MANAGING COMPLEXITY IN MASS CUSTOMIZATION
THROUGH DESIGN SPACE SUBDIVISION: 
A CASE STUDY IN CERAMIC TABLEWARE DESIGN

A Dissertation in
Architecture
by
Eduardo Raposo da Silva de Castro e Costa

© 2018 Eduardo Raposo da Silva de Castro e Costa

Submitted in Partial Fulfillment
of the Requirements
for the Degree of
Doctor of Philosophy

May 2018
The dissertation of Eduardo Raposo da Silva de Castro e Costa was reviewed and approved* by the following:

**José Duarte**  
Professor of Architecture, Stuckeman Chair for Design Innovation  
*Dissertation Advisor, Chair of Committee*

**Loukas Kalisperis**  
Professor of Architecture

**Timothy W. Simpson**  
Paul Morrow Professor of Engineering Design and Manufacturing

**Conrad Tucker**  
Associate Professor of Engineering Design

**Tom Lauerman**  
Assistant Professor of Art  
*Special Member*

**Joaquim Jorge**  
Professor of Computer Science  
*Special Member*

**Ute Poerschke**  
Professor of Architecture  
*Director of Graduate Studies in the Department of Architecture*

*Signatures are on file in the Graduate School*
ABSTRACT

Although design is traditionally the métier of designers, nowadays participation in the design process is increasingly shared with end-users. Such transfer of design responsibility happens in various creative fields such as architecture and product design, and in a number of contexts, one of which being mass customization.

In the context of mass customization, designs are generated automatically by computer-implemented configurators according to the needs and preferences of the customers, in the quality of non-expert end-users. In modern configurators, those end-users directly explore the customization design space, which corresponds to the set of all possible design solutions. For those users, the quality of a mass customization experience depends on a balanced design space that provides design variety while avoiding the burden of choice. Such balance is difficult to ensure with traditional configurators. Therefore, we suggest an innovative Design Participation Model towards a more flexible design space management, by modulating the interactions among participants of the design process, namely system designers, designers and end-users. With this model, complex design spaces can be simplified by designers and more easily manipulated by end-users.

In order to experiment with the proposed Design Participation Model, we develop a prototype of a mass customization system focused on ceramic tableware. Reasons for choosing ceramic tableware as a case study include its reduced production cost and the sector’s need for competitiveness against companies manufacturing low-cost products. Implementation of the prototypical system includes the development of generic shape grammars and parametric models, as well as their integration with user interfaces and digital fabrication technology. Gathered insights can be useful for implementing the Design Participation Model in the introduction of mass customization in other areas.

Keywords: mass customization, ceramic tableware, generative design systems, shape grammars, parametric modeling, digital fabrication
TABLE OF CONTENTS

List of Figures ......................................................................................................................... vi
List of Tables ........................................................................................................................... xiv
Preface .................................................................................................................................... xv
Acknowledgements .................................................................................................................. xvi

Chapter 1 Introduction ............................................................................................................. 1
  1.1 Mass customization .................................................................................................... 1
  1.2 Design democratization .............................................................................................. 3
  1.3 Generative design systems ......................................................................................... 5
  1.4 Problem ...................................................................................................................... 7
  1.5 Hypothesis .................................................................................................................. 8
  1.6 Methodology .............................................................................................................. 11
  1.7 Contributions .............................................................................................................. 19
  1.8 Dissertation organization ........................................................................................... 21

Chapter 2 State-of-the-Art ....................................................................................................... 23
  2.1 Mass Customization ................................................................................................... 23
  2.2 Digital Design and Generative Design Systems ........................................................ 33
  2.3 Ceramic production and digital fabrication .............................................................. 47
  2.4 Chapter conclusion ..................................................................................................... 62

Chapter 3 Encoding tableware design rules in shape grammars .............................................. 63
  3.1 Design Participation Model for the design system ..................................................... 64
  3.2 Methodology .............................................................................................................. 67
  3.3 Analysis of existing collections ............................................................................... 69
  3.4 The precedent of Digital Alberti ................................................................................ 78
  3.5 Analysis of tableware design processes .................................................................... 80
  3.6 Synthesis into shape grammars ................................................................................ 83
  3.7 Single collection shape grammar .............................................................................. 84
  3.8 Generic shape grammar from multiple collections .................................................. 95
  3.9 “Genericness” of a shape grammar ......................................................................... 105
  3.10 Chapter conclusion .................................................................................................. 107
Chapter 4 Streamlining parametric modeling for tableware designers ...................................................... 110
  4.1 Implementation architecture and tools ......................................................................................... 111
  4.2 The Grasshopper prototype ....................................................................................................... 123
  4.3 The final prototype ...................................................................................................................... 140
  4.4 Modeler Usability Testing ........................................................................................................... 155
  4.5 Chapter conclusion ..................................................................................................................... 162

Chapter 5 Enabling parametric design space exploration for non-expert users ..................................... 167
  5.1 Existing parametric configurators ............................................................................................. 169
  5.2 Vector design space exploration ............................................................................................... 173
  5.3 Navigation through interpolation .............................................................................................. 177
  5.4 Navigation through qualities ..................................................................................................... 180
  5.5 Chapter conclusion ..................................................................................................................... 188

Chapter 6 Exploring additive manufacturing for ceramic tableware .................................................... 190
  6.1 Survey on additive manufacturing applied to ceramics ............................................................. 192
  6.2 Experiments with ceramic extrusion ......................................................................................... 202
  6.3 Suggested improvements to the ceramic printer ......................................................................... 216
  6.4 Chapter conclusion ..................................................................................................................... 221

Chapter 7 Conclusion ............................................................................................................................ 223
  7.1 Discussion of results .................................................................................................................... 223
  7.2 Future work ................................................................................................................................. 230
  7.3 Final note ...................................................................................................................................... 234

Reference list ........................................................................................................................................... 235

Appendix A: Tableware Shape Grammar ............................................................................................. 262

Appendix B: Insights gathered within Contextual Inquiry ....................................................................... 278

Appendix C: Publications ...................................................................................................................... 285
LIST OF FIGURES

Figure 1 – Readjusting roles from a classic design approach (left) to a co-design approach (right) (U = end-user; D = designer; R = researcher) .................................................................3

Figure 2 – Design Participation Model for a mass customization design system .........................9

Figure 3 – Combinatorial customization of sport shoes (https://www.adidas.com/) ......................30

Figure 4 – Cell Cycle web-based interface (Nervous Systems, inc., 2012) .................................31

Figure 5 – Sake Set Creator interface (Huang & Hudson, 2013) ...............................................32

Figure 6 – Digital design models (adapted from (Oxman, 2006)) ............................................34

Figure 7 – De Casteljau construction (Source: Haui, licensed under CC BY 2.0) .......................37

Figure 8 – Example of a shape grammar (adapted from (Stiny, 1980, p. 348)) .........................41

Figure 9 – Coding processes for grammars and types (adapted from (Duarte, 2011)) ...............42

Figure 10 – Elemental shapes of the Albertian column system (left), combined into classical molds (Castro e Costa, 2012) .................................................................44

Figure 11 – Partial derivation tree of the Temple Grammar (Figueiredo et al., 2013) ...............45

Figure 12 – Grasshopper file containing all Digital Alberti parametric models .......................46

Figure 13 – Typical ceramic manufacturing process (adapted from Hamilton, 1974) ............47

Figure 14 – Handicraft ceramic modeling techniques, from left to right: pinching (Hamilton, 1974), coiling, throwing (Quinn, 2007), and slab building (Clay Pottery Slab Building, 2008) ........48

Figure 15 – Mold-based traditional techniques, from left to right: pressing by hand, by fly press (Hamilton, 1974), and slipcasting (Quinn, 2007) ....................................................49

Figure 16 – Mold-based traditional techniques, from left to right: sledging, plaster turning, jiggering and jollying (Quinn, 2007) .................................................................49

Figure 17 – Industrial techniques, from left to right: roller making, industrial slipcasting (Lippert GmbH, 2014), ram pressing (RAM Products, Inc., 2014), and isostatic pressing (SACMI IMOLA S.C., 2014) .................................................................50

Figure 18 – Terminology suggested by Pupo et al. (2009) (left) and the one adopted in this dissertation (right) ........................................................................................................53
Figure 19 – Additive manufacturing: complexity and individuality for free (Aghassi & Witzel, 2014)....................................................................................................................55
Figure 20 – Design Participation Model for the design system.................................................64
Figure 21 – Incrementally constrained design space .................................................................65
Figure 22 – Coding processes for grammars and types adapted from (Duarte, 2011)..............67
Figure 23 – Process for inferring the generic Tableware Shape Grammar..................................68
Figure 24 – General methodology for developing the design system.........................................69
Figure 25 – Romantica collection, by Matceramica .................................................................70
Figure 26 – Bisel collection, by Matceramica ............................................................................70
Figure 27 – Catarina de Bragança collection, by Matceramica ..................................................71
Figure 28 – Cotélé collection, by Matceramica ..........................................................................71
Figure 29 – Cuisine Provençale collection, by Matceramica.....................................................72
Figure 30 – Flor collection, by Matceramica ..............................................................................72
Figure 31 – Parts of a teapot (Utah teapot adapted from Nicholas Sourd) .........................74
Figure 32 – Example of a tableware collection........................................................................75
Figure 33 – Three functional parts in dinner set elements, exemplified for a soup plate .......76
Figure 34 – Parametric variations within the Romantica collection, according to Table 2 .....77
Figure 35 – The archetype and some related types.................................................................78
Figure 36 – Column system deconstruction (Castro e Costa, 2012)........................................79
Figure 37 – Section profiles of a column element (left); element combination into classical molds (right) (Castro e Costa, 2012).................................................................79
Figure 38 – A Doric capital, a classic vase (www.archiexpo.es), and two ceramic tableware elements..................................................................................................................80
Figure 39 – Romantica tableware collection by Matceramica..................................................85
Figure 40 – Overview of the soup plate derivation....................................................................85
Figure 41 – Shape grammar rules for envelope creation and functional partitioning.............86
Figure 42 – First steps in the derivation of a soup plate: envelope creation and functional partitioning .......................................................................................................................87

Figure 43 – Simplified derivation of the base shape of a soup plate ........................................88

Figure 44 – Subdivision rule and example of its application: subdivision allows for more complex shapes. ...............................................................................................................88

Figure 45 – Distortion rule and example of its recursive application .....................................88

Figure 46 – Replacing envelopes with the corresponding Bézier curves: substitution rules and examples of their application ....................................................................................89

Figure 47 – Different rule application order generates different results ...................................89

Figure 48 – Derivation of the base shape of a soup plate ........................................................90

Figure 49 – Derivation of the decoration of a soup plate .........................................................90

Figure 50 – Decoration subdivision rules and their application in a soup plate derivation ....91

Figure 51 – Motif replacement rules and their application in a soup plate derivation ..........91

Figure 52 – Example of uv mapping: rules are applied onto surfaces ....................................92

Figure 53 – Parametric model developed in Grasshopper: using rules as groups of components ......................................................................................................................93

Figure 54 – 3D printed prototypes ...........................................................................................94

Figure 55 – Partial derivation of a plate from collection B .........................................................96

Figure 56 – Examples of rule deconstruction .........................................................................99

Figure 57 – Examples of rule generalization through grouping and parameterization ............102

Figure 58 – Example of application of GRule 11 to divide the border of a plate into parts for decoration purposes .................................................................103

Figure 59 – Elements of the original collections generated by the generic shape grammar ....105

Figure 60 – Elements of three new collections generated by the generic shape grammar ......105

Figure 61 – Shape grammar “genericness” spectrum .................................................................107

Figure 62 – Comparing processes for formulating different generic grammars .................109
Figure 63 – Simplified timeline of the development and implementation of the design system .........................................................................................................................................................................................111

Figure 64 – Implementation architecture for the design system ..............................................................................................................................................................................................................112

Figure 65 – The Grasshopper prototype .............................................................................................................................................................................................................................................113

Figure 66 – Implementation architecture for the Grasshopper prototype ........................................................................................................................................................................................................114

Figure 67 – Implementation architecture for the Racket prototype: during development (left) and after compilation (right) .............................................................................................................................................................................................116

Figure 68 – Implementation of the Racket prototype, featuring an HTML toolbox as a Graphical User Interface ........................................................................................................................................................................................................................................................................116

Figure 69 – The Unity prototype of the tableware design system ....................................................................................................................................................................................................................119

Figure 70 – NURBS curve and surface example in web browser using three.js (mrdoob, 2013) ........................................................................................................................................................................................................................................................................120

Figure 71 – Implementation architecture for the Unity prototype: during development (left) and after compilation (right) ........................................................................................................................................................................................................................................................................121

Figure 72 – One possible derivation tree for the churches shape grammar (Figueiredo et al., 2013) .........................................................................................................................................................................................................................................................................124

Figure 73 – Grasshopper implementation of some rules (Figueiredo et al., 2013) .........................................................................................................................................................................................................................................................................124

Figure 74 – Parametric model of a tableware element, consisting of clustered components..............................................................................................................................................................................................................................................................126

Figure 75 – Initial steps in the derivation of a soup plate: envelope creation and functional partitioning .........................................................................................................................................................................................................................................................................126

Figure 76 – GH code for the parametric model (left) and the corresponding digital model (right) of a Doric capital (Castro e Costa, 2012) ..............................................................................................................................................................................................................................................................................127

Figure 77 – Shape grammar derivation of a soup plate’s base shape: section view .........................................................................................................................................................................................................................................................................129

Figure 78 – Shape grammar derivation of a soup plate’s decoration: plan view .........................................................................................................................................................................................................................................................................129

Figure 79 – Non-circular tableware from Art Deco (left: Tricorne collection, design by Don Schreckengost, manufactured by the Salem China Company) and Art Nouveau (right: collection manufactured by Villeroy & Boch, Mettlach factory) .........................................................................................................................................................................................................................................................................130

Figure 80 – Generating the same shape using different operations: revolve (left) and loft (right) .........................................................................................................................................................................................................................................................................130

Figure 81 – Different loft options in Rhino: loose (left) and normal (right) .........................................................................................................................................................................................................................................................................131
Figure 82 – A plaxel’s three-dimensional surface (left), and its attributes in profile (right) ... 131

Figure 83 – Grasshopper VB.NET component operating the editCoxel method (renamed editPlaxel), implementing Rule 2a of the Tableware Shape Grammar ............................................ 135

Figure 84 – Rule 2a of the Tableware Shape Grammar ....................................................................... 135

Figure 85 – Derivation of the base shape of a soup plate (Castro e Costa & Duarte, 2013) ... 136

Figure 86 – Parametric model of the base shape of a soup plate ..................................................... 136

Figure 87 – Generated digital model of the base shape of a soup plate ........................................ 136

Figure 88 – Grasshopper parametric model of the first collection (left), and resulting digital models (right) ........................................................................................................ 137

Figure 89 – Hierarchical levels of implementation scales ................................................................. 138

Figure 90 – Derivation of the decoration for a soup plate (Castro e Costa & Duarte, 2013) .. 139

Figure 91 – Implemented parametric models of both existing and original collections (left) and resulting digital models (right) .............................................................................. 140

Figure 92 – Customized Grasshopper toolbar for tableware design ................................................. 140

Figure 93 – The Modeler application running in Unity ....................................................................... 141

Figure 94 – Plaxel representing the scotia in the Doric base (adapted from (Castro e Costa, 2012)) ........................................................................................................................... 142

Figure 95 – Comparing plaxel description using dimensions and guide coordinates ...................... 143

Figure 96 – New operations in the multi-guide approach: (a) add guide, (b) delete guide, (c) split plaxel, and (d) merge two plaxels ................................................................. 144

Figure 97 – Direct guide manipulation in the Unity prototype .......................................................... 144

Figure 98 – Changing the superellipse parameters ........................................................................... 147

Figure 99 – De Casteljau construction (Haui, licensed under CC BY 2.0) ....................................... 148

Figure 100 – Hierarchical structure made explicit in Unity Hierarchy Window ............................... 150

Figure 101 – Generated tableware collection in Unity ...................................................................... 151

Figure 102 – Collection-wise editing: changes made in one element are reflected in all other elements .............................................................................................................................. 152
Figure 103 – Types of a tableware collection as distortions of the Archetype .......................153

Figure 104 – Type Table containing information about a collection's types: name, proportions, and visible parts .................................................................154

Figure 105 – XML file corresponding to a collection .........................................................154

Figure 106 – Design Participation Model for the design system (repeated) .........................163

Figure 107 – Tableware Shape Grammar: initialization rules ............................................164

Figure 108 – Default tableware collection, generated at the start of the Modeler application ................................................................................................................164

Figure 109 – Type Table ......................................................................................................164

Figure 110 – Tableware Shape Grammar: base shape rules ............................................165

Figure 111 – Guide manipulation ......................................................................................165

Figure 112 – Tableware Shape Grammar: decoration rules related to contour .................165

Figure 113 – Superellipse parameters .............................................................................166

Figure 114 – New operations in the multi-guide approach: (a) add guide, (b) delete guide, (c) split plaxel, and (d) merge two plaxels ......................................................166

Figure 115 – The Interpolator interface ...............................................................................168

Figure 116 – The Qualifier interface ................................................................................169

Figure 117 – ShapeDiver interface ..................................................................................171

Figure 118 – Thingiverse Customizer interface ..............................................................172

Figure 119 – Vectorial representation of design solutions as points ................................174

Figure 120 – Representation of a variation vector as arrow and point .............................175

Figure 121 – Definition of a design subspace through specification of limit solutions ......176

Figure 122 – Sliders for eliciting values interpolated between two solutions ...................178

Figure 123 – The Interpolator: Areal coordinates for eliciting values interpolated among three solutions .......................................................................................179

Figure 124 – The Qualifier application .............................................................................181
Figure 125 – Quality vector obtained from subtracting A from B and associated to a quality; the same quality vector is applied to solution C to generate a new solution........ 183

Figure 126 – The Survey application........................................................................................................184

Figure 127 – Design solutions used in the Survey application, represented by their corresponding soup plate ........................................................................................................185

Figure 128 – Robot configurations (sources: http://robohub.org/, http://blog.robotiq.com/, (Stan et al., 2008), http://www.elmomc.com/).......................................................................... 193

Figure 129 – Extrusion of white clay objects by Unfold (left) and extrusion of red clay object by Fabclay (right)........................................................................................................ 195

Figure 130 – Extrusion of tall ceramics objects by Jonathan Keep, United States (2013) (left) and by Olivier Van Herpt, The Netherlands (2014) (right)...................................................... 195

Figure 131 – Left: Limits of the 45 degree rule (Jokić et al., 2012). Right: the jiggering technique (Quinn, 2007) .................................................................................................................. 202

Figure 132 – Elements of a tableware collection generated in Unity, some having been selected for production through ceramic extrusion................................................................. 203

Figure 133 – Setup for ceramic extrusion................................................................................................. 203

Figure 134 – Components of the two ceramic printer prototypes.......................................................... 204

Figure 135 – Consecutive models of a customized tea cup: 1. geometric model in Unity; 2. geometric model in Rhino; 3. Cura model showing toolpaths; 4. produced physical model, glazed and filled with water ............................................................. 208

Figure 136 – Final results of each of the three iterations for producing the creamer ...................... 209

Figure 137 – Comparison between original (yellow) and reduced (cyan) models of the creamer......................................................................................................................... 210

Figure 138 – Six prototypes of the printed handled cup........................................................................ 211

Figure 139 – Two different designs for the handled cup....................................................................... 211

Figure 140 – Failed prints of the handled cups..................................................................................... 212

Figure 141 – Intermediate prints of the handled cup .......................................................................... 213

Figure 142 – Different orientations of linear support patterns according to object's rotation ......................................................................................................................... 214
Figure 143 – Prototype 6: production with longitudinal support (6a) and detail views of the handle (6b, 6c, 6d) .................................................................................................................................215

Figure 144 – Printing the saucer with different Support Patterns: Linear (left) and Concentric (Right) ........................................................................................................................................216

Figure 145 – Build sequence of a teapot's body according to the proposed printing process ...........................................................................................................................................218

Figure 146 – Left: Simplified tool path curve (red) and tangent to the surface on a curve point (green); Right: Compounded effect of rotating platform and 45 degree rule ..........219

Figure 147 – Left: Gough-Stewart platform (Stewart, 1965). Right: DMG MORI LASERTEC 45 Shape (industryarena.com) ........................................................................................................................................220

Figure 148 – Left: Rule 11b. Right: partial derivation of a soup plate, Romantica collection ...........................................................................................................................................226

Figure 149 – Roles of design participants in the process of developing a generic shape grammar (adapted from (Duarte, 2011)) ...........................................................................................................227
LIST OF TABLES

Table 1 – Categorization of digital fabrication processes.................................................59
Table 2 – General and partial dimensions of the Romantica collection .........................73
Table 3 – Digital models of generated solutions............................................................94
Table 4 – Shape rules of the six encoded collections ....................................................98
Table 5 – Shape rules after revision of the grammars following a comparative analysis......101
Table 6 – The rules and rule parameters used to encode the six collections studied.........104
Table 7 – Comparative evaluation of the three prototypes .............................................122
Table 8 – Different possible shapes for the same plaxel (adapted from (Castro e Costa, 2012)) ..........................................................134
Table 9 – Sample of abstract shapes obtained by modifying the parameter of the superellipse for positive integer rotational symmetries m from 0 to 8 (Gielis, 2003)......146
Table 10 – Summary of the designers’ responses to the forms ........................................157
Table 11 – Must-have wishlist .................................................................................158
Table 12 – Added-value wish list .............................................................................159
Table 13 – Visualization wish list .............................................................................160
Table 14 – Nice-to-have wish list .............................................................................161
Table 15 – CAD wish list .......................................................................................161
Table 16 – Summary of Additive Manufacturing equipment for ceramics ...............200
The human subjects research performed through Contextual Inquiry as reported in Chapter 3 was approved by the Pennsylvania State University Office of Research Protections (STUDY00008284) after being submitted to expedited Institutional Review Board (IRB) review.

The human subjects research performed through Usability Testing as reported in Chapter 4 was approved by the Pennsylvania State University Office of Research Protections (STUDY00008305) after being submitted to expedited Institutional Review Board (IRB) review.

The human subjects research performed through the Online Survey as reported in Chapter 5 was determined by the Pennsylvania State University Office of Research Protections (STUDY00007305) to be exempt from formal Institutional Review Board (IRB) review.
ACKNOWLEDGEMENTS

This dissertation would not have been possible if not for the support of many people to whom I owe my deepest gratitude.

I wish to express such gratitude in particular to my adviser José Pinto Duarte, for his mentoring and friendship, and for accompanying my progress throughout my career in Design Computing, and in particular throughout this research both at the University of Lisbon and at Penn State University.

I also would like to thank the members of my dissertation committee at Penn State, namely Loukas Kalisperis, for his continuous support and his insights regarding architecture and digital design; Tim Simpson and Conrad Tucker, for their perspectives from the engineering side of design; Tom Lauerman, for his enthusiasm about this project and for sharing his knowledge of ceramics and his experience with clay 3d printing; and Joaquim Jorge, for our stimulating conversations and his guidance in the field of human-computer interaction.

Additionally, I would like to thank Paulo Bártolo, for his supervision in the field of additive manufacturing, and for his support during my short stay at the University of Manchester, as well as to António Leitão, for his tireless assistance with programming in Racket.

I also wish to thank everyone at the School of Architecture in the University of Lisbon who have accompanied and supported me along a large part of this research. I owe special thanks to everybody at the Design and Computation Group, for their valuable feedback and invaluable friendship. Additionally, I thank everyone at the Stuckeman School, and in particular to my colleagues at the Stuckeman Center for Design Computing, for making me feel at home.

I would like to offer my special thanks to Matceramica, particularly to Marcelo Sousa and Gonçalo Martins, for believing in this project and for providing invaluable information.
I am indebted to the Stuckeman School for their financial support through Graduate Assistantship.

I am also indebted to the Portuguese Foundation for Science and Technology (Fundação para a Ciência e a Tecnologia - FCT) for funding this research under a Doctoral grant with reference SFRH/BD/88040/2012, through the European Social Fund (Fundo Social Europeu) and by the Human Capital Operational Programme (Programa Operacional Capital Humano - POCH).

Finally, I am deeply grateful to my Mother and Father for providing for my education and supporting me in the decisions that eventually led me to being a researcher in Design Computing.

Last but not least, I thank Françoise and Francisca for making everything worthwhile.
Chapter 1
Introduction

This dissertation is concerned with design processes in mass customization.

Mass customization is a production and management paradigm according to which products and services are tailored to meet the individual customer’s needs or preferences with efficiency close to mass production, which results in lower costs when compared with traditional customization. This seemingly self-contradictory concept has interesting repercussions in how products and services are designed, namely an increased participation of end-users in the design process, as can be attested by previous applications of mass customization to different design domains.

Despite its potential advantages, the implementation of the mass customization paradigm raises challenges in terms of design. The research presented in this dissertation focuses on finding solutions for those challenges by developing a prototypical mass customization system supported by computational tools such as generative design systems and digital fabrication.

1.1 Mass customization

Mass customization (MC) can be defined as “producing goods and services to meet individual customer’s needs with near mass production efficiency” (Tseng & Jiao, 2001, p. 685). As a production paradigm, MC features qualities from both craft production and mass production. Like craft production, MC is based on highly flexible processes, producing directly according to demand instead of estimates, thus attaining high levels of variety and personalization. As in mass production, MC tends to produce large quantities, with low unit costs, using automatized production methods (Pine, 1993).

Mass customization bears the potential to improve competitiveness of companies that adopt such paradigm, namely through differentiation and innovation. In a market composed of ever more demanding consumers, in which differentiation is increasingly valued, a sound implementation of
MC can be beneficial for improving customer satisfaction, and a subsequent profit gain (Bernard et al., 2012). Despite the fact that the concept of mass customization has been around since the 1970's (Toffler, 1971), consumers are still drawn towards the possibility of customizing their products with a sense of novelty (Franke et al., 2009). Being customized, a product is inherently differentiated from customer to customer, as opposed to mass produced products, which follow the one-size-fits-all paradigm. Moreover, a number of references in mass customization literature (Franke & Piller, 2004; Piller & Müller, 2004; Schreier, 2006) suggest that consumers can be willing to pay twice as much for a customized solution as they would for a mass produced product.

According to Piller (2004), the main enabling factors of mass customization are (1) high flexibility in manufacturing and (2) an efficient method in eliciting customers’ preferences. Recently, flexible manufacturing has been an important research focus in many fields, including mass customization research and practice – just consider the media coverage on 3d printing, additive manufacturing and digital fabrication in recent years (Chua et al., 2010; Wohlers & Caffrey, 2014). Customer elicitation has received less attention from the research community, although that has changed in the last ten years, judging by the growing number of papers on user interfaces for customer elicitation, called toolkits or configurators (Franke & Hader, 2014; Heiskala & Tiihonen, 2007; Hermans, 2012).

Configurators enable a more efficient interaction between customer and manufacturer, essential to a successful elicitation process, by integrating customers in an iterative process in which they modify properties of the customized product and are presented the corresponding design solution. According to Duray and Milligan (1999), involving the customer in the early design stages of the mass customization process leads to highly customized products. Such dialogue between customizers and customers is aligned with the collaborative customization approach introduced by Gilmore and Pine (1997). An interesting consequence of eliciting users’ preferences through configurators is that the design process is performed by the customer, thus rendering mass customization an exercise of co-design (Crayton, 2001), as shown by some examples in the fashion industry (Peterson, 2016; Ulrich et al., 2003).
1.2 Design democratization

Design is traditionally the métier of designers. However, in the last decades we have witnessed the emergence of approaches such as user-centered design, participatory design and co-design, in which the users, for whom design is performed, are brought closer to the design process with various degrees of proximity (Sanders & Stappers, 2008). ‘User-centered design’ is a broad term to describe design processes in which end-users influence how a design takes shape, from being consulted about their needs and involved at specific times during the design process such as requirements gathering and usability testing, to being involved as partners with designers throughout the design process (Abras et al., 2004). Co-design moves one step further from user-centered design, bringing the agents in the design process closer together, as well as re-interpreting their roles in the process. The classic agents in the design research process are Researcher, Designer, and User. In user-centered design research, the Researcher observes a passive User, and hands their conclusions to the Designer. In a co-design approach, the roles of all three agents are more difficult to distinguish (see Figure 1). The User plays a more active role in the design process, engaging in a dialogue with the Designer to develop the product and actually performing design tasks. The Researcher designs tools that will aid in such dialogue, trying to understand the point of views of both User and Designer. Finally, the Designer might develop some research skills deriving from a closer collaboration with the Researcher.

![Diagram showing user, designer, and researcher roles]

Figure 1 – Re-adjusting roles from a classic design approach (left) to a co-design approach (right) (U = end-user; D = designer; R = researcher)

At the same time, we have witnessed an exponential technological development, reflected in today’s omnipresent online connectivity, increased technological literacy and advancements of design and fabrication technologies. Such a development produced more accessible digital design tools, meaning they are easier to use, derived from an increasing concern for the user in software
design, but also easier to obtain, due to increased internet access and business models in which basic services are provided free of charge\textsuperscript{1}. This increased accessibility of design tools benefits designers in their professional activity, but it also empowers non-designers\textsuperscript{2} to perform design tasks. Although the legitimacy of having non-designers performing design tasks has been the subject of debate, this tendency influences many different design fields, architecture being one of them, as illustrated by initiatives such as the Wikihouse project\textsuperscript{3} (Vardouli, 2012).

Consequently, we are witnessing a process of design democratization, in a double sense. First, increased accessibility of design tools enable designers to reduce the cost of design activities, rendering design services they provide increasingly accessible to the general population. Second, the same accessibility enables non-designers to perform some design activities and, therefore, design as an activity is no longer performed exclusively by designers, being shared with non-designers. Such phenomenon raises a number of important questions, arguably the most important being how to ensure the quality of design performed by non-designers (Kolarevic, 2015).

This question is particularly relevant in the context of mass customization, which is a paradigmatic reflection of design democratization. While in a traditional customization scenario customers state their preferences to a designer, in a modern mass customization application, it is neither expected nor possible to have designers validating every customized design solution. Therefore, customers must manipulate the design themselves, at least to an extent that should be determined by the developer of the mass customization system. However, these customers might not be designers themselves and, therefore, are not expected to be proficient in performing design tasks. In fact, for a non-expert, the complexity of the design process can be overwhelming (Zipkin, 2001; Dellaert & Stremersch, 2005; Piller et al., 2005). Without proper guidance, the non-experts’ lack of design experience would lead to low quality design solutions.

\textsuperscript{1} In the freemium business model, for example, basic services are provided free of charge while more advanced features must be paid for (Oxford Dictionaries, 2017a).

\textsuperscript{2} Let us consider non-designers as individuals that perform design activities despite not having been professionally trained for such.

\textsuperscript{3} https://wikihouse.cc/
1.3 Generative design systems

A solution used in existing mass customization initiatives is to embed proper design guidance in configurators. Such guidance can be supported by generative design systems, as it was shown by Duarte (2001) in his application of mass customization to the architectural design of housing. According to the author, mass customization can be applied to such design practices through a dialogue between two systems, one concerning design and another one concerning production, moderated by the use of computational technology (Duarte, 2008). The production system enables the automatic manufacturing of customized designs that are automatically generated by means of a generative design system such as a shape grammar. This tripartite system enables the necessary automatization for end-users to efficiently develop custom design solutions.

Consider a generative design system as a tool that is responsible for producing designs in a somewhat automatic fashion. Examples of generative design systems include parametric models and shape grammars. In order for such a design system to automatically generate design solutions, it needs to be implemented as a computer program and so, the corresponding design process needs to be translated into an algorithm. Such externalization of the design process implies its visibility as if it were inside a glass box (Jones, 1970, p. 49). For this purpose, the design process needs to be deconstructed into consecutively simpler design tasks, which can be then translated into simpler, articulated algorithms. However, this deconstruction exercise is not always possible in Design. In fact, such positivistic attitude of deconstructing a problem into its atomic constituents that are easier to solve is controversial (Snodgrass & Coyne, 1996). Design problems are often considered wicked problems, or ill-defined problems, partially because the problem is not completely defined at the beginning of the design process (Buchanan, 1992; Rittel & Webber, 1973). It is common that new aspects related to the design problem emerge during the design process itself, implying an adaptation of the problem’s formulation to accommodate for new variables that need to be addressed. Therefore, it seems impossible to deconstruct a design problem into a set of atomic problems, since such deconstruction is deemed not to make sense as new variables are introduced into the design process.

Despite generally being considered ill-defined, design problems can fall into one of three categories: routine, innovative, or creative design (Brown & Chandrasekaran, 1985; Coyne et al.,
What distinguishes creative design from the other two categories is the variability of its design space, defined by the set of all design alternatives, due to the introduction of new variables along the design process:

"Routine design can be defined as the design that proceeds within a well-defined state space of potential designs. [...] Innovative design can be defined as nonroutine design that proceeds within a well-defined state space of potential designs. What distinguishes it from routine design is that the designs produced are outside the routine or normal space. This distinction is produced by manipulating the applicable ranges of values for variables. What results is a design with a familiar structure but novel appearance because the values of the defining variables are unfamiliar. [...] Creative design can be defined as nonroutine design that uses new variables producing new types and, as a result, extending or moving the state space of potential designs. In the extreme case, a new and disjoint state space is produced." - (Gero, 1990, p. 34)

According to this categorization, we might consider that only creative design corresponds to ill-defined problems, since the stable solution space that characterizes routine and innovative design presumes well-defined problems to begin with. Jones (1970, p. 50) suggests a different but related distinction between splittable and unsplittable design problems. Examples of splittable design problems include flow systems or assemblies such as a chemical plant, an electrical supply network or a telephone system. As assemblies, these designs feature a one-to-one relationship between function and physical components. Consequently, an initially planned design sequence derived from input and output specifications typically suffers only minor deviations. On the other hand, unsplittable design problems include buildings, cars and machine tools, in which function is distributed over the different components of a more integrated assembly. Merging the two categorizations, we can say that problems in routine and innovative design are splittable into simpler constituents, and thus applicable of the glass box method, whereas creative design problems are not. Therefore, routine and innovative design can be successfully implemented into a generative design system.

Implementing a mass customization system implies a fixed solution space (Piller, 2004), ruling it out as a creative design problem. Therefore, according to Gero and Jones’ previous definitions, we consider mass customization to be an innovative design problem. Considering that we are looking for ways of empowering non-expert designers to design, it seems appropriate to constrain
that power to innovative design, reserving the act of creative design and the inherent complexities of ill-defined problems to trained designers. Also, delimiting the mass customization problem to innovative design corroborates that its solution can be supported by a generative design system.

In summary, the paradigm of mass customization seems to be finally gaining economic relevance due to developments in digital fabrication and generative design systems. Nevertheless, despite its potential economic advantage and technical feasibility of its enabling factors, implementation of mass customization still presents a number of challenges, namely in terms of design.

1.4 Problem

One limitation in mass customization is the potential for mass confusion, a phenomenon that may emerge in mass customization systems from forcing customers to ponder on too many choices or design decisions (Piller et al., 2005; Zipkin, 2001). Similar phenomena emerge in contexts other than mass customization. Schwartz (2004) explores the paradox of choice that consumers face while grocery shopping or browsing through cable television channels. Although researchers argue that choice is inherently good, empirical evidence show that too much choice actually inhibits the act of choosing due to the psychological effort of evaluating more options.

The same might happen in mass customization, in which a customer is not only faced with the paradox of choice, but also with a design task. In fact, to present a design task to a non-designer can trigger a feeling of white canvas syndrome, corresponding to not being able to design something because of the difficulty of choosing from all possible design paths. Mass confusion can thus derive from an extensive design space, which might overwhelm customers and ultimately deter them from engaging into the mass customization experience entirely.

On the other hand, a successful mass customization experience needs to present enough variety in order to become interesting and respond to specific users’ needs and preferences, implying an extension of design options and consequently a more complex design space. Therefore, the success of a mass customization experience depends on finding a balance between design variety and design complexity. In other words, a balance is needed between a design space that is large enough to be interesting, but not so large that induces mass confusion. However, finding such
balance is challenging, considering that the potential for mass confusion increases with the complexity of the design space. Such challenge leads to the main research question in our research:

__________________________

**RESEARCH QUESTION:**

*How to extend the design space of a mass customization system while avoiding mass confusion?*

__________________________

### 1.5 Hypothesis

To address the posed research question, let us consider how configurators are devised. Typically, the design of mass customization solutions is divided into two phases: (1) the design of the configurator and (2) the use of the configurator by the user to customize a design. Although designers participate in the design process of a mass customization configurator, their participation is terminated the moment the configurator is deployed, permanently determining the configurator’s design space. From then on, the design process is controlled by both the configurator and its users, the customers. Therefore, in order to ensure the design quality of customized solutions, both explicit and tacit knowledge about the design domain need to be encoded in the design system. Also, the design process depends on the balance between design variety and potential mass confusion.

We proposed re-structuring this two-tier design process into three phases, reinforcing the role of the designer in the mass customization process. (1) The first phase corresponds to a general system that informs the design of the customized product. This general design system should be defined by a researcher or system designer and corresponds to a wider design space that encompasses a large variety of possible designs. (2) The second phase corresponds to the creation of a customizable design, or template, using the general design system previously created. The customizable design should be defined by a designer and corresponds to a subspace of the initial design space. (3) The third phase corresponds to the actual customization of the design, using the customizable design created in the second phase. The customized design is defined by the end user and belongs to a previously defined subspace.
We hypothesized that by extending designers’ participation beyond the development and deployment of the design system allows that only explicit knowledge needs to be encoded into the design system, while tacit knowledge can be used by designers themselves in a subsequent phase, and thus promoting both more flexibility and better quality of generated designs. Also, the designer serves as a mediator of complexity between the general design system and the non-expert end-users. By defining customizable collections, designers divide the general design space into simpler subspaces, thus reducing complexity and consequently reducing the potential for mass confusion.

Our hypothesis was materialized in a general model that defines the interactions among the different participants in the design process, hence called the Design Participation Model (see Figure 2). Also, the model also prescribes the adequate digital design models for each phase. The Design Participation Model encompasses four distinct sequential components, each targeting a different set of participants in the design process. Each component generates a different type of digital design model, which acts as input in the subsequent component.

