of parameters such as: “noise contour level” or “number of passing cars”, etc. This definition uses circle packing to find the maximum “number of units” that can be constructed on the project plot. (c) Constructed units for a specific situation (for a specific noise level, etc.) and the “number of stories per unit” in that specific situation can be seen. (d) Each time the position of the project plot changes; a different surface covering each unit is generated as well. The design of these surfaces will be explained later. (a) The scale of the design model to which the function belongs can be seen at the top in red...89-90

Fig 5.26. Significant parameterizations and significant relations of parameters, number sliders, panels and components at 1/100 scale can be seen in the Grasshopper definition. Numbers from 1-10 indicate the significant relations in the sequence that encodes the function. Letters from b-k indicate significant parameterizations, including 2 parameters, 1 panel, 6 number sliders from b-h, by which the author inputs the values to the function. The rest, being i, j, k, are the 3 panels created to parameterize the values depending on the values of b, c, d, e, f, g, h. To be specific, these are; (b) 2 number sliders and 1 point parameter that determine the effect of the noise source, (c) which has been parameterized with the curve parameter named "NOMOS99_Lijnden", indicating the station located in Lijnden; (d) the panel created to parameterize the size of one unit being 10m x 10m; (e) the number slider created for the value of offsetting to calculate the "width of ditches;" (f) the number slider created for the "number of passing cars;" (g) the number slider created for the value of "floor height;" (h) the number slider created for the value of the number of stories in units; (i) the panel created for "number of units;" (j) the panel created for "m^2 of land occupied on agricultural land or pasture;" (k) the panel created for the maximum height of the units, which should not exceed certain limits. (a) On the left, the curve parameter, which is created to parameterize "position of project plot", can be seen..91

Fig 5.27. The part of the design scenario related to the infrastructure on the site and its parameterization: This parameterization involves the translation of design reasoning into "width of ditches," "noise contour level," "distance between units," "m^2 of land occupied on agricultural land or pasture," "number of people traveling," "number of units," "number of stories per unit," "building density," 'building height," etc., and into the expressions defined between these...94

Fig 5.28. Plans, sections and internal views of the units in the new system. At the bottom, different units of the project plot can be seen..96-97

Fig 5.29. (a) Pipe work for the hydroponics can be seen. (b) The growth pattern of tulips, which has driven the design of the pipe work in which they are grown. (c) Four elements compose the units of the new system: pipe work system for hydroponics, tubular structure for the inner and outer needs of the units, slabs and Etfe panels......................................98

Fig 5.30. One of the fabrications of the first surfaces generated to cover the units can be seen on the top. Other surfaces are fabricated using different values for the parameters "radius of the openings in the surface covering the units" and with one specific growth pattern, etc., as can be seen below. 99-100

Fig 5.31. The grasshopper definition encoding the function that generates the structure of the surface covering the units can be seen. Numbers from 1-6 indicate significant sequences of relations in the definition. From a-i, except c and i, the components of the function, which embody significant parameterizations of the design scenario in their output parameters, can be seen. c and i are two number sliders and one panel by which the author inputs the values directly. The other components depend on the values of parameters, panels or number sliders that they are linked to. The sequences of relations, named “Etfe panels,” “tubular structure” and “pipe work system”, can be seen at the bottom. These names indicate the parts of the function that generates the structure of the surface. The scale of the design model to which the definition belongs can be seen in Fig 5.17 in red..103

Fig 5.32. The design of the surface is driven by the constraints that explore parameters such as “m^2 of land occupied on agricultural land or pasture,” which determines the dimensions of the surface. The sequence is programmed as such: the “growth pattern of the agricultural product” and the (c) “number of hours of sunlight” determine the (a) “length of the members of the structure of the surface covering the units” and the (b) “radius of the openings in the surface covering the unit,” respectively. (a) The structural constraints for the members of the structure of the surface covering the units are as follows: c+d = a’+b,’ where a, b, c, d and a’, b’, c’, d’ are the lengths of the members. (d) The openings in the surface can be seen below. (c) “Number of hours of sunight” are calculated in Ecotect and (e) the results are transferred back to Grasshopper using the add-on Geco, developed by [Uto], Ursula Frick and Thomas Grabner (http://utos.blogspot.com/) (f) With this add-on, the “number of hours of sunlight” for any defined period of time, is calculated. Specifically, these calculations determine the optimum values of the radius of the openings in the surface, which have to provide sunlight in the right amount and on the right day of the year considering, the needs of the tulips, for example. The lighting needs of the occupants are also taken into consideration. (g) The lighting qualities of the inner spaces can be seen in the centre...105

CHAPTER 1

INTRODUCTION: WHAT TO MASS CUSTOMIZE AND WHY TO MASS CUSTOMIZE – THE SWITCHING ROLE OF THE DESIGNER

In this thesis, mass customization is deemed to be significant on two levels: firstly, it allows designers to build their own computational design models; and secondly, it permits the introduction of computational design models into architectural design knowledge.

Beginning with the first point, in this thesis, for developing computational design models, the designer is required to relate each design constraint to every other constraint. The actions of a designer in this regard can be considered as the formation of mass customized relations between design constraints. These relations are abstract, modifiable and reusable, and relate the design constraints in sequence. For this study, in mass customization, the sequences are encoded in Grasshopper definitions, as will be explained later in the study. In summary, these mass customized relations may overcome the problems faced when developing models to address the individual and intuitive needs of the design problem; may enter general usage in architectural design problems. The designers should either take the opportunity to mass customize these relations, or a library/menu of these relations should be available to them to modify.

Coming to the second point of significance, being the introduction of these computational design models into architectural design knowledge, if put into computer – if implemented as applications or some software –, these models may be used by other designers as CAAD tools. Moreover, the developers of the design models may adopt them as well for their own use in different contexts, scales, etc. It is suggested in this study that, designers are able to develop computational design models by carrying out a mass customization of the way they design, and they develop computational design models for the mass customization of the way they design; and the resulting model may be adopted in the application of their architectural design to different situations.

Researchers such as Sait Ali Köknar and Arzu Erdem have broached the subject in the past, claiming that designers work with thinking tools and use existing packages of knowledge when designing.[1] Designers may use existing models (from the projects of different practices using computational support, etc.), and follow the sequence of relations that they have introduced. The reasoning of the architectural design problem can be externalized into existing packages of architectural design knowledge, and may draw upon existing models for reference. This is related to the significance of mass customization raised in this research when the need of translation is considered.

For this study, parametric and constraint-based design models are adopted for externalizing design reasoning. A model is built by translating the reasoning of the designer into a sequence of relations between design constraints. Such a translation is relevant for the particular method being applied by adopting parametric and constraint-based design models. Should another computational method be applied then the reasoning of the designer may be translated into something other than the relations between design constraints. It should be noted that the externalization of reasoning into existing packages of architectural design knowledge is a general issue that exists for all architectural design problems. Accordingly, designers design and communicate through "production-like encodings of extant and newly learned actions,"[2] as referred to by Robert

[1] Sait Ali Köknar, Arzu Erdem. "Can creativity be institutionalized?", *A-Z ITU Journal of the Faculty of Architecture*, 4(2), 2007, p.28

[2] Robert Woodbury, Andrew Burrow. "Whither design space?", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 20(2), 2006, p.68

Woodbury and Andrew Burrow.

There are a number of crucial questions that should be raised concerning the differences between customization and mass customization. For instance, is it possible for designers to develop a computational design model and put it into a computer from which they will be able to draw support in defining a design process that can be customized for each new design problem?[3] In “A Perspective on Computer Aided Design after Four Decades,” Earl Mark, Mark Gross and Gabriela Goldschmidt reiterate this question, which was first raised by Negroponte. This thesis researches the further question of whether it is possible to develop such a model without falling into the trap of determinism, resulting in the designer becoming a mere end user. These inquiries point out the differences between customization and mass customization.

1.1 Mass Customization Overtaking Customization

In customizing architectural design problems to achieve a simpler and easier automated generation of design solutions, determinism can be a fallback. As suggested by Alexander Koutamanis in his paper, “A Biased History of CAAD: The Bibliographic Version,” CAAD does not derive from the distinct ambition of the various attempts to automate the production of architectural designs. To back up this claim, he looks into history of CAAD,[4] and finds that while automation in design has been popular in different aspects since the 1970s,[5] the aim has always been to integrate computational methods – such as parametric and constraint-based design – into architectural design. Sometimes the intention may have been to

[3] Earl Mark, Mark Gross, Gabriela Goldschmidt. “A Perspective on Computer Aided Design After Four Decades”, *eCAADe 26: architecture ‘in computro’,* ed. M. Muylle, 2008, pg.171

[4] Alexander Koutamanis. "A Biased History of CAAD: the bibliographic version", in, J.P Duarte, G Ducla-Soares & A.Z Sampaio (Eds.), *eCAADe 23: Digital Design: the Quest for New Paradigms*, p.630

[5] *ibid.*, p.632

replace the designer, as Negroponte suggested in 1975; or sometimes only to provide support; but it would be fair to say that the conventions and assumed principles of architectural design have always been at the forefront. If the actual intention is integration concerning the conventions, the externalization of individual design reasoning may emerge as a problem. A designer drawing upon the support of computational methods to model the complexity of architectural design problems may be unable to take advantage of the components of the computational design models if they are unchangeable. The relations of design constraints are included in the computational components of the particular models under consideration, and they should be changeable and open to modification by the designer.

If the intention is to customize on a problem-by-problem basis, then the result may only be the provision of a model that the designer can play with, and may easily result in the creation of a CAAD tool that the designer is unable to modify. In this case, the result may be a deterministic model that may not be sufficient to encompass the complexity of the way each individual designer addresses architectural design problems. Additionally, it may not be compatible with the character of the architectural design problem, if the variety of parameters and constraints that can exist in a translation of a specific design reasoning of a specific architectural design problem is taken into account. Within the scope of this thesis, the translation of design reasoning into the computational components is crucial in this respect. It is how a designer defines the design parameters, constraints and the relations between them and it is considered as a part of the way they design. These components should not be customized, as resulting models may not be open to modification by the designers and thus may hinder the designers in developing design models that are able address the problems that they define.

Designers should be provided with possible sequences of relations, allowing them to develop their own models. These are called designer actions, meaning mass customized sequences of relations that are abstract, modifiable and reusable, and relate various design constraints to each other. They are not customized for specific design constraints, and thus bring a changeable character to the model. If an attempt to generate architectural design solutions is supported by a computational design model, then mass customization may prove to be advantageous for the designers in accessing the relations between design constraints, and making the designer the builder of the model. Mass customization enables designers to build models that meet the individual and intuitive needs of the design problem that they are addressing.

Since the 1980s, the intention has been to address all kinds of problems, from making a simple line drawing to explaining and supporting the use of design precedents, as stated by Koutamanis.[6] This necessitates an ability to shift between different scales of architectural design problem, for which a more comprehensive approach may be required that goes beyond customization, referred to here as mass customization. This scale shift is discussed later in depth as an important proposal of the thesis regarding mass customization. To clarify here, the shift is packaged and explained in 1/1000, 1/100 and 1/10 scales of the graduation project of the author, which is being explained in Chapter 5. In this project, rather than customization on a problem-by-problem basis, the author translates the reasoning of the design problem into the relations of design constraints through mass-customized sequences of relations. The generality and usefulness of this approach lie in the ability of the author to act as the

[6] Alexander Koutamanis. "A Biased History of CAAD: the bibliographic version", in, J.P Duarte, G Ducla-Soares & A.Z Sampaio (Eds.), *eCAADe 23: Digital Design: the Quest for New Paradigms*, p.632

builder of her own model.[7]

In this research, the intention is not to come up with a CAAD tool that is specific for one architectural design problem, but rather to offer suggestions of how designers may translate their reasoning on one or more scales of their design problem into relations between design constraints, and thus enable them to come up with a computational design model that responds to different scales of the design problem as a whole. The intention in describing the sequence of relations that may exist between design constraints in such a model is to suggest how designers can be the builders of their own models.

1.2 Externalizing Individual Design Reasoning

Within this research, it suggested that designers should define the relations of design parameters and choose how constraints are related to each other. Even though no computational method exists that is able to generalize the complexity of architectural design,[8] some may be applied to allow the building of a computational model. Axel Kilian, in his paper "Design Innovation through Constraint Modeling," claims that constraints in a design space exist in many forms, but that no master model exists that can incorporate them all.[9]

[7] In *UN Studio: Design Models - Architecture, Urbanism, Infrastructure* by UN Studio, the paradigmatic shift between the need of building a design model rather than designing a building is discussed with several different cases and categories. Ben Van Berkel, Caroline Bos, *UN Studio: Design Models - Architecture, Urbanism, Infrastructure*, Thames and Hudson, 2006. This point on "tool ward thinking" rather than "design ward thinking" may as well be related to Sait Ali Köknar's point of view in "Architectural Design Tools: Toward a non-linear design process", *Proceedings Designtrain Congress Trailer I*, 2007, pp.180-181.

[8] Generative design systems may also be an efficient method of grasping the complexity of architectural design, although there is not only one method to analyze its design space. The reason for that, as Scott Chase suggests in "Generative design tools for novice designers: Issues for selection," is that generative design tools introduce key concepts that are the foundations of design (p.697). Moreover, generative design systems work with operations that are completed by using a number of "tools that perform only one task, but doing it well" in sequence, as also put forward by Chase (p.689), and address the design problem piece by piece.

[9] Axel Kilian. "Design Innovation through Constraint Modeling", *Digital Design: The Quest for New Paradigms, 23rd eCAADe Conference Proceedings,* 2005, p.678

Individual design reasoning necessitates an individual translation. For instance, in the graduation project, the land is expressed in terms of m^3 of water that need to be drained. This is one way to translate the particular reasoning. Every designer does this differently, making each architectural design project an architectural design tool, as Sait Ali Köknar and Arzu Erdem explain in their studies.[10] This is very similar to the approach of this thesis, which seeks to assist designers in developing their own models.

1.3 Building an Intuitive Computational Design Model: The Efficient Translation of Design Reasoning

The generality and usefulness of the model developed in the author's graduation project lies in the attempt to translate design reasoning into the computational components[11] through mass customized sequences of relations. These are encoded in Rhino Grasshopper definitions.

If a translation of reasoning into parametric and constraint-based relations is required, then difficulties with incompatibilities may emerge. Most architectural design problems "do not easily translate into an equation format," as Axel Kilian explains.[12] There will always be incompatibilities when translating architectural design reasoning into computational components, simply because it defies "easy mathematical expression," as mentioned by Mark Gross, Stephen Ervin, James Anderson and Aaron Fleisher in their studies.[13] What's more, in architectural design, constraints

[10] Sait Ali Köknar, Arzu Erdem. "Can creativity be institutionalized?", *A-Z ITU Journal of the Faculty of Architecture*, 4(2), 2007, p.32

[11] The author builds relations by "expressing relationships and dependencies among quantities [parameters] in the design". Earl Mark, Mark Gross, Gabriela Goldschmidt. "A Perspective on Computer Aided Design After Four Decades", eCAADe 26: architecture 'in computro', ed. M. Muylle, 2008, pg.170

[12] Axel Kilian. *op.cit.,* p.672

[13] Mark Gross, Stephen Ervin, James Anderson, Aaron Fleisher. "Constraints: Knowledge representation in design", *Design Studies,* 9(3), 1988, p.138

and objectives are seldom stated at the outset,[14] but rather emerge during the design process. In a way, the problem space and the solution space co-evolve.[15]

The choice of computational method can directly affect the design space. Different computational methods bring with them different computational components to translate the reasoning into. If some particular CAD software is used to build the computational design model, for instance, then the entities that the software affords will shape the design space as well.[16]

The synthetic features of the design model and the different scales of the architectural design problem can be related. The development of a computational design model is possible through the full translation of design reasoning into sequences of relations, which necessitates the efficient conflict and concurrence of design constraints.[17] The model developed in the graduation project aims to find a way to keep these synthetic features of the model while shuffling between (as Christopher Alexander calls it) different scales (such as 1/1000 polder block scale or the project plot scales of 1/100 – building scale – and 1/10 – building component scale) and between different groups of constraints in the architectural design problem.

The constraints and parameters of the graduation project can be

---

[14] Earl Mark, Mark Gross, Gabriela Goldschmidt. "A Perspective on Computer Aided Design After Four Decades", *eCAADe 26: architecture 'in computro',* ed. M. Muylle, 2008, pg.171

[15] It is also believed that "if we can describe some constraints—sizes of things, spatial relationships among built and open elements in the building — then we can program a computer to manage them." *ibid.*

[16] *ibid.*, pg.169

[17] Christopher Alexander. *Notes on the Synthesis of Form*, Paperback Ed. Cambridge, MAs: Harvard University Press, 1971, Preface. This 'efficient conflict and concurrence of constraints' is exemplified on several occasions in this thesis.

exemplified as such: the “width of ditches” or ”number of units” or “m^2 of land occupied on agricultural land or pasture” or “number of hours of sunlight” or the “growth pattern of the agricultural product” on a numerical basis, including minimum/maximum distances between two seeds, tectonic properties or solid-void proportions of the same growth pattern, or the “radius of the openings in the surface covering the units” or the “length of the members of the structure of the surface covering the units,” etc. While developing a computational design model (in which an ability to shift between different groups of constraints at different scales is a must), the relating of different constraints, whether in conflict or concurrence, gains utmost importance.

The main text of the thesis begins with a comparison of the nature of programming in architecture and the nature of programming in computation to maximize designers’ involvement in the building of the design model, and to discuss a reconsideration of the word “programming”. Tool building is mostly related to programming in its conventional sense. By revisiting this term, the problem of incompatibilities between computational methods and architectural design may be resolved. The thesis proposes the sequencing of relations that is called programming and involves mass customization, as a remedy for the efficient translation of design reasoning into parameters, constraints, and the relations between them.

CHAPTER 2

THE NATURE OF PROGRAMMING IN ARCHITECTURE AND THE NATURE OF PROGRAMMING IN COMPUTATION

This chapter aims reconsidering the issue of program in architectural design, and focuses on what designers propose when suggesting a program for an architectural design problem. The intention here is to highlight the findings related to the role of design reasoning, which indicate that it may be possible to translate design reasoning into the relations between design constraints, although this translation is not the same as the mass customization of sequences of abstract, modifiable and reusable relations of these constraints.

2.1 The Nature of Programming in Architecture

The translation of design reasoning into relations between design constraints requires the perspective of the designer, and involves programming in architecture – the proposal or choice of designer actions that are needed for smaller individual tasks in a complex architectural design problem. As a prerequisite, programming requires the mass customization of sequences that relate the design constraints to each other. The outcome of this mass customization is designer actions.

The issue of program in architecture has been revisited on several occasions. Anthony Vidler states that the "[program should] be flexible and adaptive, inventive and mobile in its response to environmental conditions and technological possibilities".[18] Vidler's approach can interpreted as such: If the architectural program can be defined in terms of the parameters, constraints and actions that need to be taken to relate these constraints, then design reasoning can be translated into relations between design constraints comprehensively and efficiently.

Moreover, for defining the program of an architectural design problem in terms of parameters, constraints and the actions that need to be taken to relate them, a brief article entitled "Algorithmic Design" on the Yazdani Studio can be referenced. There, a clear description of program that can be interpreted as such can be found. In summarizing the studio's work on a concert hall, the issue of defining the program (in terms of parameters, constraints and the actions that need to be taken) is also explained:

> *"In the traditional approach, the designer sees the building envelop as "window" and "wall", and he tries to place and size the window based on his experience and his analysis of various factors (view, sun, wind, pattern, etc.). In this new approach we created software, which could analyze multiple inputs (view, sun, wind, pattern, etc.), and automate the "design" of an exterior envelop which is no longer a binary relationship of "window" and "wall" but which becomes a continuous gradient between "window-ness" and "wall-ness."*[19]

In the author's graduation project, agricultural constraints can be related to structural and lighting constraints in different ways, depending on the individual reasoning of the author. In doing so, the aim may not have been

[18] Anthony Vidler. "Toward a Theory of the Architectural Program", 2003, pg.60, Summerson also defines the architectural program as "the description of the spatial dimensions, spatial relationships, and other physical conditions required for the convenient performance of specific functions", pg.63 (the literature on architectural program is taken from the literature survey of Ülkü Özten)

[19] Yazdani Studio of Cannon Design. "Algorithmic Design", *a+u special issue on: "Architectural Transformations via BIM"* (Eds.) Tomohisa Miyauchi, Noriko Tsukui, Atsuko Nishimaki, Kei Yokoyama. August 2009, pp.34-39.

the automation of design in one single step, but rather to develop an approach to building a model that supports the author in exploring the design space for all scales of the architectural design problem. To clarify, in using the computational support of parametric and constraint-based design models, designers have to translate reasoning into the computational components of these models, and this requires the definition of the program in terms of parameters, constraints and the actions that need to be taken, which is interpreted as programming in architecture.

This thesis also aims demonstrating how the mass customized sequences of relations may form a library[20] or a menu for designer-built parametric and constraint-based models. To clarify, if designers choose to develop parametric and constraint-based models, they may also work with similar recursive functions, as these functions can be shaped through various reasoning, meaning by which different sets of constraints can be related to each other. Therefore, in the design space of parametric and constraint-based models, common recursive actions may be encoded for architectural design problems.

In every program proposed for an architectural design problem, there exist different elements of design reasoning that need to be externalized. Moreover, designer actions should to be encoded in the design space to fit the descriptions of the various design tasks of a complex architectural design problem. They should be useful and general enough to respond to the specific needs of the designers if modified. Designers decide upon the tasks to be executed, and should be able to adapt the sequences if

[20] In "Learning Computing by Design; Learning Design by Computing", *Proceedings Designtrain Congress Trailer I,* 2007, p.104, Mine Özkar states that "the device for creating the library is the medium of orthographic drawing", which in this thesis is the "plan" (of the graduation project of the author, explained in Chapter 5) but modeled in Rhino Grasshopper with an intermediary facet of parametric and constraint-based relations that translate designers' reasoning in the model.

needed. Therefore, in the menu or library of relations, the sequences must be mass customized in an abstract, modifiable and reusable manner.

2.2 The Nature of Programming in Computation

Tool building/making in architectural design is mostly related to the components of programming (languages) in computation. As such, it is necessary to discuss the nature of programming in computation in terms of what it brings to (or takes from) the operation of the designer as the builder of a computational design model.

George Stiny and after him Mine Özkar recognizes computing as "a way of thought, reasoning, beyond its mechanical usability in technological tools"[21]. In this section, the intention is not to discuss the origins of computer programming, nor to provide a history of digital computers but rather to discuss programming in computation and the translation of design reasoning into computational components.

An approach needs to be developed to address the problem of incompatibility of unchangeable entities of computation with architectural design. In building computational design models, the aim may not be easier automated generation of architectural design solutions. The fallacy of automated generation may be seeing the design solution as mass customized. However, it should be the sequences of relations that are to be mass customized rather than the design solution itself.[22]

[21] Mine Özkar. "Learning Computing by Design; Learning Design by Computing", *Proceedings Designtrain Congress Trailer I,* 2007, p.103

[22] If a model aims at the mass customization of the design solution itself, then the designer becomes a mere player. The model imposes the way of defining relations between the parameters and relating the constraints with each other – which is different for every designer.

2.3 Why Shall We Design with Changeable Entities?

In "Classical and non-classical computation", Terry Knight and George Stiny classify computational means according to the aspects of representation and process. Accordingly, classical representation is by verbal and numerical means where non-classical representation is by visual means for computation. Although the split in the representation aspect is more profound among classical and non-classical computation,[23] this might not be the case when the subject matter is the use of the computational components for the translation of the reasoning in architectural design. Moreover, the split might not be on these terms but on the differences regarding the role of the designers.

If unchangeable computational components are used for the building of a design model, then the designers become mere end users that are only allowed to externalize the design reasoning within the confines of the package in hand – into the model imposed on them, having already been built. Rather then providing a model to the designers at the outset, they should be given the chance to build their own model. If not, they will only fill in the values for already defined relations between parameters and constraints. This results in an incompatibility, considering the nature of programming in architecture because as the model is already built, the designer does not have free choice in the necessary actions for the smaller tasks in a complex architectural design problem.

The incompatibility can be explained as follows: Designer action has a crucial role in developing the model, which is of help to the designer in generating architectural design solutions. It is an unavoidable fact that in

[23] Terry Knight, George Stiny. "Classical and non-classical computation", *Architectural Research Quarterly,* 5(4), 2002, p.355

architectural design, constraints and objectives are seldom stated at the outset, and the problem space and the solution space co-evolve. If the designers do not have a hand in the modification of the relations between the design constraints, then the model imposes particular design solutions upon the designer, who is left with unchangeable components. Architectural programming does not involve unchangeable components or primitives to design with, but rather a description of the designer actions that are needed to relate the constraints with each other. In such an approach to building a model, the parameters and the constraints are not mass customized.

There is a potential danger in programmatic thinking related to CAAD and software development.[24] As raised by Robert Woodbury and Andrew Burrow, programming involves "representing and computing directly over concrete entities,"[25] and this may be the case if the nature of programming is not efficiently considered regarding the nature of programming in architectural design, as explained in 2.1.

The unchangeable components may not necessarily be to hand in an architectural design problem; and if they are used, designers may become removed from the act of building the model. If they are no longer able to affect the relations between the design constraints, they become mere players of a model that mass customizes the solutions that it generates. If this model is put into computer and implemented as an application or some software, it may impose upon designers the way to design. In such cases, the model is built without the designers' involvements; and as a

[24] This danger belongs mostly to the group of deterministic CAAD tools that are explained in Chapter 4.

[25] Robert Woodbury, Andrew Burrow. "Whither design space?", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 20(2), 2006, p.65

consequence the design solution already exists in the model, mass customized. In architectural design, rather than solutions, it should be the abstract, modifiable and reusable sequences of relations – such as mathematical, geometrical relations, etc. – that are encoded in the design space.

CHAPTER 3

TRANSLATION OF DESIGN REASONING INTO RELATIONS BETWEEN THE DESIGN CONSTRAINTS: MODELING THE ARCHITECTURAL DESIGN SPACE

This thesis adopts an approach favoring the mass customization of sequences of relations in the building of computational design models. In following this approach, the role of the model is to support the designer in generating architectural design solutions, and therefore, is mostly relevant in the design space of the architectural design problem – in allowing the designer to explore a design space. Depending upon the research conducted, this can be made possible if the model externalizes efficiently the design reasoning of the particular architectural design problem defined by the designer.

Designers need to adopt the model that best serves the needs of the architectural design problem in terms of intuitiveness and synthetic character. Every computational design model contains different components; such as relations between design constraints (in parametric and constraint-based design models), and these components should enable efficiency in the model in terms of intuitiveness and synthetic character. This research will briefly look into different available computational components; but first it is necessary to explain the necessity and means of externalizing design reasoning, which is a common act of architectural design.

3.1 The Need to Externalize Design Reasoning in the Design Space

The reason for the quest into the design space within this research is to be explained as a priority, as it is linked with the role of the design model.

This role brings with it a necessity to include the essential properties of the model that match the properties of the design space to support the designer. Robert Woodbury and Andrew Burrow also comply on this in their approach, which is nourished by the state of mind, “the design space itself is where the largest gains are to be made”.[26]

Accordingly, in this chapter a research into design space is made in an attempt to analyze it using computational thinking, and the studies of Robert Woodbury and Andrew Burrow are referenced at length. They describe their studies on design space exploration as such:

> “*Its [design space exploration] three main threads are accounts of designer action, development of strategies for amplification of designer action in exploration, and discovery of computational structures to support exploration.*”[27]

The aim of this research in referring to studies in the exploration of design space lies at a point between the second and third threads defined above.

This section of the chapter aims to lead the discussion to the significance and the need to externalize design reasoning in architectural design space. In doing so, the way a designer’s reasoning is externalized in architectural design, and the way it is externalized within the limits of the components of the computational design model, can be compared to a significant extent.

For the purpose of this thesis it is assumed that the design problem has to be addressed through the relations defined for the parameters and the

[26] Robert Woodbury, Andrew Burrow. “Whither design space?”*, Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 20(2), 2006*,* p.64

[27] *ibid.,* p.63

relations between constraints,[28] and these play a crucial role in modeling the problem space and the solution space. This crucial role exists if the computational support is used in exploring the design space, however in intuitive thinking, the imported data to the design space is modeled in such a way that it obeys no rules, and can be represented in any way, as described by Gabriela Goldschmidt,[29] who goes on the claim that representations are mostly designer dependent.[30] However, in cases where the computational support of a model is not the subject matter, the complexity of the architectural design problems may not be addressed in a single attempt. The model is of great help to the designer in addressing this complexity, and there are two important aspects of the design space in which the model should play a key role – the problem space and the design space – that should be explained first.

Goldschmidt claims that representation is the translation of one problem state into another, the "initial" to "goal state," passing through intermediate states,[31] through "operators," and there exist different types of models to achieve this. According to Goldschmidt, states in the problem space are figural representations in which the operators are the conceptual representations.[32] In a design problem, "[t]he initial state is usually vague, and the goal state is either unknown or ambiguous,"[33] as she describes,

[28] Here it should be reminded that, "There is a clear correspondence between the kind of constraints used by the designer and the types of design representations used – [in intuitive thinking these may be] words, numbers, flow diagrams, plans, sections, perspectives." Charles, M. Eastman. "On the analysis of intuitive design processes", *Emerging Methods in Environmental Design and Planning*, ed. G.T.Moore, Cambridge, MA: The MIT Press, 1970, p.21

[29] Gabriela Goldschmidt. "Capturing indeterminism: representation in the design problem space", *Design Studies*, 18(4), p.442

[30] *ibid.*, p.445

[31] *ibid.*, p.441

[32] *ibid.*, p.446

[33] *ibid.*, p.441

and therefore a full and a sufficient external representation[34] can be seen as the "goal state" of the problem space.[35]

Axel Kilian's study covers the issue of exploration of the design space through a modeling of the constraints, which he explains by following a couple of cases in his paper, "Design Innovation through Constraint Modeling". He claims that constraints in a design space exist in many forms, and that there is no master model that could incorporate them all.[36] He classifies the types of constraints into four groups: geometric, topologic, functional and quantitative,[37] which he identifies based on a review of the literature and on the case studies introduced in his paper. Kilian calls these constraints "design drivers" for the exploration of the solution space.[38] The constraints found in different scale groups in the graduation project are also referred to as design drivers. Within the developed design model (Chapter 5), they drive the design by helping the designer to generate solutions for one small design task – such as the design of the pipe work system of hydroponics for the cultivation of plants in the units – of the complex architectural design problem.

Discussing the constraints, as design drivers in an architectural design problem would be beneficial as they play a crucial role in describing the architectural design problem by translating designers' reasoning into a

---

[34] "Representations in the problem space can be internal or external. Under current theories, internal representations are the essence of cognition and imagery is the locus of much of our internal representation activity. External representations can be visual or verbal (by 'verbal' we mean expressed in words, in oral or written form). […] External visual representations are expressed mostly in drawings (but there are other modes, e.g. three dimensional models)." Gabriela Goldschmidt. "Capturing indeterminism: representation in the design problem space", *Design Studies*, 18(4), p.445

[35] *ibid.*, p.446

[36] Axel Kilian. "Design Innovation through Constraint Modeling", *Digital Design: The Quest for New Paradigms, 23rd eCAADe Conference Proceedings,* 2005, p.678

[37] *ibid.*, p.671

[38] *ibid.*

format that is compatible for processing in the model.

In "A Perspective on Computer Aided Design after Four Decades," Earl Mark, Mark Gross and Gabriela Goldschmidt define the word "constraint" as being anything the architect wants the building to satisfy, although they claim that it suggests something restrictive.[39] Moreover, in the studies of Mark Gross, the idea of designing has already been studied as an exploration of constraints. Accordingly, in the paper "Constraints: Knowledge Representation in Design," Mark Gross, Stephen Ervin, James Anderson and Aaron Fleisher see "designing as a process of expressing and exploring constraints and trying to achieve objectives".[40] Thus, the relations between the design constraints (which individual constraint is related to which other constraint) have primary significance in describing the design problem in the best way.

Mark Gross, Stephen Ervin, James Anderson and Aaron Fleisher claim that constraint models enable designers to describe and explore the relationships between design variables,[41] referred to as design parameters throughout this thesis. Accordingly, each variable stands for a number, while the network stands for a system of equations, both of which are formed by the designers' reasoning in which parameters and constraints may be written in equation format. It is useful, however, to identify different groups of constraints, as they may belong to different scales in the architectural design space.

Constraints were also grouped for the model developed in the graduation project of the author. For instance, some groups of constraints belong to

[39] Earl Mark, Mark Gross, Gabriela Goldschmidt. "A Perspective on Computer Aided Design After Four Decades", *eCAADe 26: architecture 'in computro',* ed. M. Muylle, 2008, p.171

[40] Mark Gross, Stephen Ervin, James Anderson Aaron Fleisher. "Constraints: Knowledge representation in design", *Design Studies,* 9(3), 1988, p.134

[41] *ibid.*, p.137

the scale of 1/1000 (polder block scale), such as those exploring the "width of ditches". Some belong to 1/100 scale (project plot – building scale), such as those exploring the "number of units" or "m^2 of land occupied on agricultural land or pasture"; while others may belong to 1/10 scale (project plot – building component scale), such as those exploring "number of hours of sunlight" or (the subgroup) "growth pattern of the agricultural product" on a numerical basis (min/max distances between two seeds, tectonic properties or solid-void proportions, etc., Fig 3.1) or the "radius of the openings in the surface covering the units" or finally the "length of the members of the structure of the surface covering the units". To offer an example, the relation between different groups of constraints is ensured, as it is not only the "growth pattern of the agricultural product," but also the "m^2 of land occupied on agricultural land or pasture," etc. that drive the design of the surface covering the units.

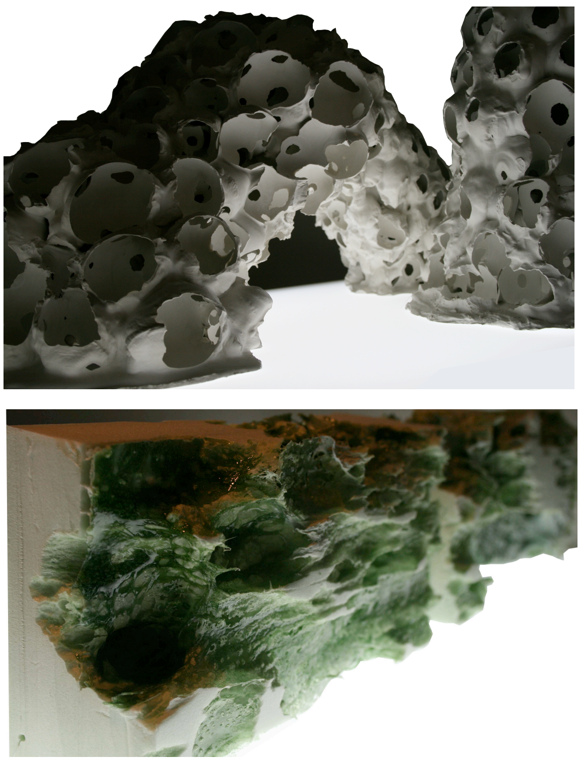


Fig 3.1. Two fabrications from the exhibition (FAB)BOTS in DHUB, curator: Marta Male-Alemany. Works by the master level students of "Machinic Control 1.0" design studio of AA and the "Digital Tectonics RS3" design studio of IAAC. These images can help to illustrate the tectonic properties of

the “growth pattern of the agricultural product”.

Mark Gross, Stephen Ervin, James Anderson and Aaron Fleisher describe the grouping issue as follows:

> *“ […] variables [parameters] and relationships that describe the properties and behavior of a single component (a beam, a room, etc.) may be abstracted as a single entity. Likewise, many local relationships may be abstracted as a single larger one”.*[42]

Grouping is given priority significance in this research as this mechanism allows designers to build a computational design model with a comprehensive design space as well. A model developed in such a way is able to respond to the whole architectural design problem, which does not permit linear sequence. “Architectural design cycles, refines, branches and backtracks”.[43]

In the exploration of a model’s design space,[44] as Kilian refers to it in the cases introduced in his study, parameters and constraints, and the relations between them, can be identified. Such identifications are crucial, as they may be used later to explain the process of building a computational design model. In this regard, they also play a crucial role in following the explanation of the model developed for the graduation project of the author. Within the scope of this thesis, design reasoning is to be translated into the relations defined for to the design parameters and the relations between design constraints. This is necessary when developing a parametric and constraint-based design model, as was the case in the

[42] Mark Gross, Stephen Ervin, James Anderson, Aaron Fleisher. “Constraints: Knowledge representation in design”, *Design Studies,* 9(3), 1988, p.137

[43] *ibid.*, p.134

[44] Axel Kilian. “Design Innovation through Constraint Modeling”, *Digital Design: The Quest for New Paradigms, 23rd eCAADe Conference Proceedings,* 2005, p.671

graduation project.

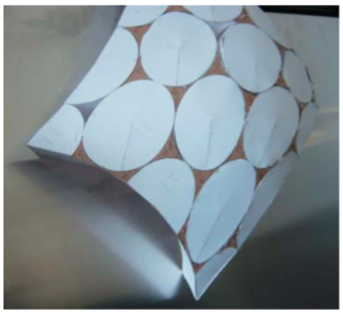


Fig 3.2. The example from Kilian's study: Cone based approximation of a double curved surface using only developable surfaces. (Taken from: "Design Innovation through Constraint Modeling", Digital Design: *The Quest for New Paradigms, 23rd eCAADe Conference Proceedings*, 2005, p.675, by Axel Kilian)

In exemplar cases from Kilian's own experience, exploring the design space is explained through a modeling of the constraints. In these cases, he explains the way relations are defined for parameters and constraints, and consequently the way problems are parameterized. In one example, featuring a cone-based translation of a free form surface into developable cone-based parts, a new approach to low cost fabrication of free from surfaces is offered.[45] This example firstly puts forward how it is possible to identify parameters, constraints and the relations among them; and secondly, it suggests possible ways of relating constraints belonging to different groups. In this example (Fig 3.2), the "degree of curvature" is one parameter that is explored through a constraint in which "curvature" is the parameter, and "degree" is the relation defined for it. To identify further, "circles" are the other set of parameters; while "spacing" and "positioning" are the relations defined for them. The designer defines these to describe the design problem in the best possible way.

[45] Axel Kilian. "Design Innovation through Constraint Modeling", *Digital Design: The Quest for New Paradigms, 23rd eCAADe Conference Proceedings,* 2005, p.675

In the same example of Killian, the act of relating various constraints with each other involves some custom design reasoning, although the act itself may be a recursive and abstract function of architectural design problems and may be one that is taken from a geometrical theorem, such as circle packing, and expressed as an algorithm. When the reasoning is translated into a compatible format using this function (in this case, circle packing), it is efficiently externalized. Moreover, as the function is mass customized in an abstract, modifiable and reusable manner, it is open to general usage, meaning to various scales, in addition to this specific design task of Killian's.

When one constraint controls the other, it is simply an externalizing of the designer's reasoning, because the reasoning is translated into the relations between design constraints. The example in previous two paragraphs, circle packing, when expressed as an algorithm, exemplifies one of the designer actions that translate the reasoning. It simply relates the "curvature of the circle" to the "spacing" and "positioning" of the "circles".

For this study, the translation of design reasoning into computational components necessitates also its quantification. To clarify, when some values are input into the functions according to some specific design reasoning they become the design rules, but these design rules are not mass customized, being rather outcomes of a customization process. That said; this is an issue that belongs in a different discussion.

Within the graduation project, the translation of reasoning into the relations of design parameters is seen as the goal of the problem space. Yet, as every designer may have a different design reasoning for the same architectural design problem, designers define "x" kind of relations for "x" kind of parameters in the problem space, for “x” kind of design problems, and this is designer dependant. Likewise, the translation of reasoning into the relations between design constraints in the generation of design solutions is seen as the goal of the solution space. For instance, the modeling of structural constraints in relation to agricultural constraints (in the graduation project) depends on the author’s choice for modeling the design space in the best way to address the design problem. Therefore, designers relate constraint “x” with constraint “y” in the solution space to generate “x” kind of design solutions, and this also depends on the choice of the designers.

By these processes mentioned in the previous paragraph, designers add their way of designing to the realm of architectural design. As Sait Ali Köknar and Arzu Erdem discuss in “Can Creativity be Institutionalized?”, almost every significant architectural design project introduces a new architectural design tool.[46] Köknar and Erdem also speculate that a general positive response to an architectural design project is highly related to this.[47] For this thesis it is related to a new addition to the realm that designers define “x” kind of relations for “x” kind of parameters in the problem space, for “x” kind of design problems, and designer relates constraint “x” with constraint “y” in the solution space to generate “x” kind of design solutions, as described previously.

3.2 Externalizing Design Reasoning in Architectural Design

In 3.1, the aim was to discuss the importance of externalizing designers’ reasoning in the design space; while in this section, the aim is to discuss

[46] Sait Ali Köknar, Arzu Erdem. “Can creativity be institutionalized?”, *A-Z ITU Journal of the Faculty of Architecture*, 4(2), 2007*,* p.32

[47] *ibid.*

how the role of the model may be closer to that of the designer.

According to conventional aspects of architectural design, the designer sketches the architectural design reasoning to integrate it into the design space. Referencing to the protocol analysis study of Charles Eastman, which broaches the task of improving a bathroom layout, Ellen Yi-Luen Do and Mark Gross state that designers use words and drawings to represent the design objects, design operations and "control mechanisms" of the design problem.[48] The external representation of a designer's reasoning by intuitive means is related to this point; and this mechanism of externalization already exists for architectural design.

As stated by Do and Gross in "Thinking with diagrams in architectural design," all external design representations in architecture are in the form of drawings – except the physical models.[49] This is a significant argument that follows the same path of "classical" representation for architectural design first discussed by Knight and Stiny.[50] Further discussions on the representation of design reasoning look into the role of the designer. Efficiency in terms of intuitiveness or synthetic character is discussed in relation to the degree to which a designer is able to operate on the translation of design reasoning into the format of the computational components of the model.

First, the visual and numerical representation of design reasoning is discussed, offering a brief introduction to the scope of externalization in architectural design problems. This is followed by an investigation of the

[48] Ellen Yi-Luen, Do, Mark Gross. "Thinking with diagrams in architectural design", *Artificial Intelligence Review 15*, 2001, p.140

[49] *ibid.,* p.135

[50] Terry Knight, George Stiny. "Classical and non-classical computation", *Architectural Research Quarterly,* 5(4), 2002, pp.355-372

act of adopting a computational design model, which is interpreted as a method for externalizing design reasoning. The intention is to show that these models have different efficiencies in terms of intuitiveness. They can be composed of different components with different levels of flexibility in allowing the designer to operating on the translation of design reasoning. Consequently, they offer different levels of efficiencies in enabling the designers to find themselves in the subject solution space.

3.2.1 Visual Representation

When discussing visual representation of design reasoning, there are a number of different references to intuitive diagrams and sketches that may be stated, but Christopher Alexander's explanation is particularly noteworthy. The attempts of designers to externalize in architectural design is mentioned in his book "Notes on the Synthesis of Form," in which he describes the activity as a shuffling back and forth in "an abstract pattern of physical relationships, which resolves a small system of interacting and conflicting forces".[51] In his opinion, this is the way of "creating designs which are whole by fusing these relationships,"[52] and the independence of visual representation is related to the designer's role.

Alexander claims that the design process has a two-fold character: the analytical phase – requirements and the program as the tree sets of these requirements; [53] and the synthetic phase – diagrams and the realization of the problem as the tree of

[51] Christopher Alexander. *Notes on the Synthesis of Form*, Cambridge, MA: Harvard University Press, 1964.

[52] *ibid.*

[53] *ibid.*, p.84

diagrams.[54] Following this approach, one can conceive the challenge of programmatic thinking, which exists for an intuitive and a more synthetic process, such as architectural design. However, this has been the subject matter of another discussion on programmatic thinking, as seen in Chapter 2 where it is mentioned that this challenge may become a problem that depends on the effectiveness and usage of the definition for programming in architecture. In short, there exist some possibilities to externalize a designers' reasoning, yet these are dealt with mostly in the analytical phase.

In Alexander's approach, there is clear explanation of the shuffling back and forth between two phases, and he puts the design solution again in the designer's mind and claims that the designer never fully understands the context – at least not as a single pattern. At this point it can be claimed that the model, which helps the designer to explore the design space, is significant in that it allows the grasping of the design space as a whole. However, to do this, design reasoning has to be put on paper; it needs to be externalized so that it is represented and processed in the model, and thus needs to be translated into the computational components of the model.

Moreover, the model has to have a basis for externalizing in a comprehensive manner, considering the diversity of design reasoning in architectural design problems.

Here, would be of benefit to raise a further question, although a

[54] Christopher Alexander. *Notes on the Synthesis of Form*, Cambridge, MA: Harvard University Press, 1964, p.84

one-word answer is not the intention: "What kind of representational means does architectural design necessities?" Such a research question is nourished by the approach of Stiny, which leads this study to an exploration of the design space in terms of a visual computation that covers all the visual representation of architectural deign rules, symbols, etc. that may be representing designer actions. Yet, this is a question that should be answered in a subsequent research, as it is not the intention of this study to answer this question with a sharp classification, being that this would undermine the complexity of architectural design.

3.2.2 Numerical Representation

Within the incoherent character of visual representations, of diagrams or sketches for instance – in other words, within their independence and specificity – a basis for interaction is crucial when considering the architectural design problems. Christopher Alexander describes this as being either for a conflict or concurrence of design constraints.[55] This need leads to a unity in a physical consideration of quantifying of the design reasoning, and is for the sake of processing in the design model.

A coherent language can contribute greatly to the understanding of the design space fully, leading to a well definition of the design problem. This can remove much of the need for the elimination of design reasoning from the design space of the model, with numerical representation being an efficient way considering this issue.

[55] Christopher Alexander. *Notes on the Synthesis of Form*, Cambridge, MA: Harvard University Press, 1964.

In the search for a unity when translating a diversity of design reasoning in the model, numerical representations may be seen as an intermediary step.[56] As explained by Ellen Yi-Luen Do and Mark Gross, in architectural design problems, “not only physical elements, but also forces and flows (e.g., forces of sun and wind and flows of people and materials)”[57] also exist. Almost all of these are computable, either geometrically or arithmetically,[58] and must all be translated in the model.

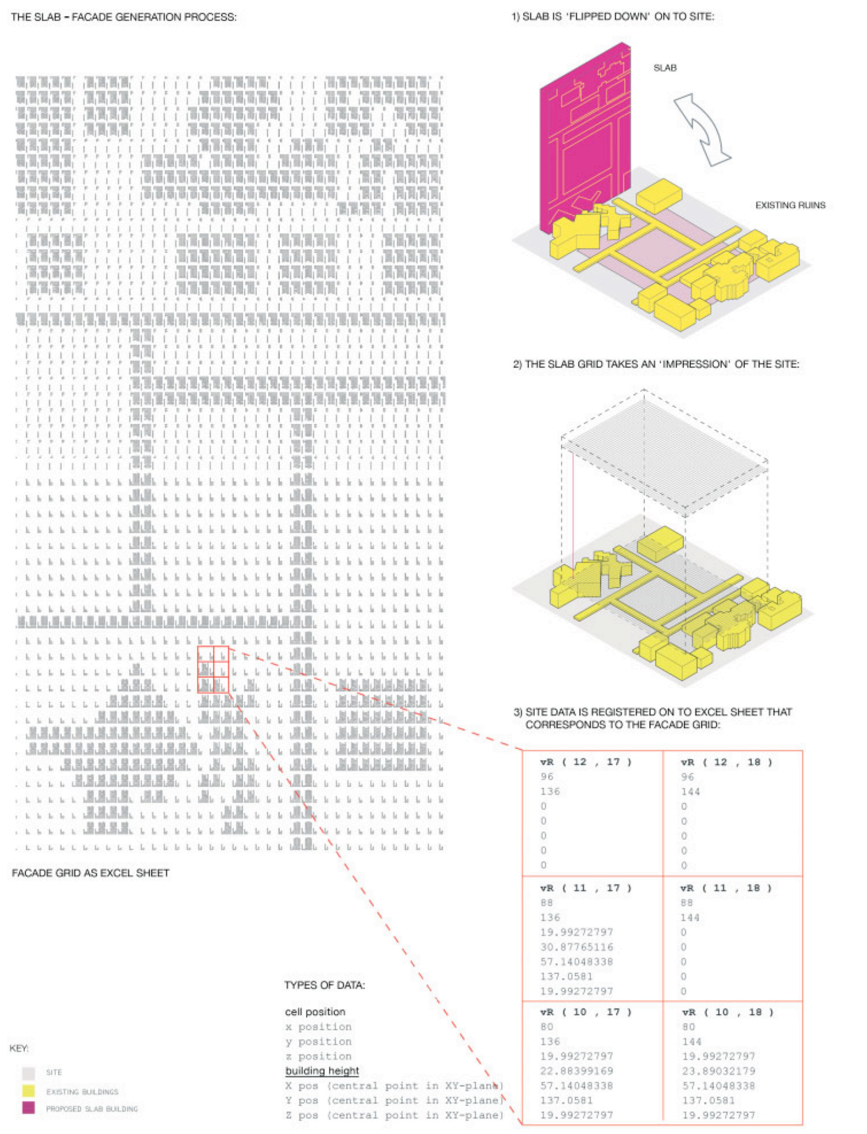
Alberti’s delineation of the city of Rome *“Descripto urbis Romae”* can be held up as pioneering case in numerical representation. His aim was to simply translate the urban environment of Rome into numerical values by measuring the physical distances to major architectural features of Rome from a specific point (also in the z-direction for their height) to provide visual information about the city. This generated maps of Rome containing selected architectural features using a numerical template.

Further discussion addresses the computational components that form the basis of the model.

[56] Here, representing different types architectural design parameters in the design space of the same model can be exemplified: “A simpler variation is parametric design in which some quantities (dimensions, positions, numbers of elements [parameters]) are expressed as a mathematical function of others. For example, the number of windows depends on the length of the wall; the longer the wall, the more windows. Parametric design sets up these dependent relationships, and when the designer changes key driving variables, dependent ones follow. It’s like design by constraints, except that the dependencies only go one way.” Earl Mark, Mark Gross, Gabriela Goldschmidt. “A Perspective on Computer Aided Design After Four Decades”, *eCAADe 26: architecture ‘in computro’,* ed. M. Muylle, 2008, pp.171-172.

[57] Ellen Yi-Luen, Do, Mark Gross. “Thinking with diagrams in architectural design”, *Artificial Intelligence Review 15*, 2001, p.135

[58] *ibid.*, p.144


THE SLAB – FACADE GENERATION PROCESS:
1) SLAB IS 'FLIPPED DOWN' ON TO SITE:
SLAB
EXISTING RUINS
2) THE SLAB GRID TAKES AN 'IMPRESSION' OF THE SITE:
3) SITE DATA IS REGISTERED ON TO EXCEL SHEET THAT CORRESPONDS TO THE FACADE GRID:
FACADE GRID AS EXCEL SHEET
TYPES OF DATA:
cell position
x position
y position
z position
building height
X pos (central point in XY-plane)
Y pos (central point in XY-plane)
Z pos (central point in XY-plane)
KEY:
SITE
EXISTING BUILDINGS
PROPOSED SLAB BUILDING

vR (12 , 17)	vR (12 , 18)
96	96
136	144
0	0
0	0
0	0
0	0
0	0
vR (11 , 17)	vR (11 , 18)
88	88
136	144
19.99272797	0
30.87765116	0
57.14048338	0
137.0581	0
19.99272797	0
vR (10 , 17)	vR (10 , 18)
80	80
136	144
19.99272797	19.99272797
22.88399169	23.89032179
57.14048338	57.14048338
137.0581	137.0581
19.99272797	19.99272797

OMA Research and Innovation Parametrics Cell, Facade Study for the Slab, 2008
This parametric facade strategy experimented with the soft data of sociocultural material through scripts that articulated the building's internal public zones and registered the site's external urban pattern while incorporating technical functional, structural and material parameters. The series of diagrams shows how the context was translated into data through a process analogous to that of creating a physical impression: 1) the facade grid was conceptually 'flipped down' on to the existing ruins of a site; 2) an impression of each building's location and height was measured; and 3) the data was registered into the corresponding cell of an Excel sheet. A special script was created that keyed this input data to a degree and type of mutation in the facade grid.

Fig 3.3. Numerical representation (taken from "The Future Information Modeling and the End of Theory: Less is Limited, More is Different", by Cynthia Ottchen, *AD issue on: "Closing the Gap"* (Guest Ed.) Richard Garber. 79(2), Wiley & Sons, March-April 2009, pp.22-27

3.2.3 Translating Design Reasoning into the Relations between Design Constraints for Developing a Computational Design Model

As mentioned previously, in conventional aspects, designers use their reasoning intuitively to explore the design space, although they do not grasp it fully in a single pattern. To develop a model that helps a designer to explore and grasp the design space as a whole, the ways of representing reasoning explicitly through the relations between design constraints, etc., are discussed here.

The ways of externalizing design reasoning embody the strategies of problem solving, as Özer Çiftçioğlu explains in “Conceptual Design Enhancement by Intelligent Technologies”. He states that such actions go beyond the limits of knowledge representation, which has a high level of uncertainty, and launch the development of strategies (i.e. rule-based systems). This may be seen as being highly analytical, as much is lost in terms of intuitiveness, and thus may be seen as a part of the analytical processes. This has been mentioned previously, giving reference to the studies of Christopher Alexander.

In CAAD theory, knowledge representation is classified as soft computing, which is an attempt to represent the human brain, as Özer Çiftçioğlu explains. As a consequence, soft computing may be seen as being closer to externalization in an intuitive manner, with one drawback being an inability to go beyond mere simulation.[59]

[59] Özer Çiftçioğlu. “Conceptual Design Enhancement by Intelligent Technologies”, *The Architecture Annual 2001-2002: Delft University of Technology*, ed. Arie Graafland, pp. 85-89.

In "A Biased History of CAAD: The Bibliographic Version," Alexander Koutamanis explains that it was in the 1990s that the earliest instances of computerization were put into practice, being the transfer of manual processes to the computer.[60] The manual processes that he mentions may be explained as the way a designer defines the design problem. Before transferring such processes into the computer, they need to be translated into the computational components of the model; after which they may be put into the computer as an application – perhaps as a "toy" application,[61] as Scott Chase refers to it. The "methodological knowledge developed in CAAD"[62] – through which design reasoning can be externalized – provides the computational components of the models that are needed in such translations.

Moreover, when considering that every architectural design project can be seen as a computational design model, the significance of the computational components necessary to translate the design reasoning should be explained.

There are significant differences between the actions of modeling and drawing a plan, and not just in the tools used – such as pens, compasses, rulers, cutters and papers, as Köknar refers to them.[63] Within the scope of this thesis there is deemed to be an

[60] Alexander Koutamanis. "A Biased History of CAAD: the bibliographic version", in, J.P Duarte, G Ducla-Soares & A.Z Sampaio (Eds.), *eCAADe 23: Digital Design: the Quest for New Paradigms,* p.634

[61] Scott Chase defines "toy" applications as such: "Many generative design tools developed to date are restricted to a limited number of design issues, or to a portion of the design. In many cases, these can be considered 'toy' applications. Some were developed as teaching aids. Others have served to demonstrate proof of concept, with the potential of being a more powerful design tool left for the future." "Generative design tools for novice designers: Issues for selection", *Automation in Construction*, vol.6, issue 4, December 2005, p.689

[62] Alexander Koutamanis. *op.cit.,* p.634

[63] Sait Ali Köknar. "Architectural Design Tools: Toward a non-linear design process", *Proceedings Designtrain Congress Trailer I,* 2007, p.178

intermediary facet in the development of a model that involves the translation of design reasoning into the computational components of the model. It is these relations that compose the plan, rather than mere lines, circles, etc. drawn with pens, compasses, rulers, cutters and papers.

The crucial aspect of such a translation lies in the ability afforded to the designer in terms of modifications. It is proposed in this thesis that the designer uses mass customized sequences of parametric and constraint-based relations in the model as recursive functions of architectural design problems. These mass customized relations are called designer actions, and are ready to be used in various contexts, and in various architectural design problems of different scales, to relate the constraints with each other.

Mass customized sequences of relations are not unfamiliar in design, being that designers are already used to working with thinking tools,[64] as stated by Köknar and Erdem. Designers use existing packages of knowledge as essential thinking tools in architectural design.[65] These thinking tools may be interpreted as design models within the scope of this thesis. With the adoption of the components of such models, design reasoning can be externalized. Designers may draw upon existing models created

[64] Sait Ali Köknar, Arzu Erdem. "Can creativity be institutionalized?", *A-Z ITU Journal of the Faculty of Architecture*, 4(2), 2007, p.28

[65] *ibid.* If designers can only describe their design problem through existing packages of knowledge, the mass customization and its outcome is no longer something unfamiliar to the new user of this mass customized relation. Others or the designers themselves can use it in other contexts for building other models. After all, as Woodbury and Burrow mention, "designers use production-like encodings of extant and newly learned actions". They also claim that designers use such encodings in communication with colleagues. Robert Woodbury, Andrew Burrow. "Whither design space?", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 20(2), 2006, p.68

in projects of different practices that use computational support, etc., and may follow the sequences of relations that they introduce (through their intermediary facet) by translating the reasoning of their own architectural design problem into the components of these existing models.[66] Existing models can be referenced when translating the design reasoning into the mass-customized sequences of relations that these models propose/program.

To be to the point, it is proposed in this thesis; designers translate their reasoning into existing components,[67] via an intermediary facet in modeling the plan. Throughout the thesis, "smaller carriers of actions"[68] are referred to as designer actions that encode recursive functions of architectural design problems. In the chapter describing the author's graduation project, they are referred to as Grasshopper definitions of recursive functions, or sometimes as mere toggles when the function is implemented as an algorithm in these toggles using programming languages (such as C# or VB).

Earl Mark, Mark Gross and Gabriela Goldschmidt, in "A Perspective on Computer Aided Design after Four Decades," describe this intermediary facet as follows: "Parametric and constraint-based approaches [defining the design problem in

[66] Although a conventional aspect, the benefit of using case studies in architectural design studios is clear; and is assured with the author's own experience.

[67] Gross, Ervin, Anderson and Fleisher explain this issue as such: "We are building computer programs: languages and tools to support the design process [CAAD tools – 'toy applications']. Computers force us to be more articulate, and therefore we formulate our theories about designing as computer programs." "Constraints: Knowledge representation in design", *Design Studies,* 9(3), 1988, p.133

[68] Sait Ali Köknar. "Architectural Design Tools: Toward a non-linear design process", *Proceedings Designtrain Congress Trailer I,* 2007, pp.176-177.

terms of parametric and constraint-based relations] add a layer in which the designer expresses relationships and dependencies among quantities (parameters) in the design".[69]

The addition of such a facet, or layer, as Earl Mark, Mark Gross and Gabriela Goldschmidt call it, is a sensible approach, considering how it enables the designer externalize the design reasoning efficiently, and in consideration of what a good design representation is. According to Woodbury and Burrow, good design representations "must afford a disciplined notion of change; otherwise, it is not a design representation and certainly not a computable one".[70] Parametric and constraint-based relations, which are the components adopted in this study (for developing the subject model), afford changes if the proposed sequences of relations that are mass customized in an abstract, modifiable and reusable manner translate the design reasoning into them.

Even if the subject matter is whether or not the representation is computable, "good design representations support designers to move from idea to idea"[71] in the architectural design space – shifting between several different scales. Such representations are possible with the addition of the particular facet being considered.

This thesis proposes that the plan is modeled using parametric

[69] Earl Mark, Mark Gross, Gabriela Goldschmidt. "A Perspective on Computer Aided Design After Four Decades", *eCAADe 26: architecture 'in computro',* ed. M. Muylle, 2008, p.170

[70] Robert Woodbury, Andrew Burrow. "Whither design space?"*, Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 20(2), 2006, p.66

[71] *ibid.*

and constraint-based relations yet there exist some differences among some of the conventional aspects when drawing it. To clarify the differences,

1. In the plan as a computational design model, architectural design reasoning is expressed in terms of parametrical and constraint-based relations. These relations are mass customized sequences that can be modified when applied on a different context/scale.

2. The model responds to different scales in the architectural design problem – meaning shifting between scales is possible. Each change (in the value of a parameter) in a particular scale causes a change in the whole design space.

3. The model is composed of mass-customized sequences of relations that encode reusable, recursive functions of architectural design problems. These relate different groups of constraints with each other – rather than mere lines, circles, etc. – without any parametric attributes. As will be explained in depth in Chapter 5, design parameters that are explored through constraints; such as noise level and "distance between units" are related with each other using a mass-customized sequence of relations that encodes the function – expressed as an algorithm – on circle packing theorem. Other parameters in this sequence include "number of units," etc.

Sequences of relations may be nourished by mathematical or geometrical theorems in architectural design problems, and implemented as algorithms. What is encoded in them is a function that gives a unique output value for every new unique input value, and they may be used in design models if they fit the small task description of designers. The recursive and reusable functions that are encoded within them allow the designers to translate the design reasoning into parametric and constraint-based relations. This thesis aims to exemplify those most commonly used in architectural design problems as well, embodying features such as being fed similar design elements (and similar constraints to relate to) that have occurred in several different cases. This may allow them to be interpreted as design patterns for architectural design.

There are benefits in developing a model. The designer's mind is not always fully efficient in comprehending the design problem fully, as has been discussed earlier, and thus may not be able to understand the problem in a single pattern. This prohibits the designer from actively manipulating the whole problem. When the mind runs out of mental space, it often resorts to externalizing reasoning with sketches, diagrams, charts, etc as J.V. Nickerson, J.E. Corter, B. Tversky, D. Zahner and Y.J. Rho discuss.[72] For this reason, the model has to embody what is put out of designers' mind. It resembles an explicit representation of reasoning and aims at the translation of all the design reasoning that is to be processed in the architectural design problem.

[72] J.V. Nickerson, J.E. Corter, B. Tversky, D. Zahner, Y.J. Rho. "Diagrams as tools in the design of information systems", *Design Computing and Cognition'08: Proceedings of the Third International Conference on Design Computing and Cognition,* 2008, p.104

There are a number of critical issues that need to be taken into account when quantifying qualitative design reasoning. This has been covered previously in the thesis, citing Axel Kilian's explanation of how designers employ constraints when constructing the design space – designers construct the way that they design. Kilian draws reference from the studies of Burrow and Woodbury on design space, claiming that constraint modeling is a powerful way to drive solution space [73].