![Design Participation Model](image)

*Figure 2 – Design Participation Model for a mass customization design system*

The first component corresponds to a Shape Grammar that encodes design rules and is the foundation of the design system. Such grammar is developed by the system designer, or researcher, and it should be supported by reference design systems and by the analysis of existing design products and processes. The resulting output of this component is a rule-based model, namely a shape grammar that can generate collections within a particular style.
The second component is called Modeler and its target user is the designer. The Modeler component enables designers to manipulate design rules encoded in the Shape Grammar to produce general parametric models, each corresponding to a customizable design. These general models can be further constrained by direct manipulation of their design space. The constrained parametric models correspond to the output of this component.

The third component is called the Navigator and its target user is the end-user. In this component, the end-user can navigate through the design space that corresponds to the constrained parametric model provided by the Modeler component, whose boundaries were defined by the designer, hence exploring variations in order to specify one customized solution. The resulting design is encoded in a digital model.

A final component is the Digital Fabrication process that can actually produce a physical object from the digital model. Although this is not part of the design system, it emphasizes its continuity, and thus it is included in the Design Participation Model.

The proposed Design Participation Model presents similarities to existing tools in other design fields. Siddique and Zhou (2002) suggest a template based approach for the automatic generation of CAD models for mass customized coffeemakers. In their approach, a template represents a product family as a parametric model. A similar approach is adopted by Cox (2000). In another example, online publishing platforms like Wordpress⁴ allow end-users to create blogs and simple websites without programming by providing customizable websites called ‘themes’, which are developed by expert users external to Wordpress, who in this case are Web-designers. After selecting a theme, end-users can customize the appearance of their website, as well as enter the content they want to publish. Considering the success of the Wordpress platform, we have adapted a similar strategy to mass customization.

---

⁴ https://wordpress.com/
1.6 Methodology

In order to test the proposed Design Participation Model and as a general methodology for this research, we developed a prototype for a mass customization system. Our research was supported by a case study of a particular design domain from which we could gather information and on which the prototype could be tested. The selected design domain was ceramic tableware, mainly due to having identified an untapped potential for the application of mass customization in this industrial sector, but also because of the low costs associated with manufacturing ceramic tableware when compared to other manufacturing sectors.

A case study in ceramic tableware

Despite the advantages of mass customization, one industry that seems reluctant to tap into its potential is ceramic tableware. Although some initiatives have been documented (Museros et al., 2004; Huang & Hudson, 2013; Vista Alegre, 2013), they lack the impact of the examples mentioned previously. Meanwhile, recent years have seen the emergence of low-cost ceramic production, namely tableware, in countries with emerging economies, making it difficult for both American and European tableware manufacturing companies to keep up in terms of competitiveness (Grahl, 2004; Ken Research, 2014). The inability to compete by lowering prices pushes those companies into improving the quality of their products through innovation and differentiation (INTELI, 2009). Mass customization can provide both.

Ceramic tableware features characteristics that render it an interesting product for mass customization. For once, it is easily manufactured, when compared for example to buildings, which are quite expensive to produce, both in terms of money and time. Moreover, recent developments in manufacturing technologies, namely in digital fabrication, have benefitted many fields of design. Often we hear of yet another addition to the list of products that have been 3D printed, which includes smartphone cases, medical prosthetics, airplane parts, and also ceramic-based products such as electronic components. Therefore, it should be possible to include ceramic tableware in that list. Another advantage of ceramic tableware is that its shape is well known by the general public, since it is used by most on a daily basis. By being familiar with the shape of an object, end-users should feel more confident to manipulate that shape. Therefore, we considered
that there is a potential economic advantage in implementing the mass customization paradigm in the ceramic tableware industry.

In the scope of the case study, an existing ceramic tableware manufacturing company was analyzed considering different aspects related to the mass customization paradigm. The research design was presented to different tableware manufacturing companies, with two objectives in mind: (1) to assess the companies’ receptivity to the concept of mass customization in general, and to this research project in particular and (2) to establish a partnership with a company that would empathize with the project, towards obtaining information crucial for the research. A Portuguese company called Matceramica expressed interest in the mass customization concept and in our project in particular and, therefore, a partnership was established.

Matceramica, the manufacturing company with which a partnership was established (henceforth referred to as ‘the Manufacturer’), played an important role in this research by providing the following information:

- documentation concerning the company’s products, such as technical drawings, digital models, actual samples, or photographs of their tableware collections;
- data about the design process for creating tableware collections, namely through direct or indirect observation of designers at work, task analysis, and interviews;
- data about the production process, from early design stages to actual manufacturing, provided through guided visits, observation and interviews, in order to assess the potential and constraints within each stage;
- feedback from the company’s staff about project results, namely about the implementation of the different systems.

**Research tasks**

Following Duarte’s implementation model (Duarte, 2008), the mass customization system comprises a design system that encompasses design rules for generating customized solutions, as well as a production system capable of actually manufacturing such solutions, while using a computational framework to articulate both systems. As researchers in a design field, our main focus was the development and implementation of the design system, although we illustrated its
relevance by integrating it with a user interface and a production system. The development of the mass customization system prototype implied performing the following five tasks:

1. to review literature to determine the State-of-the-art in related areas;
2. to develop a design system that encodes design rules for generating tableware collections;
3. to implement the design system into a computer application, capable of generating design solutions;
4. to develop user interfaces that enable both expert (designers) and non-expert (final customers) users to efficiently make use of the design system;
5. to develop a production system capable of producing the custom design solutions, considering its economic viability.

**Literature review**

In the literature review, we addressed both theoretical and technical aspects covering topics such as: mass customization, design methods, user-centered design, co-design, generative design, shape grammars, parametric modeling, user interface design, visualization technology, ceramic manufacturing technology, and digital fabrication.

**Development of the design system**

The design system should be able to generate design solutions for ceramic tableware collections in a semi-automatic fashion according to users’ preferences. We proposed that such generative design system is supported by an articulation between shape grammars and parametric models. A generic shape grammar (Beirão et al., 2011; Beirão & Duarte, 2018), henceforth referred to as the Tableware Shape Grammar or TSG, was used to encode the design rules of tableware design. Thereby, TSG constitutes the knowledge base of the design system, whereas parametric models were used to implement that design system. Typically, encoding design rules into a shape grammar encompasses three steps (Li, 2001b):

1) **Analyzing the corpus:** In shape grammar terminology, ‘corpus’ refers to a set of shapes that are known to belong to the design language encoded by the grammar. In the current research, the corpus encompasses different ceramic tableware collections produced by the
Manufacturer. Since a shape grammar can encode design rules of a particular style (Ahmad & Chase, 2012), our shape grammar corresponds to the style adopted by the Manufacturer. The shape grammar resulted from the analysis not only of existing collections, but of all the information provided by the Manufacturer, particularly about the process of designing new tableware collections. Analyzing the corpus also helped establishing a taxonomy of ceramic tableware design.

2) **Inferring the design rules:** Shape grammar rules were inferred according to the previous analysis of the various components, including the differences and commonalities among the elements that constitute the corpus. The inferred rules pertain to three-dimensional shape of the tableware elements, such as their base shape and relief-based decoration, disregarding two-dimensional decoration, such as paint-based or decal-based decoration.

3) **Fine-tuning the design rules:** The more rules were inferred, the more complex the grammar became. In order to keep such complexity manageable, inferred rules needed to be fine-tuned, so that aspects such as redundancy or versatility were controlled. In fact, each rule was fine-tuned after its inference in an iterative cycle, taking into consideration all the remaining grammar rules that had been previously inferred. Such iterative process enabled abstracting the rule set towards a generic shape grammar (Duarte, 2011).

In addition to the analysis of existing designs and design processes, development of the design system was supported by the work of Leon Battista Alberti. In his treatise on the art of building, *De re aedificatoria* (Alberti, 1485), Alberti prescribes rules that crystalize his understanding of how buildings should be designed. In Books 6 and 7 of his treatise, Alberti particularly focuses his attention on the proper design of classical columns and their composing elements. It is interesting to read Alberti’s description of the Doric capital in Book 7, Chapter 6, in James Leoni’s translation to English:

> “Some Artists therefore among the Dorians (if we may thus allow the Greeks the Honour of all Inventions) were the first that endeavoured to improve it [the capital] by making it round, so as to look like a Cup covered with a square Tile”  
> (Alberti, 1955, p. 141)
Or in the same passage in Rykwert’s translation to English:

_The inhabitants of Doron (if the Greeks are to be believed in everything) were the first to put it to the lathe, and to make it look like a round dish set under a quadrangular lid_” (Alberti, 1988, p. 201)

It should be noted that, in either version, the shape of capitals is associated to tableware, namely a cup and a dish. And in fact, classical capitals share some of their morphology with traditional design of tableware: they typically feature a round horizontal shape, and a profile composed of curved lines, which are combined into a double curved surface.

Shortly after this description, Alberti delivers his rules for designing column elements with algorithmic precision. In fact, these rules are so detailed that they could be implemented into computer programs that generate such designs. And indeed, in the scope of a research project called Digital Alberti (Krüger et al., 2011), the rules prescribed by Alberti for the design of capitals have already been scrutinized and translated into a generative design system supported by parametric modeling (Castro e Costa, 2012) and shape grammars (Quaresma, 2014).

The morphological similarities between ceramic tableware and classical architectural elements, such as capitals, associated to the algorithmic detail of the rules prescribed by Alberti to design them, suggested that those rules can be used to design ceramic tableware collections. Therefore, we adapted an existing design system to serve as the base for a new design system of a different design domain, namely ceramic tableware. This approach relieved system designers, in this case the research team, of creating a design system from the ground up, while suggesting that the new system can serve as a basis for further design systems.

**Implementation of the design system**

A shape grammar is capable of generating a large number of design solutions, implying the use of computational power in order to manage such a complexity. Therefore, the rules of the shape grammar were implemented into a computer program so that they can be more easily manipulated by designers, by combining them together and setting their parameters. Implementation of the grammar rules implied three different sub-tasks:
1) **Selecting the implementation tools**: Creating a computer program implies the adoption of a programming environment. It was therefore necessary to understand which available programming languages are more suitable for implementing design rules. Suitability factors included the ability to efficiently generate and display fairly complex geometry. Another determining factor for selecting an implementation tool was its ability to produce applications capable of running on the World Wide Web, so they can reach a wide audience of potential customers.

2) **Implementing the design rules**: Shape grammar rules were implemented by being translated into the selected programming language. Instead of developing the shape grammar prior to implementing its rules, rules were implemented as they were inferred. Therefore, the grammar could be tested incrementally, as good practice advises, in an iterative cycle between the development and implementation phases of the design system.

3) **Usability testing**: The implemented design system is to be used by designers to create customizable collections. Therefore, we asked target users to test the application, both during its development, in order to guide us through the process, and after completion, in order to assess its actual utility.

Please note that, although we implemented the rules of the shape grammar, we did not implement it as a proper shape grammar interpreter. In fact, such interpreter would need to be capable of automatically executing two different tasks: a) to recognize shapes and b) to operate on those shapes (Chau et al., 2004). The implemented system is incapable of shape recognition, and therefore we refrain from calling it a shape grammar interpreter, but rather an implementation of the grammar’s rules that enables an expedited use of the grammar. Such approach had been tested previously in the Digital Alberti project (Figueiredo et al., 2013). In fact, a set of implemented rules were also borrowed from Digital Alberti, which served as a base for implementing rules of the Tableware Shape Grammar.

The two phases concerning the design system, development of the shape grammar and implementation of computer applications, were intertwined. Nevertheless, for the sake of clarity, development and implementation are divided into their own chapters.
Development of the user interface

The ultimate user of a mass customization system is the customer. There are many kinds of customers, and while some of them might have some interest in design, and some others might be tech-savvy, the mass customization experience should be as inclusive as possible, enabling everyone to use it. Therefore, the user interface for the mass customization system, typically called a toolkit or configurator (Franke & Piller, 2002), was designed with the least experienced MC customer in mind. However, to develop a user-friendly configurator is a challenging task. As mentioned, designing can be an unsettling task for those who are not used to it (Piller et al., 2005).

In order to illustrate the relevance of the design system and its capacity to be integrated in a complete mass customization system, we explored two different approaches to user interfaces for end-users, both of them applying a strategy of design space exploration. In the first approach, which resulted in an interface called Interpolator, we explored the concept of navigating the design space through user-controlled interpolation among a number of design solutions in order to describe a custom design.

In the second approach, which resulted in an interface called Qualifier, we attempted an approximation to the end-users’ natural language, through the use of shape qualifying adjectives to drive the manipulation of the shapes of tableware elements. In an effort to come closer to the user’s perspective, implementation of the Qualifier interface included the following sub-tasks:

1) **Data acquisition:** Through an online survey, users were presented with different tableware elements, and asked to qualify them by typing a word that best described the differences between those elements.

2) **Data post-processing:** The information gathered through the survey was processed before implementation, which included identifying misspelling or responses containing qualities considered too vague.
3) **Implementation:** The qualities acquired through the survey were mapped onto the design system, associating such qualities with designs solutions and differences between them. This enabled the end-user to manipulate a collection’s shape by invoking those qualities.

**Development of the production system**

From the point of view of production, we have seen that flexible manufacturing is one of the main enabling factors for MC. This factor has been the subject of recent research on production technology, namely about digital fabrication in general and additive manufacturing in particular (Da Silveira et al., 2001; Hu, 2013, p. 6; Piller, 2004; Reeves et al., 2011). Particularly in the ceramic industry, research shows that additive manufacturing can be used for producing ceramic objects (Jokić et al., 2012; Keep, 2013; Lauerman, 2014; van Herpt, 2014) and tableware in particular (Hoskins, 2012). A question raised is whether digital fabrication, namely additive manufacturing, can be used to produce tableware elements. The particular shape of tableware elements such as plates, mugs, or teapots, represent particular challenges to different digital fabrication technologies. For example, manufacturing predominantly horizontal tableware types such as dinner plates or overhanging parts such as a mug’s handle or a teapot’s spout appears to be a challenge to using material extrusion without the use of auxiliary supports.

These issues were addressed in order to illustrate the relevance of the design system and its capacity to be integrated in a complete mass customization system, towards a potential production system. Such production system ought to be able to materialize the design solutions generated by the design system. The end product of the design system is a digital model that crystalizes the customer’s preferences for their tableware collection expressed through the customization process. The digital model is encoded into a file format that can be understood by the manufacturing equipment. In order to be cost effective, such manufacturing equipment must be flexible enough to produce different shapes while minimizing additional cost, which justified digital fabrication as a research path. The exploration of a potential production system encompassed the following sub-tasks:

1) **Technology survey:** The manufacturing of mass customized tableware is fundamentally different from its mass produced counterpart. Nevertheless, it was crucial to understand how
ceramic tableware is manufactured in both handicraft and industrial contexts. Also, we needed to learn about the available digital fabrication techniques, and assess their potential and suitability for manufacturing ceramic tableware in the context of mass customization, in both technical and economic terms.

2) **Testing the adopted technology:** After selecting a suitable candidate digital fabrication technique, a number of experiments were performed using the corresponding equipment to explore its potential and limitations for the production of ceramic tableware.

3) **Adapting equipment:** It was deemed advantageous to adapt the tested equipment to the manufacturing of tableware, considering the constraints of ceramic production such as physical integrity and shape fidelity to the customized model.

### 1.7 Contributions

The following results yielded by our research are considered generalizable contributions to knowledge.

**A model for the implementation of a design system in a mass customization context**

In a typical mass customization solution, the end-user interacts with a configurator that is able to generate and present design solutions according to the end-user’s preferences. In this scenario, the role of designers is restricted to the development of a generative design system that enables the configurator to generate valid design solutions. Compared to traditional mass customization systems, the proposed Design Participation Model extends the role of designers beyond the conception and development of the design system, allowing them to create customizable design solutions after the design system is implemented. This novel approach grants more flexibility to the design system, while granting designers a more direct quality control of design solutions.
A methodology for developing a generic shape grammar

The generative design system is supported by a shape grammar that encodes rules for the design of ceramic tableware, entitled the Tableware Shape Grammar (TSG). Development of TSG was based on research about generic shape grammars, which explores the concept of combining rules from different specific grammars into a generic grammar that encodes a broader design domain or style. The development of the TSG offered the chance of systematizing the methodology for obtaining a generic grammar, which can be generalized for other design domains. Moreover, development of the TSG illustrated that rules can be imported from external design domains, namely from classical architectural elements to tableware.

Additional results

Besides these two generalizable contributions to knowledge, our research produced a number of prototypes that can be useful for further research. In the scope of implementing the proposed design system, we developed a number of computer applications corresponding to the Modeler component with various degrees of operability. Each of these prototypes can be further developed into its original purpose, towards an industrial implementation of a mass customization system. Alternatively, they can serve as the starting point for other applications, either by recycling the code that was developed, or just as inspiration about necessary features or adopted paradigms and approaches.

Additionally, we implemented two prototypical applications corresponding to the Navigator component, focusing on user interfaces that enable non-expert manipulation of parametric design models. In the proposed mass customization system, designers create customizable collections encoded in parametric models. Customized collections are thus generated by manipulating the values of the models’ parameters, a task that is assigned to end-users. Despite being simplified design tools when compared to shape grammars, parametric models can still be overly complex for non-expert users, proportionally to the number of parameters. Therefore, two user interfaces were developed that further simplify the effort of manipulating the parametric model, shielding end-users from being overwhelmed by the design process.
Finally, prototypes of design solutions generated by the design system were produced using digital fabrication technology. Although implementing a production system was outside of the scope of our research, the work developed about that topic demonstrated the application of digital fabrication to ceramic tableware. Also, data was generated that might be useful for a future implementation in a mass customization context, as well as for other applications in ceramic manufacturing. In particular, the results of testing digital fabrication technology, both successes and failures, can prove useful for other practitioners that make use of similar equipment.

1.8 Dissertation organization

The contents of this dissertation are organized into seven chapters, beginning with this introductory chapter in which we have presented the context, problem, hypothesis, methodology and contributions of our research.

Chapter 2 presents a literature review of the topics introduced in Chapter 1, with a particular focus on mass customization, generative design systems and digital fabrication. We address the concepts that are used throughout this research by explaining their meaning and context.

Chapter 3 addresses the development of the design system by further explaining the proposed Design Participation Model, presenting the analysis of existing information and covering the development of the Tableware Shape Grammar.

Chapter 4 addresses the implementation of the design system into prototypes of computer applications for parametric modeling, through a detailed explanation of how the many components of the design system were translated into the different tools used, and focusing on the virtues and difficulties of those tools.

Chapter 5 addresses the development of user interfaces, focusing the challenges of enabling non-designers to use design tools such as parametric models. Two different approaches based on design space exploration are presented as a response to the challenge of mass confusion.
Chapter 6 addresses explorations toward the production system, which include a survey of existing technology capable of automatic production of ceramic tableware, a number of experiments with digital fabrication and a number of improvements for the used equipment.

Finally, Chapter 7 concludes this dissertation, presenting a discussion on the results, summarizing the inferred conclusions and suggesting paths for future research.
Chapter 2
State-of-the-Art

In this chapter, we review the literature of the fields of knowledge connected to this dissertation and identified in the Introduction, namely design computing, product design, architectural design, interaction design, digital fabrication and mass customization. In this literature review, we aim at establishing a primer on relevant concepts used throughout this research, by explaining their meaning and context.

We divide this chapter in three main sections, following Duarte’s implementation model for mass customization (Duarte, 2008). The first section is dedicated to the topic of mass customization, while the subsequent sections follow the model’s components. The second section is dedicated to design and design systems, while the third section is dedicated to production and manufacturing systems, namely digital fabrication. Regarding the computer system, we address various computational issues within the scope of the previous three sections.

2.1 Mass Customization

In the context of rapid prototyping, Chua et al. (2010) provide the following overview about product manufacturing:

“The competition in the world market for manufactured products has intensified tremendously in recent years. It has become important, if not vital, for new products to reach the market as early as possible, before the competitors [1]. To bring products to the market swiftly, many of the processes involved in the design, test, manufacture and market of the products have been squeezed, both in terms of time and material resources. The efficient use of such valuable resources calls for new tools and approaches in dealing with them” (Chua et al., 2010).

The 20th century featured a tendency towards a service-based economy, leading to deindustrialization. As a result, today’s American and European economies are facing growing challenges to compete with low wage regions, and therefore keeping, or bringing production back
into their markets. As such, in the Treaty of Lisbon (European Union, 2007) it is advocated that in order to regain competitiveness, the manufacturing sector must shift to high-tech industries by moving from resource-based to knowledge-based manufacturing. In the Gothenburg protocol (EU Commission, 2001) it had been previously advocated a move from mass-produced, single-use products to new concepts of higher added value, custom-made, eco-efficient and sustainable products, processes and services.

In this framework, we believe that a shift to mass customization is the way to achieve such objectives. As an emerging manufacturing paradigm, it diverges from mass-produced, single use products by adapting them to their end-user.

**From craft production to mass personalization**

Before the industrial revolution, the existing manufacturing paradigm was craft production, in which craftsmen produced objects by hand or using hand-operated utensils. This meant that products took a long time to manufacture, and at a high cost per unit. The industrial revolution in the 19th century meant a transition from handcrafted to mass-produced objects, leading to an exponential increase in production volume, as well as to a drastic decrease in cost per unit. However, this also led to a drastic decrease in variety. By contrast, craft production allowed for the product to be unique, and adapted to its user’s needs. The industrial machine would produce many more items in much less time, but they were all copies of each other (Hu, 2013).

This trend began to change in the 1960s with the introduction of lean production (Womack et al., 1991). This manufacturing and management paradigm emerged from the automotive industry, which aimed at reducing waste to increase productivity. In lean manufacturing, products would be made-to-order, which meant that they could be adapted to that order before they were produced. Such approach, therefore, led to a new increase in product variety, and paved the way to the next paradigm, mass customization, which would emphasize this tendency even more.

The concept of mass customization has been anticipated by Alvin Toffler in Future Shock (1971) and further developed in The Third Wave (Toffler, 1980). The new paradigm was coined “mass customization” by Stanley Davis (1987), but the methods to implement it were later systematized by B. Joseph Pine II (1993). Since then, the concept has evolved. One most commonly cited definition of mass customization is the one by Tseng and Jiao, as “producing goods and services
to meet individual customer’s needs with near mass production efficiency” (Tseng & Jiao, 2001, p. 2). A typical application of the mass customization paradigm is offered by sport shoes and automobiles manufacturers, which enable the customer to customize certain components from a defined set of colors. The color selection can be made online, through an automated configuration toolkit, which provides a visualization of the final product according to the customer’s selection (Fogliatto et al., 2012).

Recently, the paradigm of mass personalization has emerged, in which products can be tailor-made to individual specific requirements. Mass personalization can be considered one step further from mass customization, “whereas both of these strategies are guided by the criterion of product affordability consistent with mass production efficiencies, the former (mass personalization) aims at a market segment of one while the latter (mass customization) at a market segment of few.” (Kumar, 2008, p. 536; Tseng et al., 2010). Also, personalization involves intense communication and interaction between two parties, that is customer and company (Piller, 2014). Examples of mass personalization include manufacturing of prosthetics, which are produced to fit specifically a given patient. Typically, the region of the patient’s body that will receive the prosthetic part is 3D scanned, so that the artificial part fits perfectly (Fiorindo, 2012; Gibson et al., 2010).

Mass customization and mass personalization respond to the Gothenburg protocol (EU Commission, 2001) two-fold: by creating custom-made products and services, and by ensuring their sustainability. On the one hand, mass personalization enables competitiveness by responding to current market demands. Today consumers want to be treated as individuals - each having specific needs. Products are expected to be tailored to individual taste, “built-to-order” but at prices comparable to those of standard mass-produced goods. On the other hand, mass-personalized products are more sustainable than their mass-produced counterparts. In fact, some studies indicate that a product that is adjusted to its user will be discarded later than a mass-produced one, thereby reducing its environmental impact (Diegel et al., 2010).

**Aspects of mass customization**

As a production paradigm, mass customization (MC) combines elements of both craft and mass production. As in craft production, it features a high degree of flexibility in its processes; it builds to order rather than to plan and it results in high levels of variety and personalization. As in mass production, MC generally produces in large quantities, has low unit costs, and may rely on
automated production (Pine, 1993). In times when consumers become ever more demanding, and
differentiation becomes ever more important, a correct implementation of the MC paradigm can
boost both customer satisfaction and profit (Bernard et al., 2012).

Some definitions have been proposed for the concept of MC since its emergence. Coming from a
management background, Piller suggests an alternative definition to the one proposed by Tseng
and Jiao referred to above:

*Customer co-design process of products and services, which meet the needs of
each individual customer with regard to certain product features. All operations
are performed within a fixed solution space, characterized by stable but still
flexible and responsive processes. As a result, the costs associated with
customization allow for a price level that does not imply a switch in an upper
market segment. (Piller, 2004, p. 315)*

Such definition is more suitable for our research because of its focus on co-design, which implies
a joint design process including designer and end-user. Moreover, it constrains the MC paradigm
to a fixed and pre-determined solution space. Although it can be perceived as negative at first,
such a constraint enables the use of generative systems for the customization of design solutions.
Also according to Piller (2004), two factors determine the success of MC from the firm’s point of
view: a) high production flexibility, and b) elicitation of customer preferences. Recently, flexible
manufacturing has been an important research focus in many fields, including MC research and
practice – just consider the commotion around 3d printing, additive manufacturing and digital
fabrication in recent years (Chua et al., 2010; Wohlers & Caffrey, 2014). Customer elicitation,
however, has received less attention from the research community, although that has changed in
the last ten years, judging by the growing number of papers on user interfaces for customer
elicitation, called toolkits or configurators (Franke & Hader, 2014; Heiskala & Tiilinen, 2007;
Hermans, 2012).

For a company, whether or not to adopt and implement a mass customization strategy is
ultimately a business decision, with the intent of maximizing profit and, therefore, the costs of
such an implementation must be taken into account. Therefore, both the cost of providing high
flexibility in manufacturing and the cost of eliciting customer preferences should be considered
(Piller, 2004). Both of these costs are associated with the technologies that enable mass
customization. In fact, the advancements in digital fabrication have brought down the cost of
flexible manufacturing (Piller, 2004), while the pervasiveness of the internet and the development of generative design systems rendered the elicitation of user preferences more accessible.

Nevertheless, these technologies still come with a cost, even if a decreasing one. Digital fabrication technologies like additive manufacturing still entail a higher unit cost than equivalent industrial mass production equipment (Thomas & Gilbert, 2014). On the other hand, although the use of toolkits is suggested to improve a product’s profitability (Myrodia et al., 2017), their development still needs to be paid for. Despite the fact that customers are willing to pay a premium for a customized product when compared to a mass produced equivalent (Schreier, 2006), one of the company’s main concerns should be whether the extra costs induced by adopting those enabling technologies validate a decision of adopting mass customization.

Studies suggest that markets are receptive to MC. Some authors show that consumers look for goods and services that correspond to their particular needs, which are not fulfilled by their mass produced counterparts, and that those consumers are willing to pay a premium for a customized product (Franke & Piller, 2004; Franke & von Hippel, 2003). Although it is unlikely that MC will surpass mass production as the dominant production paradigm, it shows potential beyond just niche markets or novelty items (Piller, 2004). Despite its value, customization by itself is not enough to be a product’s differentiation factor. However, customization can be sought after by consumers as a way to fulfil one or more of three distinct needs: style, dimension, and functionality (Piller, 2004).

Style customization refers to varying a product’s aesthetic or sensorial qualities, from its visual appearance in terms of color or shape to its flavor or texture, for example. Style customization has been the most explored by companies that approach MC, probably because it is the easiest type to implement. However, one can question the value that style customization adds to a product. Studies (Piller, 2004; Tian et al., 2001) have revealed that consumers who look for something truly unique belong to a strict market. Style wise, the average consumer tends to follow a fashion trend.

In dimensional customization, a product varies in its dimensions, and consequently its geometry, according to user’s bodily measures, for instance. This is considered the most pertinent type of customization, but it is also the most difficult to implement, since it implies a great deal of flexibility, both production wise and in terms of user interaction (Piller, 2004). Recent
developments in digital fabrication and in 3D scanning potentially reduce such difficulties, creating new opportunities for MC.

Finally, functionality customization refers to variation of a product’s technical attributes. A typical example is the customization of personal computers, in which users can determine the most suitable amount of RAM or hard drive memory. Not all products are prone to this type of MC, which is the least explored of the three. A successful approach to MC should result from a combination of these three types.

The proposed mass customization of ceramic tableware might be considered belonging to the category of style customization, since it enables users to customize its base shape and relief-based decoration. Arguably, it could also be considered dimensional customization since the shape of tableware elements is being manipulated, and therefore their geometry and dimensions. As the dimension of tableware elements such as a cereal mug or a soup plate determine the amount of food they are able to contain, defining it can be associated with functionality customization for nutrition purposes. In any case, we believe value is being added to ceramic tableware through MC.

One last factor can lead the consumer towards customization, not because of the resulting product, but of the customization process itself. The fact that consumers played an active role in designing their own product is believed to add value to the product to the extent that it leads to increased satisfaction. Nevertheless, despite being instrumental for a successful user interaction, this so-called “I did it myself” effect should not be the main motivation for acquiring a customized product, for its value will wear off along with its novelty (Franke et al., 2009).

The limits of Mass Customization

Let us now take a look at the costs of MC to the consumers, which can be divided into direct and indirect. For the consumer, the direct cost of MC is the additional cost of the custom item when compared to its mass-produced counterpart, which studies show that consumers are willing to pay. The indirect cost is a cognitive one, and derives from the co-design process itself. Mass confusion, as it is commonly referred to in the MC literature, corresponds to the perceived risk and complexity of customizing a product, which might cause users to feel uncertain whether or
not their participation in the co-design process will actually cast a positive effect on the product that has just been bought (Zipkin, 2001).

Mass confusion might result from different factors. One is typically referred to as the burden of choice, that is, to have to choose from a large number of options, which may happen in any customization process. The downfall of having too much to choose from is that the consumer might end up not choosing anything at all. Moreover, consumers might simply not know exactly what they want, or what specifications are required to obtain it. Another cause for mass confusion derives from a potential uncertainty about the supplier’s behavior, namely about whether or not they will deliver the rightly customized product. Such uncertainty is justifiable, since the consumer has never actually seen the product, only its representation on a computer screen, and will only experience the product when it is actually delivered (Piller et al., 2005). Mass confusion can have a more negative effect in the MC experience than the premium cost and, therefore, it is important that such effect is minimized (Piller, 2004).

**Successful examples of Mass Customization in design**

The most successful examples of MC application can be found in consumer goods and their inherent product design. Sport shoes represent a paradigmatic example of a market segment where MC has been successfully applied. A number of brands provide MC services that allow their customers to personalize their sports shoes. This personalization consists of specifying the colors of its components, such as soles, or shoes laces, or adding customized text (see Figure 3). The success of these experiments has been praised throughout MC literature (Berger & Piller, 2003). Nevertheless, it might be argued that choosing component colors is a limited use for MC. In fact, it corresponds to a simple combinatorial schema, which we can call ‘combinatorial customization’.

Another example of MC is presented by US-based studio Nervous System⁵. In a number of projects, customers can customize different fashion items, from rings to pendants to dresses. However, more than choosing from available colors, customers can manipulate the shape of the customized elements. Shape manipulation is possible through interaction with online applications

---

⁵ http://n-e-r-v-o-u-s.com/index.php
using a number of interface elements such as sliders and buttons (see Figure 4). Such interface elements provide a more detailed control of the customization process, namely it allows users to choose any value within a continuous but well-defined range. Such continuous shape customization is supported by parametric models. Therefore, we dub Nervous System’s approach as ‘parametric customization’, as opposed to the combinatorial customization exemplified with sport shoes.

Figure 3 – Combinatorial customization of sport shoes (https://www.adidas.com/)

Both customization solutions are considered collaborative customization approaches since they foster a dialogue between supplier and customer (Gilmore & Pine, 1997). One of the main differences between the two customization paradigms resides in their design space, defined by the set of all possible solutions. In combinatorial customization, the number of solutions is finite, since it results from a combination of a finite number of components with a finite number of colors. On the other hand, in parametric customization, at least one variable is continuous, which results in an infinite number of solutions. Therefore, combinatorial customization corresponds to a discrete design space, whereas parametric customization corresponds to a continuous design space. Nevertheless, both paradigms correspond to a defined design space, in the sense that they cannot generate unexpected design solutions.

---

*https://www.adidas.com/us/mi-stan-smith*
We proceed by introducing an example of an actual application of MC to ceramic tableware. Sake Set Creator (Huang & Hudson, 2013) is a web application in which users can customize the shape of a small set of ceramic vessels, supposedly for pouring and drinking sake, the traditional Japanese drink. The application is hosted in Shapeways\(^7\), a web-based additive manufacturing service that manufactures digital models sent by their customers. In the case of the SSC, the customer is invited to customize the digital model by using sliders and manipulating control vertices, instead of sending the digital models (see Figure 5).

Sake Set Creator is an interesting example of parametric customization, as it has accomplished some of the same objectives of the proposed research: it combines a generative design system with an automated production system, thus enabling mass customization for ceramic objects, similar to tableware. However, the design system is relatively simple, not leaving much room for creativity. Also, the production system relies on Shapeways’ proprietary additive manufacturing system, which renders the sake set quite expensive.

\(^7\) http://www.shapeways.com
Architecture has also been the proving ground of mass customization experiments. We find examples of commercial experiments by architectural offices such as Blu Homes or Resolution 4 Architecture, who provide customization options for housing design based on prefabricated and modular strategies (Kolarevic, 2015). Kieran and Timberlake (2003) argue that such strategies, together with digital fabrication and building information modeling are changing the building industry towards the mass customization approach. On the academic side of architecture, a number of studies focused mass customization in the design of housing solutions (Barlow & Ozaki, 2005; Benros & Duarte, 2009; Duarte, 2005; Granadeiro et al., 2011; Piroozfar & Piller, 2013). One paradigmatic example is Duarte’s application of mass customization to the Malagueira houses of Álvaro Siza (Duarte, 2001). Malagueira was a social housing project in which dwellings were designed according to the needs of their future dwellers. Duarte’s contribution was to automatize part of the design process so that the dwellings could be delivered more quickly to their users. Despite the success of these examples, it is difficult to experiment with MC in architecture, particularly in terms of customizing shape, given the cost of producing buildings.
2.2 Digital Design and Generative Design Systems

Design practice and design research, like any other human activity, are transformed by technological advancements. Particularly in the last two decades, design has been influenced by the emergence of computational methods and tools, giving rise to what can be called Digital Design. Traditional design methodology has been dominated by logic of repetition, justifying standardization and modularity as suitable strategies for producing design in the context of mass production and industrialization. However, the emergence of computational tools that provide the means to deal with complexity enable new forms of doing design, producing more individualized responses to contextual conditions of design problems, which in turn generate new shapes. More than enabling the production of new shapes, digital design proposes meaningful alternatives to the logic of repetition (Oxman, 2006).

Oxman proposes classifying digital design models into five categories: (1) CAD, (2) formation, (3) generative, (4) performance-based, and (5) compound models (see Figure 6). The models demonstrate a successively structured development, in the sense that models from a following class make use of processes or models included in a preceding class. The fifth class relates to compound models that explicitly integrate processes from some or all of the preceding classes. The fourth class corresponds to performance-based models, corresponding to processes of formation or generative models in which performance is the driving motivation. While performance-based models are not applied in our research, the remaining three categories address models that are extensively used. Therefore, this section on digital design is structured according to those classes: CAD models, formation models, which include parametric design, and generative models, which include shape grammars. Finally, we present the precedent of the Digital Alberti project as an example of a successful integration of digital design models, with a number of repercussions in the present research.

**CAD models**

An enabling factor for the emergence of digital design is the development of Computer Aided Design (CAD) tools. CAD tools enable designers to create digital representations of geometric models in a computer. Ivan Sutherland’s Sketchpad (Sutherland, 1964), designed and built in the 1960’s, is often considered as the first ever CAD tool. In the case of the Sketchpad, the CAD tool
consisted in both hardware and software. Nowadays, CAD tools typically consist of software applications that can run in any computer, or at least one complying with the application’s minimum hardware requirements. Oxman distinguishes between descriptive and predictive models.

![Digital design models](adapted from Oxman, 2006)

CAD tools are used in a variety of design fields, from architecture and product design to mechanical engineering. Traditionally, the use of CAD systems followed the CAD descriptive model, being driven by representation of geometry, and enabling designers to describe design objects more accurately, and to perform alterations faster. A number of applications allowed designers to execute various combinations of design activities including two-dimensional drafting, three-dimensional modeling and photorealistic rendering. For the task of modeling three-dimensional objects, different applications use different modeling paradigms. Applications oriented for game development and photo-realistic rendering typically use polygonal and subdivision modeling, which aim at optimizing models in terms of rendering performance, whereas applications oriented for engineering typically apply constructive solid modeling, particularly suited for manufacturing due to its focus on closed volumes. Despite improving productivity in design tasks, CAD tools that fall within the descriptive model are little more than
an extension of the paper-based methods inherited from traditional design practices (Kalay & Mitchell, 2004).

Another application of CAD tools is the execution of evaluative analytic processes on geometric models in terms of cost estimation, structural behavior such as finite element analysis, and environmental performance. This application falls within the generation-evaluation CAD model. While the information provided by these predictive models can be used to modify designs also described by CAD tools towards better performance, the modifications are executed manually by the designer taking into account information provided by analytic tools, rather than through an automatized process.

**Formation models**

Formation models invoke a shift of design activity towards form finding, rather than form making (Kolarevic, 2003b). Such shift in design practice, from static representations to dynamic conceptualizations of shape, is what sets digital design apart from classical design techniques. One factor that contributes to the emergence of formation models is the design community increased interest and a subsequent proficiency in scripting and programming, which allows them to tap into the inner workings of CAD applications and extend their functionalities (Terzidis, 2006).

Nowadays, many CAD applications grant access to their set of functions, so they can be manipulated by the advanced user who looks for creating customized tools. However, access to the interior of the application is usually dependent on the use of programming languages. While some application use well known languages like Python or Processing, others provide their own proprietary language, such as Maya Embedded Language (MEL) or Geometric Description Language (GDL) for ArchiCAD, even if such proprietary languages are based in existing programming languages.