The modeling of constraints can take various forms, and Kilian's study can be referred to once again to provide a specific example. In his study, a different format (rather than a function) is used for the design task to "produce the basis of the circular cones and the circle center points form the tip".[74]. As Kilian explains:

> *"The interesting aspect is that the circular base is double curved following the curvature of the surface whereas the resulting cone surface is strictly developable [Fig 3.2]. The fabrication constraint is therefore embedded in the geometric property of the chosen primitive"* [75]

In this example, reasoning is externalized in the choice of the geometric primitive, and is translated into that choice for the specific design task regarding fabrication. The constraints are not related numerically, but the design reasoning is still translated: The relation of the constraints is ensured by the choice of the

[73] Axel Kilian. "Design Innovation through Constraint Modeling", *Digital Design: The Quest for New Paradigms, 23rd eCAADe Conference Proceedings,* 2005, p.671

[74] *ibid.*

[75] *ibid.*

material.

The act of relating various constraints with each other, which involves some specific design reasoning, can take various forms. It may be one recursive function – which may be expressed as an algorithm or as a formula – of a geometrical theorem, like circle packing; or it may involve the embedding of the fabrication constraint into the geometric property of the chosen primitive. The latter act can also be mass customized, just like the function, because it is the sequence of relations that is being mass customized, and relations can take various forms of actions.

It is important to retain the essential synthetic features of the model, no matter what kind of form the relating finds. If constraint models are analytical, a change in one parameter that applies to one constraint, for instance belonging to one specific scale, does not produce a result that the model continues to use as a driver in the exploration of the solution space,[76] and this would disable a shifting between different scales of the architectural design project. An example of a synthetic feature can be found in the author's graduation project. The resulting values taken from the add-on Geco, (Ecotect add-on for Grasshopper) which is used for the lighting calculations, serve their need in the rest of the design model, such as in determining the values for the "radius of openings in the surface covering the units".

The relations between design constraints are not assumed to be the only computational components that exist in CAAD theory. To

[76] Axel Kilian. "Design Innovation through Constraint Modeling", *Digital Design: The Quest for New Paradigms, 23rd eCAADe Conference Proceedings,* 2005, p.678

clarify the theories/approaches of CAAD that provide a variety of computational components, the classification of Earl Mark, Mark Gross, and Gabriela Goldschmidt can be used as a reference:

1. Parametric and constraint-based design
2. Shape grammars
3. Frame based design methods
4. Object oriented design
5. Generative systems
6. Top-down design
7. Knowledge based design systems
8. Design and cognition[77]

For instance, top-down design or shape grammars are considered as “themes and variations” for generative design, according to Scott Chase.[78] They both have similar roles, and are theories providing different computational components that the reasoning is translated into. Earl Mark, Mark Gross and Gabriela Goldschmidt refer to these as approaches “to design through the lens of different ways of constructing computer-aided design software”.[79]

Another classification is also possible here following the study of Knight and Stiny on classical and non-classical computational means. It is possible to say that genetic algorithms, as a soft computing method, is non-classical in terms of the process, but

[77] Earl Mark, Mark Gross, Gabriela Goldschmidt. “A Perspective on Computer Aided Design After Four Decades”, *eCAADe 26: architecture ‘in computro’,* ed. M. Muylle, 2008, pp.169-176.

[78] Scott Chase. “Generative design tools for novice designers: Issues for selection”, *Automation in Construction*, vol.6, issue 4, December 2005, pp.692-695.

[79] Earl Mark, Mark Gross, Gabriela Goldschmidt. *op.cit.,* p.171

classical in the representational means it uses. Their significance in a discussion related to the translation of design reasoning is as "a host of outside criteria, not given directly in the rules play a significant role in the outcomes"[80] according to Knight and Stiny.

As another method, shape grammars fall into the group in which representational means are non-classical, but the process is still classical. Their significance in the scope of this thesis is that shape grammars have neither a finite number of primitives, nor finitely defined rules. This feature makes shape grammars closer to being an effective computational design method when externalizing the designer's reasoning.

Moreover, in constraint-based automated layout programs, although the means of representation may be considered limited with respect to the ambiguity of design problems, they do respond to several different design tasks in complex architectural design problems.[81] However, among the different approaches to automated space layout, there exist various types of representations.

For instance, in a Quadratic Assignment Problem (QAP), architectural space (architectural design element) is translated into discrete objects (one-to-one assignment problem); while in a

[80] Terry Knight, George Stiny. "Classical and non-classical computation", *Architectural Research Quarterly,* 5(4), 2002, p.360

[81] This point can be clarified: "Space layout has been studied as a research problem since the 1960's. Site planning, arrangement of rooms in a building, and arrangement of furniture or equipment in a room have been solved by computer programs. In spite of being studied for a long time, the impact of this research on practice in terms of providing usable systems to professionals has been very limited". Can Baykan. "Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements" *E-Activities in Design and Design Education*. ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız, Paris: Europia Productions, 2003, p.137

stacking problem, space is represented as area (many-to-one assignment problem). What is selected as a specimen of constraint-based automated layout programs is the blocking or the floor plan layout problem (FloP), in which space is represented in terms of area and shape.[82]

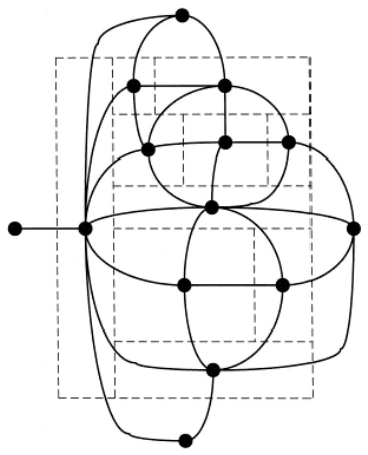


Fig 3.4. Planar adjacency graph (Taken from "Automated facilities layout: past, present and future," *Automation in Construction*, vol.9, 2000, p.199, by Robin S. Liggett)

As a difference, QAP uses a discrete grid-based system for the representation of space, with the main working principle being optimization (Fig 3.4).[83] On the other hand, as an architectural formulation, the floor plan generating program FloP works with finite, non-zero rectangles without holes, oriented parallel to the axes of an orthogonal coordinate system. As Can Baykan explains in "Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements," among

[82] Robin S. Liggett. "Automated facilities layout: past, present and future", *Automation in Construction*, vol.9, 2000, p.198

[83] This type of a representation is used according to the needs of the design problem defined by the designer. It involves adjacency relations between the parameters of the architectural design problem: "It is concerned primarily with generating a layout that meets adjacency requirements between activities. This approach requires the construction of a planar adjacency graph [Fig 3.4] where nodes represent activities to be located and edges (or links) represent a direct adjacency requirement." *ibid.,* p.199

these rectangles, the program uses unary constraints (dimensions, area and aspect ratio), binary constraints (rectangle relations, Fig 3.5; bounded difference, Fig 3.6; and orientation constraints) and global constraints to represent and maintain topological (adjacency, alignment, grouping), geometric (shape, dimension, distance) or functional relationships.[84]

The program may be seen as a model that is put into computer and implemented as some software. Using this program, the designer translates the design elements (such as the kitchen, the bedroom, or the sink, the bed, etc.) into the format that the program uses – being rectangles,[85] the size of which can be modified. The reasoning of the design problem is translated into the pre-defined relations of the constraints listed in the previous paragraph. Designers fill in the values for the program to find "satisficing" solutions (Fig 3.7), as Herbert Simon would name them. To exemplify, they list the relations that can exist between the design elements of the problem, choosing them from the 169 pre-defined relations of the rectangle algebra (Fig 3.5). These relations (binary constraints) are a part of the components that form the model, and design reasoning is translated into them.

[84] Can Baykan. "Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements" *E-Activities in Design and Design Education.* ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız, Paris: Europia Productions, 2003, pp.139-142.

[85] "The solutions are not given but are constructed by a program based on the representations that it uses. Therefore the set of significantly different solutions to a problem differ among programs." *ibid.,* p.145

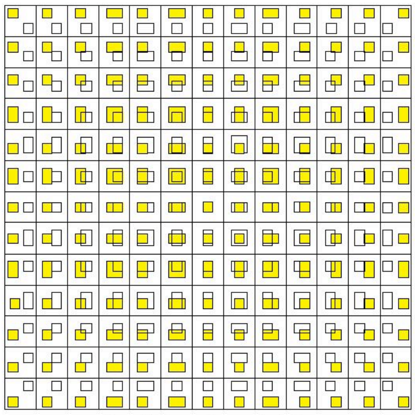


Fig 3.5. Relations of the Rectangle Algebra, RA (Taken from: “Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements,” by Can Baykan in *E-Activities in Design and Design Education*. ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız Paris: Europia Productions, 2003, p.140)

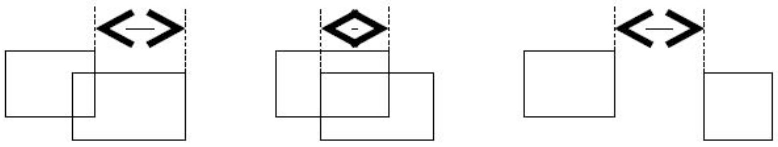


Fig 3.6. Types of distance between two rectangles (Taken from: “Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements,” by Can Baykan in *E-Activities in Design and Design Education*. ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız Paris: Europia Productions, 2003, p.141)

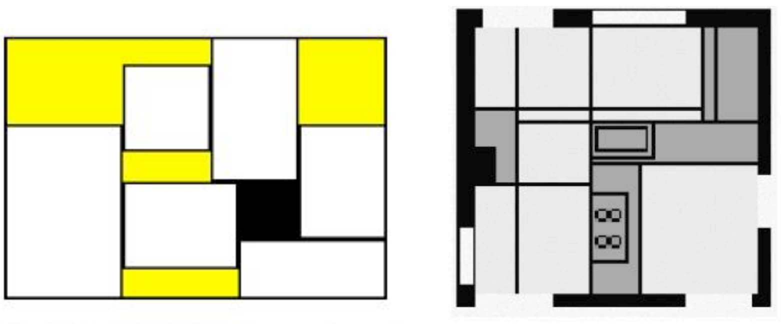


Fig 3.7. Non-trivial, trivial, internal and external holes, and a kitchen layout violating the access requirement (Taken from: “Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements,” by Can Baykan in *E-Activities in Design and Design Education*. ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız Paris: Europia Productions, 2003, p.142)

There is more to add to the usefulness and generality of the components that are used in the translation of design reasoning.

For instance, in an attempt to integrate shape grammars with FloP to develop a model for generating a sample Mamluk Madrasa design, it has been seen that shape grammars and knowledge-based systems greatly differ in terms of their computational components. If the shape rules for generating a Mamluk Madrasa[86] were to be implemented into FloP, some rigidity in formulating the design requirements in the problem space is experienced. However, this might be highly efficient in some small design tasks of architectural design problems.[87] It may appear to be more intuitive and synthetic to construct a design model using declarative means such as shape grammars, but automated systems propose a large amount of efficiency depending on the design task. This may result in an incompatibility that emerges when procedural means (i.e. the means that floor plan layout system uses) and declarative means (i.e. shape grammars) meet. This is a challenge when developing an architectural design model.

There have been recent studies concerning the translation of design reasoning that respond to the various needs of architectural design problems of different scales. These include the translation of the reasoning into various constraints defined for the small design tasks of structure, lighting, energy,

---

[86] Buthayna H Eilouti, Amer M. Al-Jokhadar. “A Generative System for Mamluk Madrasa Form-Making”, *Nexus Network Journal*, vol.9, 2007, pp. 7–29.

[87] “Formulating a layout problem as an optimization problem is not how designers intuitively define it. […] Layout problems are intuitively defined by designers as satisficing problems; automated systems should be formulated to work similarly.” Can Baykan. “Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements” *E-Activities in Design and Design Education*. ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız, Paris: Europia Productions, 2003, pp.137-138.

environment and building services. The modeling of these constraints enables a diverse range of perspectives to be quantified, as Ruwan Fernando, Robin Drogemuller, Flora Dilys Salim and Jane Burry discuss in “Patterns, Heuristics for Architectural Design Support: Making Use of Evolutionary Modeling in Design”.[88]

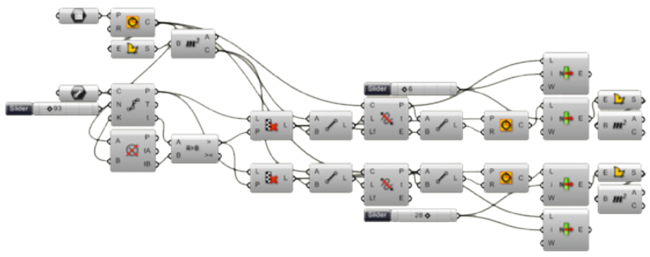


Fig 3.8. An exemplar image of a pattern definition from the design patterns study of Ramesh Krishnamurti and Tsung-Hsien Wang (http://www.andrew.cmu.edu/org/tsunghsw-design/) inspired by the design patterns study of Robert Woodbury (http://www.designpatterns.ca/)

The use of design patterns when translating a designer’s reasoning in a model contributes considerably to the quantifying of a diverse range of perspectives. As a key study, the design patterns of Ramesh Krishnamurti and Tsung-Hsien Wang (Fig 3.8) or Robert Woodbury can be referenced, among others. This part of the study is discussed at the end of Chapter 4. The need to explore the design space in terms of various advantages different computational methods bring is not avoided, yet not all of them are applied in this study.

[88] Ruwan Fernando, Robin Drogemuller, Flora Dilys Salim, Jane Burry. “Patterns, Heuristics for Architectural Design Support: Making use of evolutionary modeling in design”, *Proceedings of the 15th International Conference on Computer Aided Architectural Design Research in Asia,* (2010), pp.283-292.

CHAPTER 4

DIFFERENT TYPES OF COMPUTATIONAL DESIGN MODELS: A CLASSIFICATION FOR THE ROLE OF THE DESIGNER

Considering the issues discussed in Chapter 3, the cases detailed in this chapter have been chosen for further discussion on the externalization of design reasoning. The intention here is to look into the matter of efficient externalization in design generation and the role of the designer in using or developing computational design models. In short, this chapter will focus on both the method and the computational components.

About five years ago, Scott C. Chase stated in his paper, “Generative design tools for novice designers: Issues for selection,” that architects needed to use CAD tools effectively for design, but left programming to professional programmers. This might not be the case today, however the classification that he uses when discussing the issue offers insight into what is what in this field. For instance, one group of CAAD tools might be seen as the programs put together by professional programmers; however almost all developers of these “toy” applications (as Chase refers to them) are architects. Alternatively, another group may be classified as the “effective use of CAD tools for design”. Such classifications are one way of handling the situation; yet, it may be more effective to discuss the issue in regards to the role of the designer, which is related to the externalization of design reasoning.

When discussing the use of CAAD software to develop a computational design model (such as FloP, as discussed in Chapter 3), it is also necessary to mention the features that the particular software affords. When using a CAAD tool to generate design solutions the definer can define the design problem only within the limits of these tools. As such, dynamic features are no longer left to the intuitive choices of the designer, as the predefined relations of the design parameters control them, as E. Mark, M. Gross and G. Goldschmidt discuss. It is assumed that designers explore the design space of the model, the closer the designer is to being a tool builder, the more efficient the computational design model in terms of intuitiveness, and the more synthetic the model.

In this chapter, the means that the models use (Fig 4.1) for translating design reasoning, and further into the computational components of the method, are discussed in relation to each other. For such a discussion, it is considered necessary to follow a classification similar to the one mentioned at the beginning of this chapter.

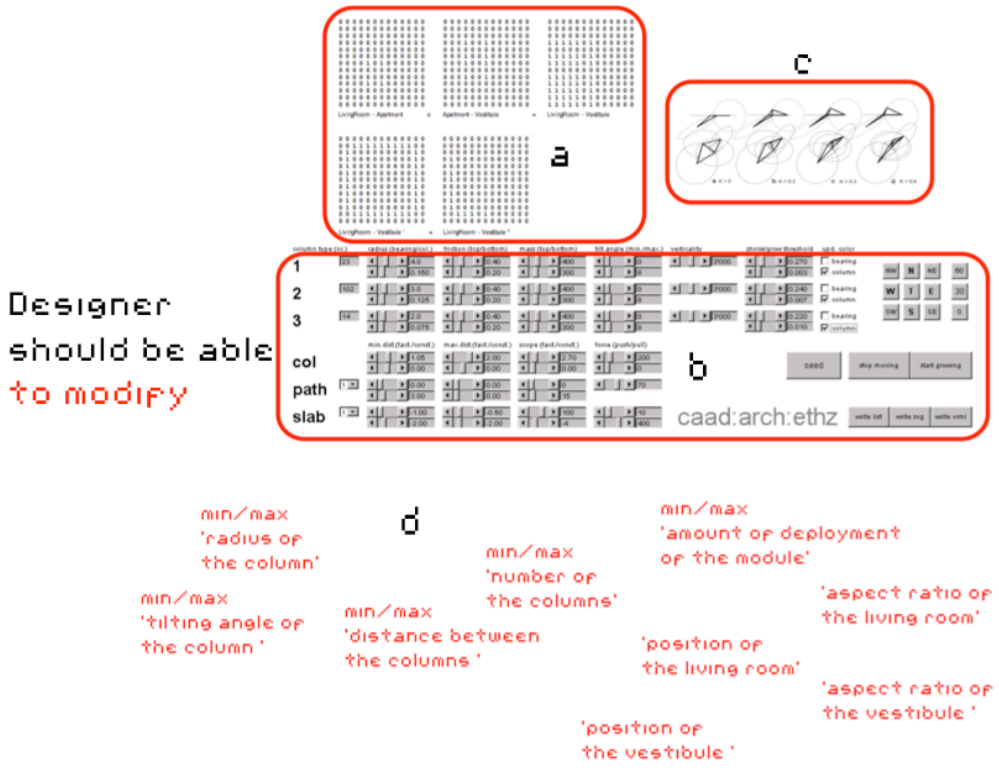


Fig 4.1. Different models that use different translations (for design reasoning) into the components that compose them. (a) The translation that FloP uses: This matrix has to be formed to define the rectangle relations between every two elements of the design problem until every element has a defined relation with the rest. (b) The translation that is used in Stadsbalkon: “designtoproduction developed a software based on artificial-life methods, which optimized the exact position, inclination and strength of some 150 irregularly placed columns according to functional and constructional requirements, and which could be utilized by the architects to generate alternative design solutions.” (Taken from: http://www.designtoproduction.ch/content/view/7/27/) (c) The translation that is used in the design research of adaptable structures: The modules that are interpreted as embodying the recursive functions used for the

parametric modeling of adaptable structures. (d) Various design parameters explored via constraints exemplified below. Design reasoning is partly translated into them in the particular models in consideration. The models that belong to the first group are bound to work with predefined parameters, constraints and their relations.

In the first group, the model is already built – modeling act is finished. Designers who generate design solutions with the model – which mostly is put into the computer as an application or some software – do not always have a role similar to the developers of it. They are unable to operate on the translation of design reasoning into the computational components that the model is composed of to develop a unique model in the particular CAAD tool. Some of the models in the group are composed of parametric and constraint-based relations, and some are not, which implies a change in the make up of the computational components. What is common is that the components are not always open to the operation of the designer. For instance, the relations between design parameters are already defined, and are not always open to the operation of the designer (Fig 4.2). Designers who generate design solutions by developing a model with the CAAD tool are only playing with it – they fill in the values via the interface, and thus have only a limited role in this deterministic system (Fig 4.3). It is not the designer who models, as the relations are already defined and the constraints are already related with each other (Fig 4.2 (d)). Fig 4.1 (d) exemplifies some pre-defined parameters explored via constraints, which designers are bound to translate their design reasoning into.

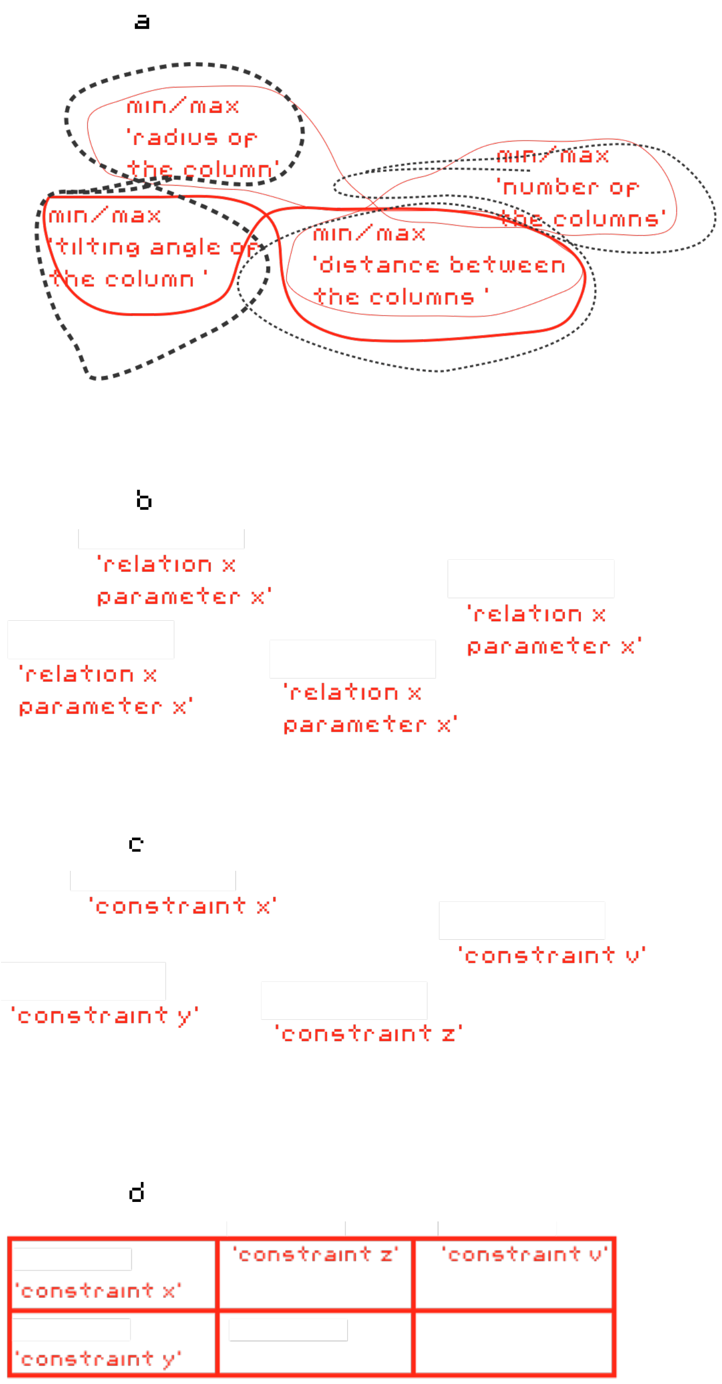
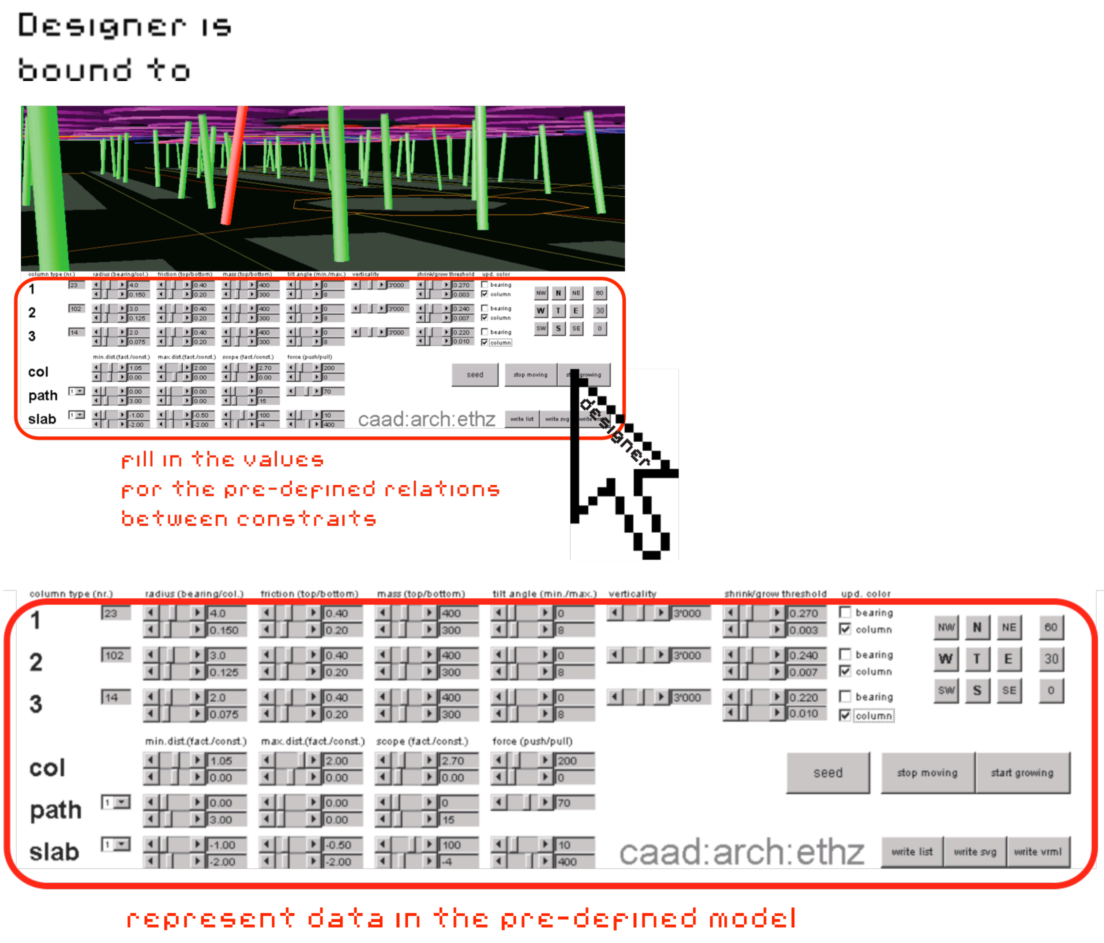


Fig 4.2. (a) Designers should be able to operate on the translation of design reasoning into the parametric and constraint based relations. They should be free to decide which constraint should relate to which other constraint. (b) For parametric and constraint based models, the reasoning is translated into the relations defined for design parameters (c) that are explored by design constraints. (d) If designers are not able to operate on the translation of design reasoning into the parametric and constraint based relations, the model might become a deterministic one.



column type (nr.)		radius (bearing/col.)	friction (top/bottom)	mass (top/bottom)	tilt angle (min./max.)	verticality	shrink/grow threshold	upd. color
1	23	4.0	0.40	400	0	3'000	0.270	bearing
		0.150	0.20	300	8		0.003	✓ column
2	102	3.0	0.40	400	0	3'000	0.240	bearing
		0.125	0.20	300	8		0.007	✓ column
3	14	2.0	0.40	400	0	3'000	0.220	bearing
		0.075	0.20	300	8		0.010	✓ column

		min.dist.(fact./const.)	max.dist.(fact./const.)	scope (fact./const.)	force (push/pull)
col		1.05	2.00	2.70	200
		0.00	0.00	0.00	0
path	1	0.00	0.00	0	70
		3.00	0.00	15	
slab	1	-1.00	-0.50	100	10
		-2.00	-2.00	-4	400

Fig 4.3. Image showing the situation for the first group. The model belongs to 'Stadsbalkon' developed by Fabian Scheurer. Stadsbalkon: "designtoproduction developed a software based on artificial-life methods, which optimized the exact position, inclination and strength of some 150 irregularly placed columns according to functional and constructional requirements, and which could be utilized by the architects to generate alternative design solutions." (Taken from: http://www.designtoproduction.ch/content/view/7/27/)

The applications or some of the software belonging to the first group may also be seen as being restricted to specific design problems, and may not always offer the designer the ability to develop their own unique models. These are new, promising developments, but are not always the result of a thorough analysis of applicability; and are unlike some of the "commercially available software," as Alexander Koutamanis calls them.[89]

[89] Alexander Koutamanis. "A Biased History of CAAD: the bibliographic version", in, J.P Duarte, G Ducla-Soares & A.Z Sampaio (Eds.), eCAADe 23: Digital Design: the Quest for New Paradigms, pp.634-635.

For example, the first group may include tools such as ‘Stadsbalkon’ (Fig 4.3) developed by designtoproduction (Java); and ‘CADenary’ developed by Axel Kilian (Processing) among others that have been developed in processing[90].

Experimental plug-ins developed for some commercial parametric software (similar to some commercial software, but not identical) may also be included in this group. Shape grammar implementations are one such plug-in, developed for AutoCAD for the generation of design solutions[91], as is the case for all of the tools in this group.

$$\underset{\text{LivingRoom - Apartment}}{\begin{array}{ccccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}}
\quad \text{o} \quad
\underset{\text{Apartment - Vestibule}}{\begin{array}{ccccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}}
\quad = \quad
\underset{\text{LivingRoom - Vestibule}}{\begin{array}{ccccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&0&1&0&0&0&0&0&0\\
1&1&1&1&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&0&1&0&0&0&0&0&0\\
1&1&1&1&1&0&1&0&0&0&0&0&0\\
1&1&1&1&1&0&1&0&0&0&0&0&0\\
1&1&1&1&1&0&1&0&0&0&0&0&0\\
1&1&1&1&1&0&1&0&0&0&0&0&0
\end{array}}$$

$$\underset{\text{LivingRoom - Vestibule '}}{\begin{array}{ccccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&1&1&1&1&1&1&1&0&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&1&1&1&1&1&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}}
\quad = \quad
\underset{\text{LivingRoom - Vestibule "}}{\begin{array}{ccccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&1&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}}$$

Fig 4.4. In FloP, constructing the automated design model is managed by defining all possible rectangle relations in terms of their constraints. This matrix has to be formed to define the relations between every two elements of the design problem until every element has a defined relation with the rest. For design generation with FloP see Fig5 in 2.2.2 (Taken from: “Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements”, by Can baykan in *E-Activities in Design and Design Education*. ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız, Paris: Europia Productions 2003, p.143)

In the second group, the model is not yet built, and the relations of parameters

[90] Daniel Davis (PhD Candidate, RMIT, Spatial Information Architecture Laboratory) compiles such examples in his blog. These include: “Dynamic Relaxation” tool (Processing): Mesh behaves like a membrane and takes on form with least stress. Or the “Photovoltaic Cell Shaper” (Processing): A tool to design the optimum photovoltaic cell based on location. http://www.nzarchitecture.com/blog/index.php/2009/11/05/beginning-programming/

[91] José Nuno Beirão (PhD Candidate, TU Delft, Design Informatics), can be referred for an implementation as such as a plug-in for AutoCAD.

and constraints are to be defined by the designer in the best way to describe the design problem; thus, designer develops the computational design model from scratch. The CAD tools that may be used in developing the models belonging to this group indicate their guiding principles in the user manuals and can be learned from user experience, rather than from CAAD theory.[92] In this way they differ from some commercial software, such as Rhino Grasshopper, that may be used to build the model. However, the computational components into which the reasoning is to be translated are based on the methodological knowledge of CAAD theory. Rather than the computational components themselves, it is the means of translating design reasoning into these components that are under discussion in this thesis; and designer actions are being proposed as a remedy, as has been discussed thus far.

The second group may be interpreted as the "effective use of CAD tools for design"[93]. Considering the model adopted to follow the computational method in consideration, this group includes parametric models developed by the designer using various forms of parametric software. Different to the "toy applications," some CAD tools offer lines, surfaces, etc. with which the designer translates the design reasoning from scratch in a non-deterministic way. In some parametric software, the designer may start modeling the constraints using sequences of relations that are grouped according to the Grasshopper definitions of recursive functions explained in Chapter 5. While "toy applications" can be considered as CAAD tools, these parametric software cannot, as they are rather for the building of computational design models.

For a clear description of the differences of each software, Earl Mark, Mark Gross, and Gabriela Goldschmidt can be referred: "The internal representations

[92] Alexander Koutamanis. "A Biased History of CAAD: the bibliographic version", in, J.P Duarte, G Ducla-Soares & A.Z Sampaio (Eds.), eCAADe 23: Digital Design: the Quest for New Paradigms, p.634

[93] Scott Chase. "Generative design tools for novice designers: Issues for selection", *Automation in Construction*, vol.6, issue 4, December 2005, p.691

built into these tools [CAD tools, or in general, some parametric software] have also advanced, from simply managing points, lines, and planes [features of CAD tools], to parametric and constraint based representations [features of CAD tools – but the ones enabling parametric modeling, such as Grasshopper] such as those used by leading practitioners such as Foster and Partners, KPF, Morphosis, and Arup" [94].

Parametric software includes stack-based commercial software, such as Generative Components, or the generative modeling plug-in for Rhino, Grasshopper (GH)[95]. Moreover, performance-testing add-ons for Grasshopper (like Kangaroo or Geco) can be added to this group, as they are effective when used by the designer to build a model of their own.[96]

To be specific, the parametric model developed for the generation of adaptable structures in the research of Michela Turrin, Axel Kilian, Rudi Stouffs and Sevil Sarıyıldız[97] can be seen as the source of modules which form the model (Fig 4.5). These modules translate design reasoning into relations in the model, which in this case may be interpreted as embodying some recursive functions of architectural design problems that are used for the parametric modeling of adaptable structures. These functions are encoded into the design space of this model in Generative Components (GC), and thus it becomes possible to generate design solutions for adaptable structures by exploring the design space of that particular architectural design problem.

---

[94] Earl Mark, Mark Gross, Gabriela Goldschmidt. "A Perspective on Computer Aided Design After Four Decades", *eCAADe 26: architecture 'in computro',* ed. M. Muylle, 2008, p.170

[95] Other kinds of commercial software in this group include the stack based 3dsMax or Maya and CATIA with the associative history. CAD tools like Revit and ArchiCAD also have parametric features. A full explanation can be found on Daniel Davis' blog: http://www.nzarchitecture.com/blog/index.php/2010/01/19/parametric-software-review/

[96] Designers can use the resulting values coming from these performance-testing add-ons to generate design solutions by relating the constraints with each other accordingly. It is explained in more depth in Chapter 5.

[97] Michela Turrin, Axel Kilian, Rudi Stouffs, Sevil Sarıyıldız. "Digital Design Exploration of Structural Morphologies Integrating Adaptable Modules: A design process based on parametric modeling," *Proceedings of the 13th International Conference on Computer Aided Architectural Design Futures*, 2009, pp. 800-814.

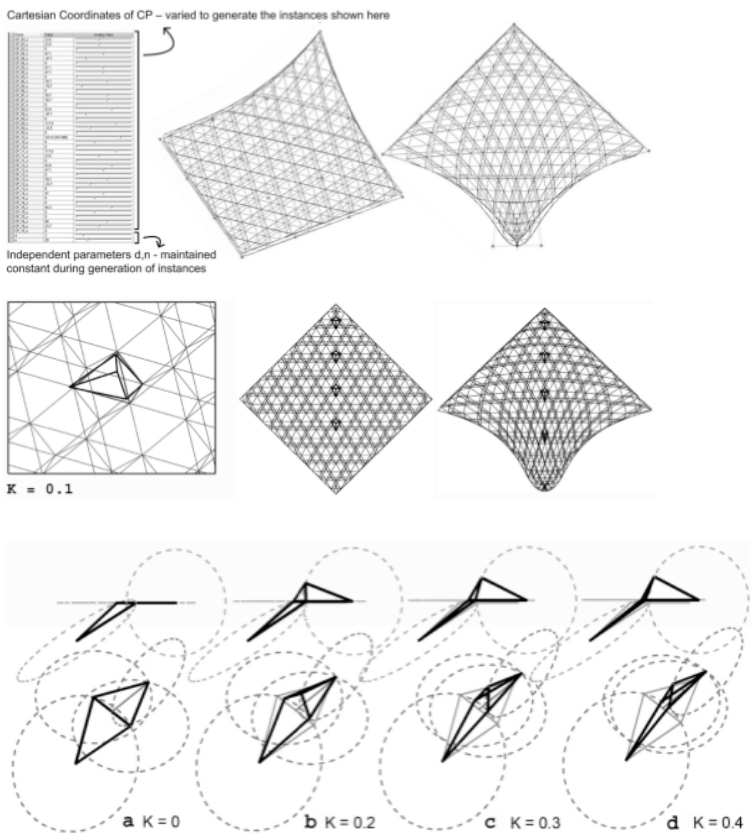

Cartesian Coordinates of CP – varied to generate the instances shown here
Independent parameters d,n - maintained constant during generation of instances
K = 0.1
a K = 0
b K = 0.2
c K = 0.3
d K = 0.4

Fig 4.5. The modules that are interpreted as embodying the recursive functions used for the parametric modeling of adaptable structures in the design research of M. Turrin, A. Killian, R. Stouffs, and S. Sarıyıldız (Taken from: "Digital Design Exploration of Structural Morphologies Integrating Adaptable Modules: A design process based on parametric modeling", *Proceedings of the 13th International Conference on Computer Aided Architectural Design Futures*, 2009, pp. 800-814)

Ramesh Krishnamurti and Tsung-Hsien's design patterns for parametric modeling on GH, and Robert Woodbury's design patterns on GC (along with many other online definitions) may be considered as key references to the functions by which the designers model their design reasoning, rather than only filling in values for already defined relations. Within the scope of this thesis, modeling the design space refers to the modeling of constraints in an intuitive and a synthetic manner, ensuring that design reasoning is translated efficiently.

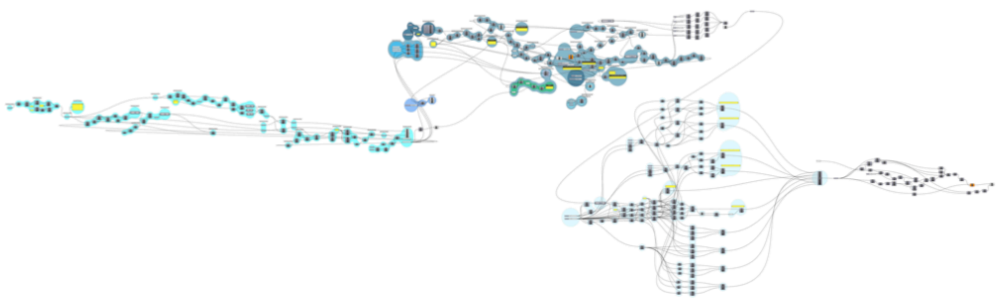
Different translations can lead to different solution spaces, just as each designer may aim at a different solution space for a specific design problem. After all, it is the design reasoning that is to be modeled when modeling the solution space[98]; and as a solution, mass customized sequences of relations may allow the designer to play a more efficient role (Fig 4.6).

To recap, mass customized sequences of relations, known as designer actions in this thesis, may include mathematical or geometrical aspects in modeling the constraints, however, as discussed in 3.2.1, there exist various ways of mass customizing relations, such as one recursive function, expressed as an algorithm, of a geometrical theorem like circle packing; or by embedding the fabrication constraint into the geometric property of the chosen primitive. The common factor when using a mass customized relation is that the model is open to the operation of the designer both before and after developing it.

The approach in this thesis, to translating design reasoning by modeling the constraints is apparent in the use of such relations for building the model of the graduation project. In Fig 4.6 (a), full Grasshopper definition developed for the model of the graduation project can be seen. Different sequences of relations can be followed. Different colors refer to different functions that belong to different scales. They embody constraints belonging to different scales. In Fig 4.6 (b) some parameters explored via constraints are exemplified. These are only one part of the parameters that the design elements are translated into. Author can modify both the parameters and the design elements. The definition is capable of changes. It responds to the changes with a new model because the functions are encoded with abstract, modifiable and reusable relations that relate the design constraints following a sequence.

[98] For instance, contemporary practices (such as Foster+Partners and Gehry Partners and FOA and Zaha Hadid Architects, etc.) which follow computational methods for modeling the design space of the architectural design projects, differ a lot in terms of their aimed solution space – differ a lot in terms of relating which design constraint to which for instance.

a



b

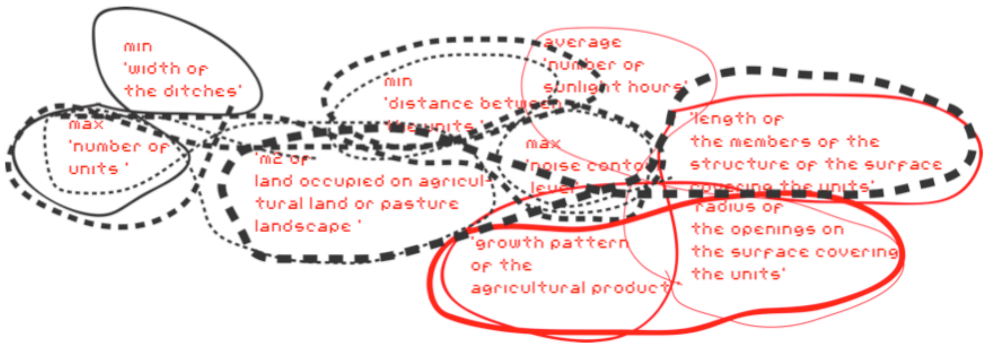


Fig 4.6. (a) Full Grasshopper definition developed for the model of the graduation project. Different colors refer to different functions that belong to different scales. They embody constraints belonging to different scales. (b) Some parameters explored via constraints are exemplified. These are only one part of the parameters that the design elements are translated into. Author can modify both the parameters and the design elements. The definition is capable of changes. It responds to the changes with a new model because the functions are encoded with abstract, modifiable and reusable relations that relate the design constraints following a sequence.

As Köknar and Erdem mention, in architectural design, tools may be borrowed from other disciplines, offering the example of "circulation as a separate and positive entity inside the body of a building," which dates back to the encyclopedia age.[99] They also point out that the "coin was imported from the respiratory system to design culture,"[100]; while also mentioning the usage of "collage as a tool to compose masses and integrate buildings to their

[99] Sait Ali Köknar, Arzu Erdem. "Can creativity be institutionalized?", *A-Z ITU Journal of the Faculty of Architecture*, 4(2), 2007, p.33

[100] *ibid.*

surroundings,"[101] which they claim was borrowed from the plastic arts of the avant-garde movements at the start of the 20th century.

In Pamphlet Architecture 27: Tooling, Benjamin Aranda and Chris Lasch follow a similar route, proposing seven "recipes," borrowed natural phenomena in the world, that embody functions (expressed as algorithms) that keep the translation of reasoning alive and allow the designer to dominate. They conduct a similar research on the methods "to better understand"[102] the designs, in other words, to better grasp the complexity of the design space of the architectural design problems with the help of the computational design model.

Aranda/Lasch propose seven algorithmic techniques that describe and simulate design reasoning, namely spiraling, packing, weaving, blending, cracking, flocking and tiling. In the foreword, Cecil Balmond says of the authors: "Their method offers endless potential for the interpretation of program,"[103] and their work is also deemed highly relevant to this thesis too, in which a reconsideration of the nature of programming in architecture is considered a priority. As a reminder, programming in architecture involves the proposal of necessary actions for the smaller tasks of a complex architectural design problem.

The seven techniques detailed in the study of Aranda/Lasch use mathematics and geometry to control the task of executing the aimed relation between the design constraints. They provide examples of various usages of these tools in different contexts, referencing various architectural design projects. For example, they describe spiraling as an essential recipe in plant growth patterns, as it allows the maximum amount of plant to grow in the least amount of

---

[101] Sait Ali Köknar, Arzu Erdem. "Can creativity be institutionalized?", *A-Z ITU Journal of the Faculty of Architecture*, 4(2), 2007*,* p.33

[102] B. Aranda, C. Lasch. *Pamphlet Architecture 27: Tooling*, New York: Princeton Architectural Press, 2006, p.94

[103] *ibid.*, p.7

space[104]; but explain that it can be used to address an infrastructural problem concerning car traffic, giving an idea of the maximum number of cars that can exist on the move, avoiding any major traffic jams that may occur (Fig 4.7). From this it can be understood that such a tool may respond efficiently to the concurring needs of agricultural and infrastructural constraints, among others, as explained in Chapter 5.

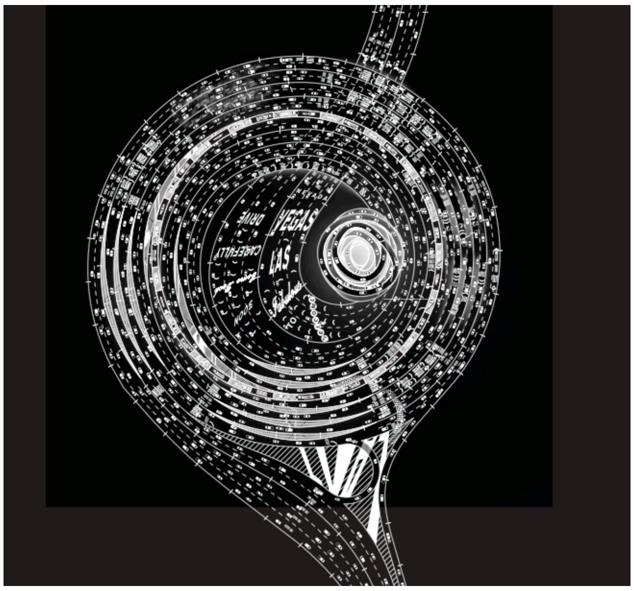


Fig 4.7. Spiraling used in an architectural design project (Taken From: *Pamphlet Architecture 27: Tooling*, by B. Aranda, C. Lasch, New York: Princeton Architectural Press, 2006, p.19)

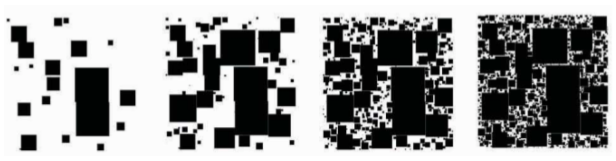
Moreover, if packing, for example circle packing, is used, as can be seen in Fig 4.8 (a), it affords "stability through adjacency,"[105] performing the task of providing a stable structure in its simplest sense. However, it can also be used to relate various constraints of various scales, depending on the designers' reasoning. It is shown in the next chapter that this technique may be used to "calculate the maximum number of units that can be constructed on the project plot," similar to that seen in Fig 4.8 (b). However in the latter example, rather

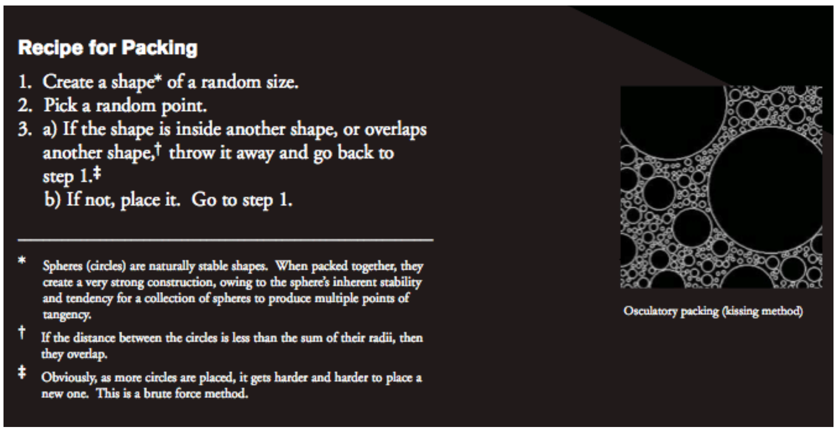
[104] B. Aranda, C. Lasch. *Pamphlet Architecture 27: Tooling*, New York: Princeton Architectural Press, 2006, p.12

[105] *ibid.*, p.22

than stability, the main intention lies in achieving the minimum distance between units to avoid high noise levels from passing aircraft.

Other architectural design projects are given in the study of Aranda/Lasch to exemplify different contexts – such as the façade of a log cabin – to suggest possible different usages of packing. The techniques are expressed in the form of algorithms, which are given in the recipes for the techniques (Fig 4.8 (c)). Scripting is used as a tooling device for the mass customizing of techniques, meaning that all seven recipes are encoded in the form of algorithms.





Recipe for Packing

1. Create a shape* of a random size.
2. Pick a random point.
3. a) If the shape is inside another shape, or overlaps another shape,† throw it away and go back to step 1.‡
   b) If not, place it. Go to step 1.

* Spheres (circles) are naturally stable shapes. When packed together, they create a very strong construction, owing to the sphere's inherent stability and tendency for a collection of spheres to produce multiple points of tangency.

† If the distance between the circles is less than the sum of their radii, then they overlap.

‡ Obviously, as more circles are placed, it gets harder and harder to place a new one. This is a brute force method.

Osculatory packing (kissing method)

Fig 4.8. (a) Packing which executes the task of bringing "stability through adjacency" (b) Square packing (c) Recipe for packing (Taken From: *Pamphlet Architecture 27: Tooling*, by B. Aranda, C. Lasch, New York: Princeton Architectural Press, 2006, p.26, 25, 23)

Accordingly, the concern of this thesis is also to exemplify the techniques/functions that are most commonly encoded in the sequences of relations of architectural design problems. For further examples of such relations, called designer actions, the author's graduation project can be referenced.

CHAPTER 5

PROGRAMMING THE DESIGNER ACTIONS: MASS CUSTOMIZING SEQUENCES OF RELATIONS AND PROCESSING PARAMETERS IN A SPECIFIC ARCHITECTURAL DESIGN PROBLEM

This chapter borrows from the author’s graduation project to suggest an approach to the defining of designer actions, and to present different examples of mass-customized sequences of relations that contain recursive functions of architectural design problems. Efficient translation of design reasoning relevant to design decisions taken at varying scales, is explained in detail. Great care is taken to ensure that no constraints are eliminated while shifting between different scales in the architectural design project.

As discussed previously, programming in architecture involves the proposal or selection of the designer actions needed for the smaller tasks in a complex architectural design problem. As a prerequisite, the proposal requires a programming of the sequences that relate design constraints with each other. This prerequisite will enable the mass customization of the relations that will result ultimately in designer actions. If this is not the case, a library or menu of mass-customized sequences of relations should be available at the outset for the designers to recreate them according to the needs of the design task. Both cases are applied by the author in the graduation project, and will be referenced accordingly while being explained.

The project proposes a new system that will allow the co-existence of agriculture (and green areas in general) and transportation infrastructure in the Schiphol region (Fig 5.1) in the Netherlands. The region contains a network of several different modes of transportation (Fig 5.2), and the dominant elements of architectural design for this region include Schiphol Airport and the adjacent agricultural fields (Fig 5.1). A project plot measuring 100m x 100m is to be selected from the area indicated in red (measuring 7.1 km^2) in Fig 5.2, depending on the author's choice. The area is located on the southeast of Haarlem, southwest of Amsterdam and on the northwest of Schiphol Airport. The reason for the parameterization of the position of project plot is related to the prior significance given to the constraints such as those exploring noise levels, climatic conditions or wind, light, etc., for different locations. Here, the parameterization of the position of the project plot is programming it as one parametrically defined square – that can be moved in x, y and z directions, etc. – in the function encoded in the Grasshopper definition, which can be seen in Fig 5.3.

To help visualize, firstly an instance from the model is given in this paragraph. The process explained here is similar to the latter that explain other instances; therefore this paragraph aims a brief and a direct introduction to the model and the sequences in it. If a project plot (indicated in green, Fig 5.3 (e)) measuring 100m x 100m is selected from the area, which is indicated in red (measuring 7.1 km^2) in Fig 5.2, the sequence encoding the function follows as such: The curve indicating project plot, which can be seen in the Rhino scene in Fig 5.3 (a), is first set in the curve parameter – by clicking "Set one Curve" toggle", Fig 5.3 (b) – that is named "project plot". This curve parameter is created to parameterize the position of project plot, which is indicated in orange in Fig 5.3 (b). One curve with different number of edges, degree, length, etc. can be set in the parameter in addition to a square. This brings the modifiable character of the function encoded. It responds to various curves indicating project plots. To translate design reasoning, "project plot" is defined as one parameter. For this parameter, the relation "position" is defined. When a curve is set into the parameter, various numbers of units are generated. In Fig 5.3 (c) two units are generated. Number of units depends on the constraints such as those exploring noise levels, climatic conditions or wind, light, etc., for different locations of the project plot. The moving is further explained in Fig 5.21. In Fig 5.3 (d), the curve parameter "project plot", can now be seen in green because a curve is now set in the parameter.

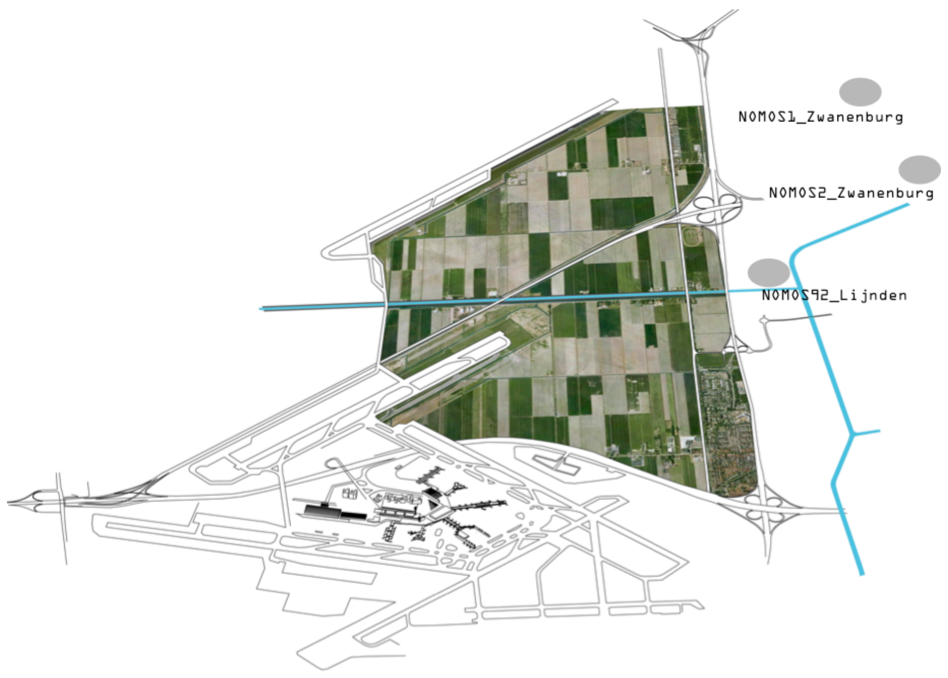


Fig 5.1. Schiphol Airport and the adjacent agricultural lands: These are the dominant land uses in the region, and are surrounded by a mobility network.

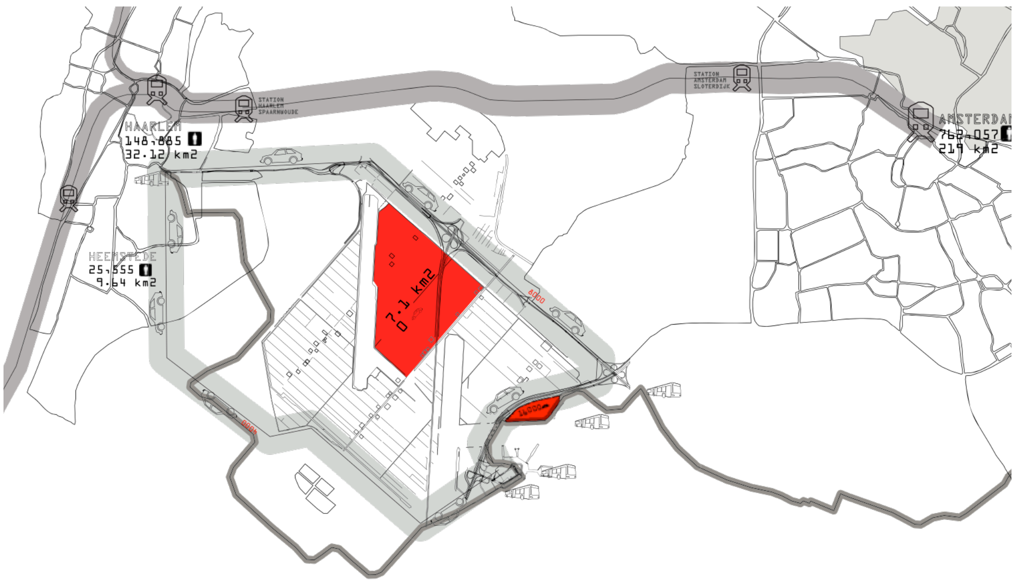


Fig 5.2. One part of the mobility network of the Schiphol region. Different means of transportation are highlighted, including the means of connecting Haarlem Station to Schiphol Airport. An area of 7.1 km^2 has been chosen for the architectural design problem, which is indicated in red. The area is located on the southeast of Haarlem, southwest of Amsterdam and on the northwest of Schiphol Airport.

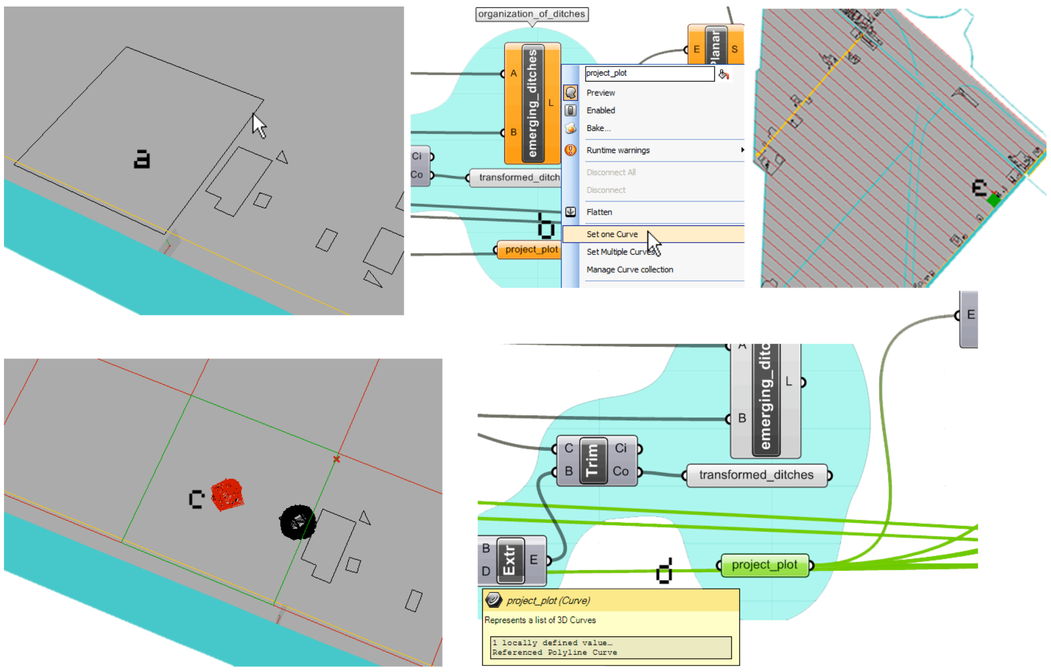


Fig 5.3. (e) A project plot (indicated in green) measuring 100m x 100m is selected from the area, which is indicated in red (measuring 7.1 km^2) in Fig 5.2. (a) The curve indicating project plot can be seen in the Rhino scene. It is not yet parameterized. (b) The curve parameter, which is created to parameterize project plot, is indicated in orange. (c) Two units are generated when a curve is set for the parameter “project plot”. Number of units depends on the constraints such as those exploring noise levels, climatic conditions or wind, light, etc., for different locations of the project plot. (d) The curve parameter, which is created to parameterize project plot, is now indicated in green because a curve is set in the parameter.

5.1 Content and the Context of the Architectural Design Problem

Mobility and infrastructure have a profound effect on the formation of the reclaimed land (Fig 5.5) and the delta metropolis in the Randstad region of the Netherlands (Fig 5.4). Although the region is actually several cities and a metropolitan area, within the hierarchy of the European mobility network, Randstad is considered to be a metropolis on its own. The name Randstad actually refers to a conurbation of cities and a metropolitan area in the region.

What needs to be considered in the Randstad is the transformation of constraints – such as those involving land transformations or various noise limits, or the changing climatic conditions in terms of wind, light, etc. The current constraints in the lowlands may not be the same in 10–50 years time. Such transformations are common in the drained and reclaimed lands of this region (Fig 5.5), which lie almost 5 m below sea level. For the transformation of the current situation in the Schiphol region (project site), noise levels, as well as climatic conditions, wind, light, etc., are considered to be crucial factors in the modeling of constraints to address the architectural design problem.

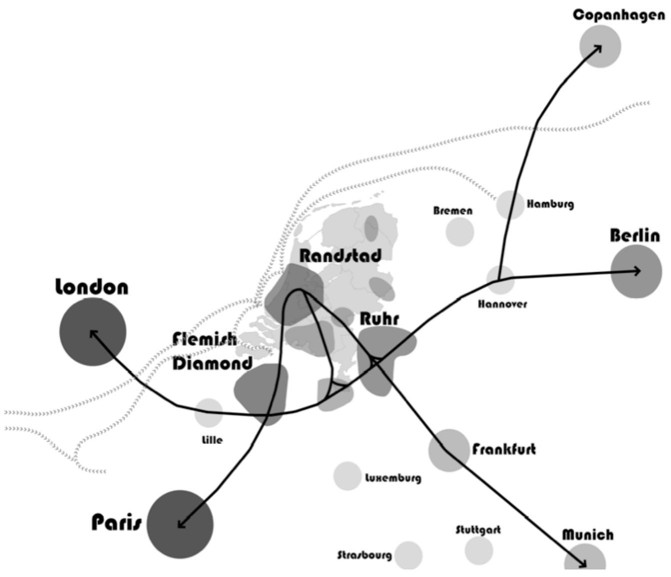


Fig 5.4. Randstad of the Netherlands within the European mobility network

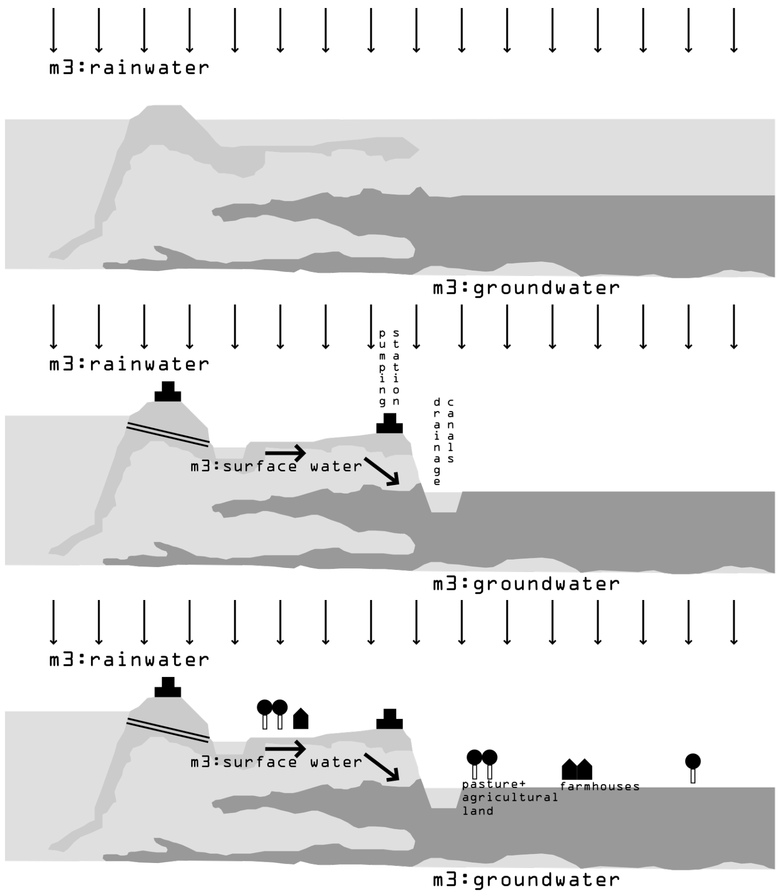


Fig 5.5. Process of reclaiming land (a) the land is below sea level, (b) pumping stations drain the seawater and (c) a polder system is introduced, including agricultural fields and farmhouses.