One common disadvantage associated with programming languages is the steep learning curve, particularly for designers who often lack any training in programming whatsoever. As a reaction to this limitation, visual programming languages (VPL) began to pervade CAD tools, such as Generative Components for Microstation, or Grasshopper for Rhinoceros. These languages relieve users from dealing with syntax errors, while at the same time providing feedback in real-
time. For this reason, VPL rapidly gained popularity among the digital design community (Leitão et al., 2012).

Formation models include topological, associative and motion-based design models. Motion-based models take advantage of animation software tools, which are used for form finding by borrowing techniques such as key-frame-animation, forward and inverse kinematics, dynamic force fields and particle emission. Compared to motion-based models, topological and associative models are closer to our research and, therefore, are discussed in the following sections.

**Topological models and freeform surfaces**

Advances in CAD tools have come to empower designers with the ability to manipulate complex shapes provided by topology and non-Euclidean geometry, and giving rise to new design tendencies. In architecture, these tendencies are associated to expressions such as blob architecture, or hypersurface design. The mathematical complexity associated to these new forms can be tackled by resorting to computation (Oxman, 2006).

A good example of such complex shapes is the so-called freeform curves and surfaces. This category includes Bézier, B-splines, and NURBS curves – the former being particular cases of the latter –, as well as their surface counterparts. Despite being popularly referred to as “freeform”, these shapes have a well-defined shape, which is determined by unambiguous mathematical expressions or computational algorithms and, therefore, have an exact graphical representation in computational applications. A considerable contribution came in the 1940’s and 1950’s from Pierre Bézier and Paul de Casteljau, working for car manufacturers Renault and Citroen respectively. Independently, they have developed algorithms for designing curved shapes (Pottmann et al., 2007).

Bézier curves can be considered the most simple of the three. Points in a Bézier curve can be calculated through repeated linear interpolation of its original control points, represented in Figure 7 by $P_n^{(0)}$. A control polygon is obtained by sequentially joining the curve’s control points. B-spline curves are an extension of the Bézier curves. B-splines allow extended control of the curve’s shape by adding knots. NURBS curves are an extension of the B-spline. NURBS allow extended control of the curve’s shape by adding weights / knots.
In the scope of this research, we make use of Bézier curves for describing the profile shape of tableware elements, since moving a reduced number of control points provides relatively simple control of the curved shape. Bézier curves hold some properties regarding their geometric continuity that make them interesting for describing tableware shape. Within the curve itself, geometric continuity is guaranteed up to three levels: positional ($G^0$), tangential ($G^1$) and curvature ($G^2$) (Farin, 2002). Moreover, when combining Bézier curves it is possible to guarantee $G^0$ continuity by coinciding the curves’ endpoints, as well as $G^1$ continuity by aligning the end segments of the control polygon.

Figure 7 – De Casteljau construction (Source: Haui, licensed under CC BY 2.0)

**Associative models and parametric design**

Associative models encompass design strategies in which relationships among design elements are explicitly defined. Such strategies are normally concomitant with parametric design, resulting in dynamic parametric models that generate a multitude of variations. Design variations are then easily manipulated by changing attribute values or manipulating inter-connected elements, although maintaining the topological relationships that have been modeled and characterize the design intention (Oxman, 2006).

Hoffmann and Joan-Arinyo (2002) address parametric modeling from the product design perspective. The main advantage attributed to parametric modeling techniques is enabling designers to define entire families of shapes and not just specific instances. Moreover, the authors classify parametric design strategies according to three categories: (1) variant, (2) constrained-based, and (3) feature-based.
In variant modeling, a generic design is represented by a symbolic system and is associated with a number of parameters that can be assigned values in different ways, generating variant designs. In constraint-based modeling, relations on or between model entities are specified into constraints that must be maintained. Such constraints include geometric relationships such as perpendicularity or concentricity, topological relations such as connectivity, among others. Constraint-based applications use solvers to ensure the constraining relationships defined by the user, allowing for an informed exploration of design alternatives. Finally, feature-based modeling can be considered an elaboration of the previous modeling types, corresponding to higher level vocabulary for specifying operations to create features. Features can be defined as generic shapes with parametrized geometry, to which attributes and constraints are added in order to embody behavior and engineering significance (Hoffmann & Joan-Arinyo, 2002).

In the scope of our research, variant design is the most relevant of the three categories, in which we include the parametric models referred to previously in the Methodology section. In fact, the definition of variant as a model that changes according to parameter values is often applied to the general concept of parametric design. The explicit variation of parametric design lends itself to be formalized into a design space. If parametric design enables the creation of families of designs, then the design space of a parametric model corresponds to the set of every design that the model can generate, thus belonging to that family. The concept of design space is borrowed from a number of authors from Design literature (Mitchell, 1990; Woodbury & Burrow, 2006; Strobbe et al., 2015), who consider design space as the set of possible design alternatives when addressing digital design processes. A more restricted interpretation can be borrowed from computer science literature, in which design space is narrowed down to digital design processes (Saxena & Karsai, 2010), and particularly to parametric models (Talton et al., 2008, 2009). In the work of Talton et al., formalization of the parametric design space enables experimentation with novel design of user interfaces. Similarly, design space exploration plays an important role in our research, namely, in the implementation of the design system and in the exploration of user interfaces for that design system.

**Generative design**

Although formation models can produce design solutions, Oxman (2006) distinguishes them from generative design. In formation models, shape generation varies within conditions of topological
control, whereas in generative design, topology is not necessarily maintained. Also, while formation models are characterized by the definition of geometrical relationships, generative design models result from the formalization of qualities and processes, and the designer is able to interact with the generation mechanism. In addition to relations, also defined in formation models, in generative design the designer manipulate rules and principles. Therefore, generative design is often associated with rule-based design.

Generative design models encompass two main approaches: (1) grammatical transformative models, which include shape grammars, and (2) evolutionary models, which include genetic algorithms. Shape grammars are mathematical expressions for driving shape generation processes through transformational rules. As mentioned, shape grammars serve as the foundation of the design system addressed in our research. Evolutionary form-generation techniques are based on evolutionary models of natural generation that can be applied to generative processes in design, and have not been addressed in our research (Oxman, 2006).

Compared to Oxman, other authors present a broader vision of generative design by including topological and parametric models, which are considered formation models by Oxman. Kolarevic (2003b) defines ‘digital morphogenesis’ as the use of digital media as a generative tool for the derivation and transformation of form, in which “instead of modeling an external form, designers articulate an internal generative logic, which then produces, in an automatic fashion, a range of possibilities from which the designer could choose an appropriate formal proposition for further development” (Kolarevic, 2003a, p. 13).

This apparent lack of consensus in term definition is recurrent in emerging fields, and more so if such fields involve digital technologies, in which change happens at a fast pace. A similar situation happens in the field of digital fabrication, in which the term ‘3d printing’ is recurrently used as a synonym of, and more used than, ‘additive manufacturing’, when in fact the former is one process within a category of digital fabrication processes defined by the latter (Wohlers & Caffrey, 2014).

Although we empathize with Oxman’s differentiation, we adopt the broader definition of generative design in which both parametric modeling and shape grammars are included, given their importance in our research. Therefore, they are part of a generative design system that is able to generate designs in an automatic or semi-automatic fashion, according to data provided by
the system’s user. In the scope of our research, the provided data corresponds to the needs and wants of the final customer.

**Shape grammars**

Shape grammars are generative design mechanisms that were invented by Stiny and Gips (1972), and are considered one of the potentially significant models of generation for digital design (Oxman, 2006).

The formalism for defining shape grammars was further developed by Stiny (1980). A shape grammar can generate designs by articulating four components: a set of geometric shapes that correspond to the design elements, a set of spatial relations that articulate those design elements, a set of if-then rules that define how spatial relations are applied, and an initial shape from which the design process is triggered. For generating designs, shape grammar rules are recursively applied, first to the initial shape, and from then on to the design that results from the previous application of a rule. The final design thus depends on which rules are applied in which state of the design process. A sequence of successive designs that result from the recursive use of shape grammar rules is called a ‘derivation’. The set of all possible final designs is called the ‘language’ of the shape grammar (see Figure 8). The shape grammar formalism allows the use of labels for improved control of design generation. Labels are symbols that can provide additional information about spatial relations, for example for defining a particular side in an otherwise symmetrical shape, or for defining different states of shape development. Another extension to the original shape grammar formalism is the concept of a parametric shape grammar, which generates families of designs rather than single designs, similarly to parametric models. This extension provides an exponential flexibility to the shape grammar formalism.

A particular contribution to shape grammar theory came from Knight (1983), who demonstrated that the transition of one particular style into a different but related one can be explained as a transformation of the grammar underlying the first style into the one underlying the second by adding, subtracting, or transforming rules.

The first architectural shape grammar encoded rules for designing Palladian villas, whose layouts could be generated through iterative application of subdivision and replacement rules (Stiny & Mitchell, 1978). Later on, the Palladian grammar has been revisited (Benrós et al., 2012), in an
effort to use its rules to encode other architectural styles for housing design (Benrós et al., 2014). Since the original Palladian grammar, a number of other architectural styles have been addressed through the shape grammar approach (Koning & Eizenberg, 1981; Flemming, 1987; Li, 2001a; Eloy & Duarte, 2014). In product design, we can find a number of rule-based design methodologies including the application of shape grammars to various product types: furniture (Knight, 1980; Barros et al., 2011), vehicles (Pugliese & Cagan, 2002; McCormack et al., 2004; Orsborn et al., 2006), among others (Agarwal & Cagan, 1998; Chau et al., 2004).

![A simple shape grammar that inscribes squares in squares](image1)

**Figure 8 – Example of a shape grammar (adapted from Stiny, 1980, p. 348)**

Despite the considerable number of studies on shape grammars, most of them focus on rectilinear shapes (Chau et al., 2004). However, designing tableware like dishes and mugs is likely to require the use of curved geometry. Chau applied the shape grammar formalism in his study on consumer products, such as soda and shampoo bottles, whose shapes are described by circumference arcs. Jowers and Earl (2010) extended the shape grammar formalism to include all types of curved shapes, focusing on the geometric aspects necessary for a precise computational manipulation of such shapes.
Generic grammars

More recently, some researchers have been focusing on the distinction between specific and generic shape grammars (Duarte, 2011; Beirão et al., 2011; Benrós et al., 2014). The concept of generic grammars is supported by the idea that through abstracting a specific design, or a set of designs of the same type, one can infer a specific grammar that encodes such a type. It is suggested that, analogously, a generic grammar can be inferred from abstracting other specific grammars. Subsequently, the generic grammar can be used to define a new specific grammar, which in turn can be used to generate new designs (see Figure 9).

![Diagram of coding processes for grammars and types](adapted from Duarte, 2011)

Figure 9 – Coding processes for grammars and types (adapted from Duarte, 2011)

For a better understanding of how generic and specific grammars function, let us look at research in which generic shape grammars are applied to housing design (Benrós et al., 2014). A generic shape grammar has been inferred by analyzing three well-known examples from shape grammar literature, all of which in the realm of housing: the Palladian grammar (Stiny & Mitchell, 1978), the Prairie house grammar (Koning & Eizenberg, 1981), and the Malagueira house grammar (Duarte, 2001). By coordinating differences and similarities among the studied specific grammars, the resulting generic shape grammar encodes three different house styles. A similar approach has been followed in the development of the design system for ceramic tableware.

The generic grammar paradigm has also been applied in generative design tools for urban design in the scope of a the City Induction research project (Beirão et al., 2011, 2012). These tools aim at facilitating the urban design process through the use of shape grammars, design patterns and parameters. Especially interesting is the application of the concept of design pattern, borrowed from Christopher Alexander et al (1977), as well as its subsequent application to computer programming (Gamma et al., 1994). In this case, rather than encoding different styles, specific grammars encode different aspect of urban design, such as street networks or building block
geometry. Furthermore, the methodology of generic urban grammars has been used to analyze existing urban plans beyond the scope of the City Induction project (Mendes et al., 2013; Mendes, Celani, et al., 2014).

Specific and generic shape grammars have also been applied to the design of furniture, namely chairs. Barros focused on mass customization for Thonet chairs (Barros et al., 2011; Barros, 2015), having developed a design system supported on shape grammars, and focusing on design manufacturing and evaluation. This research thus focuses on a very particular chair design style, encoded into a specific grammar. Later on, a generic grammar was inferred from the Thonet chair grammar, along with a set of other specific shape grammars for chair design (Garcia & Barros, 2015).

The precedent of Digital Alberti

A connection can be established between generative design tools, a product of the contemporary digital age, and the work of Leon Batista Alberti. The Italian architect marked the beginning of the Renaissance period, in particular with his treatise on the art of building, De re aedificatoria, in which rules are prescribed for designing elements of classical architecture, namely column systems, referring to what is called in other architecture treatises as the classical orders (Serlio, 1982; Vignola & Watkin, 2012), and rectangular temples.

One virtue of Alberti’s rules is their algorithmic precision. The design of columns is described as a step-by-step recipe on how to design each of its elements and how to compose them. Alberti begins by deconstructing the column system into the column itself and the entablature. Each of these parts is then further deconstructed into smaller elements, thus defining a hierarchical structure for the column system. The column itself comprises a base, a shaft (which Alberti actually calls “column” again), and a capital. On the other hand, the entablature is composed by the architrave, the frieze, and the cornice. In a lower hierarchical level, we find the composition of elements like the capitals, which can belong to different styles such as Doric, Ionic, Corinthian or Composite. Alberti deconstructs the column system recursively into deeper detail, into an elementary vocabulary comprising the simplest possible shapes, from which classical molds can be delineated (Coutinho et al., 2011) (see Figure 10). In his treatise, Alberti is equally systematic in describing the design of rectangular temples (Figueiredo et al., 2013).
This deconstruction exercise is not always possible in Architecture. Architectural design is generally considered an example of creative design, given its complexity as well as the ever-changing conditions inherent to a design activity (Jones, 1970; Gero, 1990). However, rather than a creative design problem, the particular case of Alberti’s column system can be considered a routine design problem, since the solution space for the design of classical columns is in fact well defined. It has been studied for centuries, at least since Vitruvius in the first century BC, and used before him by the ancient Greeks. Also, such design space comprises a finite number of possible solutions, corresponding to the number of classical orders or, in Alberti’s case, to all possible combinations of column elements of varying styles. Therefore, as the Albertian column system is not ill-defined, it can be deconstructed, and it can be computable.

![Image](Figure 10 - Elemental shapes of the Albertian column system (left), combined into classical molds (Castro e Costa, 2012)]

Mass customization shares this characteristic with the Albertian column system. As a design problem, MC falls under the category of well-defined problems, since according to the definition of MC presented earlier, “all operations are performed within a fixed solution space, characterized by stable but still flexible and responsive processes” (Piller, 2004, p. 315).

Alberti’s algorithmic precision in identifying the rules behind classical architecture makes it possible to accurately replicate his designs. All we must do is to follow his instructions, obtaining solutions that agree with the compositional principles and proportions Alberti defined. This exercise was performed in the scope of the Digital Alberti research project (Duarte et al., 2011, 2013; Krüger et al., 2011). The main objective of Digital Alberti was to determine Albertian influence on Portuguese churches in the Counter-Reform period (Krüger et al., 2011), namely by identifying the extent to which the Alberti’s design rules could account for the generation of Portuguese churches in that period. The fundamental theoretical reference was the translation of Alberti’s treatise into Portuguese by Santo & Krüger (Alberti, 2011). Despite focusing on such a classical subject, the project’s methodology was extensively supported by computational
technologies, namely shape grammars, parametric modeling and digital fabrication (Duarte et al., 2011). As part of the study, Alberti’s design rules were encoded into two shape grammars, one corresponding to the column system (Quaresma, 2014) (henceforth referred to as the Column Grammar), and another corresponding to the rectangular temples (Figueiredo, 2016) (henceforth referred to as the Temple Grammar).

Following the development of the shape grammars, these were converted into parametric models implemented in Grasshopper (GH), which generated digital models of columns and churches according to Alberti. These parametric models allowed testing and debugging the shape grammars as they were being developed, through semi-automatic application of rules. These parametric models comprise a sequence of operations that correspond to the application of grammar rules in a shape grammar derivation, as shown in Figure 11 for the Temple Grammar.

![Figure 11 – Partial derivation tree of the Temple Grammar (Figueiredo et al., 2013)](image)

Although the implemented parametric models do not perform shape recognition, they enable the manipulation of individually implemented grammar rules. Adopting a modularization strategy, grammar rules were translated into GH components. In the absence of a shape recognition mechanism, it was left to the (human) designer to recognize when and where to apply the rules. By manipulating the GH components that corresponded to rules enabled one to derive designs according to the grammar (Figueiredo et al., 2013).

The same approach was adopted for the tableware design system, by mapping rules of the tableware shape grammar to modules of the parametric rule implementation that executed corresponding operations.
One of the applications of the Albertian parametric models was supporting the implementation of a prototypical Interactive Tabletop, in which different interaction techniques were combined into a semi-immersive virtual reality pedagogical experience (Mendes, Araújo, et al., 2014). In this prototype, users could experiment with the Albertian rule system, namely by manipulating the shape of an Albertian temple, both in terms of its general shape, such as the number of chapels or the shape of their plan and ceiling, but also the shape of the temple’s columns. The contents of the Interactive Tabletop prototype are considered a simplification of the developed parametric models, which were too complex to be manipulated by the general public (see Figure 12) and might result in cognitive overload, causing lack of interest and focus. In fact, it was not the parametric model itself that was being manipulated, but a combination of digital 3D models that had been pre-generated from the actual parametric model (Figueiredo et al., 2014).

Having non-expert users deal with complex models can lead to phenomena of mass confusion and therefore such scenarios must be carefully addressed. Consequently, the strategy used in Digital Alberti of presenting a simplified version of an otherwise complex model is an important reference in the present research.

Figure 12 – Grasshopper file containing all Digital Alberti parametric models
2.3 Ceramic production and digital fabrication

In this section, we will focus on the production of ceramic objects, namely tableware. We take a look into the traditional and industrial techniques, as well as cover the latest technology in digital fabrication, particularly additive manufacturing, also known as 3D printing.

Figure 13 – Typical ceramic manufacturing process (adapted from Hamilton, 1974)

Ceramics manufacturing process

Either in handcrafted or industrial manufacturing, from simple vases to geometrically complex and intricately decorated sculptures, the making of a ceramic piece goes through similar steps (see Figure 13): after preparing the clay, and while it remains in its plastic state, the ceramic object is given its shape, as well as any three-dimensional decoration, such as relief-based decoration. The object is then dried and fired, resulting in what is called the biscuit. The biscuit is then covered with glaze, which grants it the shiny look and smooth texture that is typical of tableware ceramics. If the piece is to be painted or decaled, such two-dimensional decoration is applied before the glazing. After glazing, the ceramic object is fired one more time. For plain ceramic tableware, without any decoration, the manufacturing process would stop here.
Alternatively, the piece may be given on-glaze decoration, after which it must be fired one last time.

Since the objective of the proposed mass customization system is to manipulate the three-dimensional shape of the tableware elements, 2D-decoration after biscuit firing and on-glaze decoration will not be addressed, and we will focus our attention on the forming or modeling phase.

**Handicraft ceramic modeling**

A number of traditional techniques can be used for modeling ceramic objects, namely tableware elements. Some are purely manual, while others make use of mechanisms such as the potter’s wheel, or of molds that lend their inverted shape to the ceramic object.

Hand methods of clay forming include pinching, coiling, slabbing, and throwing (see Figure 14). Pinching consists of modeling a piece of clay by pressing the fingers onto a ball of clay. This technique is typically used to make small pots, called ‘thumb pots’ or ‘pinched pots’. Coiling consists of making coils out of clay, and stacking them on top of each other. The surface of an object made by coiling is revealing of the process used, and is typically smoothed by hand or using proper utensils. Slabbing, or slab building, consists of flattening clay on a flat surface. Such a technique is typically used for making flat-faced forms such as boxes. However, it can also originate other shapes, such as cylinders. Throwing is probably the most recognizable of the handicraft shaping techniques, in which hand pressure is applied on a ceramic piece that is rotating on the potter’s wheel (Hamilton, 1974).

![Figure 14 - Handicraft ceramic modeling techniques, from left to right: pinching (Hamilton, 1974), coiling, throwing (Quinn, 2007), and slab building (Clay Pottery Slab Building, 2008)](image)

Molding methods of clay forming include pressing and slip casting (see Figure 15). These methods make use of plaster molds that hold the negative shape of the desired object. Pressing
consists of manually pressing clay in its plastic state against the mold, so its shape is transferred to the clay. Pressing can also be done in a mechanical press, usually for producing tiles. In slipcasting, a suspension of clay in water, the slip, is poured inside the mold. Because the mold is porous, it will absorb the slip’s water, depositing a layer of plastic clay against the sides of the mold. When applying such molding methods, one should make sure there are no locks, meaning that it will be possible to remove the piece from the mold after it dried (Hamilton, 1974).

![Figure 15](image1.jpg)

**Figure 15** – Mold-based traditional techniques, from left to right: pressing by hand, by fly press (Hamilton, 1974), and slipcasting (Quinn, 2007)

![Figure 16](image2.jpg)

**Figure 16** – Mold-based traditional techniques, from left to right: sledging, plaster turning, jiggering and jollying (Quinn, 2007)

Other molding methods include sledging, plaster turning, and jiggering and jollying (see Figure 16). The first two methods are used for shaping plaster, which can be subsequently used as a mold for shaping the ceramic object. Sledging, or profiling, consists of applying a hard profile along the length, or the sides, of a piece of plaster. Plaster turning is also a plaster shaping technique, in which a plaster cylinder is rotated in a lathe, and the material is removed until it reaches the desired shape. Jiggering and jollying consists of forcing a piece of clay to shape against a plaster mold through rotation, similarly to throwing on the potter’s wheel. This
technique is suitable for producing revolution-based open ceramic pieces, such as circular plates in the case of jiggering and bowls in the case of jollying (Quinn, 2007).

**Industrial ceramic modeling**

Some of these traditional techniques were adapted into an industrial context, boosting their productivity (see Figure 17).

![Industrial techniques, from left to right: roller making, industrial slipcasting (Lippert GmbH, 2014), ram pressing (RAM Products, Inc., 2014), and isostatic pressing (SACMI IMOLA S.C., 2014)](image)

Roller machines make use of a heated metallic mold and a plaster die, which rotate while pressing a piece of clay into the desired shape. Due to the rotating movement, this technique is typically used to produce round ceramic objects, at least on the side of the metallic mold. Roller machines allow for low production costs per unit, making them ideal for producing large batches (Rado, 1988). The rolling process is similar to applying jiggering and jollying simultaneously on a ceramic object.

Industrial slip casting is not very different from the traditional handicraft version. Mechanized slip casting consists of automatically filling a number of molds that are placed on a conveyor belt, which slowly transports the molds until they are emptied of the excess slip. The thickness of the cast pieces depends on the time between filling and emptying. Of all industrial techniques, slip casting is the one that enables greater freedom of shape, allowing for the production of closed hollow objects, for example. However, complex shapes imply the use of several splits to remove the piece from the mold, which renders slip casting less profitable than other methods. Therefore, such technique is typically used for smaller production batches (Rado, 1988).
Plastic pressing, or “ram” pressing, consists of shaping a piece of clay between two permeable plaster dies that are pressed together. The dies’ permeability allows for air to be blown through, causing the release of the shaped piece. This technique is typically used for shaping non-round objects, which would otherwise be produced in roller machines (Rado, 1988). Such shapes, nevertheless, must allow for the vertical movement of the dies. Therefore, in terms of shape, pressing is more flexible than roller making, but less than slip casting.

Isostatic pressing, or dust pressing, is similar to ram pressing, since two dies are used to shape the piece. However, in this technique, the ceramic material is not in its plastic state, but in dry grains. Such approach has been used first for shaping ceramic tiles, and it has been adapted later on for producing plates. Isostatic pressing makes use of expensive hardware, such as pressing machines, the metallic molds and plastic membranes. However, these tools have a longer lifetime than those made of plaster used in ram pressing, which makes them more economically viable for large batches. Also, isostatic pressing does not generate waste, and dispenses drying. Nevertheless, in terms of shape, it shares its limitations with plastic casting (Rado, 1988).

These industrial methods are designed for mass production of ceramic objects. Slip casting, for example, which is typically used for producing smaller batches, is considered to be profitable for a minimum of a few hundred pieces sharing the same mold. Such quantity is still too large for mass personalization. Even if a customer is buying a large dinner set, it is unlikely that it will contain more than 24 elements of the same type, and some of them, such as trays or tureens, will be no more than three. As such, for producing such small batches, we should look into alternative manufacturing techniques.

**Digital fabrication**

‘Digital fabrication’ is used in this dissertation as an umbrella term that encompasses technologies and methodologies that make use of computers to automatize the manufacturing of objects. Digital fabrication has been hailed as the new industrial revolution (The Economist, 2012), empowering the common individual to produce objects, enabling him or her to forgo the intermediacy of factories or craftsmen (Gershenfeld, 2012). In the same way generative design enables the democratization of design, digital fabrication enables the democratization of production (Naboni & Paoletti, 2015).
Digital fabrication can be considered an evolution of rapid prototyping, which emerges within the fields of mechanical and production engineering in the 1980s, being defined as a product development process in which prototypes are manufactured faster and more accurately by Computer Numerical Control (CNC) machine tools. The development of rapid prototyping is related to the declining cost of computers, which spurred the advancement of areas like Computer-Aided Design (CAD), Computer Aided Manufacturing (CAM) and CNC machine tools (Chua et al., 2010).

Initially adopted by the aerospace, automotive, biomedical and consumer product industries, rapid prototyping and digital fabrication processes were later applied in the field of Architecture. These technologies enable the materialization of complex geometry, such as non-Euclidean shapes, present in iconic buildings such as Frank Gehry’s Guggenheim Museum in Bilbao, or the Sagrada Familia cathedral in Barcelona (Kolarevic, 2001).

Nowadays, digital fabrication is known by and available to the general public, due to a number of factors. The first one was the dramatic cost reduction in the early 2000s of prototyping equipment such as laser cutters and 3D printers. This cost reduction allowed the creation of a standardized low-cost fabrication lab (FabLab), of which the MIT’s Center for Bits and Atoms is the first example (Gershenfeld, 2005). The FabLab model would be deployed by its inventors and replicated in community centers and universities all over the world, bringing digital fabrication to academic communities but also to the general public. One consequence was the revitalization of the do-it-yourself DIY community formed by enthusiasts of making objects by themselves, reflected in the creation of publications like MAKE magazine and of events like the Maker Faire (Blikstein, 2013).

**Terminology**

Despite its present relevance, the definition of ‘digital fabrication’ is not consensual. In fact, the term is used interchangeably with a number of other terms and expressions such as ‘computer aided manufacturing (CAM)’, ‘computer numerical control (CNC) machining’, ‘rapid prototyping’ and ‘3D printing’, not only in commercial and marketing publications but also in academic literature, which leads to confusion about their meanings.
Pupo el al. (2009) tried to standardize the terminology surrounding the use of digital technologies for making models, prototypes, and buildings or building parts in the architectural field by proposing a number of definitions and categorizations, which we adopt to some extent. For example, they adopt the Volpato’s (2007) definition of ‘rapid prototyping’ “to describe technologies in which prototypes are produced specifically by the automated deposition of layers of material, which vary from liquid to solid”. Also, Pupo el al. include it in the broader term ‘digital prototyping’, which encompasses techniques for making models that include prototypes being defined as a special type of models. On the other hand, ‘digital fabrication’ is distinguished from digital prototyping regarding its purpose since it is used for producing building-parts or entire buildings, or end-products as opposed to prototypes in general. Finally, Pupo el al. propose ‘digital materialization’ as the topmost hierarchical concept that encompasses both digital prototyping and digital fabrication.

Although we agree with the distinction made by Pupo el al. between producing prototypes and end-products, we will not use some of the proposed terms, particularly ‘digital materialization’ as it has not been adopted in academic and commercial publications. Alternatively, we use a terminology that aligns with the current literature, suggesting digital fabrication as the topmost concept, which encompasses digital prototyping for making prototypes and digital manufacturing for making end-products (see Figure 18).

![Figure 18 - Terminology suggested by Pupo et al. (2009) (left) and the one adopted in this dissertation (right)](image)

**Processes**

Digital fabrication processes can be classified into additive, subtractive, and formative (Pupo et al., 2009; Chua et al., 2010).

Subtractive processes were the first to be introduced in what would later be called digital fabrication. The first CNC machine was invented at MIT in the 1950s when, for the first time, a
milling machine was connected to a computer that controlled it (Gershenfeld, 2012; Institute Archives, MIT Libraries, 2014). In subtractive processes, material is removed from a block that is larger than the final shape. Subtractive digital fabrication technology includes machining processes, such as milling and lathing, in which material is removed incrementally by a rotary cutting tool. In lathing, the object rotates around one axis typically at high speed while the tool removes the material, thereby creating objects with axial symmetry, whereas in milling the part is typically still, or moving at a slow speed. Subtractive digital fabrication technology also includes cutting technologies such as laser cutting, blade cutting, plasma cutting and hot-wire cutting (Pupo et al., 2009). Laser, water-jet and plasma cutting are considered two-dimensional processes, since they are typically used for cutting planar sheets or blocks of material using a perpendicular beam or jet. Hot-wire cutting on the other hand is able cut blocks of material along ruled surfaces through rotation of the hot-wire and, therefore, it can be considered a three-dimensional process.

In formative processes, forces are applied to change the shape of the material. Formative processes, like bending, extrusion, thermoforming, and molding, can be further categorized according to the type of forces applied, such as mechanical forces or electro-magnetic forces, the plasticity of the material when subjected to deformation, and the use of molds (Pupo et al., 2009). Under the scope of digital fabrication, we should consider CNC formative processes, which include incremental forming and reconfigurable pin tooling. Incremental forming (Hagan & Jeswiet, 2003) has evolved from spinning and shearing, and it can be used to manipulate the three-dimensional shape of metal sheets using a roller to gradually expand the metal surface. Discrete element-based tooling include reconfigurable pin tooling (Walczyk & Hardt, 1998), in which a matrix of moveable die pins is used to stamp mold a three-dimensional shape; and profiled edge lamination, in which the shape is defined by a series of assembled profiles where a laminate displays a beveled top part (Barros, 2015).

**Additive processes**

Additive processes have been responsible for bringing digital fabrication to the knowledge of the general public (Gershenfeld, 2012). Additive processes in digital fabrication are often referred to as ‘additive manufacturing’, which is the standard industrial term. According to ASTM International Committee F42 on Additive Manufacturing Technologies (2012), additive manufacturing (AM) is defined as “the process of joining materials to make objects from 3D
model data, usually layer upon layer, as opposed to subtractive manufacturing methods”. The term “3d printing” is also commonly used to refer to those technologies. In fact, a Google search for “3d printing” produces more results than one for “additive manufacturing” (Wohlers & Caffrey, 2014). Nevertheless, for the purpose of this research, the standard industrial term ‘additive manufacturing’ is used.

Recent developments in AM technology are enabling the adoption of mass customization and mass personalization strategies (Da Silveira et al., 2001; Hu, 2013, p. 6; Piller, 2004; Reeves et al., 2011). AM is revolutionizing the manufacturing industry by enabling the rapid, flexible, and cost-efficient design and production of products across different applications and industries (Wohlers, 2014), while featuring some important characteristics:

- Additive manufacturing is sustainable, since it produces much less waste than, for instance, subtractive manufacturing techniques (Diegel et al., 2010). Also, additive manufacturing allows for a product to be manufactured virtually anywhere, as long as there is raw material available, since design is shifting to digital data that can be accessed anywhere in the world in a matter of seconds. This reduces the product’s transportation costs, as well as its environmental impact;
- Additive manufacturing offers complexity for free (Hague et al., 2003), as there is no increase of cost when comparing the additive manufacturing of a simple component with a complex one (see Figure 19). Therefore, AM technology can be used to produce almost any shape;
- Additive manufacturing also offers individuality for free (Dillenburger & Hansmeyer, 2013), since the cost of manufacturing a batch of products that are different from each other is the same as if they were equal (see Figure 19). In this case, cost depends solely on quantity of material, not on variation of shape.

Figure 19 – Additive manufacturing: complexity and individuality for free (Aghassi & Witzel, 2014)
According to ASTM F42 Committee (2012), additive manufacturing processes can be classified into seven categories:

**Material extrusion**—an additive manufacturing process in which material is selectively dispensed through a nozzle or orifice. (ASTM F42 Committee, 2012).

Typically, this process uses a thermoplastic material, which is melted in the extrusion head and deposited on the build plate. Relative translation between the extrusion head and the build plate result in producing one layer of the part being manufactured. Material extrusion includes techniques such as Fused Deposition Modeling (FDM) developed by Stratasys. Material extrusion is also the process used by various low-cost equipment commonly referred to as 3D-printers. Some material extrusion machines, namely FDM, use dual extrusion, which allows extruding a secondary material, used for supporting the modeling material in particular situations such as building overhanging parts (Wohlers & Caffrey, 2014).

**Material jetting**—an additive manufacturing process in which droplets of build material are selectively deposited. (ASTM F42 Committee, 2012)

Typically, the materials used in this process are photopolymers or wax-like materials, which are dispensed in droplets using inkjet printing heads, in a process similarly to inkjet printing on paper. Like material extrusion, this process often uses two materials, one for building the object and another for building supporting structures (Wohlers & Caffrey, 2014). Examples of material jetting systems include Multi-jet modeling (MJM) and ModelMaker, developed by 3D Systems and Solidscape respectively (Chua et al., 2010).

**Binder jetting**—an additive manufacturing process in which a liquid bonding agent is selectively deposited to join powder materials. (ASTM F42 Committee, 2012)

Similarly to material jetting, binder jetting uses inkjet printing heads. However, the printing head dispenses the binder that aggregates a powder material, instead of the material itself. The material is transferred from a vat to the printing bed and rolled into an even, thin layer. Existing binder
jetting systems use different materials, namely plaster-, acrylate-, sand-, ceramic- and metal-based powders, and water- and acrylate-based binders. The first binderjetting process was developed in MIT in the 1990s, having been commercialized by ZCorporation, which was later acquired by 3D Systems. The process used by ZCorporation’s machines was curiously named 3D Printing (3DP), long before the media and the general public was exposed to additive manufacturing and began addressing it as ‘3d printing’ (Wohlers & Caffrey, 2014). Objects produced through binder jetting forgo building supporting structures, since the material that is not bonded acts as support material.

“**Sheet lamination**—*an additive manufacturing process in which sheets of material are bonded to form an object.*” *(ASTM F42 Committee, 2012)*

The most paradigmatic material used in sheet lamination is paper. Sheet lamination was originally developed by Helisys, who commercialized the process as Laminated Object Manufacturing (LOM). LOM used adhesive-coated craft paper to produce objects similar to plywood. Presently, Mcor Technologies11 commercializes machines that use standard A4 paper as their raw material. Each sheet is cut by a blade according to the shape of the object’s section and bonded to the previous layer (Wohlers & Caffrey, 2014). Color models can be produced by using regular color inkjet printing on the used paper sheets. Objects produced through sheet lamination forgo building supporting structures, since the remaining material that results from the cut acts as support material.

“**Vat photopolymerization**—*an additive manufacturing process in which liquid photopolymer in a vat is selectively cured by light-activated polymerization.*” *(ASTM F42 Committee, 2012)*

Stereolithography Apparatus (SLA) was the first patented additive manufacturing system, commercialized by 3D Systems in the 1980’s. SLA produces objects from photo-curable liquid resin. A laser beam then hardens the resin according to a two-dimensional section, in a similar process to binder jetting (Chua et al., 2010). Other suppliers of similar systems use a lamp, light emitting diodes (LEDs) and digital light processing (DLP) technology as an energy source to cure the liquid resin (Wohlers & Caffrey, 2014). Presently, some machines using this process, such as

---

Form1 from Form Labs\textsuperscript{12}, are marketed as personal 3d printers, given their reduced cost and ease of use. Objects produced through vat photopolymerization might need supporting structures, since the liquid material is unable to provide support.

\textit{“Powder bed fusion—an additive manufacturing process in which thermal energy selectively fuses regions of a powder bed.” (ASTM F42 Committee, 2012)}

In powder bed fusion processes, powder grains are fused together into a final shape by a thermal energy source such as a laser or an electron beam. Typical materials used in these processes are polymers and metals. Similarly to binder jetting processes, the powder material is transferred from a vat to the printing bed and rolled into an even, thin layer. Powder bed fusion processes include selective laser sintering (SLS) and electron beam melting (EBM), among others. The need for support materials depends on the material used. With polymers, the unfused material acts as support material for the material that has been fused into the final part. On the other hand, parts built with metal materials, which have higher melting points, typically require building supporting structures (Wohlers & Caffrey, 2014).

\textit{“Directed energy deposition—an additive manufacturing process in which focused thermal energy is used to fuse materials by melting as the material is being deposited.” (ASTM F42 Committee, 2012)}

Similarly to powder bed fusion, in direct energy deposition (DED), powder grains are fused together into a final shape by a thermal energy source such as a laser. However, the material, which is typically a metal powder, is being deposited or projected by the tool head, which is coupled with the energy source, rather than lying on a bed. Since the same tool head delivers both material and energy source, DED systems can use 4- or 5-axis motion systems or robotic arms, characterized by more degrees of freedom when compared to systems in the remaining categories, which extends the system’s capabilities beyond printing horizontal layers, making it suitable for adding material to an existing part (Wohlers & Caffrey, 2014).

Table 1 summarizes the categorization of digital fabrication processes adopted in this research.

\textsuperscript{12} https://formlabs.com/
Table 1 – Categorization of digital fabrication processes

<table>
<thead>
<tr>
<th>Digital fabrication</th>
<th>Subtractive</th>
<th>Formative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additive</td>
<td>Material extrusion</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Material jetting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Binder jetting</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sheet lamination</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vat photopolymerization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Powder bed fusion</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Directed energy deposition</td>
<td></td>
</tr>
</tbody>
</table>

**Digital fabrication workflow**

Generally speaking, every digital fabrication (DF) technique follows a similar set of steps. We adopt the process chain described by Chau et al. (2010), which encompasses five steps: 3D modeling, data conversion and transmission, checking and preparing, building and postprocessing. Although the authors describe this process chain for Rapid Prototyping techniques, which comprise mostly additive manufacturing processes, we consider that the process chain can be applied to most subtractive and formative processes.