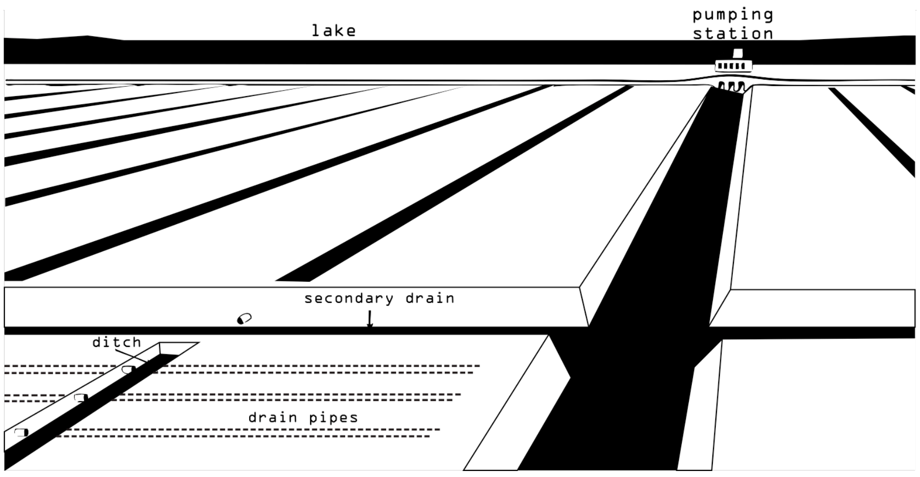


Fig 5.6. The polder system (Original image taken from: *Man-Made Lowlands: History of water management and land reclamation in the Netherlands,* by G. P. van de Ven (ed.) Utrecht, The Netherlands: Uitgeverij Matrijs, 2004)

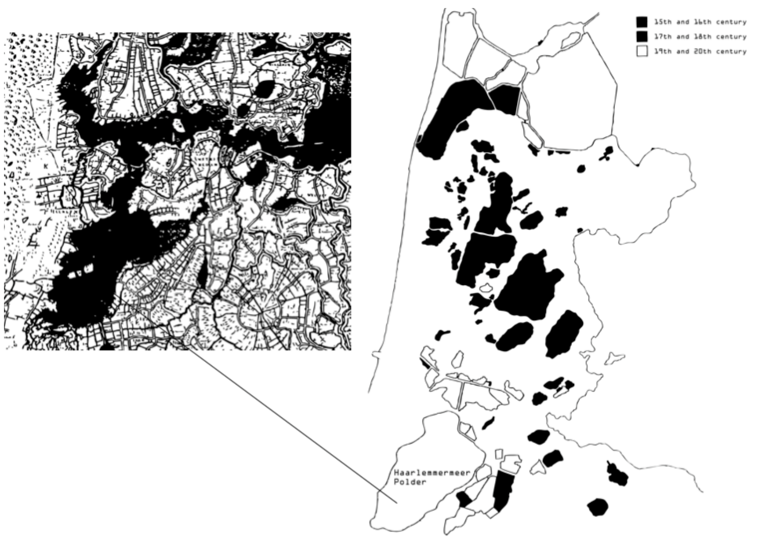


Fig 5.7. Reclamation of the Haarlemmermeer Polder in 1852 (Original images taken from: *Architecture Annual 2002-2003: Delft University of Technology*, ed. Henco Bekkering, The Netherlands: 010 Publishers, 2004 and *Sea of Land: The polder as an Experimental Atlas of Dutch Landscape Architecture,* by W. Reh, C. Steenbergen, D. Aten. Wormerveer-Zaanstad, The Netherlands : Stichting Uitgeverij Noord-Holland, 2007)

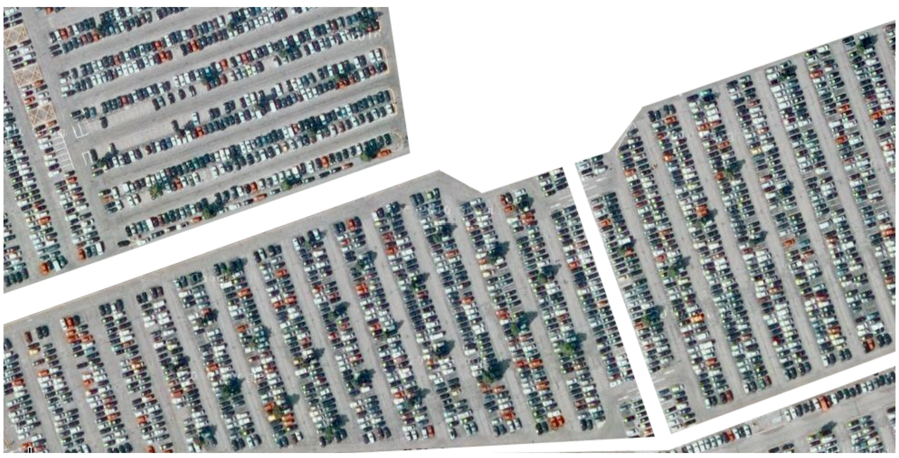


Fig 5.8. The parking lot at Schiphol Airport (Capacity: 16,000 cars)

Moreover, there are other aspects that need to be parameterized in the transformation of the region, such as the “number of passing cars,”[106] determined by the amount of people traveling, etc., which is a number that is growing due to the existence of an airport and two important cities in the region. This situation makes Haarlemmermeer, which was reclaimed in 1852, Fig 5.7 an attractive polder for both national and international companies; and for the development of different transport infrastructures, from high-speed rail tracks, to an underground logistics system. It is also attractive for the development of single-family dwellings with gardens and parking space for two cars.

New activities on this drained land require a new expression of land use and architectural space. However, spatial claims on the natural environment of the Haarlemmermeer Polder – on the agricultural land that has to be preserved within the dynamics of the reclaimed land (polder system) of the Netherlands – are threatening the existence of a

[106] The parking lot of Schiphol Airport (Fig 5.8) is especially important in this design problem, in that the growth and expansion of the infrastructure in the region is not based only to the presence of one central city. The chief driver of growth is Schiphol Airport, which drives the expansion of infrastructure, either directly or indirectly, threatening the agricultural lands. The airport both attracts and necessitates the infrastructure in the region.

variety of different agricultural products (Fig 5.9) and the landscape of the region.

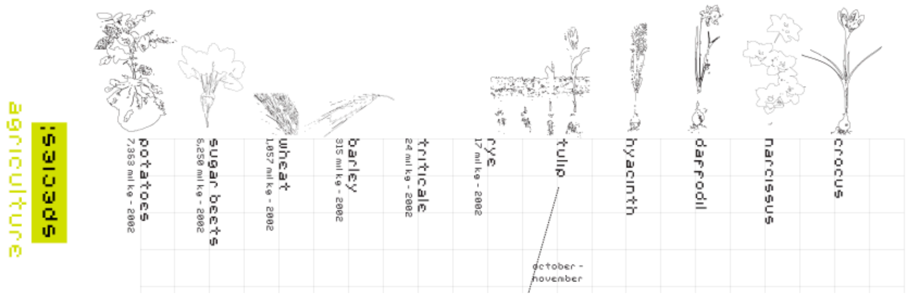


Fig 5.9. Image showing different species cultivated in the Netherlands.

To clarify, the threat to the region is more complex than just the expansion of a transport line. A definition of the modes of mobility and the densification of various elements within the Schiphol region is needed to ensure that a future scenario is addressed in the design space,[107] which shall be discussed later.

More than one architectural design solution has to be generated in order to find an optimum solution, and adopting a parametric and constraint-based design model can help the designer in this respect. Any solution should take into account the co-existence of different land uses in the Schiphol region in the future (Fig 5.10). Consequently, a solution space has to be modeled that relates different design constraints in groups, without eliminating any one of them. Only by modeling them in terms of their relations to each other can the results of the transformation drive the untransformed phases of the solution space that are related to the

[107] This is another reason for adopting a computational design model to explore the design space.

transformed groups of constraints.[108]

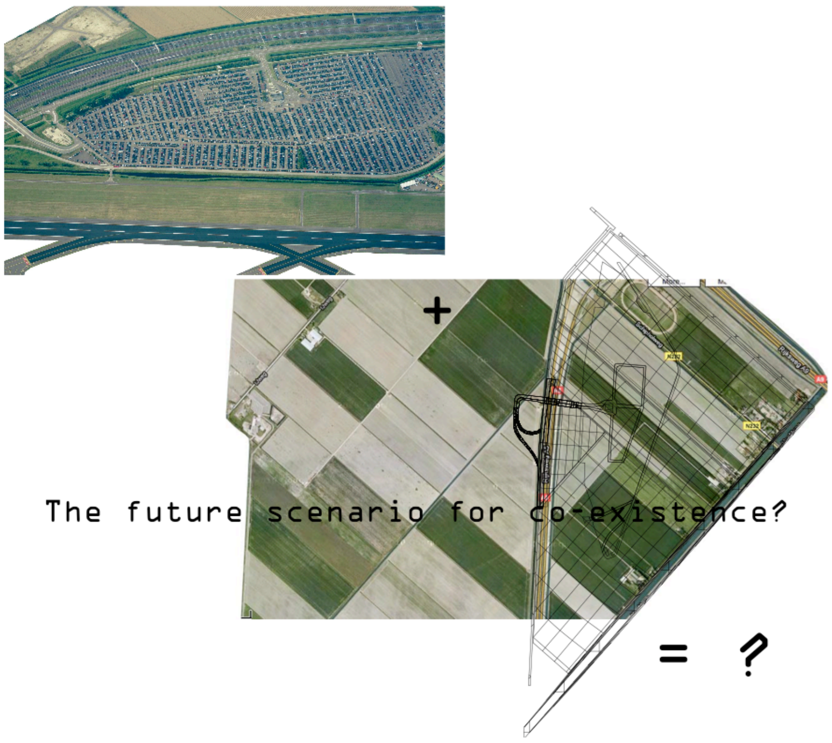


Fig 5.10. The parking lot of Schiphol Airport (top) is an important infrastructural element of the mobility network in the region, as is the agricultural land (bottom). Agricultural land is under threat from several different forces that are trying to find architectural expression in this region.

As the architectural design solutions have to be nourished by a future scenario that will allow transformation of the constraints, the new system to be implemented in the Schiphol region must be capable of the same level of flexibility in constraints, according to the planning policies and future scenarios in this region. The system needs to be adaptable in terms of different land uses and elements of infrastructure, and as a result it must be able to deal with a transformation in the constraints. To this end, abstract, modifiable and reusable sequences of relations that translate design reasoning into the components of parametric and constraint-based design is considered crucial in the development of the

[108] This also substantiates the need for a synthetic model for the exploration of the design space.

architectural design model. In summary, the design problem looks into the possibility of forming a new system comprising infrastructure, agriculture and habitation out of a hub in the Haarlemmermeer Polder.

5.2 Problem Space (Defining the Design Parameters)

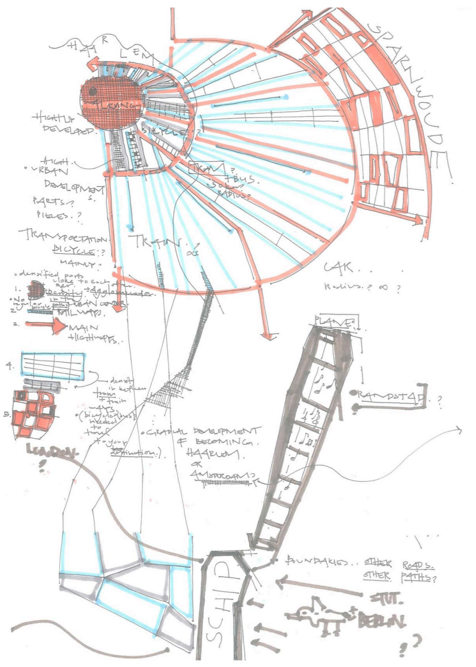


Fig 5.11. A sketch externalizing the reasoning related to the dynamics of the site, taking into account Schiphol Airport. Although the reasoning has been put to paper, it has not yet been translated into the computational components of the model.

In addressing architectural design problems, even if the reasoning has been put to paper in the form of a sketch (Fig 5.11), within the scope of this thesis externalizing design reasoning through a model refers to its translation into the computational components of the particular model. In this study, computational components of the model are the relations defined for the design parameters that are explored through the design constraints. As can be seen in Fig 5.12, several different design elements are parameterized; with different scales indicating different groups of constraints. In the same figure, the sketches positioned below different groups of constraints indicated by different scales show the instances of design reasoning that are to be translated into the particular components under consideration.

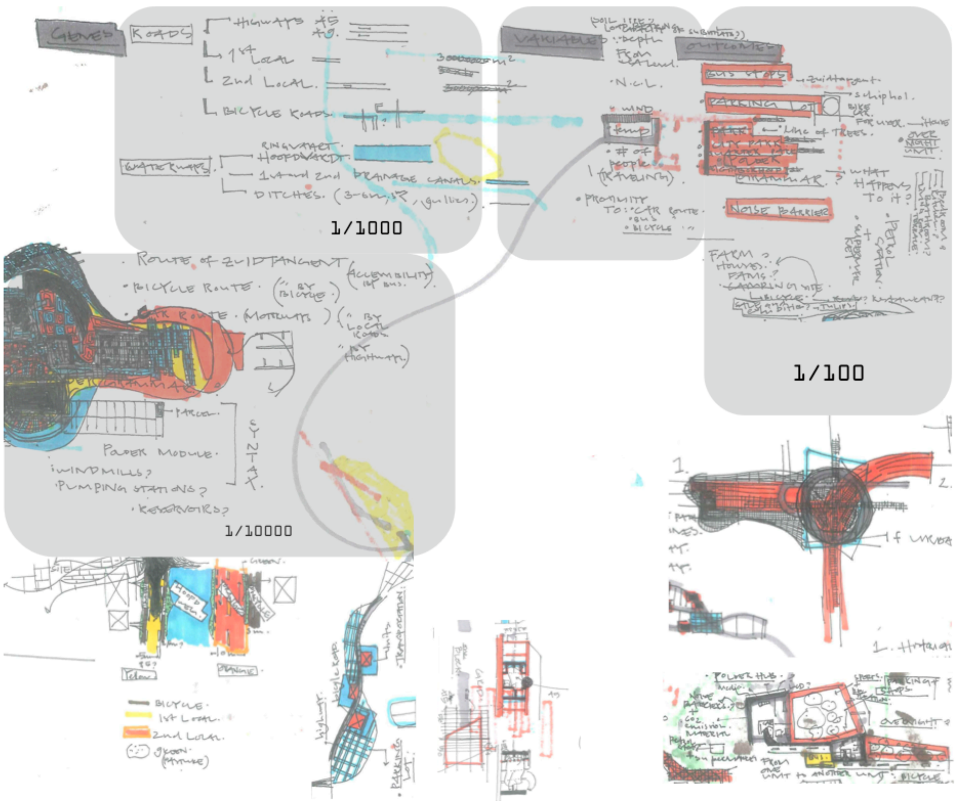


Fig 5.12. Different scales indicate different groups of constraints. The sketches positioned below different groups of constraints indicated by different scales show the instances of design reasoning that are to be translated into the particular components under consideration.

Relations are defined for the design parameters that are explored by design constraints and these constraints are related to one another to develop the model under consideration. [109] It is composed of three main groups of constrains in three main scales (Fig 5.14 – Fig 5.16): 1/1000, 1/100 and 1/10 (1/10000-scale design decisions come outside the limits of this model). The design problem is divided into these three main scales to allow a modeling of the constraints in groups.

[109] In other words, this is the translation of design reasoning into the computational components of the model developed for the graduation project of the author.

As 1/10000-scale design decisions are outside the limits of this design model, author begins exploring the design space of the model with a design decision already taken for that scale: One polder module (Fig 5.13) is chosen from the area (of 7.1 km^2, Fig 5.2) near Schiphol Airport. In that polder module, the position of one project plot – measuring 100m x 100m – is parameterized (Fig 5.16), to place it in various locations in the module. Because of the flexibility regarding its location, different values for parameters – such as different "noise contour levels;" various "m^2 of land occupied on agricultural land or pasture;"[110] different "number of hours of sunlight;" or varieties in "growth pattern of the agricultural product" – can be explored.

A definition of the relations between design parameters is explained in the following paragraphs, for 1/1000, 1/100 and 1/10 scales respectively. This translation is explained for the area allowed for the placing of the project plot, depending upon the author's choice. In the section where this translation is explained for 1/1000 scale through a grasshopper definition, the size of the project plot has a greater m^2 area than described in this section; however the design reasoning remains the same, differing only in the values of the relations defined for the parameters. Within the limits of this design model, the position of the project plot can be changed within the borders of the selected polder module (Fig 5.13). For the Haarlemmermeer Polder, one polder module, one polder block and one polder parcel cover areas of 2km x 3km, 1km x 3km and 1km x 200m respectively (Fig 5.13 – Fig 5.15). According to the decision taken by the designer taking into account the polder system of Haarlemmermeer, the project plot covers an area measuring 100m x 100m (Fig 5.16).

[110] This parameter is defined for calculating the m^2 of land set aside for infrastructure or for other land uses on agricultural land or pasture.

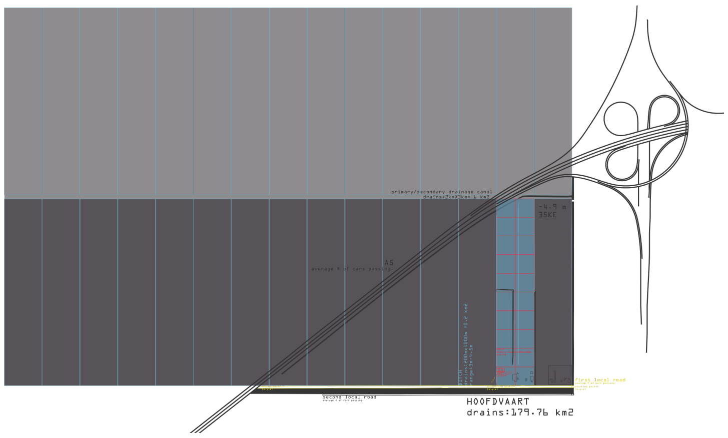


Fig 5.13. The polder module at 1/10000 scale: The group of constraints belonging to the 1/10000 scale drives the design. One polder module covers an area of 2km x 3km for the Haarlemmermeer Polder.

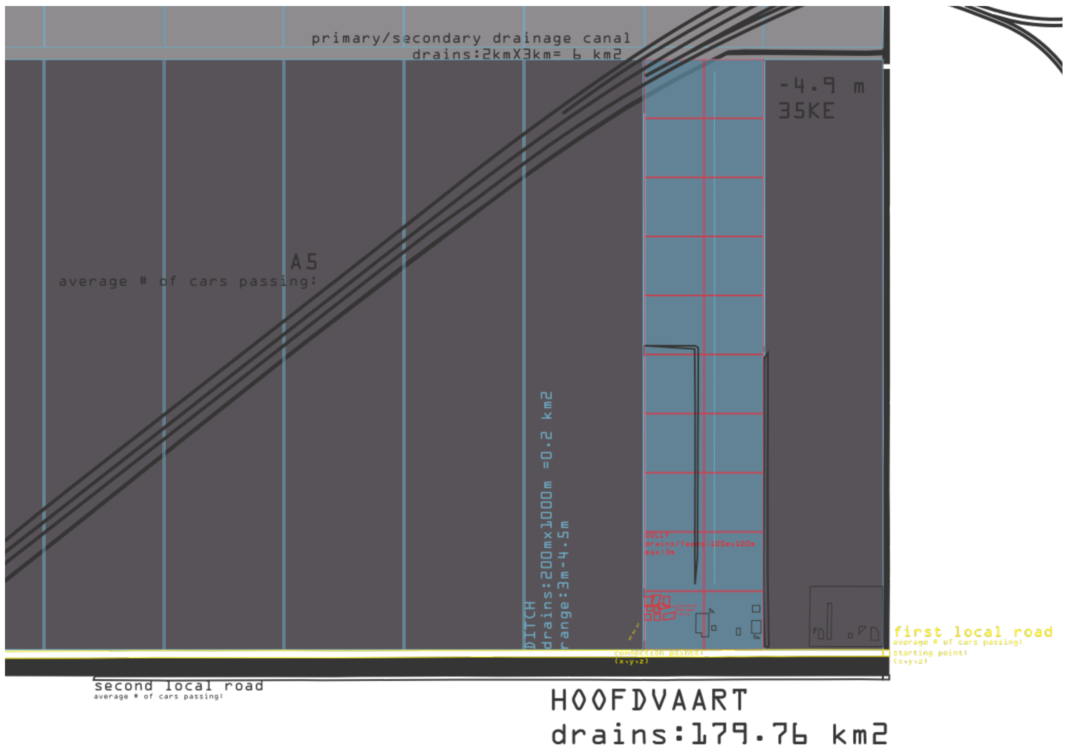

primary/secondary drainage canal
drains:2kmX3km= 6 km2
-4.9 m
35KE
A5
average # of cars passing:
DITCH
drains:200mx1000m =0.2 km2
range:3m-4.5m
first local road
average # of cars passing:
starting point:
(x;y;z)
second local road
average # of cars passing:
HOOFDVAART
drains:179.76 km2

Fig 5.14. The polder block (all of the darker gray area in Fig 5.13) at 1/1000 scale: The design is driven by the group of constraints belonging to scales 1/1000, 1/100 and 1/10, which are modeled in relation to each other. One polder module covers an area of 1km x 3km for the Haarlemmermeer Polder.

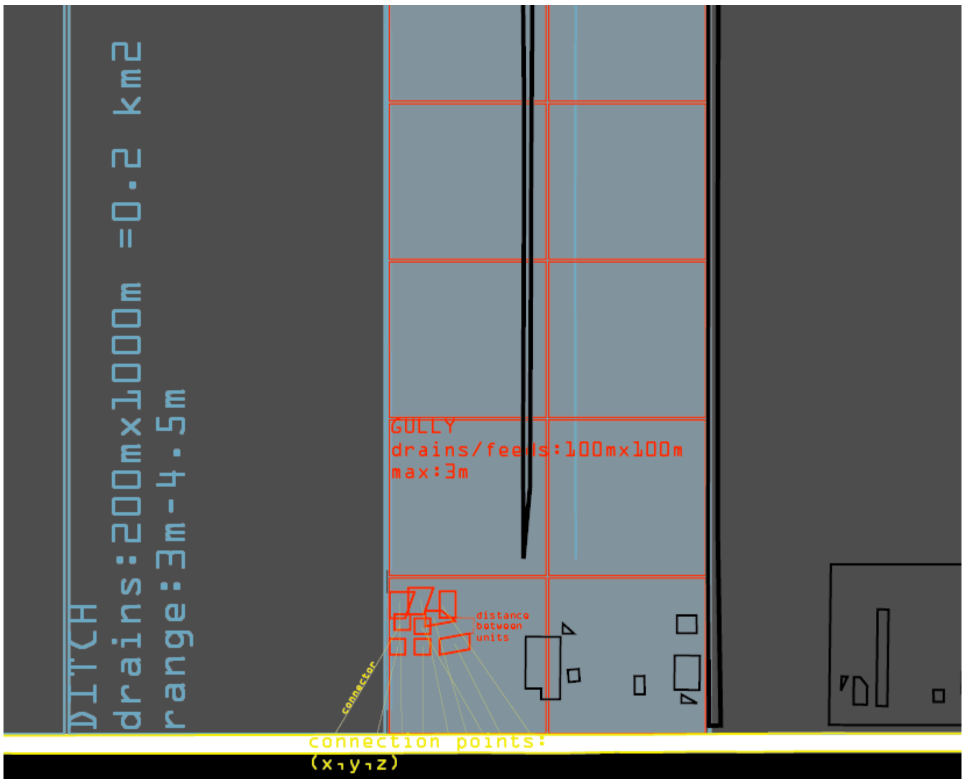


Fig 5.15. The polder parcel (all of the blue area in Fig 5.14) at 1/1000 scale: The design is driven by the group of constraints belonging to scales 1/1000, 1/100 and 1/10, which are modeled in relation to each other. One polder module covers an area of 1km x 200m for the Haarlemmermeer Polder.

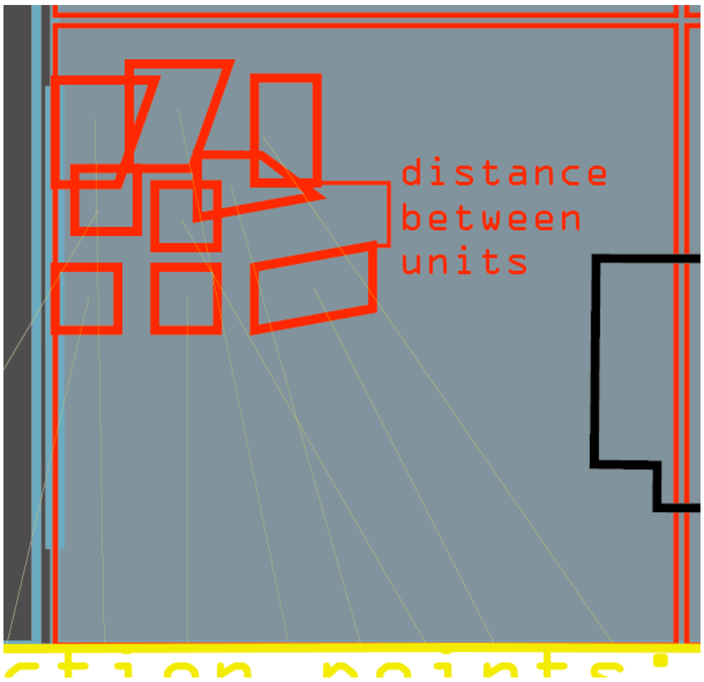


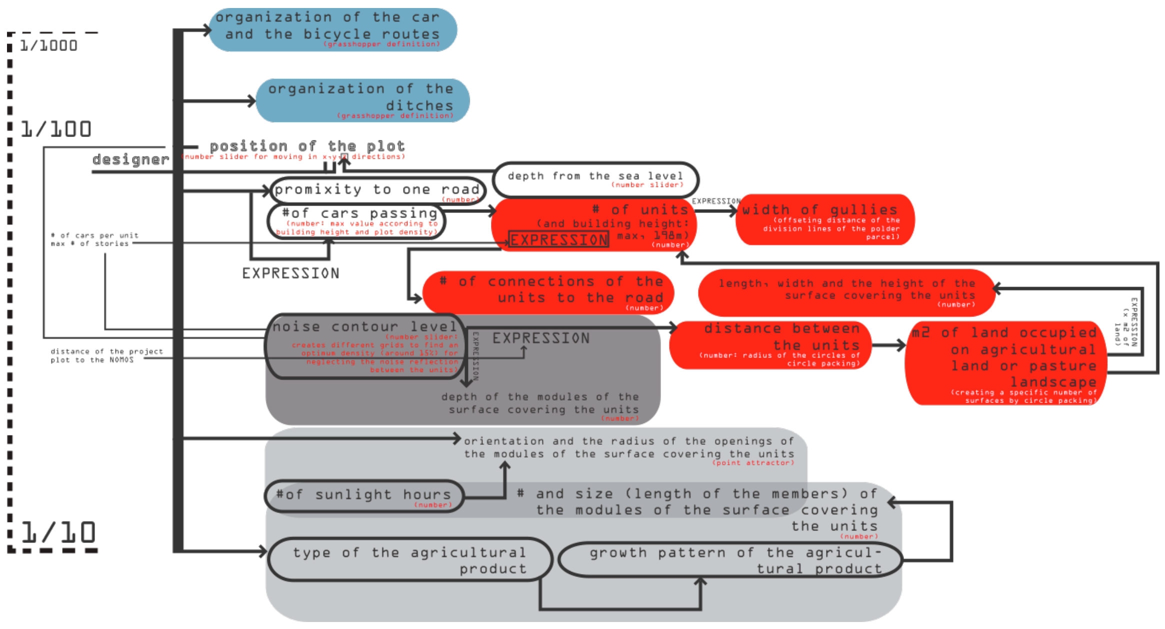
Fig 5.16. The project plot (one of the red squares in Fig 5.15) at 1/100 and 1/10 scales: The design is driven by the group of constraints belonging to scales 1/1000, 1/100 and 1/10, which are modeled in relation to each other. The project plot covers an area of 100m x 100m for every different position in which it is placed. Every

time the place of the project plot is changed, an area of 100m x100m is dealt with for the changes in the values of parameters.

While building the model, firstly, the design reasoning for the polder block scale (Fig 5.14) has to be translated in such a way that the new system – the design – can be implemented into the existing dynamics of the water management system of the Haarlemmermeer Polder. As a priority, the area in question needs to be drained. Relations are defined in such a way to best comply with the necessities of the design problem. Accordingly, at 1/1000 scale (Fig 5.14, 5.15), the parameters explored by the constraints include the "width of ditches". For the parameter "ditch," the relation "width" is defined, by which the "ditch" is most efficiently related with the other parameters explored through the constraints belonging to the other scales. This is explained further in 5.3.

At 1/100 scale (project plot - building scale, Fig 5.15), the parameters explored by the constraints include: "depth below sea level," "number of units," "length, width and the height of the surface covering the units," "distance between units" or "m^2 of land occupied on agricultural land or pasture" (Fig 5.17, in red). Some parameters may belong to 1/10 scale (project plot – building component scale), such as "noise contour levels," the "number of hours of sunlight" or (the subgroup) "growth pattern of the agricultural product" on a numerical basis (including min/max distances between two seeds, tectonic properties or solid-void proportions of the same pattern, etc); or the "radius of the openings in the surface covering the units" or the "length of the members of the structure of the surface covering the units". These can be seen marked in grey in Fig 5.17.