**3D modeling:** Digital fabrication uses computers for manufacturing objects. Therefore, for a computer to understand the shape of an object, such shape needs to be formally described as a digital 3D model, which can be created using CAD tools. Moreover, for that object to be manufactured, the model must correspond to a closed volume (Chua et al., 2010). While this condition is easily guaranteed when using solid modeling applications, users of surface or polygon modelers need to ensure that the model is devoid of surface discontinuities. Fortunately, most modern CAD applications provide tools that either identify or even correct such discontinuities (Gibson et al., 2009).

**Data conversion and transmission:** 3D models can be produced through a multitude of modeling applications and paradigms, consequently being encoded according to a number of different data structures and saved in a number of different file formats. In contrast, DF software almost exclusively uses a single standard file format, STL, which stands for STereoLithography, the first AM technique to be developed. Therefore, the 3D model resulting from the previous step
needs to be converted into STL format. Presently, most modeling applications provide algorithms for converting a model into an STL, which implies translating the object’s shape to a polygonal model, composed of information about the location of its vertices and the order by which they are related. Information about the object’s materiality, such as colors or textures, as well as more detailed geometric information contained for example in NURBS geometry encoded in IGES format, is lost in the translation.

Checking and preparing: After the automatic conversion of the 3D model into STL, the resulting model can be read by a DF application, responsible for processing the model before submitting it to the intended DF machine. These applications are usually designed by the hardware manufacturer, and therefore take into account the specificities of the machine and process they are associated with. DF applications typically perform a number of tasks, such as generating support structures, decomposing the 3D model into 2D layers, decomposing 2D layers into tool paths, among others. The result of each of these tasks is influenced by a number of parameter values, which are set by the user via the application’s settings interface. Depending on the process and the machine used, these interfaces can be as simple as having the user choose a layer thickness, which will have an impact the number of layers and subsequently on the object’s resolution; or more elaborate interfaces in which the user define parameters for each of the different tool heads used throughout the building process.

Not all processes require support structures, and they are only necessary for particular design situations in the 3D model, such as overhanging parts. A typical feature of DF applications is the ability to identify situations in which support structures are needed and to generate those structures. Whether or not the resulting 3D model features supporting structures, it must be further translated into a set of instructions that can be understood by the machine. Such instructions represent actions that the machine needs to perform, such as moving the tool head from point A to point B, or extruding material X or Y with a certain velocity.

In order to calculate such detailed actions, the 3D model must be deconstructed into simpler components which typically correspond to 2D horizontal cross-sections through a process commonly referred to as slicing. In additive manufacturing processes, such cross-sections have a direct correspondence to its consecutive layers. Nevertheless, the same process is used in subtractive and formative processes such as CNC milling or incremental forming.
The resulting horizontal layers are further processed using different approaches according to the DF process being used. For example, in 3DPrinting, the cross-section is sufficient to inform the region of the powder bed that should be printed for that particular layer. In addition, in processes such as material extrusion, sheet lamination or CNC milling, the cross-section region must be further deconstructed into toolpaths, which correspond to sets of lines along which the tool should move in order to produce the object. DF applications that support these processes use proprietary algorithms to calculate such tool paths, which can be differentiated into boundary, infill or other types of lines, depending on the process. Naturally, deconstruction of the 3D model also addresses any necessary support structures that might have been generated either by the user or by the DF application.

Deconstructing the 3D model into simpler components enables the algorithms to generate information that can be transmitted to the DF machine, but also to check for any errors in the 3D model provided by the user, as well as allow the user to visualize the resulting regions or toolpaths, thus evaluating the fabrication process beforehand.

**Building:** in many DF processes, namely in AM, the building process is automated. In fact, it is common for users to leave DF equipment on to build an object overnight, since some processes may take several hours to perform, depending on the object’s size and resolution.

**Post-processing:** As opposed to the previous step related to automated building, post-processing is generally composed of manual tasks, during which the risk of damaging the built object is relatively high. This step includes cleaning, post-curing and finishing, depending on the process used (Chua et al., 2010). In 3D-Printing, for example, cleaning entails removing the object from the build chamber and removing the unused material that is supporting the built object inside. In processes that might require support structures, cleaning includes removing these support structures. Post-curing is needed only for liquid-based techniques such as Stereolithography, in which pockets of liquid might be embedded within the part. Finishing includes processes such as sanding, hardening and painting, used to improve the surface quality or aesthetic appearance of the fabricated object.
2.4 Chapter conclusion

The purpose of reviewing the main themes that are addressed in this dissertation was to introduce concepts that are recurrently applied along our research.

In the section about mass customization we have addressed its definition and the context in which such apparently paradoxical concept emerged. We focused on enabling factors, costs and added value. We also addressed the challenges of mass customization, namely the cognitive barrier of mass confusion. Finally, we present examples that illustrate the distinction between combinatorial and parametric customization.

In the section about digital design we have addressed the classification of models in digital design, with a particular emphasis in contextualizing parametric modeling and shape grammars, the two techniques on which the design system for mass customization is supported. We also addressed the precedent of the Digital Alberti project, which has influenced some components of our research, in particular the computer implementation of the design system.

In the section about manufacturing systems we have recalled the main techniques for both handcrafted and industrially produced ceramic objects. We have also addressed the application of digital fabrication to ceramic objects, with a particular focus on additive manufacturing and the seven categories in which AM techniques can be classified.

With this revision of the state-of-the-art, we believe to have prepared the reader to better understand our research, which is presented in the following chapters.
Chapter 3
Encoding tableware design rules in shape grammars

The proposed mass customization system aims at enabling the user to manipulate the shape of ceramic tableware collections. Designing a tableware collection presents a number of particular challenges, namely to non-designers, such as manipulating the shape of tableware elements, typically represented by double curved geometry, or guaranteeing stylistic coherence. Such premise implies the existence of a design system, which has been previously defined as a mechanism that can generate designs in a more or less automatic fashion. This chapter focuses on the development of this design system.

Before we delve into the proposed design system, it is important to position it within the wider spectrum of the “more or less automatic fashion” of generative design systems. In fact, it is not our objective to create a design tool that generates all possible tableware collections. In the spirit of Mass Customization, which emerged primarily as a business strategy, it is natural that we focus on the design language of the company adopting such an approach – in our case, the Manufacturer. The proposed design system is capable of generating designs that are related to a particular design space, defined by design solutions produced by the Manufacturer.

The design system is supported by a shape grammar that encodes rules for designing tableware collections, called the Tableware Shape Grammar (TSG). This chapter presents the process used for developing TSG as a generic shape grammar, by abstracting specific shape grammars that correspond to the tableware collections within the Manufacturer’s design space. We can therefore extract two specific contributions. The first one is the Tableware Shape Grammar itself, as a tool to inform the design of ceramic tableware. The second contribution is the methodology used for obtaining the TSG, which we believe can be replicated to develop alternative design systems for mass customization in product design not necessarily related to ceramic design.
3.1 Design Participation Model for the design system

A central agent in mass customization is the customer, or the end-user (Tseng & Piller, 2011). Nevertheless, while designing a mass customization system for product design, there is another equally important user to consider: the designer. Besides these two main types of users, the proposed design system considers two other agents: the researcher, who embodies the team responsible for the design system, and the machine, which represents the production system responsible for materializing the customized design solution.

More important than to establish the agents of the design system is to clearly define the interactions among those agents. Such interactions are greatly defined by the modeling paradigms used within the system, which encompass rule-based, parametric, and digital models. Physical models are also present, as the result of the production system. Agents and interactions of the design system are explained through the model, which encompasses four distinct sequential components, each targeting a different set of users. Each component generates a different type of model, which will be used as input in the subsequent component (see Figure 20).

![Design Participation Model for the design system](image)

Figure 20 – Design Participation Model for the design system

The first component corresponds to a Shape Grammar that encodes design rules of ceramic tableware, which is the foundation of the design system. Such grammar was developed by the research team, and it is supported by the analysis of both design products and processes. As a generative design tool, shape grammars encode a design language (Stiny, 1980). In this research,
the developed shape grammar encodes the design language of ceramic tableware collections produced by the Manufacturer. The resulting output of this component is a rule-based model.

The second component is called Modeler, and its target user is the designer. The Modeler enables the designer to manipulate some of the design rules encoded in the Shape Grammar to produce parametric models, each corresponding to a family of tableware collection designs, and which correspond to the resulting output of this component.

The third component is called Navigator, and its target user is the End-User, the potential owner of the customized tableware collection. In this component, the end-user can navigate through the design space that corresponds to the parametric model provided by the previous component, and whose boundaries were defined by the designer, hence exploring variations in order to specify one favorite solution. The resulting solution is encoded in a digital model.

The fourth and final component is the Digital Fabrication processes that can actually produce a physical object from the digital model. The Digital Fabrication component is actually part of the Production System, which will be covered in depth in its own chapter, further afield. However, it is included in the Design Participation Model to illustrate the continuity of the design system. In fact, the design system encompasses only the first three components – Shape Grammar, Modeler, and Navigator. In this chapter, we will focus on the development of the first two components – Shape Grammar, and Modeler – while leaving the Navigator for the chapter on the User Interface.

![Generic ceramic tableware design space](image)

Figure 21 – Incrementally constrained design space

It is worth noting the progressive constraining of the design space along these three components (see Figure 21). In the Shape Grammar, the design space comprises the sum of design solutions that correspond to a company’s design language, whereas in the Modeler, such design space is reduced to a subspace defined by the designer. Finally, in the Navigator, the End-User specifies one single design solution. We can also add that by developing a Shape Grammar that encodes
the Manufacturer’s design language, the Research team is in fact constraining an existing – and rather large – design space, which contains every possible tableware collection design.

**Evolution of the Design Participation Model**

As research progressed, the Design Participation Model evolved and changed, and so did the roles of the different users and the modeling paradigms they were asked to manipulate.

An initial idea, dubbed the ‘design machine’ approach, consisted in developing a fully automated design system based on shape grammars, which would allow end-users to design their own tableware collections from the ground up. However, this idea was almost immediately put aside, for two main reasons, the first being an ethical one: such design machine would not consider designers as main agents in the design process. This approach would be unethical, but also impossible to implement, given the difficulties in encoding all of a designer’s knowledge into a design system. The second reason relates to the end-users: a potential effect of the so-called mass confusion (Piller et al., 2005) might deter users of ever using the design system. Not everybody is fit for design tasks, and many people feel overwhelmed with the number of decisions inherent to the task of drawing, or painting, or designing something from the ground up.

A second iteration of the design system was then considered, in which the shape-grammar-based design machine would be handed to the designer instead of the end-user. Such a tool would be based on a generic shape grammar encompassing a broad array of design rules for the designer to manipulate. By selecting specific rules from the initial generic grammar, the designer would create a specific grammar, which would enable end-users to create their own designs. However, along with the shape grammar development, its increasing complexity suggested that asking the End-User to use a shape grammar would probably be equally overwhelming. Only a shape-grammar knowledgeable – and statistically unlikely – power-user would be comfortable using such a design tool. In fact, manipulating shape grammars was considered to be challenging even for the designer.

Therefore, in the current design system, the use of shape grammars by the designer has been greatly reduced, and the interaction of the End-User with the system has been reduced to manipulating a parametric model.
3.2 Methodology

Even though it can be used to produce designs, the design system proposed in this research is itself a design product, having resulted from a synthesis process based supported on analytical procedures (Dorst & Cross, 2001). Therefore, the methodology for developing the design system encompasses both analysis and synthesis tasks.

The design system is supported by the Tableware Shape Grammar that encodes the rules for designing ceramic tableware collections. Such grammar was developed according to Duarte’s model for inferring generic shape grammars (Duarte, 2011) (see Figure 22), in which specific designs correspond to analyzed Manufacturer’s collections, which were inferred into a set of specific shape grammars, which in turn was abstracted into the generic Tableware Shape Grammar. The remainder of Duarte’s model bears a correspondence with the Design Participation Model presented above (see Figure 20). In fact, new specific grammars can be extracted from the TSG into parametric models (using the Modeler component), which are able to generate new specific designs (using the Navigator component).

![Diagram](https://example.com/diagram.png)

Figure 22– Coding processes for grammars and types adapted from Duarte, 2011

The process of inferring the generic grammar began by analyzing a set of existing tableware collections produced by the Manufacturer, focusing on aspects such as the nature of the shape of ceramic tableware elements, or the relationships among elements within a collection. A specific shape grammar was inferred for each analyzed collection, containing the design rules deemed necessary for its generation. After developing a specific grammar for one collection, each additional specific grammar was obtained as a transformation of one or more of the preceding specific grammars, following the grammar transformation methodology of Terry Knight (1994).
The resulting set of specific shape grammars was subsequently abstracted into a generic grammar (see Figure 23). Specific grammars were compared in search of common and distinguishing rules. The agglomerated set of rules corresponds to a first, rough iteration of the generic Tableware Shape Grammar. The TSG was fine-tuned by revising some of its rules, beginning by deconstructing them into simpler ones. Although some of these simpler rules could already be found in the first iteration of the TSG, the deconstruction process revealed the need to create additional rules. After the deconstruction process, rules were grouped according to similarity of their shape. Rules within the same group were generalized into a parametric rule, which replaces every rule in the group given the right parameters. The set of these parametric, so-called ‘general’ rules was considered the final iteration of the Tableware Shape Grammar.

The resulting TSG incorporates the rules that govern the design of customized tableware, and serves as a theoretical framework for implementing the computational tools that actually enable designers and end-users to customize the shape of tableware collections.

The design of ceramic tableware collections was studied both as product and process. Besides analyzing collections and inferring specific and generic grammars, we also performed a contextual inquiry on tableware designers, namely the Manufacturer’s team of designers, searching for typical design moves within the actual process of designing a new collection. Since it was performed after developing the TSG, the results of this contextual inquiry had a more significant impact on the actual implementation of the design system (see Figure 24).
3.3 Analysis of existing collections

Product analysis implied analyzing a set of ceramic tableware collections, namely focusing on dimensional, morphological, and functional aspects of those collections as a whole as well as of the elements they included. All the collections analyzed in this research are produced by the Manufacturer. In total, six collections were selected: (1) Romantica, (2) Bisel, (3) Catarina de Bragança, (4) Cotélé, (5) Cuisine Provençale, and (6) Flor (see Figure 25 to Figure 30). The analyzed collections were selected according to the following criteria:

- they should feature relief-based decoration, due to the particular interest in this research on three-dimensional (3D) shape;
- they should encompass a reasonable amount of types used on an regular basis, such as plate, bowls, or mugs;
- they should display a perceivable stylistic coherence.
Figure 25 – Romantica collection, by Matceramica\textsuperscript{13}

Figure 26 – Bisel collection, by Matceramica\textsuperscript{14}

\textsuperscript{13} http://www.matceramica.com/pt/inicio/colecoes/mat-poetry/mais-mat-poetry/romantica

\textsuperscript{14} http://www.matceramica.com/pt/inicio/colecoes/mat-cooking/bisel
Figure 27 – Catarina de Bragança collection, by Matceramica\textsuperscript{15}

Figure 28 – Cotelé collection, by Matceramica\textsuperscript{16}

\textsuperscript{15} \url{http://www.matceramica.com/pt/inicio/colecoes/mat-poetry/mais-mat-poetry/catarina-de-braganca}

\textsuperscript{16} \url{http://www.matceramica.com/pt/inicio/colecoes/mat-cooking/cotele}
Figure 29 – Cuisine Provençale collection, by Matceramica\textsuperscript{17}

Figure 30 – Flor collection, by Matceramica\textsuperscript{18}

\textsuperscript{17} http://www.matceramica.com/pt/inicio/colecoes/mat-stoneware/mais-mat-stoneware/cuisine-provençale

\textsuperscript{18} http://www.matceramica.com/pt/inicio/colecoes/mat-life/mais-mat-life/flor
Information about the six collections was retrieved from the company’s website, namely which tableware types were included in each collection. For each of the included types, an image was provided – either a top or a side view – as well as information about its overall dimensions – width and height. Despite the low resolution of the image, it was possible to trace an approximate sketch of the shape of each type, namely in terms of their profile and contour. The sketches allowed obtaining partial dimensions of each element in each collection, which were recorded in tables like the one shown in Table 2. These tables were subsequently used to compare the analyzed collections, thus obtaining variation intervals of different dimensions in different types. For example, we could determine the minimum and maximum radius of a dinner plate, or the average width of its lip, in the scope of the analyzed collections.

Table 2 – General and partial dimensions of the Romantica collection

<table>
<thead>
<tr>
<th></th>
<th>TOTAL</th>
<th>base</th>
<th>well</th>
<th>lip</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cm</td>
<td>radius</td>
<td>height</td>
<td>radius</td>
</tr>
<tr>
<td>charger plate</td>
<td></td>
<td>15,50</td>
<td>3,00</td>
<td>10,50</td>
</tr>
<tr>
<td>dinner plate</td>
<td></td>
<td>13,50</td>
<td>2,50</td>
<td>9,25</td>
</tr>
<tr>
<td>soup plate</td>
<td></td>
<td>11,50</td>
<td>4,00</td>
<td>5,25</td>
</tr>
<tr>
<td>cereal bowl</td>
<td></td>
<td>7,00</td>
<td>8,50</td>
<td>3,00</td>
</tr>
<tr>
<td>coffee mug</td>
<td></td>
<td>4,50</td>
<td>10,00</td>
<td>2,00</td>
</tr>
<tr>
<td>tea cup</td>
<td></td>
<td>3,75</td>
<td>5,00</td>
<td>1,50</td>
</tr>
</tbody>
</table>

A brief taxonomy of tableware

An important outcome of analyzing the selected collections was establishing a simple taxonomy that would support the development of the design system. This taxonomy encompasses concepts such as collection, type, element or part. Because such terms can be ambiguous, we defined their meaning in the scope of this research.

We begin by defining a dinner set as a set of objects used for meals. The concept of dinner set is useful to define other important concepts. We define each individual object within the dinner set as a tableware element, or simply, an element. For example, a dinner plate is an individual element. Although most tableware elements correspond to a single piece, some of them might
comprise two separate pieces. For example, a tureen is also an individual element, despite comprising two different pieces: body and lid. Therefore, we define a piece as a physically separate constituent of a tableware element. Pieces can be further subdivided into parts. Contrary to pieces, parts are not physically separate from each other. A teapot, for example, comprises two separate pieces: the lid, and the body. The lid is made of two inseparable parts, the lid itself, and the knob. In turn, the body comprises a number of other parts: base, belly, shoulder, lip, handle, and spout (see Figure 31). This subdivision into parts will be instrumental for defining and distinguishing among tableware types.

![Figure 31 – Parts of a teapot (Utah teapot adapted from Nicholas Sourd)](https://commons.wikimedia.org/wiki/File:Celshading_teapot_large.png)

Going back to a larger scale, a dinner set is typically composed of subsets of tableware elements that have similar shapes. For example, a simple dinner set can be composed of six dinner plates, six cereal bowls, six coffee mugs and one tray. Elements with similar shapes belong to the same type. A type is therefore an abstract entity, representing a model from which instances of tableware elements can be produced. In this research, types are differentiated according to the type of food they can carry, and to the parts they comprise. Examples of types include the dinner plate, the coffee mug, and the teapot. Although not necessarily so, a dinner set typically contains types whose shapes are correlated in one way or another. Such relations between type shapes can be defined by the style which those types belong to (Ahmad & Chase, 2012). Parts also help differentiating tableware types. For example, a teapot features a spout, without which it might be considered a mug.

---

19 https://commons.wikimedia.org/wiki/File:Celshading_teapot_large.png
A collection is defined as the set of all particular types that belong to a specific style (see Figure 32). A collection is different from a dinner set. A dinner set does not necessarily contain elements of every type that make up a collection. Also, being a set of tableware elements, a dinner set might – and usually does – include repeated elements of the same type. A collection is a set of types, not of elements. Like types, collections are abstract entities, whereas dinner sets are real entities. The dinner set is associated with the final user – the customer – whereas the collection is associated with the manufacturer. The customer acquires a dinner set, which is a combination of some or all elements of the collection, as provided by the manufacturer.

![Tableware Collection](image)

Figure 32 – Example of a tableware collection

This research deals with the three-dimensional (3D) shape of tableware parts and elements. However, for the sake of simplicity in both graphical representation and discourse, we also resort to auxiliary two-dimensional (2D) descriptions of such 3D shapes, namely profiles and contours. Although the use of these terms might seem obvious, such might not be the case for the contour, and therefore they should be properly defined. For example, some tableware designers use the term ‘outline’ to refer to a top view in shallow types such as a plate, and to a side view in deep types such as a mug. Therefore, in the scope of this research, a profile corresponds to a 2D section view of a tableware element, through a vertical plane containing the element’s symmetry axis, whereas a contour corresponds to a 2D top view.

One central aspect of the proposed taxonomy is function, which has been analyzed within two different scopes: types and parts. On one hand, we categorize types according to a) shape, which is directly related to the food they can contain, and b) the number of people they are meant to.
These categorizations were preferred since they have implications in the shape of the corresponding tableware types, and therefore should be reflected in the shape grammar. Types have been distinguished as being either deep or shallow. Deep types include elements such as a soup plate, a cereal bowl, or a mug. The common feature among deep types is that they can contain liquid food, such as soup, or milk (and cereals), or coffee, shallow types cannot, due to their shallow geometry. We can also categorize tableware types into single or collective use, corresponding for example to a dinner plate and a tray, respectively. Such categorization implies a relationship to the size of the tableware types each category contains, since collective types are expected to contain food for more than one person, whereas individual types are designed to contain an individual portion.

Parts have also been associated with the functions they perform. Again, the teapot serves as a good example, since it features a large number of parts. A distinctive part of the teapot is the spout, whose function is to pour tea, probably into a cup or mug. It also features a handle so the user can hold it, a lid for covering its contents, a body for containing the tea, and a base so it can lay on a table. Naturally, these functions – pour, hold, cover, contain, lay – are not exclusive of the teapot type, since some or all of them exist in other types of a collection, according to the parts that those types encompass. In a soup plate, for example, three functions can be found in three different parts: the well for containing food, the lip for holding the plate (and for preventing food to fall off the plate), and the base to lay it on the table.

![Figure 33 - Three functional parts in dinner set elements, exemplified for a soup plate](image)

**Metatypes and parametric variation**

Some types have similar shapes. For example, besides being smaller, a dessert plate is very similar to a dinner plate, since these two types feature the same set of parts. Therefore, we say they belong to the same metatype. The dessert and dinner plates belong to a metatype called plate, which includes also charger plates, saucers, or bread plates. A soup plate is still a part of the plate metatype. Although its taller well enables the soup plate to contain liquid food, setting it apart
from the other plates, it nevertheless satisfies the requirement of featuring the same set of parts as the other plates. Therefore, types within the same metatype might differ not only in their overall dimensions but also in the relative dimensions of each of their parts. Since such dimensions are present in every type within the same metatype, these dimensions can be considered as common parameters, whose values characterize and define each type. Therefore, types belonging to the same metatype are parametric variations of each other (see Figure 34).

![Parametric variations within the Romantica collection](image)

Figure 34 – Parametric variations within the Romantica collection, according to Table 2

Consequently, a collection can be characterized by the dimensions of all its different types, encoded into parametric configurations. We define the parametric configuration of a particular type as the set of dimensional values of all the parts that characterize the type. Likewise, we can define the parametric configuration of a collection as the set of the parametric configurations of all its constituent types.

Types with different parts can also be interrelated. Let us consider a soup plate and a cereal bowl, which belong to different metatypes – the cereal bowl can be included in the bowl metatype, being similar for example to the larger salad bowl. Unlike plates, bowls do not feature a lip. However, they share the remaining parts – the base and the well – with plates, and so they can be at least partially considered variations of each other. If we imagine that a cereal bowl features an invisible lip, we can still consider it a parametric variation of a soup plate, with a taller well. If we extend this reasoning to all the remaining types of a collection, we can consider each of those types as a variation of an archetype. The archetype would correspond to a hybrid tableware element that features every part in every type (see Figure 35). For now, consider the archetype as an abstract entity, although it will actually correspond to a shape when considering the computational implementation of the design system. A particular type can be extracted from the archetype by a) assigning that type’s dimensional values, and b) defining the parts that are not featured in that type as invisible. Establishing such direct relationships among the shapes of a
collection’s types contributes to guaranteeing a stylistic consistency across the collection in the proposed design system.

![Image](image.png)

Figure 35 – The archetype and some related types

### 3.4 The precedent of Digital Alberti

The deconstruction of the shape of tableware elements has been influenced by the work developed within the scope of the Digital Alberti (DA) project. In DA, parametric models supported the development of a shape grammar that focused on classical architectural column elements, some of them featuring very complex geometries such as the Corinthian capital (Castro e Costa, 2012). For tackling the complex shape of such elements, the column system was deconstructed into hierarchical levels of different column elements according to Alberti’s instructions. For example, a column system would be divided into column and entablature; a column would be divided into base, shaft\(^{20}\), and capital; the base would be divided into other elements, according to its style; some of those elements could be subdivided even further, in order to keep the shape complexity manageable (see Figure 36).

Another element of Alberti’s rules for the design of classical elements is the proportions of the constituent elements in relation to each other and to their hierarchic superiors, or parents. As such, each of these column elements could be characterized by common dimensional attributes, such as their distance to the column axis, their distance to the ground, and their relative height and width. Such dimensions can be represented by rectangles, which in turn can represent its many different shapes – such as convex or concave arcs, or polygonal segmented lines –

\(^{20}\)Alberti actually uses the term *column* when referring to a column’s shaft.
according to specific shape parameters (see Figure 37, left). Please note that the presented shapes correspond to section profiles of the column elements. While each of these elements can only generate relatively simple shapes, combinations of them originate more complex surfaces, such as molds, motifs like those found in columns (see Figure 37, right). Eventually, different elements could be combined into any Albertian column system.

![Diagram of column system deconstruction](image)

**Figure 36 – Column system deconstruction (Castro e Costa, 2012)**

![Profiles and element combinations](image)

**Figure 37 – Section profiles of a column element (left); element combination into classical molds (right) (Castro e Costa, 2012)**

The connection between classical architectural ornamental elements and tableware is not arbitrary. In fact, some of the classical column elements resemble, or try to mimic, the shape of tableware elements. Think for example of a classic vase and a Doric capital, in its simple, cereal-bowl like shape (see Figure 38). Or the Corinthian capital which, according to Alberti, should
resemble a cup covered with acanthus leaves (Alberti, 2011, p. 447). Such similarities in shape suggested that the design system developed for DA could be adapted for generating tableware elements. In fact, the DA approach was reflected on the development of design rules of the tableware shape grammar, as well as in the computational implementation of those rules into parametric models.

![Figure 38 – A Doric capital, a classic vase (www.archiexpo.es), and two ceramic tableware elements](image)

**3.5 Analysis of tableware design processes**

Process analysis implied observing and analyzing the process for designing new collections by the Manufacturer’s in-house design team. This analysis followed the method of Contextual Inquiry (Holtzblatt & Beyer, 1997). The successful application of the Contextual Inquiry (CI) method follows four principles: context, partnership, interpretation and context (Martin & Hanington, 2012). These principles prescribe that:

- researchers must spend time with the subjects of their research where work actually happens;
- researchers should apply a master/apprentice relationship with whom they are researching about, in which the apprentice learns about what the master does while he does it, by observing it, and asking about it;
- researchers must interpret what they learn, and get feedback about such interpretation from the researched subjects, before applying the collected data into a new design;
- finally, researchers must be ready to refocus their attention towards unanticipated aspects of the researched activity.

While mostly guided by the CI method, the analysis process was influenced by Task Analysis (Hackos & Redish, 1998). Task Analysis (TA) is a user-centered design research method in
which tasks are deconstructed for analysis purposes, typically with the intent of designing tools that can aid in those tasks (Martin & Hanington, 2012). In our research, the CI process has been supported by an experiment script that defined a set of issues that should be observed and documented. This script was based on eleven questions, suggested as standard questions for TA by educators in interface design (Fonseca et al., 2012; Landay, 2014):

1. Who is going to use the system?
2. What tasks do they now perform?
3. What tasks are desired?
4. How are the tasks learned?
5. Where are the tasks performed?
6. What's the relationship between customer & data?
7. What other tools does the customer have?
8. How do users communicate with each other?
9. How often are the tasks performed?
10. What are the time constraints on the tasks?
11. What happens when things go wrong?

While the TA method has not been actually applied, these questions have been useful to guide the observations and conversations that composed the applied CI. The CI was performed in the workplace of the design team, during five business days, during normal business hours. During that period, the design team consisted of two designers, one male and one female, with ages of 20 and 34 years old, both Portuguese, and their main task was designing new collections. At the beginning of the inquiry, the design team was briefed about the research project, namely its main objective of implementing a mass customization system for ceramic tableware, as well as the inherent need for a generative design system that would enable designers to create customizable collections.

The information gathered through observations, as well as the information provided by the design team members was first registered by the research team in notebooks, and then typed into word processing in a freeform report by the end of each day. Analysis of this report resulted in a series of important insights, which either confirmed ideas that had been previously considered by the research team, or revealed new, unforeseen aspects of the process of designing new tableware collections. Interpretation of these insights resulted in a set of internal recommendations that were
taken into account in the implementation of the design system. These recommendations ranged from broader design strategies that resulted in high-level decisions in the design system’s design, to more detailed issues that led to particular features in the design system’s implementation. In this section, we will present a summary of the insights that had more impact on the design system. The complete list of gathered insights is listed in Appendix A.

Main insights

Among the various insights gathered in the scope of the Contextual Inquiry, two have been considered as main insights, mostly due to the impact that they had on the design system’s philosophy. The first insight relates to the type of geometry used to describe tableware elements, namely its profile, which might be a relatively complex shape: typically it is a curved shape, along which its attributes might vary, such as curvature or concavity. Observation within the current CI showed that designers can describe tableware profiles using two different types of geometry: circumference arcs, and freeform curves. Due to the relative complexity of these profiles, designers take some time to find a satisfactory shape using either one of the geometry types.

It is our interpretation that designing a profile using circumference arcs exclusively requires a relatively large number of design operations because the designer struggles to control continuity. Typically, CAD applications provide functions such as snapping tools that help the designer guaranteeing positional continuity (C0). However, less support is provided for guaranteeing tangential (C1) or curvature (C2) continuity (Farin, 2002). Some CAD applications provide operations that take tangential vectors as input for creating circumference arcs, enabling the designer to create arcs with tangential continuity. However, if these arcs are edited later, no support is provided to maintain such continuity. Moreover, curvature continuity between circumference arcs is only possible if they feature the same radius.

A suitable alternative to arcs for describing tableware profiles is using freeform curves, such as Bézier or NURBS curves. If a profile is represented by a single Bézier curve, both positional and tangential continuity are guaranteed, being inherent to the construction itself, and even curvature continuity is accounted for. Learning how to design freeform curves using CAD tools requires some experience from the designer (Fisher et al., 2004). However, using freeform curves relieves
the designer of the time-consuming task of manually guaranteeing geometric continuity, and therefore may justify the time invested in learning how to work with these tools.

The initial iterations of the design system were based on a solution between circumference arcs and composite Bézier curves, in which simple Bézier curves are combined into more complex profiles. Since both approaches – circumference arcs and freeform curves – were used by the inquired designers, there was no preference for one over the other. Due to its efficiency in dealing with continuity, the Bézier paradigm has ultimately been adopted in the computational implementation of the design system.

Another main insight relates to the strategy used by designers to guarantee a stylistic coherence among the different types of a collection, by first transforming the types that were created first. Good examples are when designers transform the profile of types within the same metatype, such as a cereal bowl into a salad bowl, or a dinner plate into a dessert plate. These particular transformations are carried by first scaling up or down the original profile, and then adapting its thickness. Although designers usually proceed by applying some additional changes to the resulting profile, it appears to be useful and possible to automatize the initial standardized operations that manage size and thickness of the tableware elements.

In order to automate the mentioned standard operations, the design system should guarantee that changes in one type are replicated in all the other types of the collection, through what we call collection-wise editing. In the computational implementation, this meant that all types are generated as instances of an archetype, which informs about the shape of all of the collection’s types. Details on implementing such archetype will be discussed in the next chapter on implementation.

3.6 Synthesis into shape grammars

Findings in the analysis stage have been subsequently synthesized into the design system that supports the proposed mass customization of ceramic tableware. Such design system is composed of design rules that govern the design styles of the analyzed collections in all its different elements, and which were inferred from the preceding analysis. In our research, the design rules that compose the design system were formalized through the use of shape grammars (Stiny &
Gips, 1972; Stiny, 1980). Shape grammars are rule-based systems that can be used to generate design solutions in the same language; as such, they can generate a wide variety of designs while maintaining stylistic consistency. The design system is supported by a shape grammar, called the Tableware Shape Grammar (TSG), whose rules were derived from the preceding analysis. As mentioned above and explained below, the development of the TGS grammar includes two steps.

The first iteration of the TSG encoded just one tableware collection. By analyzing its elements, parts, components and functions, design rules were inferred and encoded into a grammar capable of generating the collection. However, since this shape grammar was parametric, we could use it to generate not just the original collection, but many design variations of it. This initial shape grammar was important to determine a preliminary set of principles for inferring specific shape grammars.

For continuing developing the design system, five additional different tableware collections produced by the Manufacturer were analyzed, and encoded into additional specific shape grammars. These six grammars were then compared in order to understand and register the similarities and differences among their rules. This analysis resulted in an extended TSG, capable of generating the analyzed collections, as well as other tableware collection within a related design space.

This extended TSG and its corresponding solution space are still bound by the analyzed collections and, therefore, it cannot generate any tableware collection. As such, the proposed design system is able to generate designs within a defined solution space and is, therefore, limited to routine design, as opposed to innovative or creative design (Gero, 1990, 2000). However, as the objective was to create a mass customization system for a particular company, not a universal tableware design tool, working within a predefined solution space was considered acceptable.

3.7 Single collection shape grammar

The proposed design system should enable non-expert users to design tableware collections so that their elements are stylistically coherent, considering both base shape and decoration. A collection’s design style can be encoded in the rules of a shape grammar, which has the potential of generating designs belonging to the same language (Stiny, 1980). Therefore each collection
features its own set of rules, which constitute its specific shape grammar, which in turn should be able to generate every type contained by the collection.

The objective of the first iteration of the TSG was to encapsulate the style of a single collection (see Figure 39). This shape grammar should be able to automatically generate the different elements of that collection, taking into account that the grammar would be subsequently extended to other collections. The selected collection for this first experiment is composed of six different types, from charger plate to coffee cup, and it features relief-based decoration, due to the research interest in studying three-dimensional shapes.

![Figure 39 – Romantica tableware collection by Matceramica21](http://www.matceramica.com/pt/inicio/colecoes/mat-poetry/mais-mat-poetry/romantica)

![Figure 40 – Overview of the soup plate derivation](http://www.matceramica.com/pt/inicio/colecoes/mat-poetry/mais-mat-poetry/romantica)

Tableware elements in the collection can be generated by applying the rules of the shape grammar. The derivation of elements after the rules is split into three phases: initialization, base shape definition, and decoration (see Figure 40). Each phase features a particular set of rules.

**Initialization**

The dimensional analysis of existing collections documented in the previous sections is incorporated into the first two rules of the shape grammar. The general dimensions of the tableware element are introduced as input parameters of Rule 1, creating an envelope in which the element is inscribed (en in Figure 41). In the two-dimensional view of the profile, the envelope is represented by a rectangle, or more generally speaking, a quadrilateral – since the rectangle might be distorted afterwards.

The initial envelope is subsequently subdivided into the element’s functional parts by Rule 2, which is parameterized according to the correspondent parametric configuration. Rule 2a subdivides the envelope into three functional parts – laying, containing, and holding – and can be applied to the shallow types. For the deep types, Rule 2b should be applied, which disregards the holding part (see Figure 41).

![Figure 41 - Shape grammar rules for envelope creation and functional partitioning](image)

This first set of rules is used in what we call the initialization phase, at the beginning of each derivation of a tableware element (see Figure 42). Derivation begins with the initial shape, which corresponds to a referential determining the center of the tableware element, as well as the upward direction (h, from height) and the outward direction (w, from width).
The grammar operates on three-dimensional shapes, namely double-curved NURBS surfaces. For two-dimensional representation purposes, rules and derivations are presented in two different ways, depending on whether they pertain to the design of the base shape or of the decoration.

Since the selected collection features circular elements, their base shape can be described as a solid of revolution, which by definition features a constant profile. Therefore, rules and derivation steps regarding the base shape are represented by two-dimensional profiles. For this first iteration of the TSG, the profile thickness was not considered, and therefore the resulting shape is more accurately described as a three-dimensional surface, rather than as a solid. Moreover, this shape grammar only encodes the shape of the main body of elements. Nevertheless, other parts, such as the handles in the cups and mugs, have been addressed in the implementation of the design system.

**Base shape definition**

The base shape is defined by manipulating the shape of the functional parts that result from the initialization phase. In the end, the base shape profile is the product of a combination of quadratic Bézier curves, controlled by their corresponding envelopes (see Figure 43). The quadrilateral envelopes are used as auxiliary shapes for defining the Bézier curves inscribed in them.

Therefore, in the base shape definition phase, three types of rules can be applied: subdivision, distortion and substitution. Subdivision rules in the base shape definition phase (Rule 3) are similar to the functional subdivision rules used in the previous phase. The only difference is that the resulting envelopes keep the type of the preceding envelope. These rules allow for more complex shapes, through subsequent combination of more than one Bézier curve (see Figure 44).
Distortion rules permit manipulating the quads that correspond to the envelopes and, therefore, manipulate the control points of the Bézier curves that will substitute them. Distortion rules can be applied recursively, allowing generating additional distortions not coded into rules through a compound effect. Figure 45 shows how to obtain this by applying Rule 4 recursively and with different transformations, which are shown under the rule application arrow. Parameters are used both for the distortion effect and for shape matching, and were omitted for clarity.

Substitution rules replace envelopes by the corresponding 3D surfaces, represented in 3D by Bézier curves. For each curve, three of the vertices of the corresponding quad correspond to the curve's control points, whereas the directions of the curve's start and end tangents are defined by the quad's edges (see Figure 46). Rules can be applied in different order. For example, we can subdivide an already distorted envelope. This is possible because of the parametric nature of the grammar, which allows for perspective transformations (Stiny, 1980, p. 351). However, the result
of applying rules in a different order is not necessarily the same (see Figure 47). In this way, some flexibility is guaranteed in the grammar. Figure 48 shows all the derivation steps for defining the base shape of the soup plate, including the initialization phase. The final step corresponds to the revolution of the profile into a set of surfaces, the actual base shape upon which decoration rules are subsequently applied.

Figure 46 – Replacing envelopes with the corresponding Bézier curves: substitution rules and examples of their application

Figure 47 – Different rule application order generates different results

Figure 48 – Derivation of the base shape of a soup plate
**Decoration**

Decoration is achieved by applying shape grammar rules to the set of surfaces that results from the base shape definition phase. In the selected collection, decoration is based on a) subdivision of surfaces, and b) application of a relief- and contour-based motif onto the resulting subsurfaces (see Figure 49).

![Figure 49 – Derivation of the decoration of a soup plate](image)

Subdivision rules in the decoration phase are similar to the ones in the base shape phase, except they operate on the surface's parametric $uv$ space. Rule 10a subdivides the surface into two parts with the same $u$ parameter differential (see Figure 50). In the derivation for the selected collection, Rule 10a is used recursively to subdivide the plate into eight parametrically equal parts. Rule 10b subdivides it into three parts, also along $u$, but in this case the first and third parts have the same $U$ parameter differential, which is different for the second independent part. The parametric relations among the parts are variable. In the selected collection, the second part is larger (see Figure 50). However, in the corresponding rule this constraint is not set, so as to allow for a wider range of variation. Rule 10b uses a label to mark the sub-surfaces to which subsequent decoration rules can be applied.

Similarly to the base shape definition, surfaces resulting from the subdivision operations are to be replaced by more elaborate ones featuring relief-based motifs. For the selected collection, two rules are defined, 11a and 11b, which are to be applied exclusively onto shallow and deep types respectively. Both rules apply a slight depression to the target labelled subsurface, a motif which is present throughout the selected collection. However, contrary to Rule 11b, Rule 11a also changes the sub-surface's contour (see Figure 51). Recursive application of motif replacement
rules to all labelled sub-surfaces is the final stage of the derivation, resulting in a design that belongs to the collection's language.

Figure 50 – Decoration subdivision rules and their application in a soup plate derivation

Figure 51 – Motif replacement rules and their application in a soup plate derivation

In the decoration phase, we make use of the parametric representation of surfaces. In this representation, the Cartesian coordinates of a surface point depend on two different parameters $u$ and $v$, allowing for a continuous mapping of a two-dimensional region onto space (Pottmann et al., 2007). In the derivation steps of the decoration phase, the tableware elements are represented in two-dimensional space as top views. In the case of the selected collection, these elements correspond to circular objects, and inherently their parts feature circular arcs. However, in representing rules, surfaces are represented by squares, a generic shape which better evokes the two-dimensional nature of the $uv$ parametric representation (see Figure 52).
Therefore, rules are mapped onto the double-curved surface of the design. These mapping operations imply the use of non-linear transformations that map straight lines into curves (Pottmann et al., 2007), which are not usually addressed in the shape grammar formalism. In fact, Stiny (1980) states that "a shape rule \( a \rightarrow b \) applies to a labeled shape \( c \) when there is a transformation \( t \) such that \( t(a) \) is a subshape of \( c \)" (Stiny, 1980, p. 347), limiting these transformations to translation, rotation, reflection, and scale (Stiny, 1980, p. 344). However, the author points out that for additional transformations, such as affine or projective transformations, a parametric grammar should be used (Stiny, 1980, p. 351). For the development of this grammar, we are assuming that, similarly to affine and projective transformations, non-linear transformations can be also be used in parametric grammars.

**Shape grammar application**

Since we are aiming at the customization of collections, the shape grammar was conceived to be parametric. Therefore, the shape grammar encodes not one collection but a class of collections, whose design space corresponds to variations of the collection selected originally. A three-dimensional parametric grammar is difficult to test without being implemented. First, the combination of several parameters corresponds to a large number of solutions. And second, some three-dimensional geometric operations are difficult to represent in two-dimensional drawings, such as surface mapping operations. Therefore, the parametric shape grammar was converted into a set of parametric models, which corresponded to derivations of that grammar, thus generating designs that would be generated by the grammar. These parametric models were developed in
Grasshopper\textsuperscript{22}, a visual programming interface that interacts with the geometric modeling software Rhinoceros\textsuperscript{23}.

It should be noted that the Grasshopper model is not considered a proper implementation of the shape grammar, but rather a parametric geometric model. However, if we consider that the result of the derivation of a parametric shape grammar is a parametric model, than we can argue that the Grasshopper model corresponds to the implementation of a derivation. In fact, the Grasshopper model was developed so that rules can be identified as groups of components, in a modular fashion (see Figure 53). The implementation of these parametric models is covered in the next chapter.

![Figure 53 – Parametric model developed in Grasshopper: using rules as groups of components](image)

With this tool, two derivations of the soup plate were implemented as two distinct parametric models. The two derivations differ slightly, having different rules applied in the base shape of the well: while one features an S shape, the other features one concave well. Then, for each derivation, two models were generated using different parameter configurations, namely in the decoration of the border part, regarding its width, number of subdivisions, thickness and depth in the relief. Therefore, a total of four digital models were generated (see Table 3), which were later materialized through digital fabrication.

The digital models generated in the Grasshopper program were materialized using additive manufacturing equipment, namely a plaster-powder-based, binder jetting ZPrinter 350\textsuperscript{24}. Prototyping the models provided for a general first impression about the models being generated,

\textsuperscript{22}http://www.grasshopper3d.com/

\textsuperscript{23}https://www.rhino3d.com/

\textsuperscript{24}http://infocenter.3dsystems.com/product-library/z-printer/zprinter-350
namely in terms of scale and weight. The 3D printed models are especially useful for communication purposes, for example to potential industry partners, permitting a better visualization of the design. Four quarters of dishes were produced, instead of four complete dishes (see Figure 54), not only for saving purposes, but also due to limitations in the equipment’s size: the maximum printable area is 20x20 cm, whereas the soup plate model measures 23 cm in diameter. This was also useful to evaluate the results in terms of their section. Further work on the production system will be covered in its own chapter.

Table 3 – Digital models of generated solutions

<table>
<thead>
<tr>
<th>Derivation 1 Parameters 1A</th>
<th>Derivation 1 Parameters 1B</th>
<th>Derivation 2 Parameters 2A</th>
<th>Derivation 2 Parameters 2B</th>
</tr>
</thead>
</table>

Figure 54 – 3D printed prototypes


3.8 Generic shape grammar from multiple collections

Despite only encoding a single collection, the initial shape grammar was essential to determine a preliminary set of principles for the design system. For its further development towards a generic shape grammar capable of generating several collections, five additional tableware collections were analyzed, and encoded into corresponding specific shape grammars, adding to the one analyzed in the previous section (henceforth referred to as “collection A”). These six collections were compared so that we could understand and register the similarities and differences among them. Expectation was that the rules inferred from these collections could be re-used to describe similarities identified in new collections, whereas differences would imply devising new rules, thus further completing the generic grammar and, consequently, the design system.

Comparing collections

The first step in this comparative analysis was similar to the single collection shape grammar, having each of the five new collections encoded into rules. However, contrary to the previous exercise, a shape grammar already existed and, therefore, rules for the new collections were inferred taking the existing grammar into account. Let us consider as an example the partial derivation shown in Figure 55, which concerns the decoration of a plate belonging to one of the five new collections (henceforth referred to as “collection B”). In this example, rules themselves are being shown instead of their corresponding numbers, since rule numbering has subsequently changed.

As expected, some rules of the grammar that generates collection A (or “grammar A”) were reused, such as the rule that subdivides a surface in equal parts in the $u$ direction (rule 10a in Figure 50), used in the first step of the derivation of collection B. However, all rules from grammar A were not sufficient to describe collection B, and so new rules were created to obtain grammar B. This method of obtaining a grammar B from grammar A follows Knight’s methodology for transforming shape grammars through rule addition, rule deletion and rule change (Knight, 1994). As such, grammar B can be considered a transformation of grammar A. This methodology was used to obtain the grammars for all the remaining collections.

After having inferred the rules for each collection, they were laid out together, including the rules of collection A (see Table 4), enabling a visual comparison of their similarities and differences. In
Table 4, columns correspond to collections, and rows correspond to rules. Rules were grouped together according to their similarity in terms of the type of design operation they correspond to, and the resulting groups are separated by horizontal lines. You will notice that Table 4 features only rules pertaining to the decoration of the ceramic tableware elements. As the new collections were analyzed, we verified that their base shape could be generated using the same rules inferred for the collection A, namely the rules governing the distribution of the functional parts, even if they were applied in different order or with different parameters.

![Figure 55 - Partial derivation of a plate from collection B](image)

After having inferred the rules for each collection, they were laid out together, including the rules of collection A (see Table 4), enabling a visual comparison of their similarities and differences. In Table 4, columns correspond to collections, and rows correspond to rules. Rules were grouped
together according to their similarity in terms of the type of design operation they correspond to, and the resulting groups are separated by horizontal lines. You will notice that Table 4 features only rules pertaining to the decoration of the ceramic tableware elements. As the new collections were analyzed, we verified that their base shape could be generated using the same rules inferred for the collection A, namely the rules governing the distribution of the functional parts, even if they were applied in different order or with different parameters.

This comparative analysis, along with the implementation of the new collections into parametric models, revealed that some rules were very similar, thereby suggesting that the grammar could be optimized and rendered it leaner and, thus, more efficient. This led to the revision of the shape grammar, which began by deconstructing some more complex rules into simpler ones, and then continued by having similar rules grouped together. Finally, each rule groups was translated into a more general rule through parametrization.

Revision of the shape grammar implied deleting the revised rules, and adding new ones, towards a transformed generic grammar. This resulted in a new arrangement of the specific grammars for each of the six collections. Table 5 presents these new arrangements in a fashion similar to the one in Table 4, laying out existing rules that were kept, along with added rules identified with a shaded background, and deleted rules identified with dotted shapes.
Table 4 – Shape rules of the six encoded collections

<table>
<thead>
<tr>
<th></th>
<th>R11a</th>
<th>R11b</th>
<th>R11c</th>
</tr>
</thead>
<tbody>
<tr>
<td>u-subdivision</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>R12a</th>
<th>R12b</th>
<th>R12c</th>
</tr>
</thead>
<tbody>
<tr>
<td>v-subdivision</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>R13a</th>
<th>R13b</th>
</tr>
</thead>
<tbody>
<tr>
<td>curved subdivision</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>R14</th>
</tr>
</thead>
<tbody>
<tr>
<td>shift</td>
<td>□ → □</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>R16a</th>
<th>R16b</th>
<th>R16c</th>
<th>R16d</th>
</tr>
</thead>
<tbody>
<tr>
<td>contour</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>R18a</th>
<th>R18b</th>
<th>R18c</th>
</tr>
</thead>
<tbody>
<tr>
<td>relief</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
</tbody>
</table>

![Image of table with shapes and rules]
Rule deconstruction

At this stage, revision of the rules consisted in deconstructing them into simpler, atomic operations, regarding shape issues such as geometry, topology, or symmetry. Figure 56 illustrates the rationale behind deconstruction actions, showing, on the left, examples of four rules from Table 4 that were subjected to revision, and on the right, how they are deconstructed into simpler rules. Beneath each example, it is demonstrated how to obtain the revised rule by applying simpler rules. Rules in Figure 56 are numbered according to Table 5. Also accordingly, a shaded background corresponds to new added rules, and dotted shapes correspond to deleted rules. Typically, revised rules were removed from the grammar, and replaced by their constituent simpler rules. Some of these rules were part of the other collections’ grammars, while other, new rules were inferred from the deconstruction process. Examples of these new rules are shown in Figure 56.

Figure 56 – Examples of rule deconstruction

The first example presents existing Rule 13b, which subdivides an input surface along two symmetrical curved lines, resulting in three different sub-surfaces. Rule 13b is deconstructed into four simpler rules, beginning with existing Rule 12, which subdivides the input surface in two parametrically equivalent parts. Each of the resulting surfaces is subdivided along a curved line
by existing Rule 13a. However, for the resulting subdivisions to be symmetrical, a rule responsible for the horizontal flipping of a surface’s parametric space must be applied previously. This new operation was therefore added to the revised shape grammar as Rule 14b. Finally, originally inexistent Rule 17b joins two of the resulting surfaces horizontally. In the second example, existing Rule 14c in deconstructed into a vertical subdivision, performed by existing Rule 11, and a new stitching operation that joins two surfaces together vertically, implying the addition of Rule 17a. In this case, the original Rule 14c was not deleted since it is expected to be more useful in the tableware design process than just its constituents. Deconstruction of the other two rules, 16b and 16c follow a similar logic to the previous examples. The deconstruction phase resulted on a re-arrangement of rules within the specific shape grammars that encoded the six analyzed collections (see Table 5).

The effort of “rule deconstruction” shed some light on what rules were “made of”, granting specific grammars more coherence among them, and making it possible to re-assemble the rules as subroutines in a computer implemented parametric model.
Table 5 – Shape rules after revision of the grammars following a comparative analysis

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R11a</td>
<td>u-subdivision</td>
</tr>
<tr>
<td>R11b</td>
<td>u-subdivision</td>
</tr>
<tr>
<td>R11c</td>
<td>u-subdivision</td>
</tr>
<tr>
<td>R12a</td>
<td>v-subdivision</td>
</tr>
<tr>
<td>R12b</td>
<td>v-subdivision</td>
</tr>
<tr>
<td>R12c</td>
<td>v-subdivision</td>
</tr>
<tr>
<td>R12d</td>
<td>v-subdivision</td>
</tr>
<tr>
<td>R13a</td>
<td>curved subdivision</td>
</tr>
<tr>
<td>R13b</td>
<td>curved subdivision</td>
</tr>
<tr>
<td>R13c</td>
<td>curved subdivision</td>
</tr>
<tr>
<td>R14a</td>
<td>edit UV</td>
</tr>
<tr>
<td>R14b</td>
<td>edit UV</td>
</tr>
<tr>
<td>R14c</td>
<td>edit UV</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R15a</td>
<td>select / delete</td>
</tr>
<tr>
<td>R15b</td>
<td>select / delete</td>
</tr>
<tr>
<td>R16a</td>
<td>contour</td>
</tr>
<tr>
<td>R16b</td>
<td>contour</td>
</tr>
<tr>
<td>R16c</td>
<td>contour</td>
</tr>
<tr>
<td>R16d</td>
<td>contour</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R17a</td>
<td>stitching</td>
</tr>
<tr>
<td>R17b</td>
<td>stitching</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R18a</td>
<td>relief</td>
</tr>
<tr>
<td>R18b</td>
<td>relief</td>
</tr>
<tr>
<td>R18c</td>
<td>relief</td>
</tr>
</tbody>
</table>

*shaded: added rules*, *dotted: deleted rules*
Rule generalization

In order to implement the revised grammar rules into parametric models, the number of rules would preferably be minimized. Some work towards this minimization had already been done. Since the beginning of the comparative study, rules have been intuitively grouped according to the type of operation they performed (see Table 4). Within the same group, rules are very similar to each other. It was quickly understood that these similar rules could be generalized into a single rule with different parameters. Therefore the number of rules could be minimized through generalization. Figure 57 shows examples of generalization of rules through parameterization.

![Diagram](image)

Figure 57 – Examples of rule generalization through grouping and parameterization

Rules 11 and 12, which govern the parametric subdivision of surfaces along u- and v-isoparametric curves respectively, are paradigmatic cases of such generalization. Rules 11a, 11b and 11c subdivide a surface into a given number $n$ of parts along a number $n-1$ of isoparametric curves, which correspond to the given parameters $u(n)$. Therefore, the parameters for a possible
general form of Rule 11 (GRule 11) could be \((n)\) numbers, corresponding to the relative sizes of the parts. In the example shown in Figure 58, given the parameters \(\frac{1}{4} \quad \frac{1}{2} \quad \frac{1}{4}\), GRule 11 returns three sub-surfaces, two smaller ones and a larger one, assuming the input surface is uniformly parameterized. Given these parameters, GRule 11 returns the same result as the previous Rule 11a. The same applies for GRule 12, which governs subdivision of surfaces along curves that are isoparametric in the \(v\) direction.

![Figure 58](image_url)

Figure 58 – Example of application of GRule 11 to divide the border of a plate into parts for decoration purposes.

GRule 13 is a special case, since the required parameter is a curve, instead of a scalar parameter. Such input curve will determine the shape of the splitting curve, and it can be represented by a mathematical function. GRule 15 is generalized from Rules 15a and 15b, which are equivalent rules, and therefore redundant. According to Krstic (2012), “two rules are equivalent if whenever applied to the identical shapes the resulting shapes are identical”. Also, two rules are equivalent if one is a transformed version of another such that the same transformation is applied to both left-hand side and right-hand side of the rule. Such is the case of Rules 15, which select or delete part of a surface that has been split along a horizontal line. In that case, mirroring Rule 15a along a horizontal axis generates Rule 15b. Analogously, a 90º rotation enables the use of general rule GRule 15 on surfaces split along vertical lines. The same applies to GRule 17, which joins two adjacent surfaces together. In the case of Rules 18, generalization permits to create different relief effects on surfaces, by manipulating parameters such as border thickness and depth. For example, assigning a positive or negative value to the depth parameter generates a high or low relief, respectively. The general rules obtained with the generalization phase constitute a first draft of the generic grammar. Table 6 shows the general rules used in each collection, along with the parameters that have been used in each rule.
# Table 6 – The rules and rule parameters used to encode the six collections studied

<table>
<thead>
<tr>
<th>GR11</th>
<th>GR12</th>
<th>GR13</th>
<th>R14a</th>
<th>R14b</th>
<th>R14c</th>
<th>GR15</th>
<th>GR17</th>
<th>GR18</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
</tr>
<tr>
<td>u →</td>
<td>u' u1</td>
<td>1:1 1:2:1</td>
<td>1:1</td>
<td>1:1</td>
<td>1:1 1:3</td>
<td>1:1</td>
<td>(gauss) (gauss) (gauss) (sine) (gauss) (gauss) (gauss)</td>
<td>(top) (bottom) (top) (bottom) (top) (bottom)</td>
</tr>
<tr>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
</tr>
<tr>
<td>v →</td>
<td>v1 v2</td>
<td>1:2:1</td>
<td>1:1</td>
<td>1:1:2 1:2:1</td>
<td>1:3 1:2:1</td>
<td>1:1:2 1:2:1</td>
<td><img src="image19.png" alt="Image" /></td>
<td><img src="image20.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image21.png" alt="Image" /></td>
<td><img src="image22.png" alt="Image" /></td>
<td><img src="image23.png" alt="Image" /></td>
<td><img src="image24.png" alt="Image" /></td>
<td><img src="image25.png" alt="Image" /></td>
<td><img src="image26.png" alt="Image" /></td>
<td><img src="image27.png" alt="Image" /></td>
<td><img src="image28.png" alt="Image" /></td>
<td><img src="image29.png" alt="Image" /></td>
</tr>
</tbody>
</table>

## Generic grammar application

Along the process of developing the generic TSG, its rules were implemented as parametric geometric operations in Grasshopper, complementing what had been already implemented after analyzing collection A. By manipulating such operations, combining them into parametric models and defining their parameters, collections can be generated as digital models. In order for this
generic grammar to be validated, it should be able to execute two kinds of tasks. First, it should be able to reproduce the original collections and their designs. This was tested successfully along with the implementation. Then it should allow to creatively generating new collections. In order to test this ability, three new original collections were generated through derivation, using the general rules, illustrated in Figure 60. The combinations that generated these collections were random, waiving any intention to resemble the original collections. Observing the new designs in Figure 60, and considering examples from the original collections in Figure 59, we can verify that a) the new designs are fairly different from the original collections, but b) their constituting elements – namely decorative ones – can be traced back to the original collections. This suggests that the generic grammar has the potential for generating new designs, while ensuring formal coherence.

![Figure 59](image1.png)
Figure 59 – Elements of the original collections generated by the generic shape grammar

![Figure 60](image2.png)
Figure 60 – Elements of three new collections generated by the generic shape grammar

### 3.9 “Genericness” of a shape grammar

The development of a generic shape grammar for the mass customization of ceramic tableware was documented, and the process used is proposed as a general methodology for developing generic shape grammars. Along this task, a question emerged: how generic is a generic shape grammar?
In related literature, the terms ‘generic’ and ‘specific’ have often been used as absolute properties of shape grammars. For example, in Benrós et al. (2014) three specific shape grammars are abstracted into one generic shape grammar. Likewise, in Mendes et al. (2013; 2014) specific grammars are analyzed for future abstraction into a generic grammar. In both cases, the reduced number of grammars in question justifies the polarization of the terms ‘generic’ and ‘specific’.

The relationships between ‘specific’ and ‘generic’ are synthesized by Duarte (Duarte, 2011) in Figure 22, explaining the states and actions for generalizing and specifying shape grammars into new designs. However, the polarity between ‘specific’ and ‘generic’ is still present in this model. In Beirão et al (2011, 2012), a hierarchy of shape grammars structures the relationships among Urban Induction Patterns and generic and specific Urban Grammars. Such a hierarchy, together with expressions like “very generic grammar”, suggests the use of ‘generic’ and ‘specific’ as relative terms.

In the case of the Tableware Shape Grammar, we anticipate the need for handling a wider range of hierarchical levels. Until now, six collections have been encoded into corresponding specific shape grammars, which in turn have been combined into the current generic grammar. However, in order to extend the design system, many more collections can be encoded. Therefore, it can be said that the generic grammar resulting from this extension is ‘more generic’, or ‘less specific’, when compared to the current one. As such, in the scope of this investigation, we consider a grammar to be ‘more generic’ when it is able to generate a wider range of collections than a ‘more specific’ grammar.

As explained above, a more generic grammar can be inferred from combining two or more specific grammars. In addition, a more specific grammar can be extracted from a more generic grammar through a process of specification, by selecting rules from the latter, and by specifying the parameter interval for these rules. We suggest the use of the expression ‘general grammar’ to designate the most generic grammar at a given time, thus corresponding to the actual state of the design system. The general grammar contains all the rules inferred from every analyzed collection, thus being able to generate each and every one of those collections. It is possible to constrain the generation of each of these collections through specification of the general grammar. We suggest designating the resulting specific shape grammars, which are only able to generate a single class of collections, as ‘elemental grammars’.
“Genericness” is the term suggested for a shape grammar’s quality of being more generic or more specific. Although “genericness” has not been quantified, it seems possible to sort two shape grammars relative to each other according to that quality. Therefore, it seems to be a relevant parameter for controlling a wider hierarchy of shape grammars, useful for extending the mass customization experience of ceramic tableware. The relations between general, generic, specific and elemental shape grammars are shown in Figure 61.

![Shape grammar “genericness” spectrum](image)

**Figure 61 – Shape grammar “genericness” spectrum**

### 3.10 Chapter conclusion

The work on the Tableware Shape Grammar is instrumental in implementing a computational application that supports the design of mass customized tableware collections, namely by driving the efforts of inferring its design rules. The next chapter will present how these design rules are implemented into computational applications, so they can be wielded by designers and end-users.

Besides the TSG itself, the process that led to its formulation as a generic grammar is also considered an important contribution from the chapter being concluded. In fact, other documented formulations of generic grammars are found to be less detailed. We examine two examples, from architecture and urban design.

The first example is a generic grammar that has been devised for single family housing design solutions (Benrós et al., 2014), for which the authors compare three existing shape grammars that encode distinct architectural styles: the Palladian villas (Stiny & Mitchell, 1978), Frank Lloyd Wright’s prairie houses (Koning & Eizenberg, 1981), and Álvaro Siza’s Malagueira houses (Duarte, 2001).
This exercise shares some of the methodological steps used in inferring the generic Tableware Shape Grammar. Similarly to our research, the three housing grammars are compared, towards identifying differences and similarities within the rule sets. Such comparison allowed the authors to conclude the feasibility of a generic grammar, by identifying a subdivision-based grammar structure as suitable for generating the three styles, and by determining the similarity of certain rules across the three specific grammars, rendering them reusable in the formulation of the generic grammar.

However, the formulation itself consisted in developing a new grammar structure and a new set of parametric shape rules. Although it is evident that the findings in the preceding comparison inform the formulation of the new generic housing grammar, its process is not documented as systematically as it has been in our research. Therefore, the methodology used for formulating the TSG can be considered an important contribution to the study of generic grammars.

The need for such systematization is justified by the larger amount of specific grammars that has been used in formulating the generic TSG. In our case, six specific grammars are abstracted into a generic grammar, compared to three in the housing grammar. Moreover, the methodology used in formulating the TSG allows for it to incorporate more collections and their respective grammars.

The concept of generic grammar has also been used in research on urban design, in which a design system is supported by shape grammars (Beirão et al., 2012, 2011). The main component in this design system is the Urban Induction Pattern (UIP), defined as a small generic grammar that encodes a typical urban design move, or a design pattern. In this definition, such grammar is considered small because it encodes a specific design move, yet generic because it is context-free. The urban design process can thus be broken down into a sequence of UIPs, and encoded into a specific grammar can be defined by combining specific UIPs and by constraining their parameters. Being defined as generic grammars, it is interesting to look at the methodology behind the formulation of UIPs.

Similarly to the TSG, such formulation departs from specific urban design solutions corresponding to four case studies. For each design, rules were inferred from interviews with the corresponding urban designer. Although not having been formalized as such, the set of rules for each design could be considered its specific grammar, in accordance to our formulation of the
TSG. Moreover, if we consider that designers have their own design styles, such styles would be reflected into those specific shape grammars, had they been formalized.

While inferring urban design rules, the researchers determined which were common across designs and which were unique of a particular design, such as in the comparison step in the TSG formulation. Moreover, it is mentioned that such UIPs have been broken down into “the smallest design moves possible so that most of the large design moves would already be a composition of smaller UIPs” (Beirão et al., 2011, p. 81), invoking the deconstruction step in the TSG formulation, although such process is not presented in detail. It is also mentioned that “[t]ogether, the UIPs form a very generic grammar with very relaxed parameters. (Beirão et al., 2012, p. 667)”, and that “[e]ach design move is encoded into a very generic shape grammar independent of context that can be applied to different contexts and customized by constraining the available parameters” (Beirão et al., 2011, p. 78). Although being presented as generic grammars, UIPs perform a similar role as the General Rules in the generalization step of the TSG formulation.

![Diagram of TSG processes](image)

**Figure 62 – Comparing processes for formulating different generic grammars**

In the formulation of the UIPs, we can identify the steps used towards a generic TSG. However, as in the housing grammar, these steps are implicit in the formulation process, rather than being explicitly systematized as for the TSG. Therefore, we believe that the methodology described in this chapter can be more easily replicated in order to develop alternative design systems for mass customization in product design. In the next chapter, we focus on how to use the design rules inferred using the described methodology to generate customizable tableware collections.
Chapter 4
Streamlining parametric modeling for tableware designers

In the previous chapter, we described the development of the design system from a conceptual point of view: we explained the ideas behind it, as well as the shape grammar that supported the design system. In this chapter, we discuss the implementation of the design system into computational tools, taking advantage of the computer’s ability to generate large numbers of solutions, in a run-time user interaction scenario. In the scope of the design system implementation, run-time user interaction means the user’s ability to participate in the design process of a tableware collection while the process is running. In fact, the idea was for the user to manipulate the shape of a tableware collection through a sequence of design moves, corresponding to shape operations provided by the design system. The intention was to enable the kind of user interaction described by Schön as a see-move-see cycle in his study of the design process (1984). It was therefore crucial to guarantee that, after applying each of the provided shape operations, the user could to see the result, and subsequently operate on that result.

This approach implied the use of programming languages for the development of computer applications that could generate visual representations of design solutions. Along the system’s implementation three prototypes were produced, using three different programming environments: (1) Grasshopper, (2) Racket, and (3) Unity. These technology shifts were induced by one of the main intentions of the project, which was to implement the design system as a web application, so it could be tested by a large number of users with diversified backgrounds. Difficulties in using the first two prototypes online led to the implementation of the third one.

Another main concern during the implementation of the design system for the mass customization of ceramic tableware was that it should be user-oriented, considering the two main user groups foreseen in the Design Participation Model presented at the beginning of the previous chapter (see Figure 20). Recall that the goal is first to allow tableware designers to create customizable collections in the form of parametric models, and second to allow end-users to manipulate the shape of such customizable collections. Reflecting this Design Participation Model, the design
system was implemented into two different components: the Modeler, which targets designers, and the Navigator, which targets end-users.

It is noteworthy to mention that the development and implementation of the design system were intertwined, particularly in the Grasshopper prototype (see Figure 63). For example, in order to test hypotheses formulated along the development of the shape grammar, its rules were computationally implemented through programming. Testing those rules brought up hidden design problems that could then be addressed.

![Development Timeline](image)

Figure 63 – Simplified timeline of the development and implementation of the design system

### 4.1 Implementation architecture and tools

The implementation of the design system resulted in three different prototypes, which were developed consecutively. Each of these prototypes were developed using different environments and programming languages. Nevertheless, the three prototypes share the same architecture, which can be represented by the Model-Viewer-Controller (MVC) pattern (Taylor et al., 2009, p. 95), comprising three components (see Figure 64):

- the **model component** constitutes the core of the design system and is responsible for generating the tableware geometry, according to the rules established in the design system;
- the **viewer component** enables the user to see the geometry as it is generated by the model component, for example on a computer screen or in virtual reality environment;
- the **controller component** corresponds to the User Interface, which enables interaction with the design system, so that the user provides the information required to generate the geometry.
The suitability of the three different prototypes was evaluated using four criteria: portability, CAD capability, performance, and user-friendliness.

We defined the portability of a prototype as its ability to be run on various platforms, that is, either as a standalone application, on a web browser, or even on mobile devices as an app, so it could be used independently by the end-user, without the help of a salesperson. Although all prototypes are developed using integrated development environment (IDEs), they should result in a portable application, independent of any third party application, or at most dependent on an application that any user is likely to have, such as a web browser. For this purpose, the ability of an IDE to compile the developed applications was considered when choosing the tools for implementing the design system. We will see that this has been the most influential aspect in evaluating the prototypes, leading to migrating twice to a different programming environment.

The CAD capability of a prototype concerns its ability to deal with shape. Surprisingly, not all prototypes are associated to a CAD application and, therefore, may lack the necessary functions for shape manipulation. These functions can be more or less sophisticated, and used for modeling and viewing the manipulated shapes, as well as encoding, saving and opening a modelled shape, either as a static digital model, or as a dynamic parametric model.

The performance of a prototype is related to the time it takes to perform certain tasks, such as generating or recalculating solutions. Rather than being commensurated in absolute terms, performance was compared empirically by the researchers.

The last criterion used to evaluate the prototypes was user-friendliness, not of the prototype itself but of the technology used in the implementation. In this case, the target users were the researchers who have developed the prototypes.
**Grasshopper**

The first prototype of the design system was implemented using Grasshopper and Rhino (see Figure 65), and its primary objective was to test the tableware shape grammar, described in the previous chapter. One of the reasons to use Grasshopper was the experience gained in implementing the Digital Alberti project, with the expectation that it would permit to implement and test the tableware design system easily and quickly. Moreover, part of the code used in DA was used to kick start the implementation of the Tableware Design System (TDS).

![Grasshopper Interface](image)

**Figure 65 – The Grasshopper prototype**

Grasshopper (GH) is a modeling application used to develop parametric models through visual programming, or as their developers describe it, “a graphical algorithm editor” (Rutten, 2007). GH has been progressively embraced by the creative community, namely designers and architects, mostly because it is a visual language, closer to the one used by such professionals (Leitão et al., 2012), enabling them to create generative algorithms by visually connecting operational nodes. Such algorithms, or definitions as they are called in GH jargon, can generate designs. Although its visual nature is usually associated with user-friendliness, GH requires its own management techniques in order to keep a program tidy. As a GH program grows more complex, it is fairly easy to entangle all its wires, rendering it undecipherable.
Designs coded in GH may be designed as digital models in Rhinoceros (McNeel, 1998) (Rhinoceros, for short), a modeling application capable of representing complex geometry such as NURBS surfaces, which are particularly suited for describing the double-curved shapes of ceramic tableware elements. As Rhino has been widely used by the design community, it gave rise to a growing number of plug-ins to extend its design coding capabilities, GH being probably the most popular. In this prototype, Rhino provides CAD capability with both modeling and visualization operations, whereas GH acted as the parametric modeler, as well as the user interface (see Figure 66). Despite the versatility of Rhino and GH, some limitations were found, which were overcome by developing scripts for GH, written in VB.NET, a textual programming language.

![Diagram](image.png)

**Figure 66 – Implementation architecture for the Grasshopper prototype**

Regarding performance, the GH prototype became slower to generate solutions in real time as the models became more complex. In fact, it was considered too slow to provide a successful mass customization experience, as the prototype depended on a number of different applications: it was built in GH, which ran on Rhino, making use of VB.NET classes. Therefore, time spent on communication between applications added to the actual geometric computation. These performance issues are also originated because some operations target complex geometric CAD objects that require a fair amount of calculation power – such as, for example, mapping operations on surfaces, which were required in the decoration phase.

In addition to performance limitations, the portability of the GH prototype also compromised it as a suitable solution for implementing the design system. At the time of its implementation, we had not found a solution for running GH programs as standalone applications. Currently, a number of solutions can be found that present models developed in Grasshopper online and explore
variations through the manipulation of slider controls, such as Speckle\textsuperscript{25}, or ShapeDiver\textsuperscript{26}. In future research, we expect to test the GH prototype using these solutions.

The portability and performance limitations identified in the GH prototype led to the development of a second iteration of the design system in a different programming environment. Nevertheless, it served its purpose as a stepping stone for the subsequent implementations in Racket and Unity.

**Racket**

The second prototype of the design system was written in Racket, a LISP-based programming language. Racket is considered a powerful and versatile language, and benefits from the availability of modules that extend Racket beyond its native functionalities (Leitão, 2014). One of them is Rosetta (Lopes & Leitão, 2011), a module that enables the generation of geometry into a growing number of CAD environments, so called back-ends. Presently, Rosetta is able to communicate with AutoCAD, Rhinoceros, OpenGL, Sketchup, ArchiCAD, and Revit. Another interesting aspect of Rosetta is that it understands a number of programming languages, so called front-ends. Apart from Racket, the native language in which it is programmed, Rosetta can read AutoLISP, Python, JavaScript, and Processing, thus relieving its users from learning a new programming language should they already know one of these.

The main objective of migrating from Grasshopper to Racket was to keep options open as Rosetta enabled the use of different CAD applications as visualization components, permitting the designer to choose a suitable one. In addition, one of the planned backends for Racket was WebGL, considered the de facto standard for generating 3D content for web browsers (Parisi, 2012), thereby facilitating the implementation of the design system as a web application.

In the implementation architecture of the Racket prototype (see Figure 67, left), the model component was implemented using the Racket programming language, the language in which Rosetta is written. Rhino was used as a backend, therefore corresponding to the viewer component. In this prototype, the DrRacket IDE was used for writing the code. In the resulting

\textsuperscript{25} http://beta.speckle.xyz/

\textsuperscript{26} https://www.shapediver.com/
system, the user provides instructions and parameters via an HTML page running in a browser (see Figure 68), which is thus considered the controller component. Regarding portability, the Racket prototype, like its Grasshopper predecessor, depended on third party applications, such as DrRacket or Rhino, to run properly.

Figure 67 – Implementation architecture for the Racket prototype: during development (left) and after compilation (right)

Figure 68 – Implementation of the Racket prototype, featuring an HTML toolbox as a Graphical User Interface
However, this implementation architecture refers to the development setup of the Racket prototype. The intended implementation for the final Racket prototype should run exclusively on a web browser, waiving its dependency on additional applications. It would make use of Rosetta’s planned WebGL backend as the viewer component, producing results in a web browser and, therefore, dispensing with using Rhino. Moreover, the model component would correspond to a compiled version of the Racket program, capable of running on its own, instead of depending on DrRacket, fulfilling the portability requirement. In the intended online usage scenario, the compiled application would have to be downloaded and installed in the customer’s computer. Such is a relatively common practice – typically it is what happens with Java applications made available online. However, the idealized experience of using the design system online should be as simple as accessing a website. An alternative would be to translate the model component into Javascript, enabling it to be embedded into and run from inside a webpage (see Figure 67, right). Translating from Racket to Javascript could be accomplished by using a third party compiler called Whalesong (Yoo & Krishnamurthi, 2013)\(^27\).

However, implementation was compromised by a number of complications, such as a delay in the development of a WebGL backend. Needed to implement the Viewer component, such backend was not yet available by the time the remaining components were implemented. Such a setback meant the prototype would not run exclusively on a browser, thus compromising the main purpose of the migration to Racket.

Another shortcoming of implementing the Racket prototype was related to user interaction. In a typical Racket workflow, the user must run the program from the beginning in order to generate different results, as opposed to Grasshopper, in which a program is compiled at runtime, meaning that the results produced by the program are always visible and updated after each change in the program. In order to apply changes to the design in an iterative fashion in the Racket prototype, a number of additional and considerably complex functionalities had to be implemented, suggesting that Racket was not an adequate solution for the intended run-time interaction. Finally, the implementation of the Controller component in HTML proved to be somewhat cumbersome, also requiring a number of workarounds in order to work properly, thereby reinforcing the inadequacy of the Racket prototype.

\(^{27}\) https://www.hashcollision.org/whalesong/
Despite its shortcomings, the Racket prototype features considerable CAD capability, since the Rosetta module wields the design functions of different CAD tools. To this extent, Rosetta is more CAD capable than Grasshopper, since GH is limited to Rhino’s CAD library. On the other hand, Racket can be considered less user-friendly than Grasshopper. Racket is a textual programming language (TPL), as opposed to a visual programming language (VPL) such as Grasshopper. TPLs typically have a steeper learning curve, and imply that its users must have background knowledge about the language, such as function syntax rules (Leitão et al., 2012). However, “TPLs are considerably more productive for dealing with large-scale and complex problems” (Leitão et al., 2012, p. 143), without suffering from performance issues like its predecessor Grasshopper. However, because Rosetta produces results in a CAD application, its performance might be clamped by the application’s own limitations. In the implemented Racket prototype, which uses Rhino as the Viewer component, one could witness an increase in processing speed by switching visualization mode from shaded, where surfaces can be seen, to wireframe, where only lines are displayed, and even more if the Rhino window were minimized. Although incomplete, development of the Racket prototype pushed the tableware design system forward. Migrating from Grasshopper to Racket was useful for debugging the system itself, as well as to translate it into a textual language, making it considerably easier to translate into another textual programming language, as it would happen later in the Unity prototype. In addition, some new features were implemented and then passed on to the next prototype developed in Unity.