5.3 Solution Space (Relating the Design Constraints with Each Other)


1/1000
1/100
1/10
designer
organization of the car and the bicycle routes (grasshopper definition)
organization of the ditches (grasshopper definition)
position of the plot (number slider for moving in x,y,z directions)
depth from the sea level (number slider)
promixity to one road (number)
#of cars passing (number: max value according to building height and plot density)
# of cars per unit
max # of stories
EXPRESSION
# of units (and building height: max, 198m) (number)
width of gullies (offseting distance of the division lines of the polder parcel)
# of connections of the units to the road (number)
length, width and the height of the surface covering the units (number)
noise contour level (number slider: creates different grids to find an optimum density (around 15%) for neglecting the noise reflection between the units)
distance of the project plot to the NOMOS
distance between the units (number: radius of the circles of circle packing)
m2 of land occupied on agricultural land or pasture landscape (creating a specific number of surfaces by circle packing)
EXPRESSION (x m2 of land)
depth of the modules of the surface covering the units (number)
orientation and the radius of the openings of the modules of the surface covering the units (point attractor)
#of sunlight hours (number)
# and size (length of the members) of the modules of the surface covering the units (number)
type of the agricultural product
growth pattern of the agricultural product

⬜ already existing parameters represented via constraints for 1/100 scale (and for 1/10 and 1/1000 scales)

▩ already existing parameters for NCL represented via constraints for 1/10 scale (and for 1/100 and 1/1000 scales)

▢ already existing parameters for agriculture and pasture landscape represented via constraints for 1/10 scale (and for 1/100 and 1/1000 scales)

🟦 design parameters defined and represented via constraints (design drivers) for 1/1000 scale (and for 1/10 and 1/100 scales)

🟥 design parameters defined and represented via constraints (design drivers) for 1/100 scale (and for 1/10 and 1/1000 scales)

⬛ design parameters defined and represented via constraints (design drivers) for 1/10 scale (and for 1/100 and 1/1000 scales) (related with NCL)

▨ design parameters defined and represented via constraints (design drivers) for 1/10 scale (and for 1/100 and 1/1000 scales) (related with agriculture and pasture landscape)

Fig 5.17. This image explains the model developed by the author for the graduation project in which design parameters are explored via constraints. The various constraints belonging to different groups are shown in different colors. The relations formed between the constraints that belong to the same or to different groups are shown with lines and arrows.

To relate the first group of constraints at 1/1000 scale (Fig 5.17, in blue), a designer action in the form of a function is encoded in a Rhino Grasshopper definition (see appendix for definition). This function translates the reasoning of the designer in implementing this new system on the polder system of Haarlemmermeer, with acting in accordance with the dynamics of the existing water management system (Fig 5.21). At this scale, the constraints (Fig 5.18) are used to ensure a continuation of the existing rules of the water management system when draining one project plot, and to support author's scenario (in such actions as changing the position of the project plot). In this way, the author is able to build on the chosen project plot without having to deal with the water management system every time a change is made in the position of the plot or in the scale of the design problem. This is ensured as well for any other change that may take place on the plot – such as an increase in the "number of units," which brings more construction, more weight and larger volumes of water to be drained.

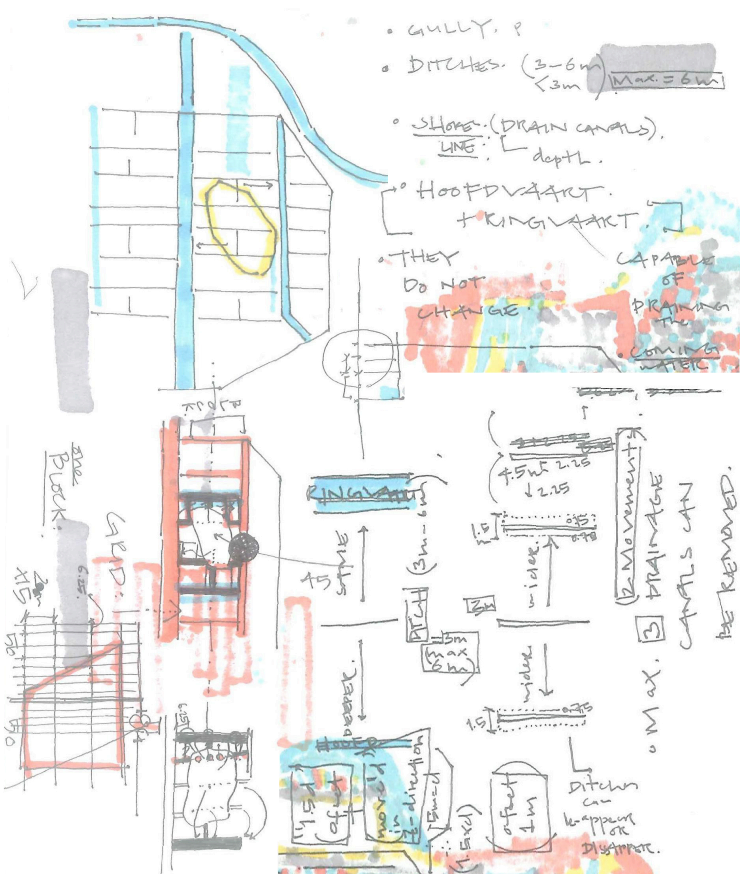


Fig 5.18. This sketch shows the constraints that are used to ensure a continuation of the existing rules of the water management system when draining one project plot, and to support author's scenario (in such actions as changing the position of the project plot).

The reasoning for the implementation to the existing water management system has now been put to paper, meaning that it has been externalized as defined in this thesis (Fig 5.22). It has been translated into the relations defined for the parameters explored by the constraints and related to the other groups of constraints to ensure that the translation has not been broken in any part of the architectural design problem. It is to ensure that the reasoning is fully translated and the model addresses the complexity of the problem. Moreover, if there is a change in the design parameters and constraints (if one constraint or parameter transforms into one another), it affects the whole design space, and the model is still processing. The consequences of this continuity may be effective in giving the model a synthetic character.

5.3.1 Relating Number of Units to Water Management System

The first sequence of relations that relate two different groups of constraints at 1/1000 scale (exploring "width of ditches") and at 1/100 scale (exploring "number of units") is as follows: If the project plot is positioned in the selected project site (Fig 5.2, in red), it is to be drained accordingly. The "number of units" in the project plot determines the weight of the construction, which then determines the amount of water to be drained. As it can be seen in Fig 5.19 (b), if 6 units are to be constructed on the project plot, its physical reflection on the overall design is that, the existing ditch line is offset to a specific value of width – that is 1.2m – according to the increasing weight of the construction to be made on the plot. The existing ditch line and the offset line are indicated in red and in green, respectively in Fig 5.19 (b). In Fig 5.19 (a), the existing ditch line is indicated in green. If 2 units are to be constructed on the project plot, then the width of the ditch is calculated accordingly. The effect of the change in the "number of units” on the “width of ditches” can be seen if Fig 5.19 (a) and Fig 5.19 (b) are compared. Grasshopper definition that encodes the function can be partially seen in Fig 5.19 (c). There, the value of width is calculated by multiplying offset value (2m) by the “number of units” and by dividing by 10, which results in 0.6. Offset value, and the “number of units” are two of the design elements that are parameterized in the function. Both the slider and the panel are indicated in green in Fig 5.19 (c). Offset value is parameterized with a number slider in the function, by which the author sets different floating points.

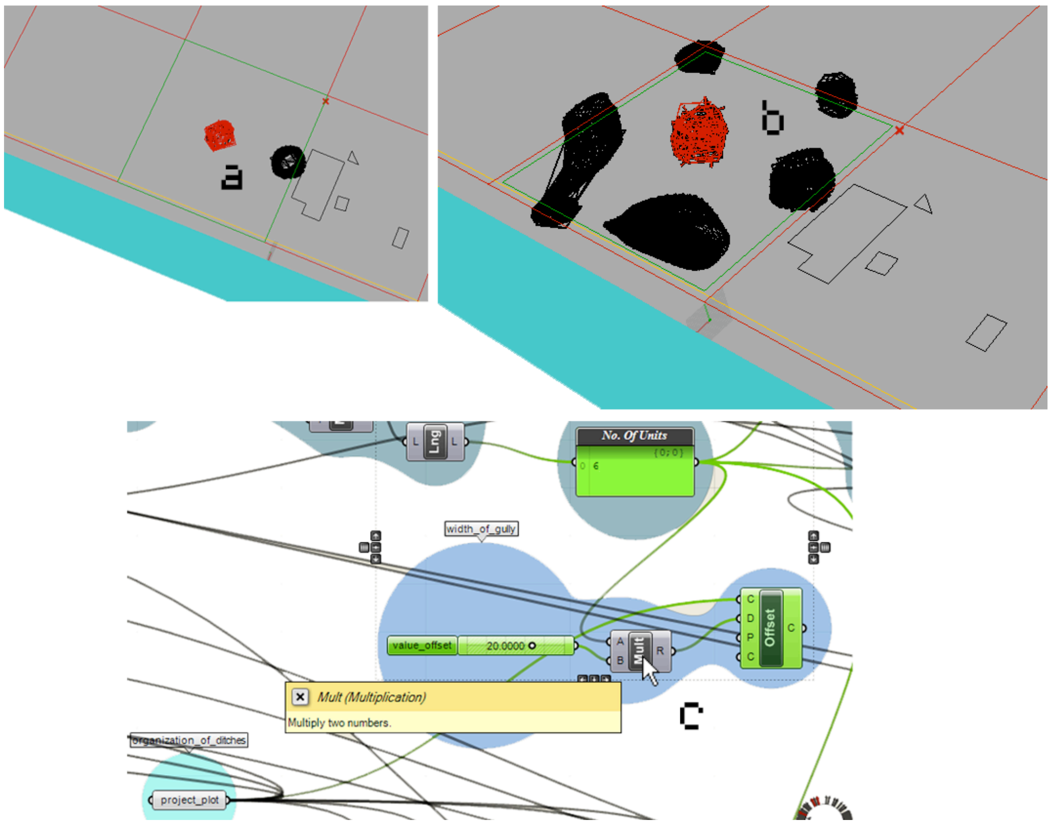


Fig 5.19. The “number of units” in the project plot determines the weight of the construction, which then determines the amount of water to be drained. If 6 units are to be constructed on the project plot, its physical reflection on the overall design is that, corresponding to the function developed and encoded in the Grasshopper definition, (b) the existing ditch line is offset to a specific value of width – that is 1.2m – according to the increasing weight of the construction to be made on the plot. Existing ditch line and the offset line are indicated in red and in green, respectively. (a) Existing ditch line is indicated in green. If 2 units are to be constructed on the project plot, then the width of the ditch is calculated accordingly. The effect of the change in the “number of units” on the “width of ditches” can be seen. (c) The value of width is calculated by multiplying offset value (2m) by the “number of units” and by dividing by 10, which results in 0.6. Offset value, and the

"number of units" are two of the design elements that are parameterized in the function. Both of the slider and the panel are indicated in green. Offset value is parameterized with a number slider in the function, by which the author sets different floating points.

The "number of units," is determined according to the constraints that explore the parameters "noise contour level" and "number of passing cars," etc., which will be explained later in greater depth. In this particular function (Fig 5.22), the excess water has to be transferred to the primary or secondary drainage canals (Fig 5.20, in blue on the left) via the ditches (Fig 5.20, in blue on the right). Therefore the excess water has to be transferred to the neighboring ditches first, to keep the land drained (Fig 5.21). However, within the hierarchy of the water management system of the Haarlemmermeer Polder, the drainage canals and the main Ringvaart and Hoofdvaart canals are assumed to be capable of handling the excess water feeding from the smaller ditches. As such, this function does not necessitate a change in these two main canals (Fig 5.20, in black on the left).

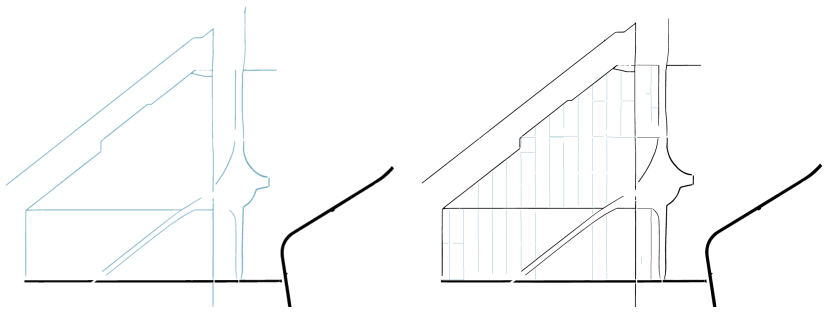


Fig 5.20. Image showing the primary and secondary drainage canals (in blue on the left) and ditches (in blue on the right) of the selected site of 7.1 km^2, shown in red in Fig 5.2.

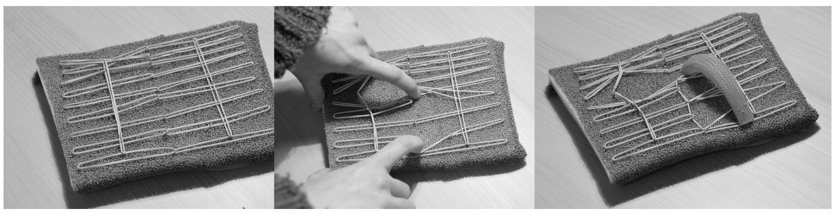


Fig 5.21. Design reasoning regarding the dynamics of the water management system, externalized in a physical model

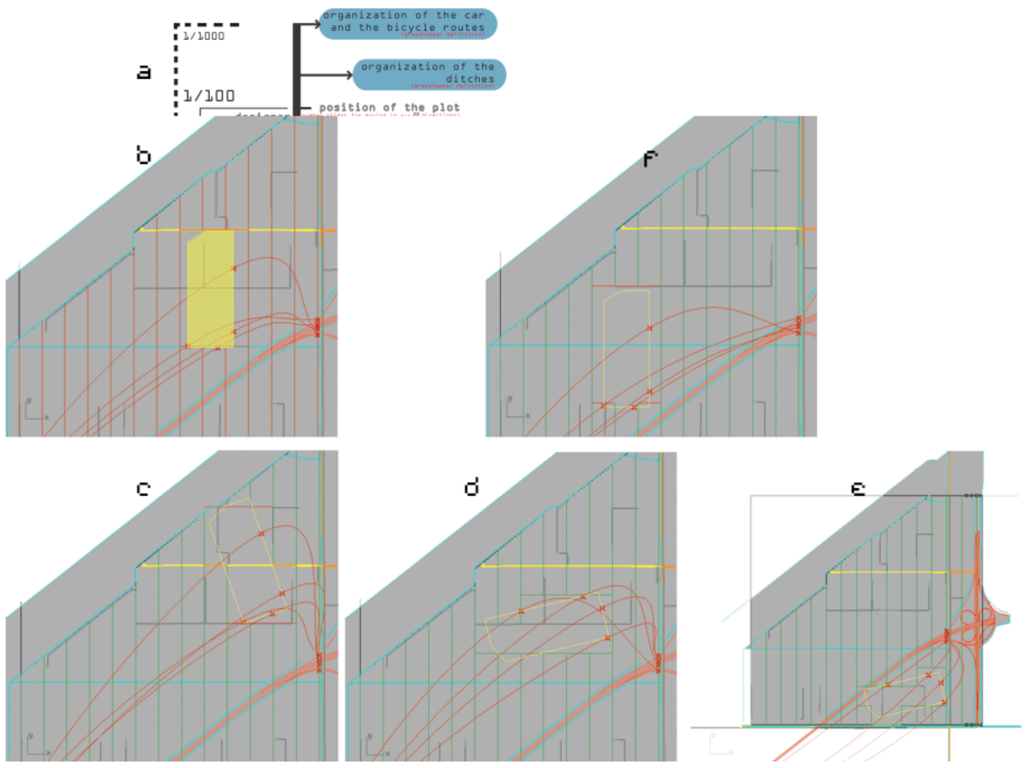

a
1/1000
1/100
organization of the car and the bicycle routes
organization of the ditches
position of the plot
b
f
c
d
e

Fig 5.22. The Grasshopper model of a re-forming of the ditches after different changes in position of the project plot can be seen. In the created definition, the ditches are re-formed in each case 100m from the borders of the project plot (this interval is slightly larger in these examples owing to the larger size of the project plot used). (b) The initial position of the project plot is shown in yellow hatching. Each of the above examples show different changes in position, being: (f) moving the plot along the xy plane or (e) rotating the plot in a counter clockwise direction or (d) moving the rotated plot again along the xy plane or (c) rotating the relocated plot in a clockwise direction. (a) The scale of the design model can be seen at the top.

5.3.2 Relating the Position of the Project Plot to Water Management System

In the function, which manages the transfer of excess water (Fig 5.22), the ditches are modeled according to parametrical and constraint-based relations that form the sequences created in Grasshopper. To clarify the function, the sequence in the Grasshopper definition can be explained as follows: Each time the position of the project plot changes, this function creates a drained plot on the site by transferring the excess water to neighboring ditches (Fig 5.21) and by creating new ditches according to an optimized grid. The effect of optimizing the grid to

100m and to 50 m on the overall design can be seen in Fig 5.23. In Fig 5.23 (c) or (d) the panel, which is created to parameterize the value of optimizing the grid, can be seen in green. In Fig 5.23 (c) the value is set to 100m and in Fig 5.23 (d) it is set to 50 m by the author. The value could be parameterized differently such as with a number slider or etc. The relation ("divide" component in Fig 5.23 (c) or (d)), which links this parameter to the rest of the definition, is modifiable and open to changes or transformations. This grid is optimized according to the constraints of the water management system in the Haarlemmermeer Polder, being the size of one polder parcel or minimum and maximum values of m^3 of water in the ditches or canals, and those exploring the "width" or "depth" of the ditches and canals, etc.

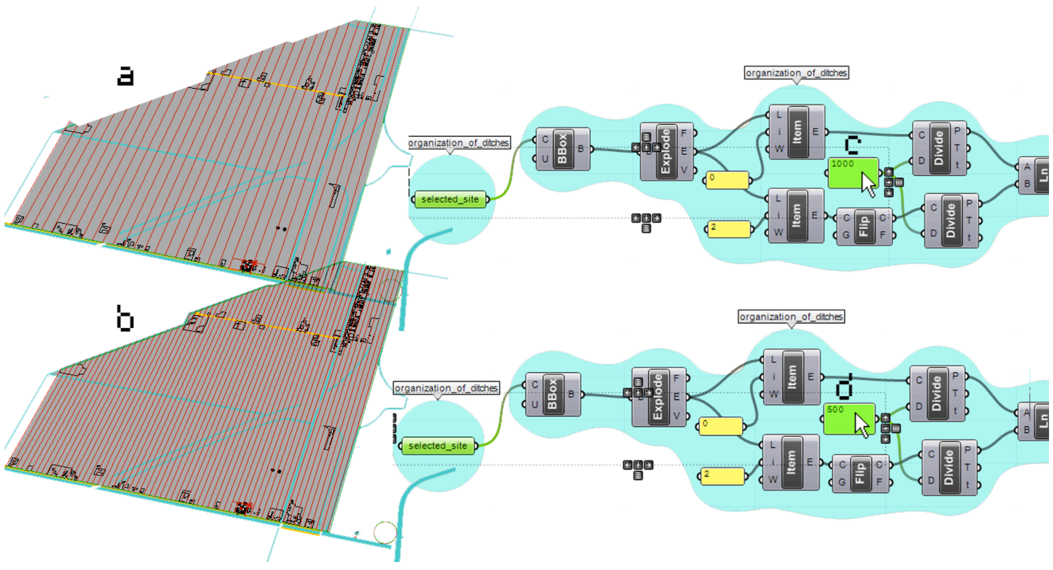


Fig 5.23. The difference between optimizing the grid (a) to 100m and (b) to 50m can be seen. (c) The panel, which is created to parameterize the value of optimizing the grid, can be seen in green. The value is set to 100m. (d) The panel, which is created to parameterize the value of optimizing the grid, can be seen in green. The value is set to 50m.

According to the optimized grid, the ditches are re-formed in every 100m from the borders of the project plot. This value of

100m should be changed in if the constraints change – if the polder changes for instance. This interval is slightly larger in Fig 5.22 (b), (c), (d), (e), (f) due to the larger size of project plot. The initial position of the project plot is shown in yellow hatching in Fig 5.22 (b). Various changes in the position of the plot are shown, such as: (f) moving the plot along the xy plane, (e) rotating the plot in a counter-clockwise direction or (d) moving the rotated plot again along the xy plane or (c) rotating the relocated plot in a clockwise direction. It is suggested that the function encoded here may be used in other contexts or in other scales in the project only by feeding some different elements of the design problem into it. This is a mass customized sequence of relations that respond to general usage. The function encoded in the sequence, which has been explained so far, can be seen in Fig 5.24. In this image, the significant parameters and the panel, by which the design elements are parameterized at 1/1000 scale, can be seen (Fig 5.24 (a), (b), (c)). If a transformation is needed, these three should be transformed into new ones and the sequence, which is indicated with the lightest tone of blue in Fig 5.24, responds with a new function. A full definition can be found in the appendix.

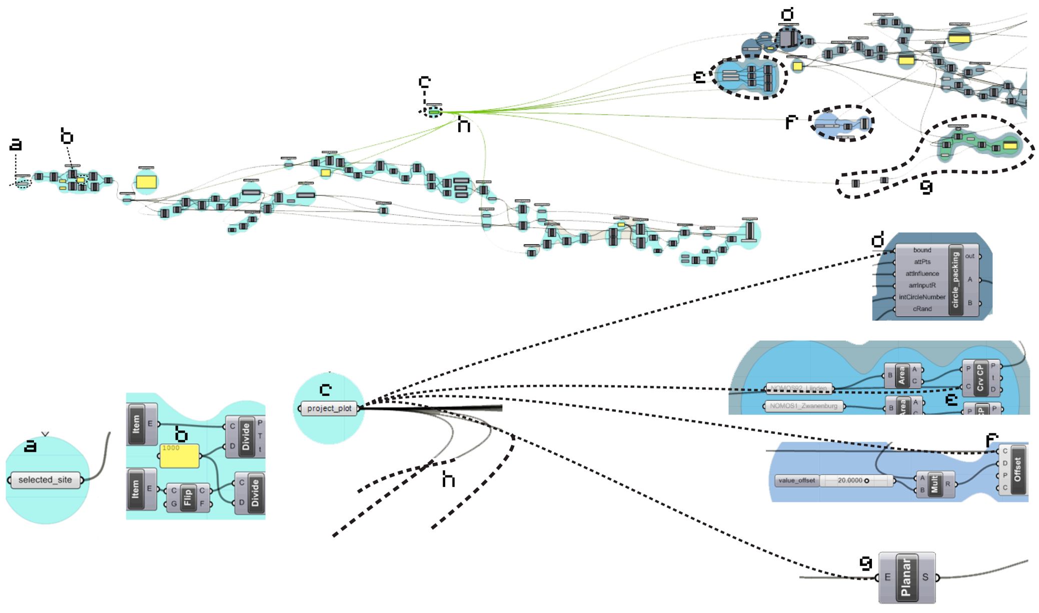
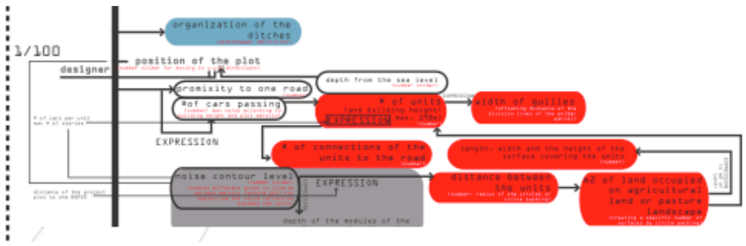


Fig 5.24. Significant parameterizations at 1/1000 scale can be seen: (a) "Selected site" in the Schiphol region, (b) the optimizing of the grid of the "selected site", (c) and the "position of project plot". "Position of project plot" is related to the other constraints at 1/100 scale. It links the two scales. In 1/100 scale it is linked (d) to the circle packing function, (e) to the function that calculates the effect of the noise level, (f) to the function that calculates the "width of ditches", and (g) to the function that calculates the building density in the plot. (h) It is also related to other parameters in the same scale that is 1/1000. In all, the particular curve that has been set into the "project plot" parameter, is also set into the "curve" input parameters' of the various components of these functions.

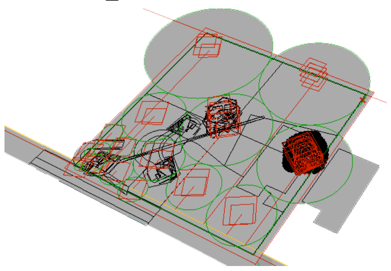
Author operates on the values and relations of the parameters shown in Fig 5.24 (a), (b), (c) for changes in the water management system of the selected project site, or changes in the size of the project plot, etc. Only by modeling the constraints in terms of their relations to each other can the results of a change drive rest of the solution space that are related to the changed groups of constraints. "Position of project plot" is related to the other constraints at 1/100 scale – it links the two scales. In 1/100 scale, it is linked to the circle packing function (Fig 5.24 (d)), to the function that calculates the effect of the noise level (Fig 5.24 (e)), to the function that calculates the "width of ditches" (Fig 5.24 (f)), and to the function that calculates the building density in the plot (Fig 5.24 (g)). It is also related to other constraints in the same scale that is 1/1000 (Fig 5.24 (h). In all, the particular curve that has been set into the "project plot" parameter, is also set into the "curve" input parameters' of the various components of these functions. If there is a change in the parameter "position of project plot", it is to ensure that the result of this change drives the design of the parts shown in Fig 5.24 (d), (e), (f), (g), (h) which are linked to the parameter "project plot". The consequences of this continuity may be effective in giving the model a synthetic character.

a

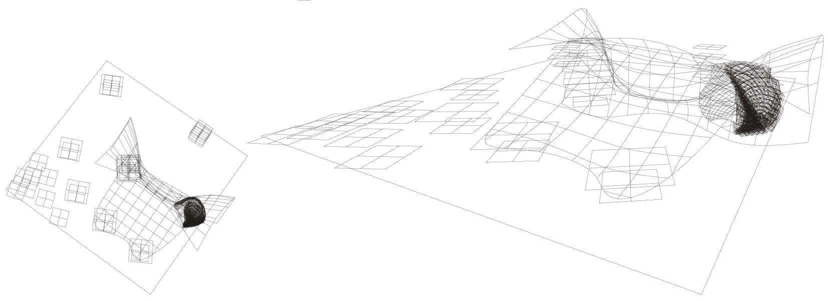


1/100
designer
organization of the ditches
position of the plot
depth from the sea level
proximity to one road
#of cars passing
# of units
width of gullies
EXPRESSION
EXPRESSION
# of connections of the units to the road
length, width and the height of the surface covering the units
noise contour level
EXPRESSION
distance between the units
#2 of land occupied on agricultural land or pasture landscape
depth of the modules of the

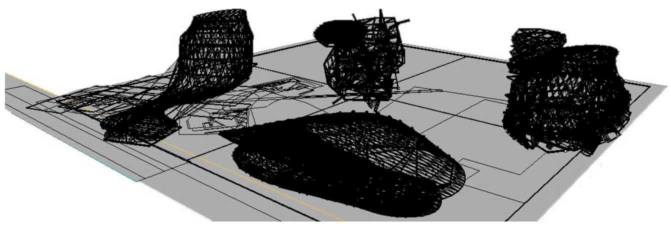
b



c



d



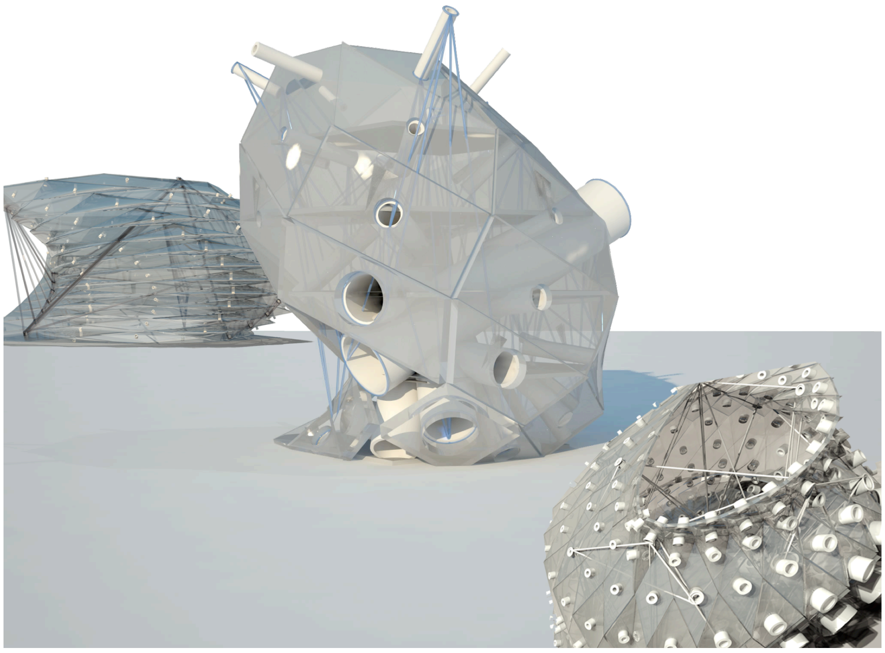


Fig 5.25. (b) The Grasshopper model of the plan in the event of a change in the “position of project plot”. The project plot can also be seen in Fig 5.15. The function encoded in the definition calculates the “number of units” every time the position of the project plot changes - yet the position change is explored not by moving the project plot to various locations in the site, but by changing the values of parameters such as: “noise contour level” or “number of passing cars”, etc. This definition uses circle packing to find the maximum “number of units” that can be constructed on the project plot. (c) Constructed units for a specific situation (for a specific noise level, etc.) and the “number of stories per unit” in that specific situation can be seen. (d) Each time the position of the project plot changes; a different surface covering each unit is generated as well. The design of these surfaces will be explained later. (a) The scale of the design model to which the function belongs can be seen at the top in red.

<u>5.3.3 Relating the Position of the Project Plot to Noise levels and the Number of People Traveling</u>

It is not only the first group of constraints belonging to 1/1000 scale that drives the design of the project plot in 1/100 and 1/10 scales. For instance, as the value for the parameter “number of passing cars” changes, which is determined according to the value “number of people traveling” – or the value for the parameter “noise contour level” changes, the “number of units” to be constructed on the project plot changes accordingly (Fig 5.25 (b), (c), (d)). Circle packing is used when modeling the particular constraints (Fig 5.25 (b)); and is used to calculate the minimum “distance between units” to ensure the maximum “number of units” in the project plot. In the calculation of finding the maximum “number of units”, the “number of passing cars” is the parameter explored by the concurring constraint, while the “noise contour level” is the parameter explored by the conflicting one.

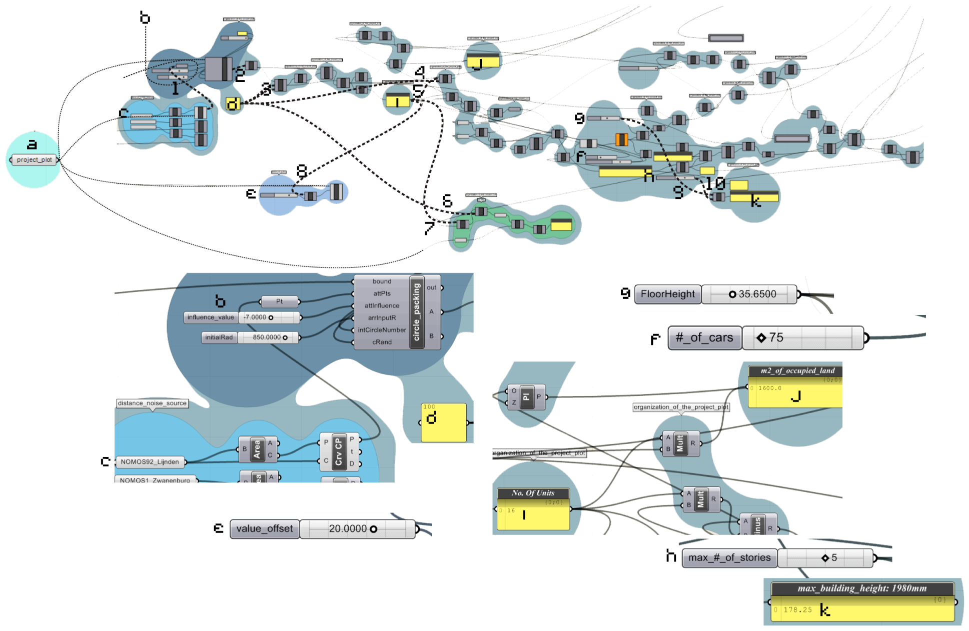


Fig 5.26. Significant parameterizations and significant relations of parameters, number sliders, panels and components at 1/100 scale can be seen in the Grasshopper definition. Numbers from 1-10 indicate the significant relations in the sequence that encodes the function. Letters from b-k indicate significant parameterizations, including 2 parameters, 1 panel, 6 number sliders from b-h, by which the author inputs the values to the function. The rest, being i, j, k, are the 3 panels created to parameterize the values depending on the values of b, c, d, e, f, g, h. To be specific, these are; (b) 2 number sliders and 1 point parameter that determine the effect of the noise source, (c) which has been parameterized with the curve parameter named “NOMOS99_Lijnden”, indicating the station located in Lijnden; (d) the panel created to parameterize the size of one unit being 10m x 10m; (e) the number slider created for the value of offsetting to calculate the “width of ditches;” (f) the number slider created for the “number of passing cars;” (g) the number slider created for the value of “floor height;” (h) the number slider created for the value of the number of stories in units; (i) the panel created for “number of units;” (j) the panel created for “m^2 of land occupied on agricultural land or pasture;” (k) the panel created for the maximum height of the units, which should not exceed certain limits. (a) On the left, the curve parameter, which is created to parameterize “position of project plot”, can be seen.

To clarify in the Grasshopper scene, significant parameterizations and some significant linking between parameters, number sliders, panels and components at 1/100 scale can be seen in Fig 5.26. In this image, numbers from 1-10 indicate the significant relations in the sequence that encodes the function. In the same image, letters from b-k indicate significant parameterizations, including 2 parameters, 1 panel, and 6 number sliders from b-h, by which the author inputs the values to the function. The rest, being i, j, k, are the 3 panels created to parameterize the values depending on the values of b, c, d, e, f, g, and h. To be specific, This parameterization involves the translation of design reasoning into: 2 number sliders and 1 point parameter that determine the effect of the noise source (Fig 5.26 (b)), which has been parameterized with the curve parameter named "NOMOS99_Lijnden," indicating the station located in Lijnden (Fig 5.26 (c)); the panel created to parameterize the size of one unit being 10m x 10m (Fig 5.26 (d)); the number slider created for the value of offsetting to calculate the "width of ditches" (Fig 5.26 (e)); the number slider created for the "number of passing cars" (Fig 5.26 (f)); the number slider created for the value of "floor height" (Fig 5.26 (g)); the number slider created for the value of the number of stories in units (Fig 5.26 (h)); the panel created for "number of units" (Fig 5.26 (i)); the panel created for "m^2 of land occupied on agricultural land or pasture" (Fig 5.26 (j)); the panel created for the maximum height of the units, which should not exceed certain limits (Fig 5.26 (k)); and the expressions defined in the particular components of these.

Significant relations in the sequence that encodes the function are indicated with numbers, from 1-10, in Fig 5.26. "Number of units" (Fig 5.26 (i)) is determined taking into account "noise contour levels" in order to ensure the project plot is suitable for inhabitation. The noise level at a specific location determines the minimum "distance between units" (Fig 5.26 (1)) in a project plot of 100m x 100m positioned in that specific location. The size of one unit in the plot is set 10m x10m (Fig 5.26 (d)). The aim here is to prevent the reflection of noise from passing aircraft between densely packed units of inhabitation. In this way, noise level is a determining factor in the maximum “number of units” to be built on the plot, alongside other constraints.

In Fig 5.26 (2) and (3) the linking of the output parameter of “circle packing” component and the “size of unit” panel, respectively, to the components that result in “number of units” panel, can be seen. Here, one thing should be noted: The new system, which proposes a new mode of mobility, needs to be adaptable in terms of different land uses and elements of infrastructure, and as a result it must be able to deal with a transformation in the constraints. For the system to be adaptable to different land uses designer should be able to parameterize different elements of infrastructure, etc. if needed – the model should be open to the operation of the designer. In the function encoded in Grasshopper definition (Fig 5.26), it is possible to parameterize different elements of design, rather than noise level for instance, for calculating the maximum “number of units” to be constructed. Maximum noise level can transform into maximum sea level if it is needed. This new constraint can also be related to the ones which noise level has once been related because the function is encoded with abstract, modifiable and reusable relations. It responds to changes with a new model for a new design scenario. The change is effective on the whole design space and the model is still processing.

The constraints in Fig 5.26 also include those exploring “building density,” “building height” and “number of stories per unit,” etc. In this particular case, the ranges of these values are determined according to the allowed values for construction in the site adjacent to Schiphol Airport. In Fig 5.26 (4) and (5) the linking of the panels “size of unit” and “number of units,” respectively, to the component that results in the panel “m^2 of land occupied on agricultural land or pasture”; and in Fig 5.26 (6) and (7) the linking of the same panels to the component of the function that calculates the “building density” in project plot, can be seen. The panel “number of units” (Fig 5.26 (i)) is also linked to the function that calculates the “width of ditches” (Fig 5.26 (8)). In Fig 5.26 (9) and (10), it can be seen that, number sliders “floor height” and “number of passing cars” are linked to the components that result in the panel “maximum building height.” With the function encoded in this definition, the part of the design scenario at 1/100 scale – constructing units as the new mode of mobility according to the amount of people traveling and noise levels, etc. – is parameterized and explored (Fig 5.27).

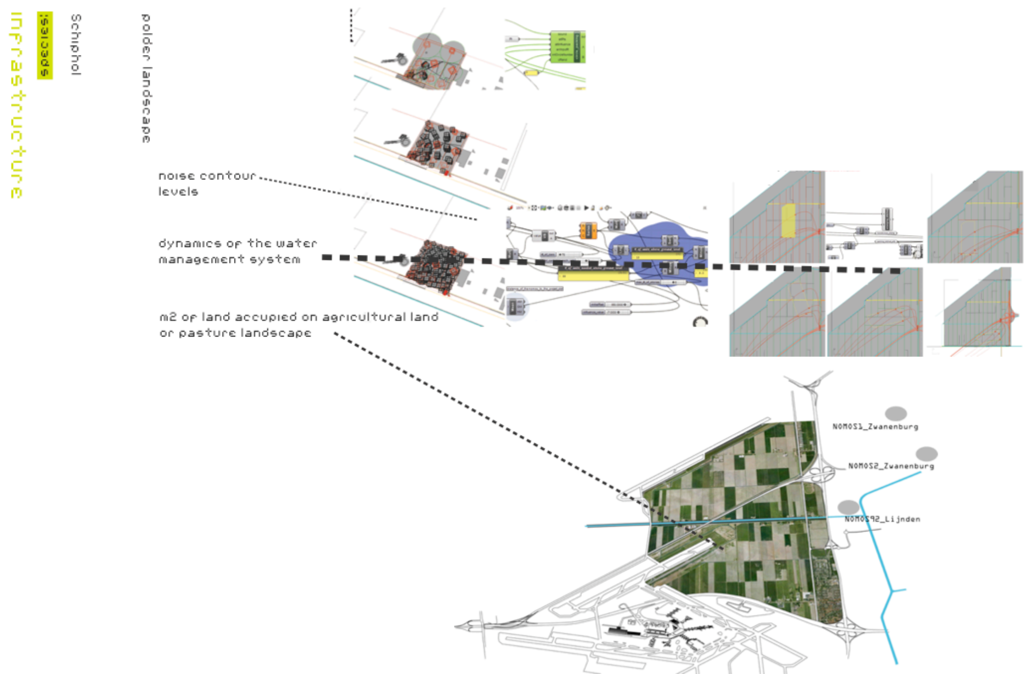


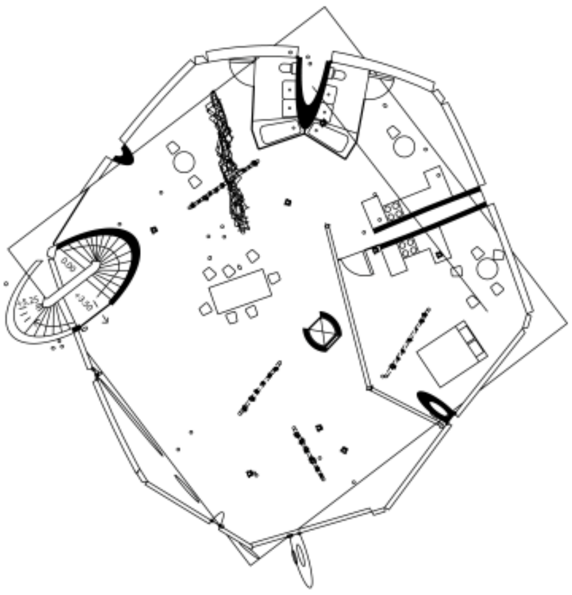
Fig 5.27. The part of the design scenario related to the infrastructure on the site and its parameterization: This

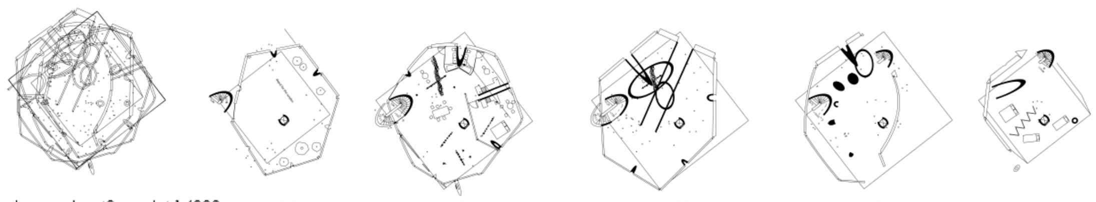
parameterization involves the translation of design reasoning into “width of ditches,” “noise contour level,” “distance between units,” “m^2 of land occupied on agricultural land or pasture,” “number of people traveling,” “number of units,” “number of stories per unit,” “building density,” ‘building height,” etc., and into the expressions defined between these.

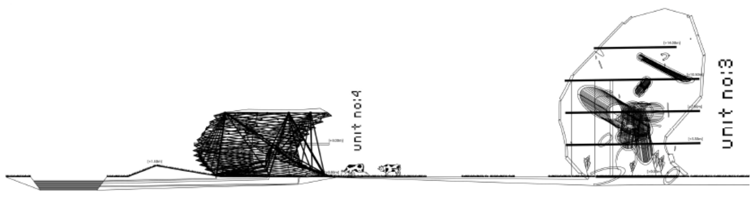
The number of people traveling is a determinant of the “number of units” on the project plot, because these units are temporary homes (Fig 5.28) for people requiring access to Schiphol Airport, and are willing to purchase “closeness” to the most important hub in the mobility network. For such people, the main priority is purchasing the least possible traveling time rather than the unit. What matters is not the unit itself, but its proximity to Schiphol Airport. This purchase is seen as a new mode of mobility, as discussed in 5.1, and this new mode of mobility is proposed as a norm of inhabitation in various locations (hubs) of the Randstad, rather than ownership in the conventional sense.

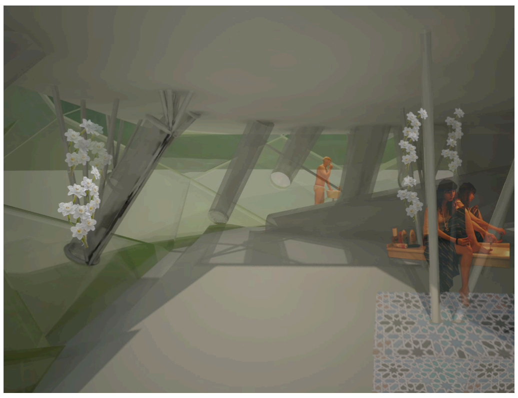
5.3.4 Relating the Structure of the Surface Covering the Units to the Growth Pattern of the Agricultural Products

Mobility in the Randstad nourishes the design of a system for people who may need to be in one place at one moment and in another place after very little traveling time. With this new system, the time lost through traveling in the Schiphol region might be avoided by defining the mode of mobility in these terms – and seeing the new mode mobility as purchasing closeness though the temporary units for rental, etc. (Fig 5.28). By cultivating agricultural products within the unit, the agricultural land use is preserved as well, which may be lost for this new mode of mobility. Therefore, in the program suggested, infrastructure, agriculture and inhabitation can co-exist. To do this, each unit is modeled in such a way that makes it possible to cultivate agricultural products within the unit through hydroponics (Fig 5.29).









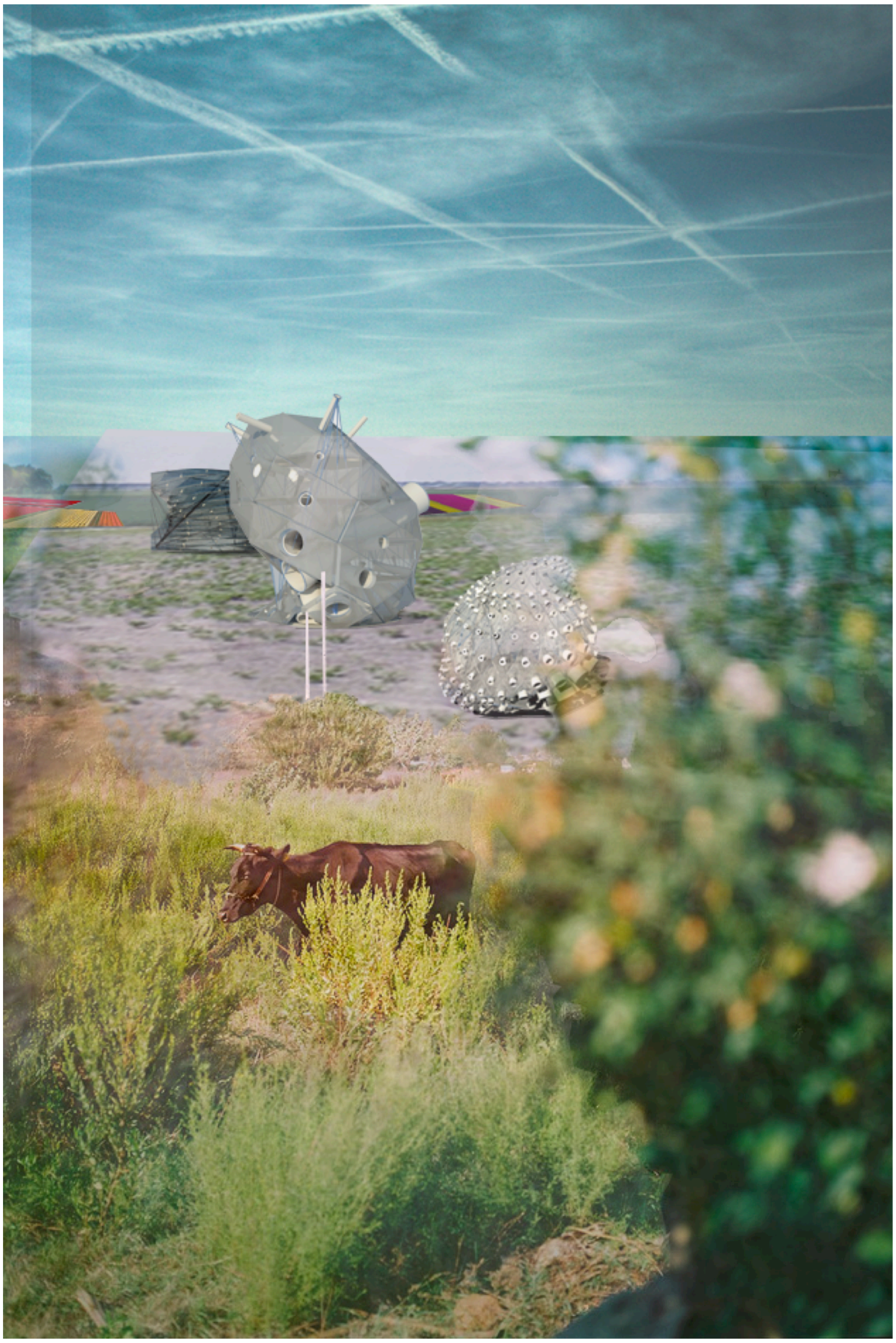
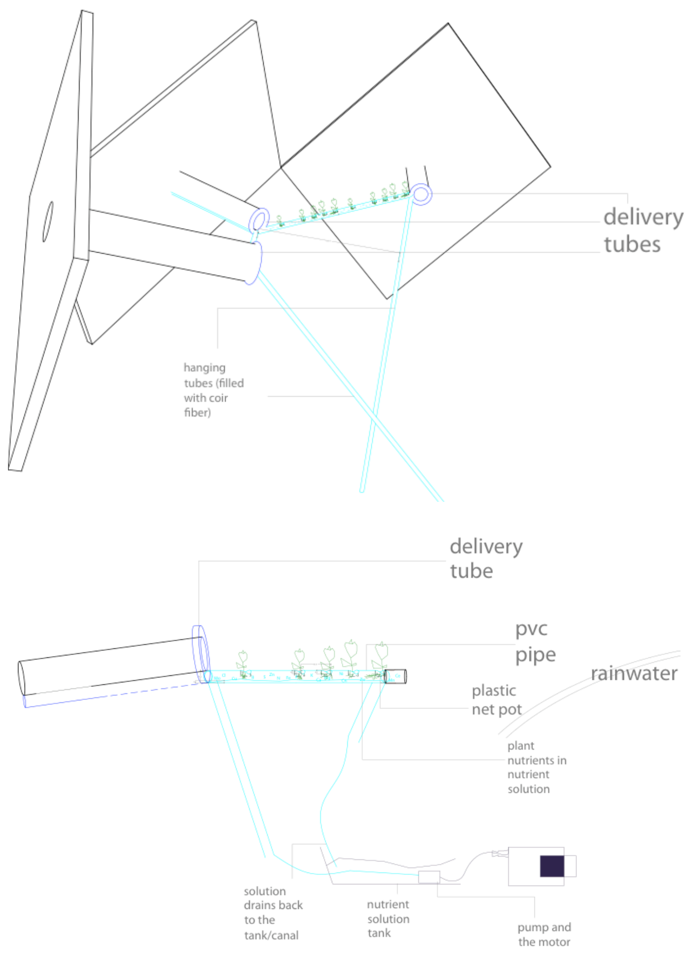
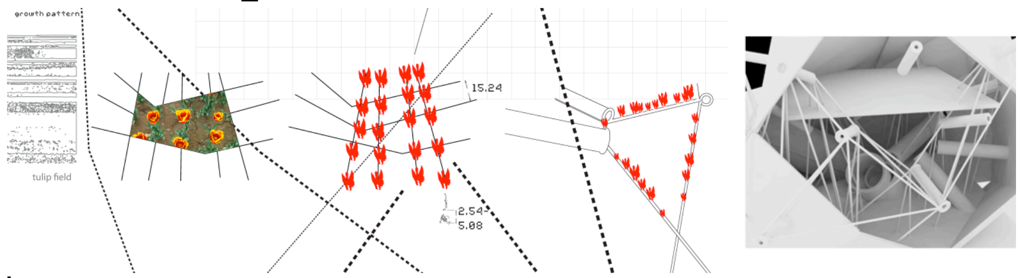


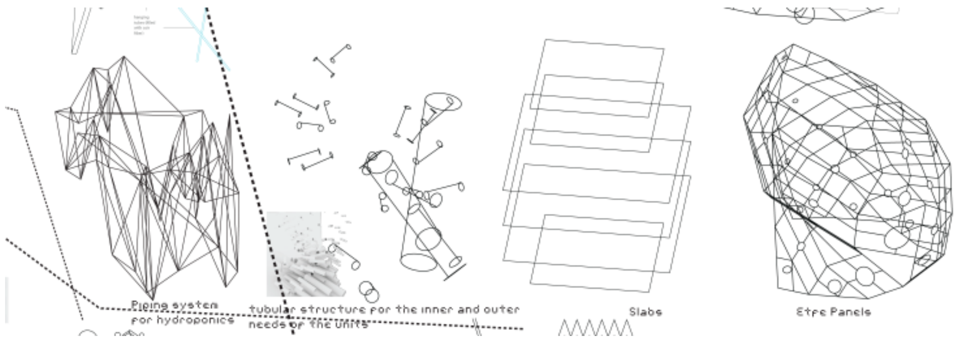
Fig 5.28. Plans, sections and internal views of the units in the new system. At the bottom, different units of the project plot can be seen.



a



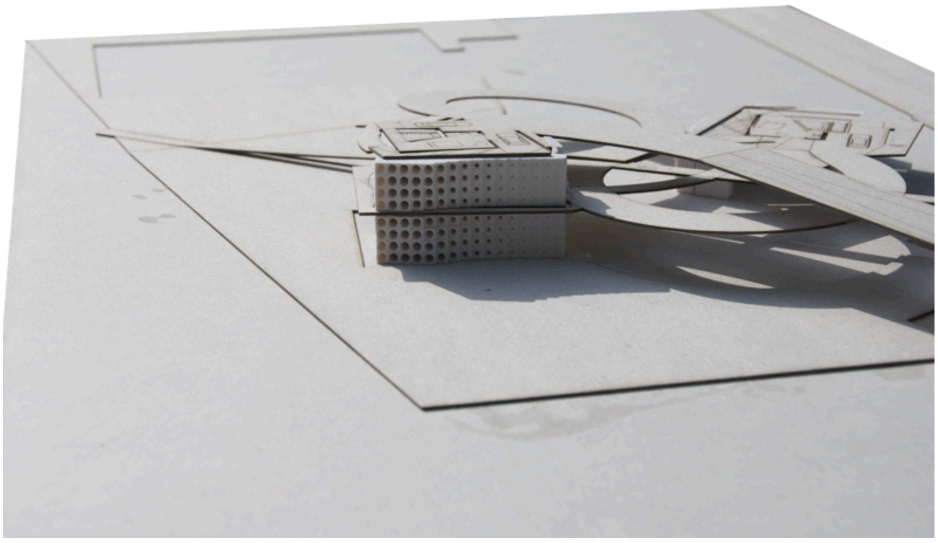
b



c

Fig 5.29. (a) Pipe work for the hydroponics can be seen. (b) The growth pattern of tulips, which has driven the design of the pipe work in which they are grown. (c) Four elements compose the units of the new system: pipe work system for hydroponics, tubular structure for the inner and outer needs of the units, slabs and Etfe panels.

The pipe work for hydroponics can be seen in Fig 5.29 (a). In Fig 5.29 (b), it can be seen how the growth pattern of the tulips has driven the design of the pipes in which they are grown. The four elements that compose the units of the new system are: pipe work system for hydroponics, tubular structure for the inner and outer needs of the units, slabs and Etfe panels In Fig 5.29 (c). The structure of the surface is composed of Etfe panels, the tubular structure and the pipe work system. There are other constraints that drive the design of the unit, rather than just the growth pattern and the irrigation needs of the plants. Different surfaces that have been fabricated with various values for the parameters that these constraints explore can be seen in Fig 5.30.



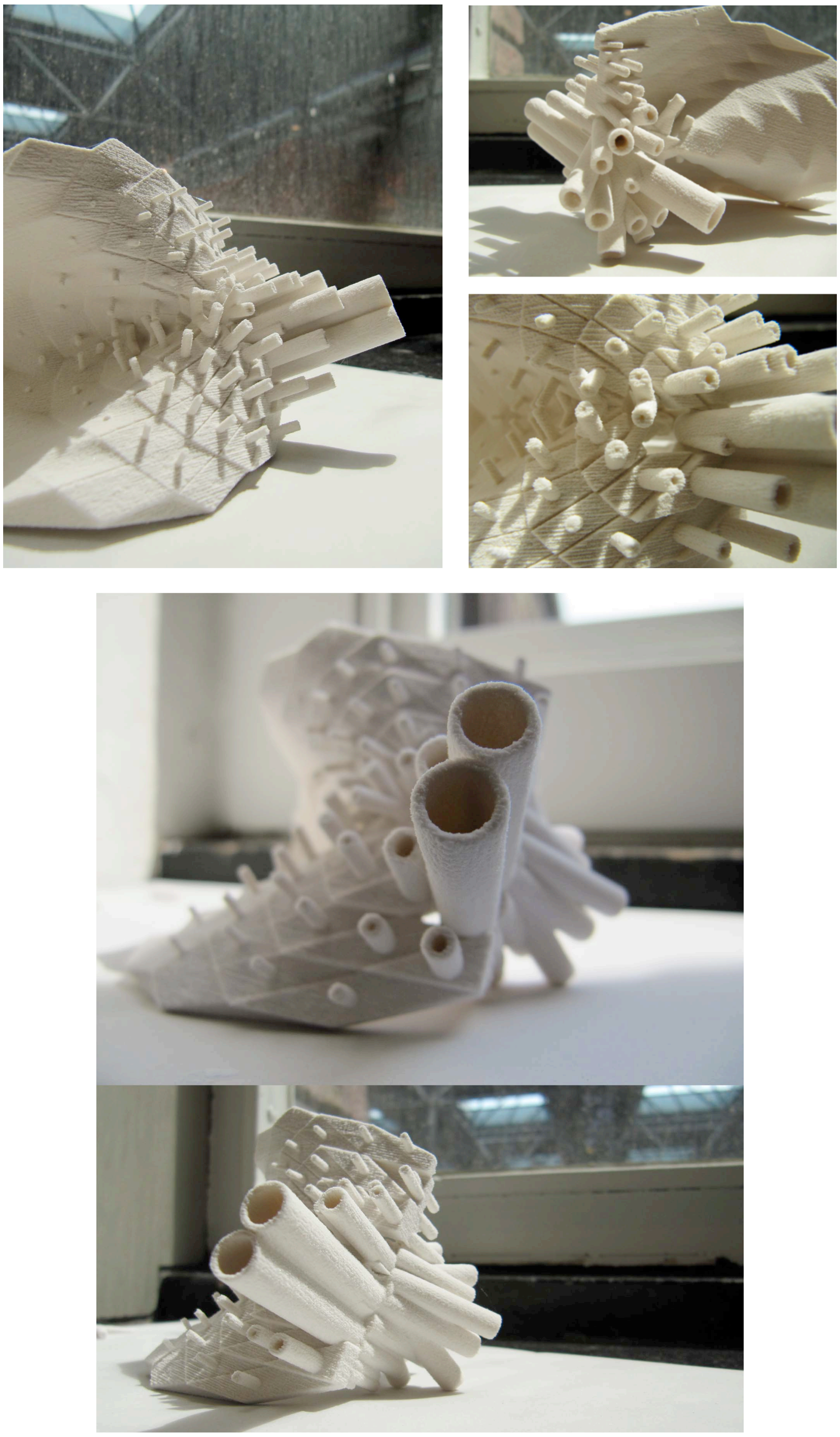


Fig 5.30. One of the fabrications of the first surfaces generated to cover the units can be seen on the top. Other surfaces are fabricated using different values for the parameters “radius of the openings in the surface covering the units” and with one specific growth pattern, etc., as can be seen below.

In modeling the surface, the relations between different groups of constraints of different scales are ensured as follows: the group of constraints at 1/100 scale that explore such parameters as "m^2 of land occupied on agricultural land or pasture" (being the "number of units" multiplied by the area of one unit, which is approximately 10m x 10m), determine the value of the dimensions of the surface covering the units. This surface (Fig 5.31 (b)) aims to preserve the agricultural land that may be lost in introducing this new mode of mobility to this region. As such, the quantity of land that is lost in m^2 determines the dimensions of the surface to cover these units and the pipe work for the cultivation of plants – the amount of pipe work necessary for the hydroponics.

The grasshopper definition encoding the function that generates the structure of the surface covering the units can be seen in Fig 5.31. Numbers from 1-6 indicate significant sequences of relations in the definition. Letters from a-i, except c and i, indicate the components of the function, which embody significant parameterizations of the design scenario in their output parameters. c and i are two number sliders and one panel by which the author inputs the values directly. The other components depend on the values of the parameters, panels or number sliders that they are linked to. The sequences of relations, named, "Etfe panels," "tubular structure" and "pipe work system", can be seen at the bottom in Fig 5.31. These names indicate the parts of the function that generates the structure of the surface.

The panel, which is created for the design of the pipe work system, can be seen in Fig 5.31 (i). The author inputs a series of numbers into this panel according to the “growth pattern of the agricultural product”. Joining the points that this series of numbers set, generates the pipe work of hydroponics in which the plants are grown. What is more, as can be seen in Fig 5.31 (1), the lofted surface is set to the input surface parameter of the component, which results in the merge component “members of the structure of the surface”. 6 different surfaces have been set through this linkage (at different times) to generate the units that can be seen in Fig 5.25 (d).

In Fig 5.31, it can be seen that, after being processed in (c) and being resulted in (d); (2) links the output surface parameter of (d) to the components that are linked to (e). (e) is the circle component that creates the openings according to constrained values of radius, etc.. The openings are linked (Fig 5.31 (3)) to the part of the function, which generates the Etfe panels. The radius (of the openings) is the relation that links the surface to the tubular structure (Fig 5.31 (4)), as these tubes should have the same radius with the openings that they are positioned on – the tubes are constrained to have the same radius with the openings. In Fig 5.31 (5) and (6) the linking of the tubular structure to the pipe work system of hydroponics, which is physically attached to these tubes (5.29 (a)), can be seen. This is the sequence to be followed for the particular reasoning of preserving agricultural land.

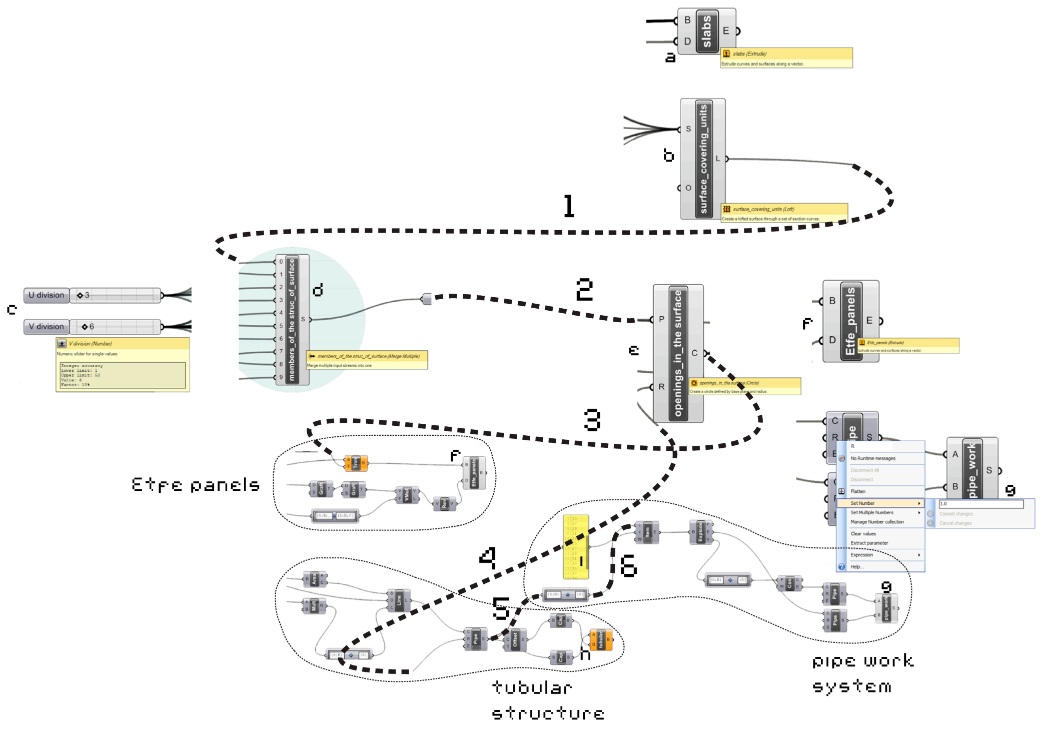


Fig 5.31. The grasshopper definition encoding the function that generates the structure of the surface covering the units can be seen. Numbers from 1-6 indicate significant sequences of relations in the definition. From a-i, except c and i, the components of the function, which embody significant parameterizations of the design scenario in their output parameters, can be seen. c and i are two number sliders and one panel by which the author inputs the values directly. The other components depend on the values of parameters, panels or number sliders that they are linked to. The sequences of relations, named "Etfe panels," "tubular structure" and "pipe work system", can be seen at the bottom. These names indicate the parts of the function that generates the structure of the surface. The scale of the design model to which the definition belongs can be seen in Fig 5.17 in red.