**Unity**

The limitations in deploying the prototype as a web app and limitations in user interaction led to the implementation of a third and final prototype of the design system using a tool that addressed them both: the Unity game engine (see Figure 69).

Unity\(^{28}\) is a cross-platform game engine, mainly used for developing video games, but it can be used to create other types of application. Nevertheless, since it is primarily oriented for game development, it features a strong emphasis on interaction. Moreover, it can easily compile

\(^{28}\)https://unity3d.com/
projects into web applications. In fact, projects developed in Unity can run on many different platforms, for both workstations and mobile devices, namely Windows, MacOS, Android or iOS – and browsers. In Unity’s later versions, projects can be compiled in HTML5, the current standard platform for web applications (Kril, 2014). This means that the most recent browsers can play Unity applications directly as if they were Web pages. Such features overcome limitations found in the previous prototype, and fit perfectly the project’s original intentions.

Figure 69 – The Unity prototype of the tableware design system

Unity’s versatility rendered it as suitable tool for implementing the tableware design system. However, its advantages would be counter-balanced by an obvious disadvantage: Unity does not have a native CAD library, besides a small set of geometric primitives such as cubes, spheres, or cones, thereby making Unity the weakest technology in terms of CAD capability when compared to its predecessors. Such limitation is natural because as a game engine its primary focus is on interaction, rather than generating geometry in real time. In a typical Unity development workflow, 3D models are created using a third party modeling application like Rhino, and then imported into Unity. However, generation of geometry in real time was needed for implementing the design system.

Such need was shared with Tecton 3D (T3D), a research project that focused on finding new interaction paradigms directed at architectural design. The T3D research team, composed of both
computer scientists and architects, developed prototype 3D modeling applications for immersive Virtual Reality (VR) environments, specifically aimed at the use by architects in early stages of the design process (Mateus et al., 2015). The T3D prototypes were developed in Unity, due to its potential for user interaction. However, there was no intention of limiting the creativity of target users to platonic solids and so one of the tasks in the project was the implementation of additional geometry representations in Unity, including curved surface geometry such as NURBS surfaces. The custom-built functionality developed in the scope of T3D was used in the implementation of a Unity prototype for the mass customization of ceramic tableware.

Unity Asset Store provides some options for extending Unity’s geometry generating capabilities, such as plug-ins that generate custom procedural geometry, waiving the need to model geometry in a third-party application. However, such tools are not able to generate geometry at run-time, but rather to create new shapes in Editor mode. Another option would be to adapt an existing function library, such as three.js (mrdoob, 2010), which enables run-time generation of 3d models in a web browser. Among other 3D modeling functions, three.js implements NURBS surfaces (see Figure 70).

Figure 70 – NURBS curve and surface example in web browser using three.js (mrdoob, 2013)

29 https://www.assetstore.unity3d.com/en/#!/list/28330-procedual
Since three.js is a Javascript implementation and because UnityScript is based on Javascript, we considered the possibility of adapting these libraries to Unity. However, because UnityScript is not exactly the same as Javascript, three.js was not directly usable, and would have to be extensively edited. Considering that revising the whole three.js implementation would be a time-consuming and cumbersome task, and that the TDS only needed a small set of 3D modeling functions, namely lofts and sweeps, it was decided to implement only the needed functions from the ground up.

Because the Unity prototype is actually built into executable applications, its architecture should be analyzed in its two stages: development and compilation. The Unity prototype was developed and tested within the Unity Editor, which is considered as the IDE. In this case, all of the MVC components are embedded in the IDE itself (see Figure 71, left). Typically, developing a Unity application implies articulating a set of objects, to which certain behaviors are assigned. Such behaviors are defined by attached scripts, which can be written in one of three programming languages: UnityScript (a proprietary derivation of JavaScript), C#, and Boo. In implementing the tableware design system, scripts were written in C#, so that they could be integrated into other T3D prototypes that had been previously developed in C#.

![Diagram showing the implementation architecture for the Unity prototype: during development (left) and after compilation (right)](image)

Figure 71 – Implementation architecture for the Unity prototype: during development (left) and after compilation (right)

When compiled, Unity applications are built as HTML5 applications through translation into JavaScript (independently from whether the scripts used in the development phase are written in UnityScript, C# or Boo), and make use of WebGL for displaying the solutions (Unity3D, 2016a). The compilation process generates all the files needed to run the app as a web page and,
therefore, all the corresponding MVC components are provided by the web browser, as initially intended (see Figure 71, right).

Regarding user friendliness, Unity is between Grasshopper and Racket. Although it relies on textual programming languages to implement behaviors, Unity is equipped with devices such as the Hierarchy Window, which enable developers to see, access, and properly organize the many entities of an application. Moreover, Unity is keen on user interaction, which can only add to its user-friendliness on the actual end-user’s side. In terms of performance, Unity relies on textual programming languages, making it possible to develop relatively complex programs. Moreover, implemented applications run without any significant delays both in the editor and as compiled versions, either online or as standalone applications. Its versatility in compiling applications into a number of different platforms renders Unity the front-runner of the three prototypes in terms of portability.

A summarized comparison of the three implementations using the four criteria outlined at the outset of this section is shown on Table 7.

Table 7 – Comparative evaluation of the three prototypes

<table>
<thead>
<tr>
<th></th>
<th>portability</th>
<th>CAD capability</th>
<th>performance</th>
<th>user-friendliness</th>
</tr>
</thead>
<tbody>
<tr>
<td>GH</td>
<td>LOW: Dependable on third party applications</td>
<td>HIGH: Link to Rhino CAD library</td>
<td>LOW: Poor in complex models</td>
<td>HIGH: Visual programming language</td>
</tr>
<tr>
<td>Racket</td>
<td>MEDIUM: WebGL set back</td>
<td>HIGH: Link to many CAD libraries</td>
<td>HIGH: Escalates well</td>
<td>LOW: Textual programming language</td>
</tr>
<tr>
<td>Unity</td>
<td>HIGH: compiles executables into multiple platforms</td>
<td>LOW: No CAD library; functions had to be implemented</td>
<td>HIGH: Escalates well; applications respond quickly</td>
<td>MEDIUM: Textual programming language; but has Visual Hierarchy</td>
</tr>
</tbody>
</table>

In the next sections, we focus on the final prototype developed in Unity, yet refer to the influence of the preceding two prototypes on the implementation design. Documenting the Grasshopper
prototype is important to understand the extent to which the shape grammar influenced the implementation of the design system. However, as discussed in the previous chapter, the shape grammar was adapted while being converted into parametric models, which was reflected on the implementation of the Racket and Unity prototypes.

4.2 The Grasshopper prototype

Let us introduce the implementation of the Tableware Design System (TDS) by going back to the Tableware Shape Grammar (TSG). As we have seen in the previous chapter, TSG holds rules for the design of tableware collections.

Despite being considered computational models, shape grammars are not necessarily implemented on a computer. In fact, they can be conceived, developed and tested by hand using pencil and paper. However, as a shape grammar grows more complex, it becomes impractical to test it by hand, requiring a computer implementation. To implement a shape grammar on a computer is not a trivial task. Such implementations, called shape grammar interpreters, need to perform two different tasks: a) to recognize shapes, and b) to operate on those shapes (Chau et al., 2004). Since automatic shape recognition was beyond the scope of the project, the TSG was translated into parametric models in Grasshopper (GH), enabling a user to test the shape grammar by selecting and applying its rules.

From shape grammars to parametric models

The conversion of a shape grammar into parametric models approach was adopted from Digital Alberti, where a shape grammar encoded the rules for designing churches according to Alberti (Figueiredo, 2016). Church designs can be generated by applying rules of the shape grammar recursively, creating what is called a derivation, like the one shown in Figure 72. Being a parametric shape grammar, each derivation has the potential to generate a family of different solutions that can be obtained by manipulating the parameters in the rules. To this extent, a derivation can be considered a parametric model, and consequently can be translated into a Grasshopper program.
Parametric models are implemented in Grasshopper by articulating components that perform operations, typically but not exclusively geometrical, which affect the shape of the final design. Some of these operations depend on parameters and so, by assigning different values to these parameters, we can produce variations of the final design. Considering a derivation as an articulation of specific shape grammar rules, these rules were translated into programming modules in GH (see Figure 73).

![Figure 72](image1)

Figure 72 – One possible derivation tree for the churches shape grammar (Figueiredo et al., 2013)

![Figure 73](image2)

Figure 73 – Grasshopper implementation of some rules (Figueiredo et al., 2013)

The set of implemented rule modules corresponds to a design system that enables a user to generate derivations of Albertian churches by combining the rule modules and by assigning different values to its parameters. Because the implemented design system cannot perform shape recognition, it is left to the human designer to recognize when and to which design elements to apply the grammar rules. Therefore such a system cannot be considered a true shape grammar interpreter. We can however consider the parametric models produced using the system to be implementations of derivations following the shape grammar rules (Figueiredo et al., 2013).

The same strategy was adopted in the Tableware Design System, by translating rules from the Tableware Shape Grammar into rule modules that execute the corresponding shape operations. However, the implementation of the rule modules in the TDS was different from the one in the
Digital Alberti grammar. Whereas the rules of the church shape grammar were translated into groups of components, rules in the TDS were translated into clusters. The difference is clarified below.

Presently in GH, groups of components can be transformed into custom components through clustering. Custom components work like standard GH components: they accept one or more inputs, and perform a particular operation that results in an output, which typically but not necessarily depends on the input. In custom components, the operation is defined by the developer as a GH definition. Clustering takes advantage of abstraction by storing and hiding code inside a component, allowing for a visually cleaner, more elegant program. Another advantage of clustering is that copies of the resulting custom component are instances of each other, which means that changes made in one of the instantiated copies are replicated in all the other instances of the clustered component. Nevertheless, each instance can generate different results, depending on the input. Therefore, clusters are the GH equivalent of subroutines, or functions, in traditional programming languages.

Instantiation of clusters is a relatively new feature in GH and an important one. By the time the church design system in DA was implemented, instantiation of clusters was not available, so groups were used instead (see Figure 73). Groups of components could be clustered into custom components, allowing for visually tidier programs, but copies of those custom components were independent of each other, and changes in one copy were not replicated in the others. In DA, this did not present a problem because the shape grammar rules had been defined previously and their translation into GH did not require alteration when implementing the derivations. However, cluster instantiation was crucial to the implementation of the TDS, since rules were translated into GH clusters while the shape grammar was being developed (see Figure 74) and, therefore, were more prone to changes. Moreover, as the number of collections increased, so did the complexity of the TDS and so the number of rule instances increased exponentially.
An atomic representation of tableware elements

To understand how rules were translated in GH clusters, we should refer back to the previous chapter, namely to another strategy used in DA, the deconstruction of the Albertian column system. Likewise, tableware collections were also deconstructed into several hierarchical levels: collections are composed of elements, which are composed of pieces, which are composed of parts. Such deconstruction was used in some rules of the Tableware Shape Grammar. For example, Rule 2 subdivides a quadrilateral envelope representing an element of a tableware collection – in this case a soup plate – into three different envelopes, each representing a different part of the element.

These envelopes represent the boundary of sections of objects on any level of the hierarchical structure. For example, as shown in Figure 75, both the high-level element and each of its parts are represented by envelopes. Despite representing objects of different levels, envelopes are similar in shape, which is determined by the object’s properties, namely its dimensions such as height and width. Moreover, as we have seen in the Tableware Shape Grammar (TSG), the left
side of rules in the initialization and base shape definition feature a similar envelope, suggesting that they can be subjected to similar operations. These two aspects led to the conceptualization of such generic envelopes as a class of objects, borrowing a concept from Object Oriented Programming:

“Classes describe the type of objects, while objects are usable instances of classes. So, the act of creating an object is called instantiation. Using the blueprint analogy, a class is a blueprint, and an object is a building made from that blueprint.

[...]

Each class can have different class members that include properties that describe class data, methods that define class behavior” (Microsoft, 2015)

Such class was named Plaxel (capitalized), which stands for PLAte ELeament\textsuperscript{30}. Objects of this class are called plaxels (non-capitalized), and can represent the base shape of any object of the TDS, from top to bottom of the hierarchical structure. The strategy of thinking about objects of the design system as instances of a class was also borrowed from DA. In fact, the Plaxel class is a derivation of a class that had been implemented for the parametric models of the Albertian column system. The predecessor class of the plaxel was called coxel, which stood for COLumn ELeement (Castro e Costa, 2012) (see Figure 76).

Figure 76 – GH code for the parametric model (left) and the corresponding digital model (right) of a Doric capital (Castro e Costa, 2012)

Moreover, although the conceptual class was renamed Plaxel, the code for its implementation is the one originally developed for the Coxel class in the Digital Alberti project – in fact, the file being used is called Alberti.dll. However, for the sake of simplicity, from now on both the conceptual and the implemented classes will be referred to as Plaxel.

\textsuperscript{30} Similarly to \textit{pixel} (Picture ELeement) or \textit{voxel} (VOlume ELeement), terms used in computer graphics.
Thinking about these objects as instances of a class, as well as developing a symbolic representation for the Plaxel class, helped rationalizing the TDS, and thereby simplifying its implementation into GH parametric models. However, GH does not provide direct support for creating classes, or other custom data types. Nevertheless, GH does allow its users to create custom components, either through clustering, as we saw before, or via scripting using textual programming languages. Natively, these so-called scriptable components can be written in either VB.NET or C#.NET. Moreover, additional languages are supported via third party plug-ins, such as Python, for example.

The Plaxel class was implemented in VB.NET, a textual programming language, developed by Microsoft as a successor of Visual Basic. VB.NET was preferred over C#.NET or Python due to previous knowledge of other Basic-related programming languages, including Visual Basic, but also for being an object-oriented programming language, which makes it particularly suited for implementing classes. The Plaxel class implementation was developed in Visual Studio.

**Geometry and behavior of a plaxel**

The Plaxel class is defined by its properties, related to the geometrical attributes, and its methods, which allow operating the instantiated plaxels, defining their behavior. Let us begin by analyzing the geometry of its shape. Although the shape description of a tableware element (TWE) can be divided into base shape and decoration, the Plaxel pertains exclusively to the base shape of TWEs. The shape of a plaxel corresponds to a three-dimensional object, namely a double-curved Bézier surface. Such surface is generated by, and described in terms of,

- its **profile**, which can be better visualized in section view, such as in the base shape derivation in Figure 77, and
- its **contour**, which corresponds to its horizontal shape, and can be better visualized in plan view, such as in the decoration derivation in Figure 78.

31 [http://www.food4rhino.com/app/ghpython](http://www.food4rhino.com/app/ghpython)
For visualizing the generation of a plaxel’s surface, let us think of a plate with a round contour, such as the one depicted in Figure 77 and Figure 78. The plaxels of such round plate can be generated by revolving their profile around the plate’s radial axis. However, such approach would limit the potential of the design system to circular tableware designs, ruling out many other interesting design solutions – think for example of the ones produced by the art deco or the art nouveau movements (see Figure 79).

The same problem arose in DA. In the first iterations of the Coxel class, the corresponding surfaces were generated by revolving profiles around the column’s axis (see Figure 80, left). However, such option would not be able to generate non-circular elements, such as the abacus in the Doric capital (see Figure 76). As an alternative, the Rail Revolve operation was tested in Grasshopper, which is similar to generating a surface by moving a profile around an axis along a closed curve, which is not necessarily circular. However, Rail Revolve would not be able to generate linear shaped entablature elements, and therefore such option was also discarded.
Therefore, in the Grasshopper implementation of both Coxel and Plaxel, the corresponding surfaces are generated through a loft operation using horizontal closed curves instead of vertical profiles (see Figure 80, right).

![Figure 79 - Non-circular tableware from Art Deco (left: Tricorne collection, design by Don Schreckengost, manufactured by the Salem China Company) and Art Nouveau (right: collection manufactured by Villeroy & Boch, Mettlach factory)](image)

Figure 80 – Generating the same shape using different operations: revolve (left) and loft (right)

Please note that the loft operation used in Rhino for generating a plaxel uses the *loose* option. A loose loft presents a similar behavior to a Bézier curve, whereas a normal loft behaves somewhat like an interpolated curve (see Figure 81). In Rhino, a user defines a Bézier curve by specifying its control vertices (CVs). However, by definition, the Bézier curve does not contain all of its CVs, just the first and the last one. Similarly, a loose loft only contains the first and last input curves, whereas the remaining curves, despite containing the necessary CVs for defining the lofted surface, are merely used to inform the loft creation. On the other hand, a normal loft in Rhino behaves like an interpolated curve, in which the points defined by the user are contained on the curve. Similarly, in a normal loft, all curves are contained in the resulting lofted surface.

---

32 Images from Etsy.com
For defining the loft that corresponds to a plaxel’s shape, let us look at the example in Figure 82. Since a plaxel is a double-curved NURBS surface, its profile results from sectioning such surface with a vertical plane containing its central axis, corresponding to a quadratic Bézier curve (see Figure 82, right). A plaxel’s profile is inscribed in a convex quadrilateral called envelope. In the plaxel shown in Figure 82, the corresponding envelope is rectangular, and it is represented by dotted lines in the section on the right. Originally, in the Grasshopper prototype, a plaxel is characterized by attributes related to the dimensions of its envelope. The dimensions of the envelope’s rectangular shape correspond to two of the plaxel’s attributes: its relative height (δH) and its relative width (δW). The remaining attributes refer to the plaxel’s absolute position, namely its distance to the axis (Wo) and its distance to the horizontal plane on which the tableware element is sitting, like a table for example (Ho).

These four attributes correspond to the main properties of the Plaxel class, but they relate to dimensions in section. However, we have decided to use horizontal closed curves instead of vertical profiles. We call these auxiliary horizontal curves guides. For understanding the
generation of such guides, let us again consider the circular plate as the simplest example. The guides that generate the surface of plaxels for a round plate are circles (represented in blue in Figure 82, left). Each of these circles is contained in a horizontal plane since, by definition, guides are horizontal curves. Moreover, the center of all circles is positioned in the plate’s central axis, a vertical straight line that intersects the parallel horizontal planes (represented in green in Figure 82, left). Therefore, each of these circular guides is defined by two parameters: the height of the plane it is contained in, and its radius. In the 2D section, the guides correspond to the vertices of the envelope’s rectangle.

The parameters for generating each guide are extracted from the values of the plaxel’s properties. In 3D space, the first guide (A) is generated according to the plaxel’s coordinate for location \((H_0, W_0)\), by generating a circle in the horizontal plane with height equal to \(H_0\), and with radius equal to \(W_0\). The second guide (B) is generated by offsetting guide A by the value of \(\delta W\), whereas the third guide (C) is generated by vertically translating guide B by the value of \(\delta H\). Finally, the fourth guide (D) is generated by vertically translating guide A the same way as for generating guide C. In the 2D section, each offset corresponds to a horizontal translation, while vertical translations are the same in 2D and 3D, and the coordinates for the four guides can therefore be calculated by the following ordered pairs:

\[
\begin{align*}
A &= (H_0, W_0) \\
B &= (H_0, W_0 + \delta W) \\
C &= (H_0 + \delta H, W_0 + \delta W) \\
D &= (H_0 + \delta H, W_0)
\end{align*}
\]

Also in the 2D section, we can see that the plaxel’s guides correspond to the four vertices of that rectangle. However, not all the guides are used for lofting the plaxel’s surface. In Figure 82, the surface’s profile corresponds to a quadratic Bézier curve whose shape is determined by its control vertices (CVs), corresponding to guides A, B, and C. Having the four guides will nevertheless provide plaxels with some flexibility.

\[33\text{In fact, the contour of the tableware elements generated by the Grasshopper prototype was actually limited to round shapes, and so the generation of non-circular tableware designs will only be addressed in the section dedicated to the final prototype.}\]
Regarding its behavior, the Plaxel class defines a set of methods that can perform specific operations to an instantiated plaxel, including:

- creating a plaxel (createPlaxel);
- distorting plaxels, by changing its parameters (editPlaxel);
- subdividing one plaxel into smaller, sequential plaxels (subdivPlaxel); and
- rendering a plaxel, i.e. replacing it by its corresponding surface (renderPlaxel).

The createPlaxel operation creates an instance of the Plaxel class. Such instance is in fact an abstract entity, which consists of information pertaining to a potential shape. Besides creating a new instance, createPlaxel also sets the attributes of the newly created plaxel to arbitrary values. The editPlaxel operation enables the alteration of a specified attribute of a given plaxel instance. Since most of a plaxel’s attributes relate to its shape, it is fair to say that the editPlaxel method provides a means to distort the affected plaxel. The subdivPlaxel operation replaces a given plaxel by a number of smaller plaxels, according to parameters provided by the user, in the form of a list of numbers.

Finally, the renderPlaxel operation generates a CAD object, thus enabling the user to actually see the shape of the current plaxel. Recall that before being rendered, a plaxel instance is fundamentally an abstract entity, consisting of information about its attributes’ values. The term “render” is borrowed from computer graphics - “render v.t. […] 2 : to convert a graphical primitive into individual pixels.” (Latham, 1995, p. 115) The plaxel primitive is rendered into a surface, rather than into individual pixels. Such surface corresponds to a loft, generated by the corresponding Grasshopper commands, invoked through VB.NET, according to the plaxel’s attributes.

Recall that the plaxel’s attributes define first and foremost the shape of its envelope, and subsequently the coordinates of its four guides. However, the guides by themselves are not sufficient to define the plaxel’s shape, since many shapes could be generated from lofting all the possible combinations of those four guides. For the renderPlaxel operation, we narrowed the admissible shapes down to sequences of adjacent vertices of the envelope (see Table 8). Therefore, besides the plaxel’s attributes, the plaxel’s shape depends on additional parameters, such as the number of envelope sides used (N), the type of resulting geometry (D, pertaining to a
loose or straight loft), and the sequence in which guides are used (R, standing for rotation). These parameters must be fed to the renderPlaxel method, for it to render the plaxel instance.

Table 8 – Different possible shapes for the same plaxel (adapted from (Castro e Costa, 2012))

Despite being written in a textual language, the Plaxel class, its instances and methods could be used in GH through the referred scriptable components. A different component was created for each Plaxel method, containing a simple VB.NET script that invokes the corresponding method, passes the component’s input as the method’s arguments, and passes the method’s output as the component’s output (see Figure 83). Scriptable components could, therefore, act as containers for Plaxel methods, and be subsequently connected to other GH components. We could then make the most out of the visual nature of Grasshopper, while at the same time taking advantage of the class approach.

Finally we could translate rules into GH clusters. Figure 83 shows the implementation of Rule 2a of the Tableware Shape Grammar (see Figure 84). The Grammar rule divides an envelope into three smaller envelopes placed diagonally, with particular functions. The GH cluster divides a plaxel using the custom subdivPlaxel component, then selects the diagonally placed plaxels via standard List Item components, then edits the names of the resulting plaxels via custom editPlaxel components, and finally it generates the corresponding surfaces via custom renderPlaxel components.
Figure 83 – Grasshopper VB.NET component operating the editCoxel method (renamed editPlaxel), implementing Rule 2a of the Tableware Shape Grammar.

Figure 84 – Rule 2a of the Tableware Shape Grammar

**Generating tableware collections**

So far, we have looked into the finer structure of the Grasshopper implementation, which generates the smaller elements of the TDS. Let us now look at the higher levels of its hierarchical structure, namely TWEs, and collections. As seen in the previous chapter, development of the TSG began with the analysis of a single collection, Romantica, during which rules were designed and implemented in GH. These initial GH rules were assembled into a parametric model of one of its types, the soup plate, replicating its shape grammar derivation. Such derivation consists of a sequential application of the shape grammar rules (see Figure 85). As described in the section
entitled “From shape grammars to parametric models”, shape grammar rules have a direct correspondence to GH groups or clusters containing components that manipulate plaxels (see Figure 86). Therefore, by combining such plaxel-manipulating groups in the same order as the rules in the derivation, we can obtain the same resulting TWE (see Figure 87).

Figure 85 – Derivation of the base shape of a soup plate (Castro e Costa & Duarte, 2013)

Figure 86 – Parametric model of the base shape of a soup plate

Figure 87 – Generated digital model of the base shape of a soup plate
In this section, generation of a tableware collection and its different phases has been illustrated by one of its types, the soup plate. However, according to the strategy adopted in the shape grammar development, all types of a collection can be described as variations of the same initial shape. Therefore, the parametric model into which the Romantica collection was translated is capable of generating all the analyzed types: charger, dinner and soup plates, cereal bowl, mug and cup. One model representing different types meant that these types are represented by parametric variations of a more generic meta-type. Such approach has been adopted throughout the whole implementations of the design system. In the first Grasshopper prototype, two meta-types were actually implemented, one comprising the shallow types, and one the deep types (see Figure 88, left). Shallow types comprise the charger, dinner and soup plates, whereas deep types comprise the bowl, the mug and the cup. In the mug and cup, the parametric model only represented the main body, since the elements’ handles had not yet been encoded (see Figure 88, right).

![Figure 88 - Grasshopper parametric model of the first collection (left), and resulting digital models (right)](image_url)

The separation into two meta-types is reflected in the diverging branches of the implemented Grasshopper model. Regarding shape, the main difference between the shallow and deep meta-types resides in the base shape of the generated elements: besides the base plaxel, shallow types featured a concave border plaxel, whereas the deep types feature a so-called ‘wall’ plaxel, following the containing part. Another difference was where decoration could be applied: in shallow types, decoration could be featured on the inside of the border plaxel, whereas in deep types, it could be featured on the outside of the wall plaxel.
Looking back at the Grasshopper implementation of the design system, we can identify the following hierarchical relationships among the different design elements (see Figure 89):

A. The Plaxel class was written in VB.NET using Visual Studio 2010, defining properties and methods for a new type of object, the plaxel;
B. Plaxel class methods for manipulating instantiated plaxels are implemented in Grasshopper through its custom components into elemental design operations;
C. Such elemental operations are grouped and clustered together into more abstract, higher order operations, which correspond to shape grammar rules;
D. Finally, such shape rules are combined together into parametric models corresponding to types and meta-types.

Implementing decoration

The Grasshopper implementation of the TDS focused not only on the base shape of the analyzed collections but also on its relief-based decoration. As in the shape grammar derivation, the decoration phase occurs after the base shape of a TWE is determined. However, the decoration phase does not take advantage of the Plaxel class, since the purpose of this class is limited to the development of the base shape, which is accomplished once plaxels are rendered into NURBS surfaces. Such surfaces are subsequently detailed following the application of decoration rules of the shape grammar (see Figure 90).
Contrary to the base shape rules, decoration rules were implemented using standard GH components, instead of the custom-made, plaxel-manipulating components. Nevertheless, such as in the plaxel rules, the principle of modularity was maintained, by grouping the GH standard components to match the rules concerning decoration. Modularity was particularly useful in the subsequent development of the shape grammar, when five additional collections were analyzed (see section entitled “Generic shape grammar from multiple collections”).

Accordingly, the GH prototype was extended based on the parametric models of the single-collection and on the new rules inferred from analyzing the new collections. Likewise, new parametric models were implemented by combining GH component groups that matched shape grammar rules. Such parametric models corresponded to the additional collections of ceramic tableware and later to original collections (see Figure 91).

Recurring use of the custom components in the extended GH prototype prompted the creation of a custom toolbar (see Figure 92) that facilitated the instantiation of such components, by avoiding copying them across the GH definition. Such toolbar comprised operations regarding the base shape (black buttons) as well as the relief decoration (yellow buttons). This tool bar was the seed for the development of a graphical user interface (GUI) aimed at the designer.
Figure 91 – Implemented parametric models of both existing and original collections (left) and resulting digital models (right)

Figure 92 – Customized Grasshopper toolbar for tableware design

4.3 The final prototype

In this section, we present the current state of the Tableware Design System implemented in Unity and, when appropriate, compare it with its previous iterations. Although this implementation is the final one in the scope of this dissertation, there is room for improvement.

The implementation of the Tableware Design System underwent significant changes from its first iterations in Grasshopper to its current state implemented in Unity and C#, including an intermediate implementation in the Racket programming language. Racket and C# are textual programming languages (TPL), as opposed to a visual programming language (VPL) like GH. Migrating to a TPL meant to waive the previously referred to advantages of a VPL, and required
a relatively literal translation of the VB.NET code that supports the Plaxel class, but also the less obvious translation of GH’s graphical components and connections into actual lines of code. However, it also meant to take advantage of the scalability provided by TPLs, the same scalability that had compromised GH as a fit solution for implementing the design system.

Besides being written in a TPL, the Unity prototype results from additional changes regarding its predecessors, namely in the way the shape of tableware elements is represented and manipulated. Moreover, the current prototype features a number of new functionalities that were deemed necessary, or at least useful, during implementation of the iterations subsequent to Grasshopper, such as non-circular contours, or collection-wise editing. It should be mentioned that most of these changes have addressed the base shape and decoration outline of tableware elements, whereas some of the decoration rules, namely the ones based on relief, have not been implemented in the final prototype.

The Unity prototype of the TDS served as the foundation to developing two distinct main applications, according to the Design Participation Model presented in the previous chapter. The *Modeler* application (see Figure 93) targets tableware designers, enabling them to create customizable tableware collections. The *Navigator* application targets end-users, enabling them to customize tableware collections.

![Figure 93 – The Modeler application running in Unity](image)

In this chapter, we mainly focus the implementation of the Modeler, whereas the Navigator application will be further addressed in the next chapter about the User Interface, where we discuss methods of eliciting the client’s wishes.
**From plaxel dimensions to guide coordinates**

The final implementation encompassed a paradigm shift in the way the shapes of tableware elements (TWEs) are represented and manipulated. In the first iterations of the TDS, a plaxel’s shape was defined by parameters corresponding to the dimensions of its envelope, such as Ho, Wo, δH, and δW. This design option was inherited from DA. For certain elements, such as the torus, two differential dimensions, δH and δW are enough to define the corresponding plaxel. But in the case of the scotia two differential widths are necessary. The scotia links the two tori in the Doric base and so must take into account their different radii (see Figure 94).

![Figure 94 - Plaxel representing the scotia in the Doric base (adapted from (Castro e Costa, 2012))](image)

Consequently, a plaxel representing the scotia depends on one differential height (δH) and two differential widths: δW0 for the bottom, and δW1 for the top. This approach was used in the first two implementations of the TDS, but in the third implementation it was shown to be inadequate when applying deformation operations. It was then decided to describe a plaxel through the coordinates of its guides, instead of the coordinates of one guide and many differential dimensions (see Figure 95).

Another limitation of the initial implementation of the TDS was the fact that a plaxel shape was determined by a set of four guides, constraining it to relatively simple shapes. Should a plaxel feature more complex shapes, it had to be subdivided into a number of smaller plaxels. The subdivision approach had been inherited from the Digital Alberti project, but presented two issues:

- Representing a part with a complex shape required multiple plaxels, which implied using the two hierarchical levels “part” and “plaxel” of the tableware structure, but if one of these levels could be eliminated, the complexity of the hierarchical structure could be reduced;
• An external mechanism was necessary to ensure continuity between the surfaces of two subdivided plaxels.

Such limitations were overcome by allowing a plaxel to be defined by more than four guides. Plaxel definition required a minimum of two guides, but there was no maximum. The profile of a plaxel could thus be controlled with multiple control vertices, as in Bézier curves. One of the reasons that supported this change was that some designers actually use Bézier curves to model the profile of TWEs, as observed during the Contextual Inquiry.

Figure 95 – Comparing plaxel description using dimensions and guide coordinates

So, in the final implementation, a plaxel’s shape is controlled by the coordinates of its guides. This new approach induced writing a number of specific functions for manipulating guides. Some of the functions used previously to manipulate plaxels were adapted to manipulate guides. For example, in the initial two implementations a plaxel’s differential width δW could be changed through an editPlaxel function used to change the plaxel’s corresponding attribute, whereas in the final implementation such effect is obtained by changing the attributes of the plaxel’s guides. New guide-manipulating functions provide greater control over the plaxel’s shape and, at the same time, enabled increased modularity of the TDS code, thereby simplifying it.
Adopting the multi-guided plaxel addressed the two issues referred to above: a Part can now be composed of only one plaxel, thereby ensuring positional and tangential continuity. Should the designer wish to create a discontinuity in a specific Part, it can be split into two or more Plaxels. However, it is no longer necessary to subdivide a Part in order to increase the complexity of its shape. This change also motivated the implementation of new operations on plaxels (see Figure 96), which rendered the subdivision operation unnecessary:

- to add or delete guides, to increase or decrease the complexity of the plaxel’s shape;
- to split one plaxel in two at a given guide, as well as merge two plaxels into one, provided that they share a guide.

![Figure 96](image)

Figure 96 – New operations in the multi-guide approach: (a) add guide, (b) delete guide, (c) split plaxel, and (d) merge two plaxels.

Another functionality included in the final implementation is the direct manipulation of guides. In the preceding prototypes, a user changed the profile shape of a plaxel numerically, by applying operations that manipulated coordinates of its guides according to numeric values. The Unity prototype enables the user to directly and visually manipulate the coordinates of guides, by dragging and dropping handles in 3D space (see Figure 97). The movement of these handles is actually constrained to a plane that contains the vertical axis of the designated TWE, which we call the handle plane, and dragging is accomplished by projecting the mouse position onto the handle plane.

![Figure 97](image)

Figure 97 – Direct guide manipulation in the Unity prototype
Non-circular contour

Let us focus now on the description of TWEs shapes from the top view. We have previously stated that the TDS should be able to generate TWEs with horizontal contours other than circular. This problem was addressed in the implementation of the final prototype by using a more versatile shape, the superellipse, to define the contour of TWEs.

The use of a superellipse to describe the contour shape was inspired in the implementation of the Lamé curve in Racket (Leitão, 2014). The Lamé curve is a generalization of the ellipse to \((\frac{x}{a})^n + (\frac{y}{b})^n = 1\) and it has been used by designers like Piet Hein and Gerald Robinson (Gridgeman, 1970). This curve has the virtue (and the limitation) of being able to represent both an ellipse and a rectangular shape as well as an infinite number of shapes in between, by varying its parameters \(a\), \(b\), and \(n\). In reality, the Lamé curve cannot produce the right angles of a rectangle, which is not a limitation in the context of this research since in ceramic design it is difficult, and often not even desired, to produce hard edges. The Lamé curve is a particular case of the superellipse, which is generalized to polar coordinates (Gielis, 2003; Weisstein, 2014), increasing shape flexibility. The superellipse corresponds to a wide variety of shapes according to parameters \(a\), \(b\), \(m\), \(n_1\), \(n_2\), and \(n_3\) in the following relation of radius in function of angle, dubbed “superformula” by its inventor (Gielis, 2003):

\[
r(\theta) = \left[\left(\frac{\cos(\frac{1}{4}m\theta)}{a}\right)^{n_2} + \left(\frac{\sin(\frac{1}{4}m\theta)}{b}\right)^{n_3}\right]^{-1/n_1}
\]

However, the relationship between the variation of these parameters and its effects on the shape of the resulting curve is not straightforward. The effects of varying the value of \(m\) are easily identifiable: the formula generates curves with \(m\)-fold rotational symmetry. However, the same cannot be said for the other parameters. For example, while for \(m = 4\), \(a\) and \(b\) are related to the proportion between the length and width of a rectangle or ellipse, for different values of \(m\) such relation does not apply, since the generated curve is no longer a rectangle or an ellipse.

Despite this slightly erratic behavior, the superellipse presented many advantages for the design system, namely:
• it can be described parametrically as a function of the angle, rendering it appropriate for a representation in polar or cylindrical coordinates;
• it can produce curves with different degrees of symmetry;
• it produces curves that are continuous and periodic;
• it can generate a wide range of shapes described by only 6 parameters.

Table 9 – Sample of abstract shapes obtained by modifying the parameter of the superellipse for positive integer rotational symmetries \( m \) from 0 to 8 (Gielis, 2003)

<table>
<thead>
<tr>
<th>Rotational symmetry ( m )</th>
<th>( m = n_2 = n_3 = 1 )</th>
<th>( n_1 = 1000 )</th>
<th>( n_1 = n_2 = n_3 = \frac{1}{2} )</th>
<th>( n_1 = 30 )</th>
<th>( n_1 = 80 )</th>
<th>( n_1 ) as column 3 when ( \alpha = 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The supershape was implemented in the TDS at the guide level. Instead of corresponding to a horizontal circle, a guide is henceforth a horizontal supershape. In the final prototype, the designer controls the supershape through a set of sliders that define its parameters \( a \) through \( n_3 \) (see Figure 98). However, the non-linear correspondence between parametric variation and its effects in the curve’s shape justifies thinking about alternative modes of interaction, which will be addressed in the next chapter on the User Interface.
In the previous prototypes, TWEs are represented as NURBS surfaces, generated at run-time by lofting the guides of a plaxel in Rhino. However, by default, run-time geometry generation in Unity is limited to simple geometric primitives, namely cubes, spheres, capsules, cylinders, planes and quads. Through scripting however it is possible to generate additional geometric shapes through the use of mesh objects. Meshes are the only way to represent 3D geometry in Unity. According to Unity’s reference, “aside from some Asset store plugins, Unity does not include modeling tools. Unity does however have great interactivity with most 3D modeling packages. Unity supports triangulated or Quadrangulated polygon meshes. NURBS, Nurms, Subdiv surfaces must be converted to polygons.” (Unity, 2015) Unity allows the creation of custom polygonal meshes through scripting by means of the Mesh class in its API. The geometrical information necessary for describing a mesh follows the face-mesh data structure (Smith, 2006), consisting of an ordered list of vertices, each represented by its 3D coordinates, and a list of faces, each represented by three integers, corresponding to vertices in the vertex list.