In the encoded function, which is partly shown in Fig 5.31, there exist more parameters that are explored through constraints. As can be followed in the lines and arrows in Fig 5.17, "growth pattern of the agricultural product" and the "number of hours of sunlight," etc. drive the design of the surface covering the units, in addition to "m^2 of land occupied on agricultural land or pasture". Moreover, it is ensured that the tectonic properties and the solid-void proportions of the same growth pattern of the agricultural product also drive the design of the same surface in terms of structural behavior (Fig 5.32 (a)), and in terms of the lighting qualities of the inner spaces of the units (Fig 5.32 (g)).

These relations exist in the model as components into which the reasoning is translated, generating the surface in favor of inhabitation. What is more, all constraints related to the design of the structure of the surface, including the min/max distances between two seeds, etc., conflict and concur for the co-existence of cultivation and inhabitation. To clarify, in Fig 5.31 (c), number sliders named “u division” and “v division” can be seen. These determine the number of divisions on the surface in u and v directions[111]. These sliders are created for the “length of the members of the structure of the surface covering the units” and the distance between two seeds, etc. It is the author who determines the values in these number sliders according to the “growth pattern of the agricultural product”, or the structural constraints, etc.

5.3.5 Relating the Radius of the Openings in the Surface Covering the Units to the Number of Hours of Sunlight

To ensure the conflict and concurrence are driven by the results of the calculations, (Fig 5.32 (c), (d), (e), (f)), firstly the parameter sunlight is defined in terms of the number of hours. The openings in the surface (Fig 5.32 (b)) through which the plants and inner space of one unit receive sunlight are defined in terms of their radii. These are related in the Grasshopper definition as follows: the “number of hours of sunlight” – which is calculated in Ecotect and transferred back to Grasshopper as a list of numerical values using the add-on Geco developed by [Uto], Fig 5.32 (c), (e), (f) – determines the “radius of the openings in the surface covering

[111] This sequence of relations is taken from http://www.co-de-it.com/wordpress/code/grasshopper-code as a mass-customized designer action and modified to serve the needs of the design task of this part of the scenario.

the units." This takes into account the needs of the agricultural products to be cultivated in the units and the required lighting qualities of the inner space (Fig 5.32 (g)).

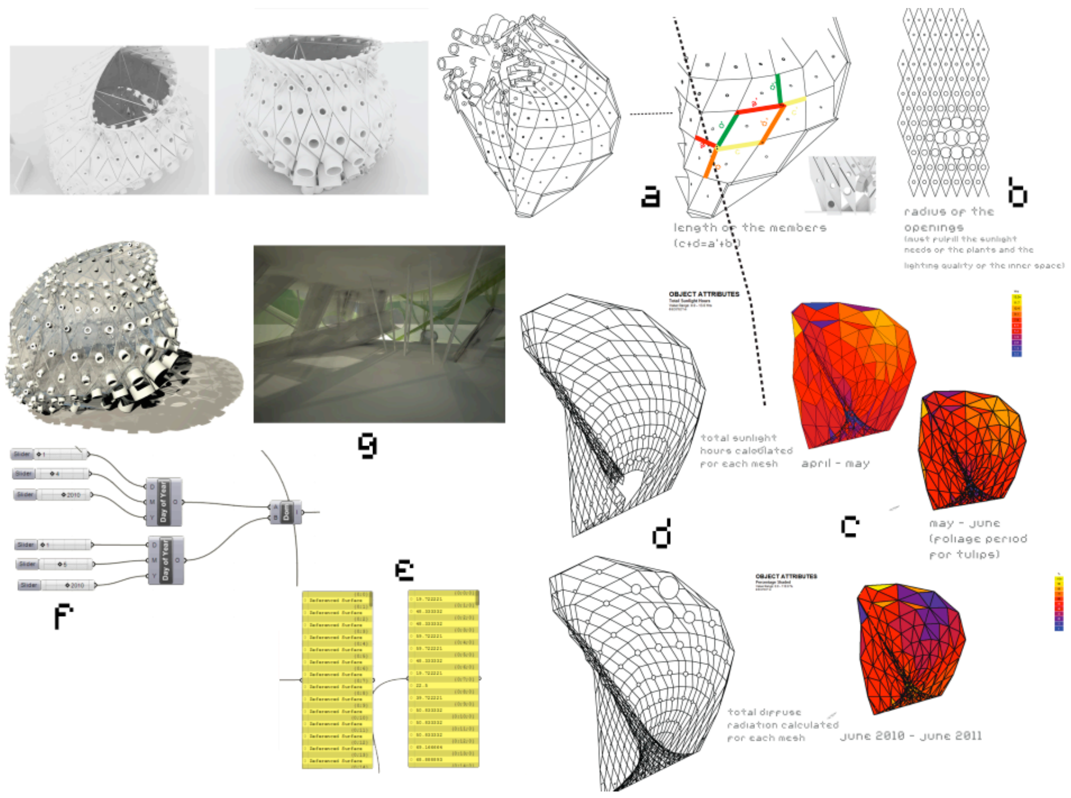

a
length of the members
(c+d=a'+b')
b
radius of the openings
(must fulfill the sunlight needs of the plants and the lighting quality of the inner space)
OBJECT ATTRIBUTES
total sunlight hours calculated for each mesh
april - may
c
may - june
(foliage period for tulips)
d
g
e
f
Day of Year
total diffuse radiation calculated for each mesh
june 2010 - june 2011

Fig 5.32. The design of the surface is driven by the constraints that explore parameters such as "m^2 of land occupied on agricultural land or pasture," which determines the dimensions of the surface. The sequence is programmed as such: the "growth pattern of the agricultural product" and the (c) "number of hours of sunlight" determine the (a) "length of the members of the structure of the surface covering the units" and the (b) "radius of the openings in the surface covering the unit," respectively. (a) The structural constraints for the members of the structure of the surface covering the units are as follows: c+d = a'+b,' where a, b, c, d and a', b', c', d' are the lengths of the members. (d) The openings in the surface can be seen below. (c) "Number of hours of sunlight" are calculated in Ecotect and (e) the results are transferred back to Grasshopper using the add-on Geco, developed by [Uto], Ursula Frick and Thomas Grabner (http://utos.blogspot.com/) (f) With this add-on, the "number of hours of sunlight" for any defined period of time, is calculated. Specifically, these calculations determine the optimum values of the radius of the openings in the surface, which have to provide sunlight in the right amount and on the right day of the year considering, the needs of the tulips, for example. The lighting needs of the occupants are also taken into consideration. (g) The lighting qualities of the inner spaces can be seen in the centre.

To summarize, the design of the surface is driven by the constraints that explore parameters such as "m^2 of land occupied on agricultural land or pasture," which determines the dimensions of the surface. The sequence is programmed as such: the "growth pattern of the agricultural product" and the "number of hours of sunlight" (Fig 5.32 (c)) determine the "length of the members of the structure of the surface covering the units" (Fig 5.32 (a)) and the "radius of the openings in the surface covering the units" (Fig 5.32 (b)), respectively. The structural constraint for the members of the structure of the surface covering the units is as follows: c+d = a'+b', where a, b, c, d and a', b', c', d' are the lengths of the members (Fig 5.32 (a)). The openings in the surface can be seen in Fig 5.32 (d). The "number of hours of sunlight" are calculated in Ecotect and the results are transferred back to Grasshopper using the add-on Geco (Fig 5.32 (c), (e)), developed by [Uto], Ursula Frick and Thomas Grabner[112]. With this add-on, the "number of hours of sunlight" in any defined period of time is calculated. Specifically, these calculations determine the optimum values of the radii of the openings in the surface, which have to provide sunlight in the right amount and on the right day of the year, considering the needs of the tulips, for example. The lighting needs of the occupants are also taken into consideration, as can be seen in Fig 5.32 (g).

[112] Thomas Grabner and Ursula Frick. *@ [UTO]: A COLLECTION OF PAST AND ONGOING WORKS*, http://utos.blogspot.com/

CHAPTER 6

THE PROPOSED REMEDY IN MASS CUSTOMIZING PARAMETRIC AND CONSTRAINT BASED REALTIONS

This study researches ways to enable the designer to build computational models through mass customization. The solution can be found in programming sequences of relations that encode designer actions in the form of functions, by which the designer relates the design constraints with each other to develop a computational design model.

As a supportive showcase, the graduation project of the author is used to experiment with various designer actions in an architectural design problem, and to suggest several examples of recursive functions in architectural design. Designer actions may exist in various forms in addition to functions. Within the limits of this study, they are encoded in Rhino Grasshopper definitions, with parametric and constraint-based relations. A full definition can be found in the appendix, where definitions of various functions are shown in different colors.

6.1 A Brief Summary and the Discussion Path

If computational support is used when modeling the design space, the degree of determinacy of the design solutions depends on the degree of the relations that are open to the operation of the designer. If the constraints and relations are predefined and hidden in the model,[113] out

[113] In http://www.nzarchitecture.com/blog/ Daniel Davis discusses that in Grasshopper, relationships are not hidden. In graph-based parametric modelling tools like Grasshopper and GC, which graphically represent the relationship between nodes, the means may be considered more architect-friendly when compared to other means, although design problems might be similar.

of reach of the designer, this blocks the dynamic features of the model.[114] When generating design solutions using computational design models, it should not be ignored that dynamic features depend no longer directly on the reasoning of the designer, but on the efficiency of the translation of this reasoning into the components of the model.[115] Therefore, the solution space is dynamic only as much as these relations are dynamic, and as much as these relations are open to the operation of the designer. This part of the discussion was covered in the classification in Chapter 4, in which the differences in the role of the designer were highlighted in various cases.

In the model developed by the author for the graduation project, this dynamic character can be achieved, to a considerable extent. To clarify, the addition or preservation of the constraints when shifting between different scales in the architectural design project is possible with the crucial support of Rhino Grasshopper. There have been recent researches into the ease that the graphical interfaces of such software bring to the designers to make them the builder of the models,[116] however this falls outside the scope of this thesis.

---

[114] In design models of generative systems built with Rhino Grasshopper or Generative Components, "the user defined transactions and parameters generate forms, and then allow the designer to make dynamic modifications in order to adjust the results," as discussed by E. Mark, M. Gross, G. Goldschmidt. "A Perspective on Computer Aided Design After Four Decades", eCAADe 26: architecture 'in computro', ed. M. Muylle, 2008, p.173

[115] *ibid.*, p.175

[116] Changing relations is relatively easier with the means that Grasshopper offers if the way of representing relations in this parametrical software is also considered. M. Maleki and R. Woodbury state that in terms of accessing and modifying object properties and dependencies to bring the model closer to the designer: "In parametric modeling, an object may be dependent on other objects or variables if other objects or variables are used to determine the value of its properties. In programming in the model, these dependencies are shown by links connecting object properties to each other. For each object, inputs come from the left side and outputs exit from the right. This is similar to Generative Components' symbolic view and Grasshopper's graphic representation of the model. In fact, they represent the same information. The main difference is that the graph is embedded in the model". M. Maleki and R. Woodbury, "Programming in the Model: Combining task and tool in computer-aided design", Proceedings of the 15th International Conference on Computer Aided Architectural Design Research in Asia, (2010), p.121

This research also aims to point out that if the model is already built, one particular design reasoning in a specific design problem may be imposed upon the designer that may not be compatible with architectural design problem in hand. There may be a number of possible parametrical relations that can be defined when addressing a specific architectural design problem with a parametric and constraint-based design model, and designers have their own ways of doing this. To clarify this problem, architectural design reasoning may not be quantified easily or in a single step, and for this reason the different means of representation have been researched and discussed in Chapter 3. Yet, in that particular discussion, a need for a basis – for translating design reasoning without eliminating any part of it – in the model was highlighted. The contribution of this discussion may be linked to the attempt to build a comprehensive model that aims to support the designer in exploring the architectural design space in all scales of the project, taking into account also any changes in the constraints of the architectural design problem.

6.2 Mass-customized Sequences of Relations Overtaking Pre-defined Relations in Parametric and Constraint-Based Design Models

The design reasoning needs to be modeled with parametric and constraint-based relations in the graduation project because the conditions are changing in Randstad of the Netherlands – in the project region. The design scenarios are undergoing continuous transformation in the particular region, resulting in a need for design models that are open to transformation. The parametric design model has been of great help to the author to generating various design solutions under changing conditions by allowing the comprehension of the architectural design space as a whole. It is impossible to grasp the complexity of all the transformations of the constraints that result from the changing conditions, and their impact on each other, without modeling them using an intermediary facet that includes parametric and constraint-based relations.

Mass-customized sequences of relations can be used in different contexts or in different scales of an architectural design problem, just like using the conventional line tool found in some CAD software. These sequences do not belong to one specific type of design problem, as they encode reusable designer actions. However, exactly what can be referred to as a designer action is of utmost importance, and the author's graduation project is used as a showcase for this very reason. This research may be expanded further by encoding them one by one in a comprehensive study and in an experimental project to come up with a solid proposal for the library of some software – such as the generative modeling plug-in of Rhino, Grasshopper, which enables designers to develop computational design models.

The designer actions that Grasshopper definitions encode are seen as a potential library/menu of the software by which the author can model the design space of the architectural design problem. It has been discussed throughout the text that designers use thinking tools; and they communicate via encodings to externalize their reasoning. Reasoning is externalized through the existing packages of knowledge, and this externalization and the encodings already exist in architectural design. This is the evidence that mass customization of sequences of relations by encoding designer actions in them is not alien to architectural design. Using mass-customized sequences of relations to address architectural design problems is seen as a way of making the designer the builder of the model; as otherwise, modeling may become a deterministic act that is closed to further modifications or operations by the designer.

It is not the intention to suggest that designer actions are not researched in general while developing models to be put into the computer as CAAD tools. Before models are implemented as applications or software, most of them are also built on theories that look into what kind of designer actions exist for the defined architectural design problem. Designers model the relations for design parameters and the relations between design constraints accordingly using them. However, there exists one problem, in that the model is already built, and the designer is not always able to operate on it, only being able to fill in the values for the already defined relations. In this case, the design elements have also been previously defined. This is mostly seen in the second group of models, which are discussed in Chapter 4. In these models, designers cannot reach the translation of reasoning, which they need to be able to do to enable the transformation of constraints. In this case, the models may easily become deterministic, considering the limited role of the designers when using them.

Designers should be able to access the components that make up the model, and change them. For example, they should be able to change the design elements that are fed into the parameters. For this reason, the sequences of relations should be available to designers to play with and modify,[117] maybe as a menu or library, and designers should be allowed to develop their own design models to address the design problems that they define.

[117] Or they should be let to program them.

6.3 Conclusions and Contributions

There are various ways of externalizing reasoning in architectural design. Throughout this thesis, this externalization is seen as the translation of reasoning into the computational components of the design model that are parametric and constraint-based relations. It is proposed in this thesis that, this may be achieved by feeding the design elements, etc., into the recursive functions of architectural design that the mass-customized relations encode. These functions respond to general usage and can be modified to translate various design reasoning.

Externalization is done synthetically and intuitively through the mass-customized sequences of relations. They open up the design constraints to which they relate to changes according to the changing values of the parameters that these constraints explore. This change may as well be feeding a different design element into one parameter of the model. As the constraints are also open to transformation, a different model may be achieved using the same sequence. Simply, the designer is able to operate on the sequences.

In the graduation project, circle packing has been used in one of the functions that embody a reusable designer action that exists recursively in architectural design problems. To be specific, for 1/100 scale of the project there exists the problem of calculating the maximum "number of units" that can be constructed on the project plot. The author needs to make this calculation for the defined design problem, as can be seen in Chapter 5. At this scale, design reasoning is translated into "number of units," "distance between units," "number of passing cars (determined by the amount of people traveling)," "noise contour level", etc., and into the expressions that are defined between these. The function, in which circle packing is used, embodies all of these and translates the design reasoning to execute the defined task, which is to find the maximum "number of units" that can be constructed on the project plot with the least possible noise levels. In addition to this usage of circle packing in the graduation project, it can also be used to afford "stability through adjacency" as it has been discussed at the end of Chapter 4 referencing to the study of Aranda/Lasch. In the graduation project, rather than stability, the main concern lies in achieving the minimum distance between units to avoid high noise levels, but the function that circle packing is used in, can respond to several usages in several different scales. Therefore, it is proposed that this function allows the parametric and constraint-based modeling of various design reasoning. In the Grasshopper definition of the project, it can be modified according to the particular design task because it has been mass customized beforehand with abstract, modifiable and reusable relations as a Grasshopper component.

6.4 Assessment of Limitations as a Further Study

One limitation of this study is that not all of the computational methods listed in Chapter 3 are applied for the graduation project. The potential of parametric and constraint-based design models have been tested, but a further study might look at possibilities of coming up with a methodological study – of the know-how – to categorize the way different practices use computational support in architectural design, and may reference such practices as Gehry Partners, Foster+Partners, Zaha Hadid Architects, FOA, etc. For instance, the mathematical model that was developed for the structure of the Great Court at the British Museum by Foster + Partners is significantly different to any of the models' of Gehry Partners, because the practices have significantly different reasoning behind their designs and they model different solution spaces. Practices introduce new ways of reasoning in their design projects through their models. It is assumed that with every architectural design project that uses computational support, a new way of defining “x” kind of relations for “x” kind of parameters in the problem space for “x” kind of design problems, and a new way of relating constraint “x” with constraint “y” in the solution space for generating “x” kind of design solutions are added to architectural design knowledge.

The discussion on predefined relations in models criticizes approaches that do not allow the designer to operate on the model; but at the same time, it does not totally aim to discount this suggestion of possible frequent design reasoning and elements, etc. that may exist for some specific design problems. For instance in parametric and constraint-based design models, some frequencies or similarities might occur when relating design constraints to each other. This is not to suggest a deterministic approach, yet such an approach may lead to the identification of these frequencies and similarities as design patterns for architectural design problems. After all, the mass-customized sequences that are studied in this thesis can be fed similar design elements and similar reasoning in general at varying frequencies. This may cause them to be interpreted as design patterns.

The reasoning introduced by different practices[118] can be considered as reusable patterns, and to understand the reason behind this, the explanation of Mark Garcia in the Design Patterns issue of AD magazine can be referenced:

[118] What these practices have in common is that they all model their design space using computational support for the exploration of the design space.

> *“The etymology of ‘pattern’ is from the Latin pater, or patronus, meaning father, patron, god or master, from which is derived the notion of pattern as a model, example, matrix, stencil or mould. The contemporary concept of pattern is as a sequence, distribution, structure or progression, a series or frequency of a repeated/repeating unit, system or process of identical or similar elements”.*[119]

A study of patterns in the models of different practices is suggested as a further research; yet first it is necessary to make a comprehensive analysis of how designer actions make the emergence of these patterns possible when dealing with architectural design problems.

[119] AD issue on: “Patterns of Architecture” (Guest Ed.) Mark Garcia. 79(6), Wiley & Sons, November-December 2009, p.8

REFERENCES

Alexander, Christopher. *Notes on the Synthesis of Form*, Cambridge, MA: Harvard University Press, 1964.

Aranda, Benjamin., Lasch, Chris. *Pamphlet Architecture 27: Tooling*, New York: Princeton Architectural Press, 2006.

Baykan, Can. “Spatial Relations and Architectural Plans: Layout problems and a language for their design requirements”, *E-Activities in Design and Design Education*, ed. Bige Tunçer, Şaban Suat Özsarıyıldız, Sevil Sarıyıldız, Paris: Europia Productions, 2003, pp.137-146.

Carpo, Mario., Furlan, Francesco. (eds.) *Leon Battista Alberti’s Delineation of the city of Rome* (Descripto urbis Romae), critical edition by Jean-Yves Boriaud & Francesco Furlan, English translation by Peter Hicks.

Chase, Scott. “Generative design tools for novice designers: Issues for selection”, *Automation in Construction*, vol.6, issue 4, December 2005, pp.689-698.

Çiftçioğlu, Özer. “Conceptual Design Enhancement by Intelligent Technologies”, *The Architecture Annual 2001-2002: Delft University of Technology*, ed. Arie Graafland, The Netherlands: 010 Publishers, 2003, pp. 85-89.

Do, Ellen. Yi-Luen., Gross, Mark. “Thinking with diagrams in architectural design”, *Artificial Intelligence Review 15*, 2001, pp.135-149.

Eastman, Charles. “On the analysis of intuitive design processes”, *Emerging Methods in Environmental Design and Planning,* ed. G.T.Moore, Cambridge, MA: The MIT Press, 1970, pp.21-37.

Eilouti, Buthayna H., Al-Jokhadar, Amer M. “A Generative System for Mamluk Madrasa Form-Making”, *Nexus Network Journal*, vol.9, 2007, pp. 7–29. (automated generation will be discussed in relation to the implementation of the shape rules to FloP)

Eisenman, Peter. *Diagram Diaries*, New York: Universe Publishing, 1999.

Fernando, Ruwan., Drogemuller, Robin., Salim, Flora Dilys., Burry, Jane. “Patterns, Heuristics for Architectural Design Support: Making use of evolutionary modeling in design”, *Proceedings of the 15th International Conference on Computer Aided Architectural Design Research in Asia,* 2010, pp.283-292.

Frazer, John. *Themes VII: An Evolutionary Architecture*, AA Publications, 1995.

Garber, Richard. (guest ed.) *AD issue on: “Closing the Gap”* , 79(2), Wiley & Sons, March-April 2009.

Garcia, Mark. (guest ed.) *AD issue on: "Patterns of Architecture",* 79(6), Wiley & Sons, November-December 2009.

Goldschmidt, Gabriela. "Capturing indeterminism: representation in the design problem space", *Design Studies*, vol.18, n.4, October 1997, pp.441-455.

Gross, Mark., Ervin, Stephen., Anderson, James., Fleisher, Aaron., "Constraints: Knowledge representation in design", *Design Studies,* 9(3), 1988, pp. 133-143.

Kilian, Axel. Design Exploration through Bidirectional Modeling of Constraints, MIT, February 2006.

Kilian, Axel. "Design Innovation through Constraint Modeling", *Digital Design: The Quest for New Paradigms, 23rd eCAADe Conference Proceedings,* 2005, pp.671-678.

Knight, Terry., Stiny, George. "Classical and non-classical computation", *Architectural Research Quarterly,* 5(4), 2002, pp.355-372.

Koutamanis, Alexander. "A Biased History of CAAD: the bibliographic version", in, J.P Duarte, G Ducla-Soares & A.Z Sampaio (Eds.), eCAADe 23: Digital Design: the Quest for New Paradigms

Köknar, Sait Ali. "Architectural Design Tools: Toward a non-linear design process", *Proceedings Designtrain Congress Trailer I,* 2007, pp.174-182.

Köknar, Sait Ali., Erdem, Arzu. "Can creativity be institutionalized?", *A-Z ITU Journal of the Faculty of Architecture*, 4(2), 2007, pp.27-37.

Liggett, Robin. "Automated facilities layout: past, present and future", *Automation in Construction*, vol.9, 2000, pp.197–215.

Maleki, Maryam., Woodbury, Robert. "Programming in the Model: Combining task and tool in computer-aided design", *Proceedings of the 15th International Conference on Computer Aided Architectural Design Research in Asia*, 2010, pp.117-125.

Mark, Earl., Gross, Mark., Goldschmidt, Gabriela. "A Perspective on Computer Aided Design After Four Decades", *eCAADe 26: architecture 'in computro',* ed. M. Muylle, 2008, pp.169-176.

Miyauchi, Tomohisa., Tsukui, Noriko., Nishimaki, Atsuko., Yokoyama. Kei. (eds.) *a+u special issue on: "Architectural Transformations via BIM"*, August 2009.

Nickerson, Jeffrey., Corter, James., Tversky, Barbara., Zahner, Doris., Rho, Yun. Jin. "Diagrams as tools in the design of information systems", *Design Computing and Cognition'08: Proceedings of the Third International Conference on Design Computing and Cognition,* 2008, pp.103-122

Özkar, Mine. "Learning Computing by Design; Learning Design by Computing", *Proceedings Designtrain Congress Trailer I,* 2007, pp.101-111.

Özkar, Mine. “Sayısala Sayısal Olmayan Bir Arayüz: Temel Tasarım”, *Mimarlık, Ocak-Şubat 2005"*, 2005, pp.31-32.

Reh, Wouter., Steenbergen, Clemens., Aten, Diederik. *Sea of Land : The polder as an Experimental Atlas of Dutch Landscape Architecture,* Wormerveer-Zaanstad, The Netherlands : Stichting Uitgeverij Noord-Holland, 2007.

Stiny, George. *Shape: Talking about seeing and doing*, Cambridge, MA: The MIT Press, 2006.

Terzidis, Kostas. *Algorithmic Architecture*, 2nd Ed. Elsevier, 2007.

Turrin, Michela., Kilian, Axel., Stouffs, Rudi., Sariyildiz, Sevil. “Digital Design Exploration of Structural Morphologies Integrating Adaptable Modules: A design process based on parametric modeling”, *Proceedings of the 13th International Conference on Computer Aided Architectural Design Futures*, 2009, pp. 800-814.

Van Berkel, Ben., Bos, Caroline. *UN Studio: Design Models - Architecture, Urbanism, Infrastructure*, Thames and Hudson, 2006

Van Berkel, Ben. “Diagrams, Interactive instruments in Operation”, *Any Magazine 23,* 1998, pp. 19-23.

Ven, G. P. van de. (ed.) *Man-made lowlands: History of water management and land reclamation in the Netherlands,* Utrecht, The Netherlands: Uitgeverij Matrijs, 2004.

Vidler, Anthony. "Toward a Theory of the Architectural Program", *Octobe*r, 2003, pp.59-74.

Woodbury, Robert., Burrow, Andrew., “Whither design space?”*, Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 20(2), 2006, pp.63-82.

Davies, Daniel. *Digital Morphogenesis: Evolving architecture through computation*, http://www.nzarchitecture.com/blog/, last accessed on: 23.08.2010

Del Giudice, Davide., Erioli Alessio., Graziano Andrea. *Co-de-iT: computational design italy* http://www.co-de-it.com/wordpress/, last accessed on: 23.08.2010

Grabner, Thomas., Frick, Ursula. *@ [UTO]: A COLLECTION OF PAST AND ONGOING WORKS*, http://utos.blogspot.com/, last accessed on: 23.08.2010

Kilian, Axel. *Axel Kilian: PhD Computation and Design, Dipl.-Ing.,* http://www.designexplorer.net/, last accessed on: 23.08.2010

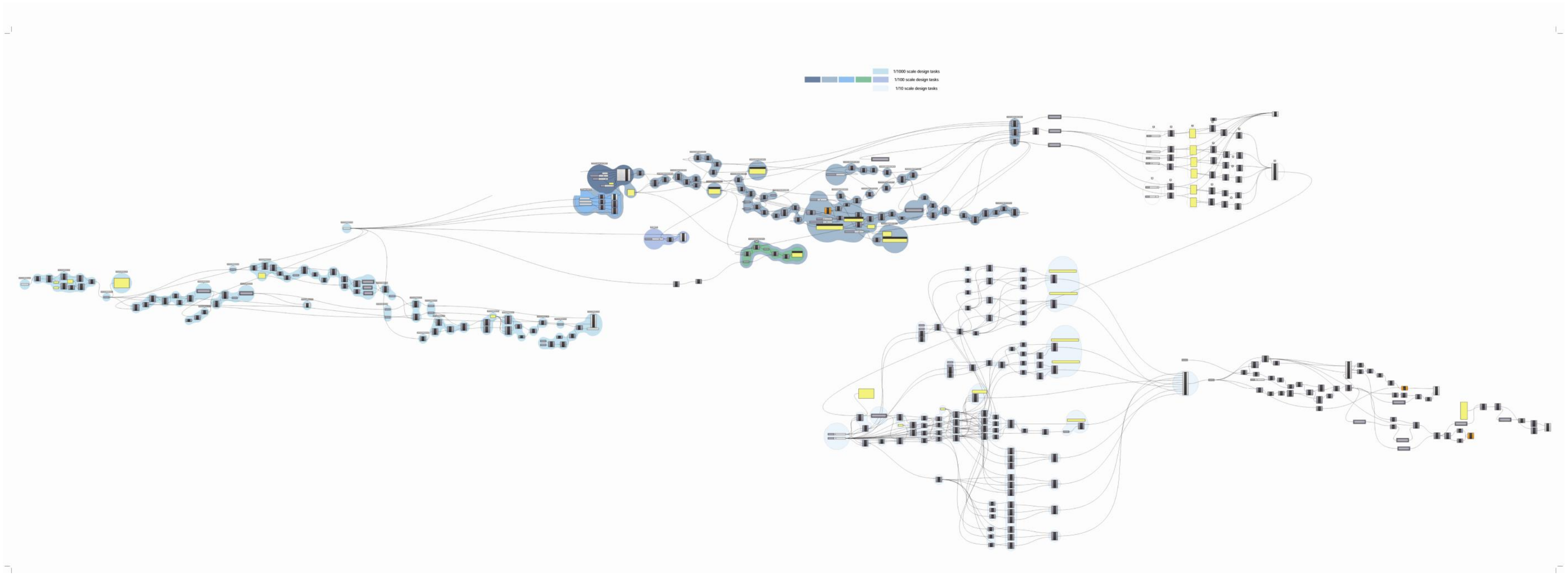
Krishnamurti, Ramesh., Wang, Tsung-Hsien. *Design Patterns for Parametric Modeling in Grasshopper: New tools and ideas for design,* http://www.andrew.cmu.edu/org/tsunghsw-design/index.html, last accessed on: 23.08.2010

Scheurer, Fabian., Walz, Arnold. *designtoproduction*, http://www.designtoproduction.com/, last accessed on: 23.08.2010

Woodbury, Robert. *Design Patterns for Parametric Modeling: New tools and ideas for design,* http://www.designpatterns.ca/, last accessed on: 23.08.2010

APPENDIX A

GRASSHOPPER DEFINITION OF THE GRADUATION PROJECT: (Various functions belonging to different design tasks at different scales are shown in different colors.)


1/1000 scale design tasks
1/100 scale design tasks
1/10 scale design tasks