To enable run-time generation of double curved geometry in Unity, an algorithm was implemented to generate the mesh corresponding to a loft using the plaxel’s horizontal guides. Contrary to the NURBS representation, the number of guides that define a plaxel’s shape is not enough to guarantee a smooth mesh. Therefore, it was necessary to determine a set of additional curves through which the surface is lofted. Let us call such curves sub-guides. The coordinates for each sub-guide are interpolated from the coordinates of the actual guides, in the same way points in a Bézier curve are interpolated from its control vertices (see Figure 99).
Instead of tracing the profile curve of the plaxel using the obtained set of interpolated coordinates, these will be used to generate sub-guides that, like guides, correspond to super-elliptical shapes. Then, each superellipse is sampled according to the superformula. In summary, the points of the plaxel’s lofted surface follow the following cylindrical representation:

\[
\begin{aligned}
    x &= W \star \left[ \frac{\cos\left(\frac{1}{4}m\theta\right)}{a} \right]^{n_2} + \left[ \frac{\sin\left(\frac{1}{4}m\theta\right)}{b} \right]^{n_3} - \frac{1}{n_1} \right] \star \cos \theta \\
y &= H \\
z &= W \star \left[ \frac{\cos\left(\frac{1}{4}m\theta\right)}{a} \right]^{n_2} + \left[ \frac{\sin\left(\frac{1}{4}m\theta\right)}{b} \right]^{n_3} - \frac{1}{n_1} \right] \star \sin \theta
\end{aligned}
\]

in which H and W are the guides’ height and width respectively. Note that the H is associated to y instead of z, because in Unity the Y axis is associated to the vertical direction. By feeding the equation with a set of angle values \( \theta \), corresponding to the division of the trigonometrical circle by the number of desired sampled points, we obtain the set Cartesian points needed for generating the mesh.

The number of sub-guides and sampled points in each sub-guide will determine the resolution of the generated surface and, consequently, its smoothness. In general, if the curves and points are well distributed spatially across the surface, then the smoothness of the surface is proportional to the number of sampled curves and points. While setting the resolution parameters, a balance was
sought between satisfactory smoothness and program performance. In the current implementation, the number of sub-guides is set to 10 and the number of sampled points is set to 24.

Regarding navigation, the interface was kept simple, by implementing a user-controlled camera that rotates around a specified object and zooms in to and out of it. The final prototype supports mouse control, in which rotation is constrained to azimuth and zenith, mapping the horizontal and vertical pixel differential of a drag and drop operation, and zooming is controlled by scrolling the mouse’s middle-wheel, mimicking the default navigation standard in Rhinoceros. Panning was not implemented to minimize the likelihood of the user getting lost in the visualization space. The user can, nevertheless, have the camera targeting each of the different objects of the collection by pressing the directional left and right arrows.

**Datatype structure**

Guides and plaxels are considered the building blocks of the TDS. Let us now look at how they can be articulated into a customizable tableware collection. Recalling the taxonomy of tableware design presented earlier, a *collection* corresponds to the set of all available *types* in a ceramic tableware dinner set, such as the dinner plate, the cereal bowl, the tea cup, and so on. Each type comprises a set of *parts* according to its type and each part corresponds to a surface, which results from lofting its *guides*. We can articulate these concepts hierarchically as shown in Figure 100, on the left.
Such hierarchical structure is visible in Unity’s Hierarchy Window, a Graphical User Interface component that enables users to organize objects in the scene (GameObjects) by establishing parent-child dependency relationships (Unity3D, 2016c). In Figure 100, on the right, we can see the Hierarchy Window containing components of a generated tableware collection, namely the names that have been assigned to the various instances of the different hierarchical classes. For example, the object “blankCollection” represents an instance of the Collection class. Likewise, “ChargerPlate” represents an instance of a TWE, and because each generated TWE presents a different type in the collection, they are actually grouped into a GameObject named “types”. “Body” and “Border” are instances of the Part class. GameObjects of the Part class contain a set of objects named “inside”, “outside”, “top”, and “bottom”, which correspond to instances of the Plaxel class. A Part object also contains a GameObject called “handles”, which in turn contains a set of objects called “sphere”. Each of these spheres serves as a handle for manipulating the coordinates of a guide instance that controls its part’s shape.

Although a Part has four different plaxels, only the outside plaxel is defined by the part’s guides, while the shapes of the remaining plaxels are calculated from the shape of the outside plaxel and the part’s thickness, set as a constant value of 4 mm. Therefore, the Plaxel was not considered as a hierarchical level in the Unity implementation of the TDS.
Besides making the hierarchical structure explicit, it is necessary to define the behavior of each of the TDS’s objects, in the same way class methods do for objects in the earlier Grasshopper prototype. In Unity, the behavior of GameObjects is defined by programming scripts assigned to them (Unity3D, 2016b). Each script implements a class, with its properties and methods, and these methods define the behavior of GameObjects to which an instance of such class is assigned. For implementing the TDS, a class was scripted for each level in the tableware hierarchy, from collection through guide. Since the behavior for each class had been previously modelled in Racket as structured data types, the Unity scripts were translated from their Racket counterparts.

When the Unity implementation of the TDS is run, it begins by generating all of the GameObjects corresponding to every component and sub-component of a collection. To each GameObject, the corresponding script is assigned. Although every component is associated to a GameObject, not all of them are visible. In fact, only plaxels and guides will be associated with a shape, and thus visible. Plaxels are associated with meshes, which are generated by the custom loft function presented earlier, while guides are represented by black spheres, dubbed ‘handles’. By the end of the generation procedure, all GameObjects are represented in the Hierarchy Window, and the visible ones are displayed in the Game or Scene Views, where the user can see a customizable tableware collection (see Figure 101).

![Figure 101 – Generated tableware collection in Unity](image_url)
Collection-wise design

The final prototype implemented in Unity allows the user to change the design of a whole collection simultaneously, so that when a user changes the shape of a part in a specific tableware type, this alteration is propagated to all the other types (see Figure 102). Such feature, dubbed collection-wise editing, facilitates guaranteeing formal coherence among the collection’s different elements, and was considered a significant improvement from the Grasshopper prototype, in which it was only possible to combine plaxels into individual types.

![Figure 102 – Collection-wise editing: changes made in one element are reflected in all other elements](image)

Implementation of collection-wise editing implied a unifying mechanism that would enable variation in a manageable fashion. Going back to the shape grammar development, it had been suggested that the shapes of a tableware collection’s types vary according to a) the parts each type comprises, and b) the proportions of those parts. This had been the rationale behind the implementation of the shallow and deep meta-types into corresponding parametric models, in the Grasshopper prototype. In the final prototype, this approach was taken further by adopting an Archetype, a meta-type that features all possible parts, and from which it is possible to derive all types by removing (or hiding) parts and changing the proportions of the parts that are kept. The first Archetype was the teapot, because it is composed of many parts, such as a body, a lid, a handle or a spout. However, as new types were added to the design system, it became obvious that the Archetype should feature additional parts, such as a foot or a border.
A collection can thus be considered a set of instances of the Archetype. Each instance corresponds to a type, in which only the parts corresponding to that type are present. Also, the proportions of such parts are tweaked according to the represented type. For example, a soup plate would correspond to a “squashed” version of the Archetype, in which only the base, body and border parts are visible (see Figure 103).

In terms of its implementation, visibility of a part is determined by the Boolean value (true or false) of the visibility attribute in the part class definition. As for the different proportions corresponding to the different types, they have been inferred from observation and analysis of the case-study collections. As the collection is created, the dimensions of each composing part are calculated from the inferred proportions, although they can be later altered by the designer. Information about part visibility and dimensional proportions is stored on a table within the programming code of the TDS implementation, named Type Table (see Figure 104). Currently, such table is inaccessible to the system’s user, although it is possible to make it available to the designer in future developments thus providing more control over the collection.

The collection-wise paradigm simplified serialization, “the automatic process of transforming data structures or object states into a format that Unity can store and reconstruct later” (Unity3D, 2017). We use serialization for storing and retrieving information about collections created by or manipulated in the TDS, namely the parametric configuration of the archetype. The archetype’s shape depends on two factors: its contour shape, which is determined by the associated superellipse, and its profile shape, which is determined by the position of the guides of all its
composing parts. Since both of these factors depend on given parameters, namely the supershape parameters, and the guides’ coordinates, a collection can be reproduced if such parametric configuration is known.

Figure 104 – Type Table containing information about a collection’s types: name, proportions, and visible parts

```
// creates all types of PCU
public List<PCUtype> types = new List<PCUtype> {
    new PCUtype("ChargerPlate", new Vector(1.10F, 0.20F, 1.10F), new List<Vector> {"Base", "Body", "Border"}),
    new PCUtype("DinnerPlate", new Vector(1.10F, 0.20F, 1.10F), new List<Vector> {"Base", "Body", "Border"}),
    new PCUtype("TeaPlate", new Vector(1.10F, 0.20F, 1.10F), new List<Vector> {"Base", "Body", "Border"}),
    new PCUtype("TeaCup", new Vector(1.10F, 0.20F, 1.10F), new List<Vector> {"Base", "Body", "Handle"}),
    new PCUtype("CoffeeCup", new Vector(1.10F, 0.20F, 1.10F), new List<Vector> {"Base", "Body", "Handle"}),
    new PCUtype("TakePlate", new Vector(1.10F, 0.20F, 1.10F), new List<Vector> {"Base", "Body", "Handle"})
};
```

Figure 105 – XML file corresponding to a collection

The implemented solution consisted in developing a set of functions that translated a collection’s parametric configuration into XML format (see Figure 105). XML format was selected because of its hierarchical nature, which renders it efficient in representing the tableware hierarchy. Besides enabling saving and loading collection designs, the XML translation played an important
role in the implementation of alternative user interfaces, which will be covered in the next chapter.

4.4 Modeler Usability Testing

As designers were the target users for the Modeler tool, its development took into account the input of the design team of the Manufacturer. Their input was considered in two moments. First at the beginning with an exercise of Contextual Inquiry, in which the designers’ way of working was observed and analyzed, as described in the previous chapter. Then, as the TDS prototype’s development reached a later stage, designers were asked to test the prototype, to get feedback and finalize the prototype’s development.

This second experiment consisted in a preliminary usability test, in which the Manufacturer’s designers tried the application and provided feedback. Each participant started the experiment by responding to a form, containing an informed agreement and a small questionnaire required to identify the user profile. Of the four subjects, one was male and three were female. Three of the subjects had ages between 20 and 34 years old, while the other had between 35 and 49 years of age. All of them hold a Bachelor’s in Design. Regarding their exposure to digital technology, all of them use 3D modeling CAD tools at least once a week, and played videogames at least once a month. All four designers stated they use the mouse with their right hand.

After a brief introduction by the interviewer, each designer experimented with the Modeler application. The designers’ feedback was recorded in a number of different media as they were using the application: video capturing of the screen, video recording of the designers, audio recording of their comments, as well as note taking by the interviewer. Each experiment took 25 minutes on average.

At the end of each experiment, the designer answered to a questionnaire about the experience of using the application. It consisted of several questions to assess how easy it was to perform tasks, using a 6-point Likert scale. It also featured open questions about more general aspects of the application, such as navigation or editing, although they would also be asked to express their opinions orally.
From the forms we could obtain quantifiable information regarding how the designers felt about the Unity prototype (see Table 10). Regarding visual navigation, rotating around a tableware element (TWE) was considered the easiest action to perform, followed by zooming in and out. Changing the focus to the next tableware element was considered the least successful navigational feature by the designers. Collection-wise editing was a praised feature, though they suggested viewing TWEs individually as a desired feature.

Regarding editing the tableware collection, opinions diverged regarding how easy it was to manipulate the contour and profile shapes. Three designers did not find it easy to change the contour parameters, which determine the shape of the superellipse, and two of them considered it compromised their ability to design the contour to their liking. About editing the profile, results also diverged: two designers found it easy to move the guide-controlling handles, while the other two did not; one designer ended up not adding or deleting guides, and of the remaining three, two did it with ease. Such ambivalence is reflected on their opinion regarding how well this feature could help determining the TWE profile. A particular comment suggested the need for higher precision, as well as the implementation of production-related constraints, while questioning whether the tool limited creativity.

Overall, the application performed well when evaluated on general qualities. All designers responded positively when asked about the learning curve of tool use, both in terms of visual navigation and shape editing. When comparing it to the CAD tool that they most commonly used, all the designers considered the Modeler application to be quicker in producing results, and easier in visualizing those results. In terms of formal freedom, only one designer considered that Modeler was no better than his usual CAD tool, and one was dissatisfied with the diversity of the designs. According to the responses, all designers used Rhino, two also use Solidworks, and one of these also utilized AutoCAD.
Table 10 – Summary of the designers’ responses to the forms

<table>
<thead>
<tr>
<th>I consider it was easy to use the following features:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[zooming in and out]</td>
<td></td>
</tr>
<tr>
<td>[rotate camera around elements]</td>
<td></td>
</tr>
<tr>
<td>[switch camera across elements]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I consider it was easy to perform the following actions:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[visualize a specific element]</td>
<td></td>
</tr>
<tr>
<td>[visualize the collection as a whole]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I consider it was easy to use the following features:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[changing contour’s parameters]</td>
<td></td>
</tr>
<tr>
<td>[moving the profile’s control points]</td>
<td></td>
</tr>
<tr>
<td>[adding or deleting control points]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I was able to generate a solution according to my design intentions:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[in terms of contour]</td>
<td></td>
</tr>
<tr>
<td>[through its profile]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I consider it is easy to learn how to work with the system namely:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[in terms of navigation]</td>
<td></td>
</tr>
<tr>
<td>[editing a collection]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I consider this application to be better than the CAD tool typically used:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[in the time spent in that task]</td>
<td></td>
</tr>
<tr>
<td>[in the allowed formal freedom]</td>
<td></td>
</tr>
<tr>
<td>[in visualizing results]</td>
<td></td>
</tr>
<tr>
<td>[in the diversity of generated results]</td>
<td></td>
</tr>
</tbody>
</table>

1 - Strongly disagree  2  3  4  5  6 - Strongly agree

Reviewing the videos of the designers as they experimented with the application and the written notes helped to further understand the designers’ evaluation of Modeler, namely some of their answers obtained from the questionnaire. Most of their complaints, requests, suggestions and comments pertained to features of the application, namely existing ones that might be improved, or potential ones that could be implemented. All the designers’ suggestions were compiled and divided into the following five lists of requirements, or wish lists, according to implementation priority:

- **must have**: features that should be implemented for the application to function properly;
- **added value**: features that add real value to the application, either in terms of perceived innovation, or interest demonstrated by the designers;
- **visualization**: similar to the previous list, contains features deemed important for visualizing the collection being modeled;
• **nice to have**: features that add to the tool’s comfort, typically because users are accustomed to them in the CAD applications they use frequently;

• **CAD**: features that are useful in CAD applications, but that go against the objectives set for the TDS.

It should be noted that many of the suggested features point towards a CAD tool that could be used to design in general, including tableware collections. Although the Modeler’s purpose is to design tableware collections, it was never intended or designed to directly compete with or replace existing CAD tools in the task of designing a tableware collection, but rather to enable the personalization of such collections. Nevertheless, it is understandable that designers benchmark the application against their everyday CAD tool. Features identified in such case fall under the “nice to have” or “CAD” categories. Although features in the CAD category counter the objectives set for this research, and hence were not considered for implementation, listing them is potentially useful should the philosophy of the Modeler tool shift towards an actual CAD tool. Also, they might help explain some negative responses to the questionnaire, namely those concerning design freedom.

Table 11 – Must-have wishlist

<table>
<thead>
<tr>
<th>Must have</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Control dimensions by setting limits</td>
</tr>
<tr>
<td>• Highlight elements which are “out of bounds” in red</td>
</tr>
<tr>
<td>• ensure positional continuity</td>
</tr>
<tr>
<td>• ensure tangential continuity</td>
</tr>
</tbody>
</table>

Of all the wish lists, Must-have is arguably the most important, since its features are considered by the research team important for the success of the implemented design system. The first two items indicate that the system should limit the dimensions of the modelled elements, taking into account production specifications, such as general dimensions or surface curvature. If such values fall out of the recommended interval, then the user should be warned either by a written message or by the change of color to red, for instance, of the TWE in question. Although such a feature is categorized as Must-have, it does not have to be implemented, as it does not concern the concept itself. Since designers are the target users of the application, they can rely on their expertise to constrain the shape of a collection to something that can be produced. Eventually, this approach
can be followed to guarantee compliance with other design limitations or to control the impact of shape on the price of the dinner set, for instance.

The latter two items in the Must-have list relate to surface continuity. It is necessary to ensure that the different parts of a TWE can be perceived as a single object, by enforcing positional continuity between adjacent parts. This can be accomplished by linking overlapping guides together, by ensuring that if the position of one guide changes, the position of the other guide changes with it. Also, it is desirable that tangential continuity is maintained. One possible solution is to implement functions that merge two plaxels into a larger one, since plaxels feature a tangentially continuous surface by definition. Naturally, an opposite function for splitting plaxels should be implemented. Although the need for these features has not been addressed directly, it has emerged from discontinuities that occurred during the usability tests with the designers.

Table 12 – Added-value wish list

<table>
<thead>
<tr>
<th>Added-value</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>New customizable types from hiding parts and/or scaling</td>
</tr>
<tr>
<td></td>
<td>Hideable types</td>
</tr>
<tr>
<td></td>
<td>Different contours per individual guide</td>
</tr>
<tr>
<td></td>
<td>Different materials for production (earthenware, stoneware, porcelain)</td>
</tr>
<tr>
<td></td>
<td>Undo – action history</td>
</tr>
</tbody>
</table>

The Added-value list contains features mainly related to the modeling of the tableware collection itself. The first two entries in the list pertain to collection-wise editing. It has been suggested that it should be possible to create a new type or to apply changes to an existing type within the collection. A given example was the American mug, which is larger than its European counterpart, thus it should be possible to change the scale of the mug type according to its target audience. Moreover, a designer should also be able to change which parts are visible in a particular type, as well as which types exist in a collection. Currently, every type is visible, but it might be useful to exclude certain types from a collection. All these features might be implemented by granting the Modeler’s users access to the collection’s Type Table, so they could change a type’s scale parameter and the Boolean parameters that control the visibility of its parts. Also, a new Boolean parameter could be added to the types themselves to control their visibility within the collection.
Other suggestions included allowing different contour shapes within the same type, as well as different kinds of ceramic material, such as earthenware, stoneware or porcelain. The latter suggestion implies a different set of global shape parameters such as minimum thickness or maximum curvature. Such parameter values would however need to be properly assessed. Last but not least, the undo function is always mentioned as a necessary feature. In our case, it could be implemented as a list of parametric configurations corresponding to the successive states of the parametric model.

Table 13 – Visualization wish list

<table>
<thead>
<tr>
<th>Visualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Navigation: Pan</td>
</tr>
<tr>
<td>• Layout modes: inline vs carrousel vs stacked (same type)</td>
</tr>
<tr>
<td>• Visible supershape parameter values</td>
</tr>
<tr>
<td>• Bézier polygon in handles</td>
</tr>
<tr>
<td>• Highlighted focused type, highlighted handle</td>
</tr>
<tr>
<td>• Different materials for visualization</td>
</tr>
</tbody>
</table>

Many of the designers’ requests focused on visualization features that could enhance Modeler. Panning was one of them. Panning, rotating and zooming exist as visual navigation functions in virtually every CAD tool. However, panning had been intentionally left out on the assumption that users with less experience in exploring 3D space might get lost in such space, for example panning further away from the collection and not being able to go back. However, since Modeler is targeted at designers, a panning function is indeed necessary. Moreover, it would make sense to implement a way to reset the view, bring the user’s view back to the collection’s center.

Another interesting remark concerned the collection’s layout. Although agreeing that the circular layout was a good solution, designers suggested toggling among additional layouts, such as a straight line, or stacking two or more elements of the same type, therefore simulating how the actual objects could be stored in a cabinet.

Other suggested visual aids include: displaying the current values of the supershape parameter sliders; displaying straight lines connecting the guides’ handles, simulating the corresponding Bézier polygon; highlighting the type that is focused on at a given time, as well as the guide
handle as it is being used to manipulate a profile; and simulating different finishing materials, rendering the collection’s elements with other colors or textures, like *craquelure* for example.

Table 14 – Nice-to-have wish list

<table>
<thead>
<tr>
<th>Nice-to-have</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Viewing modes: Isometric vs perspective vs profile vs plan</td>
</tr>
<tr>
<td>• Rendering modes: shaded vs semitransparent vs wireframe</td>
</tr>
<tr>
<td>• Viewing modes in palette</td>
</tr>
<tr>
<td>• 4 viewports</td>
</tr>
<tr>
<td>• Grid for “ground”</td>
</tr>
<tr>
<td>• Show dimensions</td>
</tr>
<tr>
<td>• Gizmo for Move and Rotate</td>
</tr>
</tbody>
</table>

Most of the items in the Nice-to-have list actually concern visualization aspects of the application, and might fall under the previous category. If implemented in Modeler, they would provide increased comfort for the user, but they do not fundamentally render the application innovative, nor does their absence compromise the ability to use Modeler. Nice-to-have features include different viewing modes, such as perspective or parallel projection, isometric, plan, side or section view, or different rendering modes, such as shaded, semitransparent or wireframe, as well as a toolbar that lets the user switch between these modes. There were also suggestions for implementing multiple viewports, thereby providing the user with plan and profile views simultaneously, as well as displaying a horizontal grid marking the plane where the collection stands, displaying dimensional values for its many elements and parts, and displaying a gizmo as a visual aid for eventual moving and rotating operations.

Table 15 – CAD wish list

<table>
<thead>
<tr>
<th>CAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Editing individual points in guide, instead of general guide value</td>
</tr>
<tr>
<td>• Isolate specified type from collection-wise editing</td>
</tr>
<tr>
<td>• Export to IGES for further editing in a CAD tool</td>
</tr>
<tr>
<td>• Shape recognition of imported 3D model</td>
</tr>
<tr>
<td>• Geometric primitives for differentiation</td>
</tr>
<tr>
<td>• Parts library: extra handles</td>
</tr>
</tbody>
</table>
In the CAD wish list we find features that fall out of the scope of this research. For example, it was suggested that it should be possible to edit individual points along a particular guide, rather than defining the whole shape through parameters, as well as to change a particular type individually, without reflecting such changes in all other types.

It was also suggested to enable exporting the resulting model to IGES, or any other format, so it could be further developed in a CAD application. Currently, it is possible to export the resulting collection into the OBJ mesh format through a third party plug-in ExportOBJ (DaveA et al., 2014), so it can be produced via digital fabrication. It is possible to implement conversion algorithms into other formats.

An interesting comment suggested implementing an algorithm to take a 3D model of a TWE, for example, and match the implemented parametric model to the input model’s shape, thereby automatically generating the whole corresponding collection. Such an algorithm would likely imply shape recognition, which is definitely outside the scope of this research, but it is nevertheless an interesting possibility for future work.

The CAD wish list also includes suggestions about adding geometric primitives such as cubes or spheres as differentiating elements, or adding existing objects such as handles. Although the latter feature seems closer to the implemented paradigm, it would still require considerable further development.

4.5 Chapter conclusion

In this chapter we have described the implementation of the Tableware Design System, whose objective is to enable designers to create customizable tableware collections. According to the Design Participation Model presented at the beginning of the previous Chapter (see Figure 106), a customizable collection corresponds to a parametric model that is handed to end-user as part of the mass customization experience. This premise has been achieved as Modeler is capable of generating models of tableware collections represented by changeable parameters.
According to the Design Participation Model above, Modeler receives a rule-based model as input, which can be manipulated by the designer into a parametric model. As originally intended, this rule-based model corresponded to a shape grammar and Modeler would enable the designer to manipulate the design rules that composed the Tableware Shape Grammar.

In the Grasshopper prototype, this intention was implemented. The designer is handed a set of components that correspond literally to TSG rules, each performing different design operations. By combining these operations, the designer creates parametric models that represent families of tableware collections. In the Unity prototype, however, correspondence to the TSG rules is less obvious, and therefore, it was deemed useful to make such a correspondence explicit.

In the Unity prototype, some of these rules are embedded into the system as automatic features. The initialization rules, inferred for the first shape grammar, created a TWE and divided it into functional parts according to its type (see Figure 107). Such rules are implicit in the automatic generation of collection types when the Unity prototype is run. Parameters for the application of these rules to each different type are contained in the Type Table.
Another example is the replacing rule (see Rule 6a in Figure 110) in the base shape phase of the initial shape grammar. The replacing rule makes sense while using the shape grammar, where the envelopes representing plaxels are actually drawn, so that other rules can be applied. The replacing rule is also explicit in the Grasshopper implementation, being encoded in the renderPlaxel method towards a more agile computation process. On the other hand, in the Unity prototype, plaxels are abstract entities, whose manifestation in their corresponding surfaces is permanent. Therefore, the replacing rule is considered embedded as an automatic feature.
Others rules are embedded in the system in functionalities that can be directly manipulated by the user. A good example is the distortion rule (see Rule 4 in Figure 110), whose effects can be replicated by manipulating the guide-controlling handles, and subsequently altering the shape of the corresponding plaxel. Another example lies in the superellipse parameters, whose manipulation applies decoration design rules related to contours (see Figure 112). Rule GR11 subdivides a plaxel along the $u$ direction according to the number of user-specified $u$ values, whereas the $m$ parameter of the superellipse determines the $m$-fold symmetry, and at the same time determines a secondary symmetry in alternating surfaces, mimicking Rule 14a (see Figure 113). Rule GR11 can also be used for dividing a plaxel in two different parts, which can be accomplished in the superellipse by setting different values to the $a$ and $b$ parameters. Finally, the result of applying GR13 can be reproduced by manipulating parameters $n2$ and $n3$, respectively dubbed “sinusoidal 1 and 2”.

Figure 110 – Tableware Shape Grammar: base shape rules

Figure 111 – Guide manipulation

Figure 112 – Tableware Shape Grammar: decoration rules related to contour
Figure 113 – Superellipse parameters

Finally, some rules were implemented as design operations that can be invoked by the designer, similarly to their use in a shape grammar derivation, or in the Grasshopper prototype. The rules that replaced the original plaxel subdivision operation (see Figure 110, see Rule 3a) are included in this category, namely adding or subtracting guides, and merging or splitting plaxels (see Figure 114).

In summary, we have implemented the design rules inferred in the earlier development of the design system both in the Grasshopper and the Unity prototypes, but while the former corresponds to a more literal implementation of those rules, the latter embeds most of those rules in semi- or fully-automated features. In the next chapter, we focus on how the design system can be used by its ultimate users.

Figure 114 – New operations in the multi-guide approach: (a) add guide, (b) delete guide, (c) split plaxel, and (d) merge two plaxels.
Chapter 5
Enabling parametric design space exploration for non-expert users

Recall that according to the Design Participation Model shown in Figure 106, the Tableware Design System (TDS) is divided in two components, Modeler and Navigator. Modeler enables designers to create customizable tableware collections in the form of parametric models, while Navigator enables end-users to customize those tableware collections by exploring their corresponding parametric models. Each component was implemented in a corresponding computer application, also named Modeler and Navigator, and each application corresponds to a distinct user interface, adapted to the different cognitive requirements of their target users.

The reason for separating the two components is related to the cognitive difference between designers and end-users. The Modeler application allows changing a considerable number of aspects of a parametric model, namely in terms of its topology, as well as to directly change the values of its many parameters. Each of these potential changes corresponds to a design option. However, to cope with this many design options towards a satisfying solution can be a difficult task for potential customers of customized tableware, who are not expected to have any design experience. Nevertheless, an engaging interface is needed for these end-users to comfortably manipulate the shape of their future collections. The Navigator addresses this need.

The Modeler component, its corresponding application and user interface have been addressed in the previous chapter. In this chapter we will describe the design and implementation of the Navigator application, focusing on its user interface.

One of the main challenges of designing the Navigator’s user interface was to avoid mass confusion, a phenomenon typically associated with mass customization strategies that results from an excessive burden of choice (Zipkin, 2001, p. 13). The difficulty in dealing with a wide range of possibilities might actually deter the user of engaging in the mass customization experience (Piller et al., 2005). In order to prevent the effect of mass confusion, the design
strategy for the Navigator adopted the design space exploration paradigm (Berger & Piller, 2003; Woodbury & Burrow, 2006; Strobbe et al., 2015). Rather than letting users to directly manipulate parametric models, the Navigator enables end-users to explore, or navigate through, design spaces that correspond to customizable tableware collections. These design spaces are previously defined in the Modeler application by the designer, when defining a parametric model, and who then sets specific variations as the limits of its design space. Navigator generates new collection designs within the restricted design space in real-time according to the end-user’s input.

For obtaining this input, we explored two different approaches. In the first approach, a new design is defined by interpolating among previously designed solutions, resulting in an alternative hybrid solution that shares design qualities with the interpolated collections. In this case, the end-user manipulates the interpolation weights of the existing solutions. An alternative approach is to have the end-user manipulating qualitative descriptions in order to iteratively apply changes to one previously designed solution. This quality-based approach is an attempt to get closer to the users’ natural language. The interpolation approach was implemented into a computer application named Interpolation Navigator (or Interpolator, for a shorter term) (see Figure 115), whereas the quality-based approach was implemented into a computer application named Quality Navigator (or Qualifier, for a shorter term) (see Figure 116). Both options are considered prototypes of the Navigator application, and their implementations are described in this chapter.

![Image of the Interpolator interface](image-url)
Presently, one can find a large number of examples of websites and mobile applications that enable users to manipulate shapes. Contrary to CAD suites, these mobile and web-based apps have the advantage of specifically targeting non-expert users. A subset of these applications is developed specifically for mass customization purposes, and capable of generating design solutions within more or less constrained design spaces. In the Mass Customization literature, these applications are often called toolkits or configurators (Franke & Piller, 2002; Hermans, 2012), or configurators (Heiskala & Tiihonen, 2007). A particular subset of configurators is supported by parametric models, which enable continuous shape manipulation. Let us call these parametric configurators, as opposed to combinatorial configurators, in which products are customized through combining existing design elements. miAdidas is a paradigmatic example of combinatorial customization, in which users assign selected colors to different elements of a sports shoe (Berger & Piller, 2003).

Navigator, the interface developed for the design system, falls into the category of parametric configurator because it allows users to manipulate parametric models, generating solutions in real-time within a continuous design space. In order to contextualize the Navigator application as a parametric configurator, let us look at some of the existing parametric configurators and

**5.1 Existing parametric configurators**

Figure 116 – The Qualifier interface
understand their differences. Some of the applications are not aimed at mass customization, but they constitute examples useful for our analysis.

We begin by distinguishing between specific and generic configurators, invoking the use of these terms relative to shape grammars. Specific configurators are applications that were developed for a specific product, such as the sport shoes example. Among specific parametric configurators, we find Cell Cycle (Nervous Systems, inc., 2012), which enables users to define the shape of a piece of jewelry through a web-based interface34 (see Figure 4, page 31). This application has been developed for the purpose of designing a particular kind of jewel, defined by a specific parametric model. Its designers later deployed similar applications for different products, but each of them only generates one type of product. Specific parametric configurators are supported by a single parametric model, which is specifically implemented into an application. An example closer to our research is the Sake Set Creator (Huang & Hudson, 2013), which enables users to customize the shape of a small set of ceramic vessels35 (see Figure 5, page 32). Sake Set Creator is hosted on the website of online digital fabrication service Shapeways, the manufacturers of the sake set. In fact, the Sake Set Creator is one of various applications called Creator Apps, which let users customize products manufactured by Shapeways. Each of these apps can generate one specific type of object, and therefore they are considered specific configurators.

Both the CellCycle and the SakeSet configurators feature a similar set of control elements for manipulating the shape of the objects they customize. These control elements can be divided into direct and indirect controls. Indirect controls, such as sliders, enable manipulating the object’s shape by changing values in the parametric model. On the other hand, direct controls let the user manipulate shape directly, such as exposed control points that let the user change the object’s defining curves.

Comparing both configurators, we realize that the number of controls is proportional to the potential complexity of the controlled shape. Besides the handles for controlling the vessel’s profile, Sake Set Creator features a very simple interface with only two sliders, which limits the design space of generated solutions. An additional indirect control element consists of a set of

34 https://n-e-r-v-o-u-s.com/cellCycle/

35 https://www.shapeways.com/creator/sake-set
predefined shapes that can be used as a starting point, which correspond to a set of predefined parameter values. In Cell Cycle, a similar function is performed by a drop-down menu in which the user chooses the overall dimensions of the product, between the size of a ring or a bracelet. Additionally, the Cell Cycle interface features a larger number of indirect controls, which include not only sliders but also material swatches among others, allowing for a more diverse design space.

![ShapeDiver interface](image)

**Figure 117 – ShapeDiver interface**

As opposed to specific configurators, generic configurators are applications that can be used to manipulate multiple parametric models, typically corresponding to different products. One example is ShapeDiver\(^{36}\) (see Figure 117), a Web service for publishing parametric 3D data, although not necessarily for mass customization purposes. In ShapeDiver, designers can upload parametric models developed in Grasshopper, which are exposed to end-users as a set of sliders, which control the model’s parameters and a visualization window for presenting the results of changing these parameters.

Similarly, Thingiverse’s app Customizer\(^{37}\) (see Figure 118) lets designers offer customizable designs by publishing parametric models on Thingiverse. Designers must build their parametric

\(^{36}\) [https://www.shapediver.com/](https://www.shapediver.com/)

\(^{37}\) [https://www.thingiverse.com/apps/customizer/](https://www.thingiverse.com/apps/customizer/)
models in OpenSCAD, an open-source solid-modeling CAD application. Contrary to Grasshopper, OpenSCAD’s parametric models are written in textual programming language. Special comments in the scripts enable Customizer to expose the user to the model’s variable parameters through sliders but also other HTML form elements such as text fields and drop down menus.

![Thingverse Customizer interface](image)

Figure 118 – Thingiverse Customizer interface

A common aspect is that both configurators receive parametric models created in third party modeling and scripting tools, namely Grasshopper and OpenSCAD, which imply designers must have some programming proficiency in these applications in order to create parametric models. Alternatively, MatterMachine\(^{38}\) offers an online implementation of a Visual Programming Interface, similar to Grasshopper. This tool allows the user to build a configurator that is able to control everything from the geometry of the customizable object to the information that is displayed to the final client – such as the cost of the customized object. Still, similarly to ShapeDiver and Thingiverse’s Customizer, it requires the user to be able to build a parametric model through programming.

In the generic configurators mentioned above, manipulation of the object’s shape is limited to indirect control elements, such as sliders or other HTML form elements, lacking direct shape

\(^{38}\)http://mattermachine.com/
manipulation capabilities found in the specific configurators. Moreover, both in specific and generic configurators, the number of variable parameters and corresponding sliders is proportional to the complexity of the underlying parametric model. According to Miller (1956), the amount of information units that can be handled simultaneously by the human brain is limited to the order of magnitude around the number seven. Therefore, we frequently witness a trade-off in mass customization configurators between shape complexity and interface simplicity, being difficult to find a satisfying compromise between the two.

In the scope of this research, we challenge this trade-off by proposing a simple interface that lets end users to manipulate the relative complexity of the shapes comprised in a tableware collection, by drastically minimizing the number of parameters manipulated by the user, while avoiding to sacrifice shape complexity. Moreover, our proposed design system is in-between generic and specific configurators: despite being constrained to the customization of tableware collections, it enables the manipulation of multiple parametric models.

5.2 Vector design space exploration

Navigator adopts the design space exploration paradigm, and so its implementation requires a formalization of design spaces. We have previously defined design space as the set of all possible design solutions that can be generated by a design system. The term solution space is often found in Mass Customization literature (Berger & Piller, 2003; Hermans, 2012), having a similar meaning:

“A solution space encompasses all the possible designs a toolkit [...] can produce.” (Hermans, 2012, p. 205)

However, we will use the term design space, which is borrowed from Design literature (Mitchell, 1990; Woodbury & Burrow, 2006; Saxena & Karsai, 2010; Strobbe et al., 2015).

The implementation of the Navigator as a toolkit or configurator is supported by parametric models, which determine the geometrical shape of design solutions for tableware collections. Therefore, we are particularly concerned about parametric design spaces, defined as the set of all possible design solutions generated by the parametric model, and whose dimensionality is defined by the number of parameters of the corresponding parametric model (Talton et al., 2008, 2009).
Each specific design solution is generated by feeding the parametric model with a corresponding ordered set of parameter values, which is the solution’s parametric configuration. Therefore, there is a biunivocal correspondence between generated design solutions and parametric configurations. If we consider a parametric design space as a vector space, we can represent each of its design solutions as a solution vector. Each component of a solution vector corresponds to a value of the design solution’s parametric configuration.

Like the parametric design space it belongs to, solution vectors are \( n \)-dimensional, in which \( n \) corresponds to the number of parameters necessary to generate (and to describe) solutions of the parametric model. Recall from the previous chapter that a specific design solution corresponds to a tableware collection. Moreover, one customizable collection corresponds to a defined parametric model, which in turn corresponds to a parametric design space with a defined dimensionality. Nevertheless, the number of parameters can vary from collection to collection, since a collection’s parametric configuration depends on the number of guides of its Archetype. For illustration purposes, solution vectors are represented by points in a two-dimensional referential (see Figure 119), although these points exist within an \( n \)-dimensional space.

![Figure 119 – Vectorial representation of design solutions as points](image)

Exploring a design space implies the possibility of changing from one design solution to another. Therefore it is important to represent this variation, which corresponds to the variation between their corresponding parametric configurations. Like design solutions, variation between solutions can be also represented by a vector, which we denominate as a variation vector, which can be calculated as the vector difference between two solution vectors. Like solution vectors, variation vectors can be represented graphically, either as an oriented arrow connecting the two points corresponding to the solution vectors, or as an equivalent point (see Figure 120). Since these vectors represent variation departing from a particular solution, the arrow representation seems
more suitable. Either way, please keep in mind that, in our tableware multi-dimensional parametric models, these are vectors in $n$-dimensional space.

![Figure 120 – Representation of a variation vector as arrow and point](image)

Solution vectors and variation vectors belong to the same vector space, and so vectorial operations can be performed between them, with some operations being more relevant for generating variations in the Navigator implementation. We have seen that subtracting two solution vectors results in a variation vector. Conversely, adding a variation vector to a solution vector results in a new solution vector. It also makes sense to multiply solution and variation vectors by a scalar value, as it will be necessary for interpolation operations.

**Design subspace**

Finally, we introduce the concept of design subspace as a subset of a design space. According to the previous chapter, Modeler enables the designer to create a customizable tableware collection, which is represented by a parametric model. For defining a customizable collection, the designer modifies the parametric model by applying design operations such as adding or deleting guides, thus modifying the model's topology. It is also possible to edit the collection's profile and contour shape by specifying parameter values, namely the guides’ positions or values in the superellipse’s parameters, in which case the designer is not actually modifying the parametric model, but only changing values in the model’s parameters, and therefore specifying a design solution.

When modifying the parametric model’s topology using the Modeler, the designer inherently defines its corresponding design space. However, the designer might consider this design space to be too broad for the end-user, and that only a subset of the solutions in that design space is suitable for mass customization. For example, some proportions among guides might lead to
unwanted designs, or the number of sides in the superellipse should be set between two values. Therefore, only a subspace of the model’s design space should be made available to the end-user. That subset can therefore be specified as a design subspace by imposing restrictions on the values of the variables defining the model.

Instead of making these constraints explicit, the designer can define a design subspace in the Modeler by specifying a set of design solutions, which we refer to as limit solutions (see Figure 121). Defining the design sub-space should be done after the designer defines the design topology. While specifying the limit solutions, the designer cannot change the topology of the design, for example by adding or deleting guides. That would result in design solutions whose corresponding parametric configurations would have different structures, and therefore those solutions could not be considered variations of the same parametric model. We have constrained the subspace definition to three limit solutions due to the interface paradigm selected for Interpolator, explained in the next section. In Modeler, the three limit solutions are stored together, so they can be loaded together in Interpolator. The constraints are therefore implicit in the set, which facilitates the act of defining the sub-space of the customizable collection.

![Figure 121 – Definition of a design subspace through specification of limit solutions](image)

Figure 121 – Definition of a design subspace through specification of limit solutions
Please note that the term subspace is used in a broad sense, and as an extension of the term design space as used in Design and Mass Customization literature, rather than related to vector subspaces found in Algebra. In Algebra, for a subset of vectors to be considered a subspace of a vector space, it needs to verify all three conditions: non-emptiness, closure to addition, and closure to scalar multiplication (Blyth & Robertson, 2013). We can easily verify that any vector of a constrained parametric design space falls out of that space when multiplied by a large enough scalar, violating the condition of closure to scalar multiplication.

5.3 Navigation through interpolation

According to the Merriam-Webster dictionary, interpolation is defined as the act of “estimate[ing] values of (data or a function) between two known values”\(^{39}\). Interpolator is an implementation of the Navigator component that enables the end-user to generate a new collection by interpolating between initial pre-designed collection designs, resulting in a new hybrid solution that shares design qualities with the initial ones. The only variable the user needs to control is the interpolation values, reflecting whether the new solution is more similar to one or the other of the initial designs.

For determining a new design, consider two different pre-designed tableware collections, corresponding to design solutions \(S_1\) and \(S_2\). Each solution corresponds to a parametric configuration \(C\), containing the same number \(i\) of parameters \([P_1, P_2, \ldots, P_i]\). In order to interpolate solution \(S'\) from \(S_1\) and \(S_2\), we obtain the corresponding parametric configuration \(C'\), by calculating a weighted average for each and every parameter in \(C_1\) and \(C_2\), such as:

\[
P_i' = P_{i1} \cdot W_1 + P_{i2} \cdot W_2 \quad (3)
\]

in which \(W_a\) and \(W_b\) are weights that reflect the aforementioned interpolation values. Considering that parametric configurations can be represented by solution vectors, we can generalize the previous expression into

\[
S' = S_1 \cdot W_1 + S_2 \cdot W_2 \quad (4)
\]

\(^{39}\)https://www.merriam-webster.com/dictionary/interpolate
The method for calculating the weights can be related to sliders, graphical user interface (GUI) elements that are typically used to elicit values from the user. Imagine users are asked to place a slider’s handle between two solutions $S_a$ or $S_b$, in order to determine which solution the hybrid solution $S'$ is more similar to. Typically, users would place the slider’s handle closer to their favorite solution. Therefore, we can use the distances between the slider’s handle and each solution to determine the weight to use in the aforementioned weighted average. As such, the closer the handle is to a particular solution, the higher the corresponding weight (see Figure 122).

![Figure 122 – Sliders for eliciting values interpolated between two solutions](image)

Considering $W_1$ as the weight corresponding to $S_1$, whose value equals the distance from the handle to $S_2$, and $W_2$ as the weight corresponding to $S_2$, whose value equals the distance from the handle to $S_1$, we can interpolate the parametric configuration of the new solution using the following weighted average:

$$S' = S_1 \ast \left( \frac{W_1}{W_1 + W_2} \right) + S_2 \ast \left( \frac{W_2}{W_1 + W_2} \right) \quad (5)$$

This approach to interpolation uses barycentric coordinates of the slider handle’s position relative to two solutions (Coxeter, 1989). Let us extend it to a more complex example by considering three tableware solutions. In this case, weights for each solution cannot be calculated from the distance to the other solution, since there is more than one other solution, causing ambiguity. Instead, these weights $(W_i)$ are proportional to the area of a triangle whose vertices encompass the handle’s position, and the positions of the remaining two solutions (see Figure 123). Since the areas of the three triangles are proportional to the handle’s barycentric coordinates, we can extend the previous expression to three solutions like this:

$$S' = S_1 \ast \left( \frac{W_1}{W_1 + W_2 + W_3} \right) + S_2 \ast \left( \frac{W_2}{W_1 + W_2 + W_3} \right) + S_3 \ast \left( \frac{W_3}{W_1 + W_2 + W_3} \right) \quad (6)$$

The barycentric interpolation was implemented in the Interpolator prototype (see Figure 123), in which three design solutions specified by the user are positioned on the vertices of an equilateral triangle.
triangle. By moving a draggable black sphere as the interpolation controller, the end-user can determine the interpolation weights of the new design. The resulting new tableware collection is displayed in a circle around the three pre-existing solutions.

Barycentric coordinates have been used in interfaces for shape interpolation, namely applied to mesh deformation (Von-Tycowicz et al., 2015; Wang et al., 2015), whereas in our prototype it is applied to a parametric model.

**n-solution interpolation**

The weighted average approach can be generalized to a larger number $n$ of solutions.

$$S' = S_1 \ast \left( \frac{W_1}{W_1 + W_2 + W_3} \right) + S_2 \ast \left( \frac{W_2}{W_1 + W_2 + W_3} \right) + S_3 \ast \left( \frac{W_3}{W_1 + W_2 + W_3} \right) \iff S' = \frac{S_1 \ast W_1 + S_2 \ast W_2 + S_3 \ast W_3}{W_1 + W_2 + W_3} \iff \sum_{i=1}^{n} S_i \ast \frac{W_i}{\sum_{i=1}^{n} W_i}$$

However, the visual representation of extended number of solutions presents some problems, namely when determining the interpolation weights using the barycentric coordinates. In the previous example, three solutions were placed in the vertices of an equilateral triangle. Analogously, if we were to interpolate among four solutions, then we could lay them out as vertices of a square. However, in this case, the order in which the solutions are placed on the vertices counts. For example, if the handle is placed closer to a particular solution, the solution
placed on the opposite vertex will have less influence than the solutions placed in adjacent vertices, not because of the user’s preference, but because of the layout order. This problem does not exist when using the equilateral triangle, in which each vertex is an equal distance from all other vertices. It is however aggravated as the number of solutions and, therefore, the sides of the interpolation polygon increase beyond three.

The order problem is easily solved by using additional spatial dimensions. For example, instead of laying out four solutions on a two-dimensional square, we can place them on the vertices of a three-dimensional solid, such as a regular tetrahedron. This platonic solid obeys the same premise as the equilateral triangle, by having each vertex at an equal distance from all other vertices. In fact, barycentric coordinates are originally meant to be used in simplexes (triangles, tetrahedrons, and so on), although there have been efforts to generalize them to polygons and polyhedral (Floater, 2015). In that case, interpolation weights are proportional to smaller tetrahedrons defined by a vertex corresponding to the handle position and the vertices of one of the tetrahedron’s faces. However, having users place a handle in a specific and meaningful position within a three-dimensional tetrahedron using two-dimensional displays like a PC monitor or a mobile device is not an easy task (Geng, 2013). An alternative would be to use technologically advanced solutions, such as an immersive virtual reality environment, or a stereoscopic display, which provide the user with the perception of 3D space. However, that would be too much of an effort to interpolate among four instead of just three, and it would be insufficient for interpolating among five solutions, as they would need to be laid out on the vertices of a pentatope, a simplex in four spatial dimensions. Since the Interpolator is meant to be used primarily in a two-dimensional display, the number of solutions to interpolate was set to three.

5.4 Navigation through qualities

An alternative interface was considered for exploring the design space of ceramic tableware collection designs, in which the end-user is able to transform an initial solution through the manipulation of adjectives that qualify tableware designs, which we call qualities. This interface was, thus, dubbed Qualifier and it is an attempt to bring the design system closer to the target users’ natural language.
The implementation of the Qualifier interface in Unity derives from the Interpolator interface based on interpolation. Although both applications enable users to apply design changes to tableware collections, there are some noticeable differences between them. When Qualifier is run, the user is asked to select one starting collection, while Interpolator starts with three collections. The Qualifier’s Graphical User Interface (see Figure 124) features two control elements: a dropdown menu, from which the user can select a quality to apply to the predefined collection, and a slider, with which the user can control “how much” of that quality is applied.

When the end-user applies a specified quality, by selecting it from the dropdown menu, the system generates a new collection by adding the corresponding variation vector, which was previously surveyed and calculated, to the solution vector of the specified collection. By moving the slider, the end-user controls the magnitude of the quality vector that is being added, thus controlling the “amount” of quality that is being applied.

![Figure 124 – The Qualifier application](image)

**Related work on linguistic descriptions**

A number of experiments focusing the approximation to users’ natural language can be found in Human Computer Interaction literature, two of which served as an inspiration for conceiving the Qualifier interface: AttribIt and Body Talk.

AttribIt is described as “an approach for people to create visual content using relative semantic attributes expressed in linguistic terms” (Chaudhuri et al., 2013, p. 193). The AttribIt interface
enables non-expert users to modify the design of objects in particular design domains, namely animals, airplanes and ships, by selecting from a set of semantic attributes, or qualities, applied to different parts of those objects. One of the steps in obtaining such an interface was to obtain relevant qualities from users, making use of crowdsourcing. Later, also through crowdsourcing, partial designs were ranked according to the obtained qualities. This information was processed and used in the implementation of a graphical user interface.

In BodyTalk (Streuber et al., 2016), a similar approach is used to produce human avatars. BodyTalk is supported by a parametric 3D model of the human body shape, which can be manipulated through pre-determined linguistic descriptions of shape attributes. Again, these linguistic descriptions were obtained through crowdsourcing, by asking human participants to tag photographs of female bodies using adjectives. Likewise, the mapping between linguistic body space and a geometric body space was learnt by having human users rating images of the synthetic human bodies using those adjectives.

Inspiration for the Qualifier came from these experiments in terms of their objective, namely to facilitate the manipulation of parametric models to non-expert users, and some of their methods, such as crowdsourcing for semantic attributes.

**Methodology**

Developing the Qualifier interface encompassed three consecutive tasks: (1) data acquisition, (2) data post-processing and (3) implementation.

In order to implement the intended interaction paradigm based on qualities, it is important to obtain a significant amount of such qualities. For that purpose, a survey was conducted in which non-expert users were asked to qualify design variations, which worked as follows. Survey-takers were shown a randomized set of pairs of tableware elements of the same type, corresponding to design solutions generated in Modeler. For each pair, survey-takers were asked to write one adjective that best distinguishes one solution from the other, and to indicate which solution is least and most of that adjective. For each pair of design solutions, a variation vector were calculated by subtracting the “least” solution vector from the “most” solution vector. The resulting variation vector was associated to the quality suggested by the user, in turn resulting in a quality vector. Finally, each specified design solution was transformed according to the
corresponding quality by adding the corresponding quality vector to the corresponding solution vector (see Figure 125).

![Figure 125 – Quality vector obtained from subtracting A from B and associated to a quality; the same quality vector is applied to solution C to generate a new solution](image)

**Data acquisition**

For acquiring a substantial number of qualities, online surveying tools were used, since they facilitate contact with a large number of potential users, and therefore have the potential to provide a large quantity of data. A number of services allow conducting online surveys, most of them for free. However, these services typically run pre-defined surveys consisting of a single set of questions to be asked to multiple users. In our survey, the same question is asked about multiple entries of a data set, namely randomly pairs of tableware collections. Therefore, standard online surveying tools are unfit for our purpose.

An alternative option would be to turn to citizen science, defined as “a type of crowd-sourcing, where volunteers collaborate with professional scientists to conduct scientific research” (Jennett et al., 2013, p. 1). Online citizen science services are more suitable than surveys for gathering information about large data sets, by asking volunteers to perform a particular task onto a number of items, which is typically large. Examples of these tasks include identifying features in many different photographs of galaxies (Raddick et al., 2010), or reading VAT numbers from scanned shopping receipts. Services like Zooniverse and Sensr abide to the open science movement, by asking volunteers to perform the tasks needed to a certain research project, whereas other services, such as Amazon Mechanical Turk, pay its workers for performing those tasks. However, in the citizen science tools considered for running the survey, questions about items in the data set need to be closed questions, with predetermined answers. Therefore, survey-takers would not be able to write spontaneous distinguishing qualities, but rather be forced to select them from a set of predefined adjectives. Zooniverse allows for open questions in custom-built projects, but they
need to be developed in Zooniverse’s programming environment, called Panoptes (Bowyer et al., 2015).

Another limitation in both surveying tools and citizen science services is that data sets correspond to predetermined items, such as photos or sound clips, whereas the intended survey should display randomly coupled pairs of design solutions. Additionally, in order to better evaluate morphological differences between collections, survey-takers should be able to explore the 3D models of presented design solutions, for example by rotating them in 3D space and cycling through types. Since 3D exploration is already possible in Modeler, it was considered to implement a simplified version of Modeler, which would serve as an online individual visualizer for pre-generated collections. However, none of the considered surveying tools or citizen science services allows embedding third party applications in their surveys.

**The Survey application**

The adopted solution was to develop a custom application for conducting the intended online survey. This application, called Survey, materialized the idea of the individual visualizers as a simplification of the Modeler application (see Figure 126). In this application, two tableware collections are displayed side by side. Like in Modeler, it is possible to rotate the models in 3D space, and to view different collection elements by pressing the arrow keys, but unlike in Modeler, in each collection only one type is displayed.

![Image of the Survey application](image-url)

*Figure 126 – The Survey application*
For implementing the Survey, the research team used the Modeler application to previously generate fifteen design solutions (see Figure 127), which can be combined into 105 different pairs.

\[
C(15,2) = \frac{15!}{(2!(15-2)!)} = 105
\]  

(6)

Within the application, each solution corresponds to a) a solution vector encoded in XML format, and b) an identifying number for tracking the solution during the survey. When the application is run, all the 105 pairs are generated, and presented in random order to the survey-taker, who is asked to write a quality that differentiates the two displayed collections in each pair. The Survey application’s graphic user interface (GUI) includes web form items for gathering such qualities, namely a text field, and a dropdown menu with the words ‘more’ and ‘less’. Therefore, each response is composed of:

- a value corresponding to the language in which the survey was taken, either Portuguese or English;
- two numerical identifiers, corresponding to the left and the right collections;
- a Boolean value, obtained from the more or less dropdown, more being the true value;
- a string, corresponding to the linguistic description, or quality, suggested for that pair.

The response submitted by the survey-taker about a pair is encoded into an XML file, and uploaded into a specific folder in the server where the application is hosted, to be later converted into a quality vector.

Figure 127 – Design solutions used in the Survey application, represented by their corresponding soup plate
The application was hosted in a server of the School of Architecture at the University of Lisbon, and the corresponding link was disseminated through e-mail and social media, namely Facebook. The results were satisfactory, totaling 980 responses in 23 days.

Before actually beginning the qualifying tasks, the survey-takers were presented with a webpage informing them about the project, namely about the purpose of the survey. They were informed about the total number of presentable pairs (105), and that not every pair needs to be qualified. In fact, survey-takers are asked to answer a minimum of 10 pairs each, although nothing prevents them of quitting the application before answering the minimum amount required. Survey-takers were also asked to answer a few personal questions for demographic characterization purposes. Also, a unique user identifier is generated in order to associate survey-takers with their responses.

**Data post-processing**

The information gathered through the Survey application needed to be processed before being used in the implementation of the Qualifier.

The first step is to convert the survey-takers’ responses into quality vectors, by subtracting the solution vectors corresponding to the two solutions in each pair presented in the Survey application. If the dropdown value is ‘less’, the left collection is subtracted from the right collection. If the dropdown value is ‘more’, the subtraction is reversed. Recall that the structure of a quality vector is exactly the same as of a solution vector. The resulting quality vector is associated with the quality suggested in that response.

After converting responses into qualities, it is necessary to “clean” the gathered data, namely by correcting any occurring misspellings, but also by identifying responses containing qualities considered too vague, such as such as “ugly”, “beautiful”, “simple”, or “complex”.

After “cleaning” the data, two situations should be taken in consideration. The first is that very likely the same variation vector is associated with different qualities. Consider the case that two survey-takers qualify all of the 105 pairs, and that they associate at least one of those pairs with different qualities. However, this eventuality does not compromise the concept of the Qualifier interface. Consider the pair depicted in Figure 126. It is actually interesting that we can obtain a
transformation from the left collection to the one on the right by invoking different qualities, such as “less twisted”, or “less spiky”, rather than to limit one particular transformation to one quality.

However, the opposite situation is problematic. It is also likely that different variation vectors are associated with the same quality, which in this case we call an ambiguous quality. Since the Qualifier is given qualities to return variation vectors, if given an ambiguous quality, the Qualifier cannot determine which variation vector should be used for transforming a design. Therefore, it is important to search for and deal with ambiguous qualities among the obtained quality vectors. The simplest solution is to discard ambiguous qualities from the dictionary. A more elaborate alternative would be to keep the association with the most frequent vector. Imagine that a particular quality has been associated with one variation vector by two survey-takers (having a frequency of 2) and with a different vector by another survey-taker (having a frequency of 1). Therefore, that particular quality could be associated the first vector, with the highest frequency. Such an approach would actually reinforce the association validity for qualities associated to vectors with large frequency disparities. On the other hand, qualities associated with the same frequency to both vectors should be discarded.

In the end of post-processing, we have a dictionary of qualities that correspond to potential transformations to design solutions, which can be used in the implementation of the Qualifier as an alternative Navigator.

**Current state of the Qualifier interface**

Most of the tasks documented for implementing the Qualifier interface have been executed, namely its implementation in Unity. The interface is working with a basic quality dictionary, created by the research team as a placeholder before a more elaborate dictionary is generated. The generation of this dictionary depends on carrying out the survey, and the subsequent data processing. The Survey application needed for the survey is also implemented and a preliminary test was carried out, which permitted to validate the process. The next step is to undertake more extensive testing, with a larger pool of users, which will be covered in future work.
5.5 Chapter conclusion

We have covered the last component of the Tableware Design System. While Modeler enables designers to create customizable tableware collections in the form of parametric models with a defined design subspace, Navigator enables end-users to customize those tableware collections by exploring that subspace. We have studied two possible implementations of the Navigator component: Interpolator, which enables interpolating a hybrid design solution among existing tableware collections, and Qualifier, which invites users to use their natural language in order to apply design changes to an existing collection.

The implementation of the Navigator prototypes underlined a number of issues surrounding user interface. A well-known obstacle to mass customization is mass confusion, which emerges when the user is imposed with the burden of choosing among too many options. In designing a configurator, this phenomenon relates to the apparently direct proportion between the complexity of configurator interfaces and the complexity of the shapes they generate, which requires the designer to choose between engaging users either with a simple interface and therefore fewer customization options, or more customization options and, therefore, a more complex interface, risking mass confusion. Navigator challenges that tradeoff, resulting in a very simple interface for controlling a relatively complex set of shapes, in which the user manipulates the complex double-curved shapes of all the elements of a ceramic tableware collection by moving a single handle.

More than developing a standalone application, the purpose of implementing the Navigator prototype was to demonstrate that the Tableware Design System and its implementation into the Modeler application could be extended to target non-expert end-users. This extension is made possible by the strategy adopted in the development of the Tableware Design System of translating shape grammars into parametric models for representing tableware collections. Contrary to shape grammars, designs generated by a parametric model feature a fixed topology, which allows representing generated solutions by a defined data structure, which in turn can be translated into an XML file, or an $n$-dimensional vector. Defining this data structure positively affected a number of aspects of the TDS, such as:

- facilitating the creation of a parametric model for the specific purpose of customized tableware;
• enabling the designer to set the limits to the parametric model’s design space simply by selecting limit solutions;
• enabling the communication between Modeler and Navigator; and
• expediting the calculation of interpolated solutions within Navigator.

Being a prototype, Navigator presents a number of limitations, some of which should be mentioned. For example, in order to use Navigator in a real mass customization scenario, it is important to show that all interpolated designs are acceptable design solutions. Alternatively, means must be devised to identify and communicate unacceptable designs. The second alternative might suggest the implementation of an evaluation module, which has not been considered, since this research is focused on shape generation rather than evaluation.

Also, the concept of design subspace can be further refined, namely in the Interpolator. As mentioned previously, the term is used broadly, since the barycentric interpolation generates solutions limited to the interior of a triangle defined within the design space. However, we have not explored the consequences of extrapolating solutions beyond the triangular boundary. Moreover, due to the formalization of the vector design space, the three solutions defined by the designer can be used to define a geometric plane in the design space containing the triangular boundary, which would be considered a proper subspace according to its algebraic definition. Determining this plane might help explain why a designer selects a particular set of limit solutions. Finally, the triangular boundary can be used to geometrically determine whether a design solution is inside or outside the designer’s determined subspace.

In future research, these shortcomings should be addressed. Additionally, we intend to test the usability of the proposed interfaces, in order to validate and quantify the advantages attributed to our approach.

In the next chapter, we focus on the link between the design system as a whole, which ultimately enables end-users to customize the design of a tableware collection, and the production system, which is responsible for materializing customized designs.
Chapter 6
Exploring additive manufacturing for ceramic tableware

In this chapter we focus on the production capabilities necessary for manufacturing customized tableware collections. The work on the production system begins with an assessment of suitable production technologies for manufacturing customized tableware, with a special focus on additive manufacturing, from which material extrusion was selected as the best candidate for the implementation of a production system. A number of experiments were performed using the selected technique in order to explore its potential and limitations. Finally, a technical improvement is suggested for some of the identified limitations.

Techniques used in mass production of tableware are unfit for mass customization, as their profitability depends on producing large quantities of copies of the same product, in the order of hundreds. Since our aim is to provide a mass customization experience to individual customers, who probably only need to acquire a single tableware collection, the production system must be designed to produce a smaller number of units of the same type, namely in the order of tens. Therefore, we must resort to digital fabrication, which offers individuality for free (Dillenburger & Hansmeyer, 2013), meaning that manufacturing a batch of units that are different from each other has a similar cost as if they were identical, provided that they use similar material quantities.

Digital fabrication techniques can be divided into additive, subtractive and formative, according to the manufacturing process used (Kolarevic, 2001; Pupo et al., 2009). Regarding their suitability for manufacturing ceramic tableware elements, subtractive technologies, which include CNC milling and lathing, limit the geometric complexity of the produced objects (Newman et al., 2015), especially in areas in which the machining head would have difficult access like, for example, the interior of a teapot. Compared to subtractive manufacturing technology, additive manufacturing allows for greater geometric complexity. On the other hand, the use of formative digital fabrication techniques, which include the use of molds, is constrained by cost. Mold-based techniques are traditionally used in ceramic manufacturing, such as slipcasting and pressing. The digital fabrication equivalent to these techniques would imply manufacturing the necessary molds
using additive or subtractive digital fabrication. However, the manufacturing process itself would be subject to the same limitations identified in mass production, rendering them unprofitable for batches smaller than a few hundred units.

Consequently, additive manufacturing (AM) was selected as the most suitable technology for mass customization of ceramic tableware. We began by conducting a survey on currently available AM techniques applied to ceramics and corresponding equipment, with special focus on information about equipment and operation costs and maximum dimensions of the built parts. By comparing the surveyed techniques in terms of cost and build dimensions we concluded that material extrusion technology is a suitable solution for implementing the production system for the developed mass customization system, because of its low cost and large build envelope when compared to the remaining technologies. Nevertheless, two main challenges have been identified in ceramic extrusion: (1) the need for supporting structures and (2) layer expression.

Experiments with the production of mass customized tableware were performed by using the ceramic extrusion equipment developed by Tom Lauerman in the School of Visual Arts at the Pennsylvania State University. The existing technology was used for producing design solutions generated by the design system. The manufactured prototypes helped confirming the technology’s suitability for mass customized tableware, as it has been suggested by the survey analysis. Also, the equipment itself could be tested for the specificities of tableware, such as double curved surfaces or the existence of handles.

Finally, we propose two alterations to the ceramic extrusion equipment used in the experiments, towards resolving some of the issues related with the need for supporting structures. The first alteration is implementing dual extrusion, which allows using two different materials, one for modeling and one for support. The second alteration is implementing a rotating build plate, in order to maximize the verticality of the object’s wall as it is being built. We hypothesize that the need for supporting structures can be overcome by rotating the part in space, in order to keeping the extrusion head vertically aligned with a vector that is tangent to the part’s surface. The use of NURBS surfaces to describe ceramic tableware, proposed in the Tableware Design System, facilitates calculating the tangent vector.
6.1 Survey on additive manufacturing applied to ceramics

A survey was conducted on the state-of-the-art concerning AM technology, specifically technology that has been used for the production of ceramic objects. The equipment identified in the survey has been analyzed according to criteria stemming from the requirements of the proposed customization system to determine which equipment would be more suitable for our purpose, with an emphasis on the maximum build size. Moreover, we sorted the surveyed equipment according to the ASTM F42 Committee’s seven categories: 1) Material extrusion, 2) Material jetting, 3) Binder jetting, 4) Sheet lamination, 5) Vat photopolymerization, 6) Powder bed fusion, and 7) Directed energy deposition. Except for directed energy deposition, every listed AM process has been applied to the manufacturing of ceramic objects, and most of them are commercially available. Although most of the AM technologies are applied to ceramics for the production of precision components, such as electronics, two processes emerged as the most adequate for producing ceramic tableware: material extrusion and binder jetting.

Kinematic structure

Besides sorting in terms of AM process, we also categorized the surveyed equipment in terms of its kinematic structure, a term that can be borrowed from robotics literature (Craig, 1989; Tsai, 1999) if we consider a piece of AM equipment to be a robot. The Oxford English Dictionary defines robot as “[a] machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer” (Oxford Dictionaries, 2017b). A robot’s kinematic configuration determines in what way its end-effector is positioned by the mechanical manipulator. In AM equipment, we consider the end-effector to be the printing head. According to its kinematic structure, a robot can be either serial or parallel, whether it is served by an open- or closed-loop mechanical manipulator. A robot can also be served by a hybrid manipulator, consists of both open- and closed-loop chain (Tsai, 1999). In the surveyed ceramic AM equipment we find examples of serial and parallel kinematic configurations (see Figure 128).
The Cartesian robot is an example of a serial robot, whose manipulator features the simplest serial kinematic structure, made up of three mutually perpendicular prismatic joints corresponding to the X, Y, and Z Cartesian directions, and for which the inverse kinematic solution is trivial. In fact, the Cartesian manipulator is the most common solution found in commercial solutions for AM. Moreover, this configuration lends itself to stiffer structures, in turn allowing for the production of larger robots (Craig, 1989). The Articulated robot is another example of a serial robot, with an open-loop manipulator. This configuration is often found in robotic arms typically used in the automotive industry. Articulated robots feature revolute joints, which renders calculation of inverse kinematic solutions more complex than in Cartesian robots. However, its construction requires less structure, rendering articulated robots less expensive for applications needing smaller workspaces (Craig, 1989). As an example of a parallel manipulator we find the Linear Delta robot, also known as Linapod. In this kinematic configuration, the end-effector is positioned by three vertical linear actuators (Merlet, 2006). Since the actuators are identical, such a machine is actually easier to implement (Wurst, 1999), although calculation of inverse kinematics is more complicated when compared with the Cartesian robot. Recently, the Linear Delta configuration has been introduced in the market of AM equipment, some of the resulting equipment being referred to as ‘Delta printers’. However, this denomination can be misleading, since the terms ‘Delta’ and ‘Linear Delta’ correspond to two different kinematic configurations (see Figure 128, right).

Most of the surveyed equipment relies on a Cartesian manipulator for positioning its printing head. The only exceptions are found in equipment of the Material Extrusion category. Besides Cartesian manipulators, we also find examples of Articulated Robots and Linapod Robots.
Material extrusion

Material extrusion is probably the best known AM technology. The expiration of patents on material extrusion, namely of Fused Deposition Modeling, is behind the current dissemination of low-cost 3D printers (Wohlers & Caffrey, 2014). Extrusion of ceramic material has been explored both in academia and in design practices. Different projects have explored the constraints and the potential of controlled extrusion of ceramic paste through a nozzle to build objects. In a way, ceramic extrusion invokes the traditional techniques of coiling and throwing.

Belgium-based design practice Unfold (2009) developed a system from a RepRap 3D printer, which features a Cartesian kinematic structure. Originally, the RepRap machine was designed to extrude plastic materials, like many of the 3D printers that have emerged in recent years. By modifying the extrusion head in order to extrude white clay, they have managed to produce ceramic objects from digital models through additive manufacturing (see Figure 129, left). Ceramics artist and educator Tom Lauerman also made use of a RepRap printer, the Prusa i3, for producing ceramic objects (Lauerman, 2014).

A similar approach was tested at IAAC by a team of students in their project Fabclay (Jokić et al., 2012). In this project, an articulated robot was used to position the extrusion head, depositing several layers of clay, thus producing the intended ceramic object (see Figure 129, right). Despite the many degrees of freedom provided by the use of a robotic arm, in this project this equipment was used to replicate the movements of a Cartesian robot, i.e., XYZ translation. In this case, red clay was selected as the material for extrusion. More recently, ceramic designers experimenting with additive manufacturing have tried with linear delta printers. Both US-based Jonathan Keep (2013) and the Netherlands-based Olivier van Herpt (2014) have used printers based on the Linapod design, which enables them to produce large ceramic objects (see Figure 130).

Alongside these projects, equipment suppliers have developed ceramic extrusion machines based on the Linapod robot configuration. Italy-based company WASP produces extrusion machines for a number of materials, including clay. A particular solution is the WASP Clay Kit, which includes the DeltaWASP 20 40 machine. As the name implies, the DeltaWASP features a Linapod configuration, working in a similar way to the machines used by Jonathan Keep and

http://www.wasproject.it/w/en/
Olivier van Herpt. In fact, previous collaborations between WASP and Jonathan Keep (Keep, 2014) suggest that the artist used prototypes of the DeltaWASP. This machine can produce objects as wide as 200 mm, and up to 400 mm tall, by positioning the extrusion head along the successive layer paths that build the object.

![Figure 129 – Extrusion of white clay objects by Unfold (left) and extrusion of red clay object by Fabclay (right)](image1)

![Figure 130 – Extrusion of tall ceramics objects by Jonathan Keep, United States (2013) (left) and by Olivier Van Herpt, The Netherlands (2014) (right)](image2)
US-based DeltaBots\textsuperscript{41} produces the Potterbot. Contrary to the DeltaWASP, the Potterbot’s design features a static extrusion head, as it is the object being printed and the build plate that are positioned according to calculated toolpaths. While in the original Potterbot the build plate was manipulated by a Linapod type manipulator located underneath it, in its latest iteration it has changed into a Cartesian configuration. Potterbot 7 can print objects up to 19 inches (483 mm) tall, within a rectangular envelope of 17 by 14 inch (432×356 mm).

Netherland-based VormVrij\textsuperscript{42} provides ceramic AM equipment featuring a more conventional Cartesian configuration. Their LUTUM clay printer is available in three different sizes, the largest of which LUTUM MXL has a maximum print envelope of 45×44 cm by 75cm tall.

### Material jetting

France-based company CeraDrop\textsuperscript{43} supplies a machine called CeraPrinter, which produces Printed Electronics through material jetting. The system is able to deposit a wide range of inks, although none of them are ceramic-based, despite what the name suggests. It is able to print on ceramic substrates. Nevertheless, the system is unable to produce ceramic objects. Also, the maximum dimensions of the produced parts must fit in a 305 mm wide, 10 mm thick square, dimensions which are adequate for manufacturing electronics, but not for tableware. Nevertheless, CeraPrinter is the only surveyed equipment using material jetting technology and ceramic materials, and therefore it was included in this analysis.

### Binder jetting

In recent years, different research teams have been exploring the possibility of using the binder jetting principle for manufacturing ceramic objects, either on the hardware side (Hoskins, 2012; Universidade de Aveiro, 2013) or on the material side (Figulo, 2013; Tethon 3D, 2014). The common starting point for these projects was the ZPrinter, a machine developed at MIT in the

\textsuperscript{41} http://www.deltabots.com/

\textsuperscript{42} http://lutum.vormvrij.nl/

\textsuperscript{43} http://www.ceradrop.com/en/
1990s that led to the spin-off ZCorporation, and is now part of the 3DSystems\(^{44}\) product family. Originally, the ZPrinter produced objects by jetting a cyanoacrylate-based binder onto layers of a plaster-based powder. By using the same process but replacing the plaster-based material with ceramic powder, it is possible to produce detailed green ceramic objects, which can then follow their typical production process, through firing in the kiln, glazing, and so on (see Figure 13).

Eventually, ceramic binder jetting showed signs of a transition from research to the commercial domain. In the 2014 edition of Consumer Electronics Show (CES), US-based company 3DSystems announced the launch of CeraJet, a “3D printer” that produces ceramic objects (3DSystems, 2014). This equipment was advertised as suitable for producing large ceramic objects, such as tableware, although the dimensions of its build envelope have not been revealed. Shortly before announcing they would commercialize the CeraJet ceramics 3d-printer, 3D Systems acquired Figulo (3DSystems, 2013), a US-based company that produced ceramic parts using binder jetting (Wohlers & Caffrey, 2014). Despite commercialization of CeraJet having been announced for the second half of 2014, 3DSystems appear to have discontinued the CeraJet project, since we could not find additional information since then.

### Lamination

US-based company CAM-LEM\(^{45}\) applied lamination technology to the production of ceramic and metal components for microfluidic applications and technologies, through a machine called CL-100. When the CAM-LEM process is applied to ceramics, each of the object’s layers is laser cut from a sheet stock of “green” ceramic tape. After assembly, the layers are laminated by warm isostatic pressing or other suitable method to achieve intimate interlayer contact, promoting high-integrity bonding in the subsequent sintering operation. The laminated “green” object is then fired into a monolithic structure. The result is a 3D part, which exhibits the correct geometric form as well as functional structural behavior (Chua et al., 2010). However, the maximum build size is a 150 mm cube. As such, it was considered inadequate for producing tableware.

---

\(^{44}\) https://www.3dsystems.com/

\(^{45}\) http://www.camlem.com/
Vat photopolymerization

Several companies are currently supplying equipment for ceramic AM using vat photopolymerization.

CeraFab 7500 is supplied by Austria-based company Lithoz. This printer is advertised as being suitable for producing high-performance ceramic objects, namely cost-effective prototypes, small scale series and complex parts, through its proprietary LCM technology (Lithoz GmbH, 2012). The build envelope is rather small – 76×43×150 mm.

CeraMaker uses laser stereolithography to produce ceramic parts, and it is supplied by France-based company 3DCeram. According to the company, this printer is particularly suitable for manufacturers of ceramic parts, integrators or end-users of ceramic parts, players in the luxury or biomedical industries, but also in the industry in general (3DCeram, 2014). In comparison with the previous example, the build envelope is larger – 300×300×110 mm.

Italy-based company Digital Wax Systems supplies the DWS XFab model, which applies laser stereolithography to several materials, including Therma 289, described as a nano-ceramic composite. Despite its ceramic composition, its application is recommended for thermal resistance tests and high definition models for vulcanized rubber molds, rather than for producing tableware. Furthermore, its work area is limited to a cylinder that is 180 mm wide and tall. This machine is marketed as a consumer-oriented 3d printer, costing around US$ 5,000 (Wohlers & Caffrey, 2014).

ProMaker V6000 is a piece of equipment that applies laser stereolithography to composite materials of ceramics and metal, supplied by France-based company Prodways. The maximum build area is a box of 120×500×150 mm. The focus on composite materials suggests that such equipment might be less suitable for producing mono-material ceramic objects.

46 http://www.lithoz.com/
47 http://3dceram.com/
48 http://www.dwssystems.com/
Finally, FormLabs\textsuperscript{50} recently launched a Ceramic Resin that can be used with their Form 2. This model is commercialized as a desktop 3d printer, namely in terms of equipment cost. However, the machine’s building envelope measures 145×145×175 mm, rendering it too small for manufacturing tableware elements.

**Powder bed fusion**

Phenix\textsuperscript{51}, a France-based company recently acquired by 3D systems, supplies equipment for laser sintering. Though most Phenix machines use metal powders, the ProX DMP 200 is capable to work with ceramic powders. However, its build envelope of 140×140×100 mm is considered too small for tableware.

Chinese company Wuhan Binhu\textsuperscript{52} develops sintering systems for several different materials including three ceramic powders (Wohlers & Caffrey, 2014). The maximum build size for producing ceramic parts is a 500 mm square, 400 mm high. Although this build envelope is large enough for producing most of the elements of a complete tableware set, costs typically associated to sintering technology suggests that this option would not be competitive.

**Directed energy deposition**

In this survey, we have not found any application of directed energy deposition in ceramics.

**Comparative analysis of AM technology**

From the survey presented above and summarized in Table 16, we verify that most AM processes and equipment are focused on the production of high-performance precision parts, limiting their suitability for production of tableware.

\textsuperscript{50} https://formlabs.com/materials/form-x/#ceramic
\textsuperscript{51} http://www.phenix-systems.com/en
\textsuperscript{52} http://www.binhurp.com/en/
The first limitation is set by the dimensions of the machines’ building envelopes. For example, a standard dinner plate, which typically measures 27 cm in diameter and 3 cm in height, would not fit in more than half of the surveyed machines. Those where a plate would fit couldn’t accommodate a standard 35 cm tray, or a teapot, which can be 16 cm tall. The only exception is the Laser Sintering equipment from Wuhan Binhu.

Table 16 – Summary of Additive Manufacturing equipment for ceramics

<table>
<thead>
<tr>
<th>Process</th>
<th>Supplier</th>
<th>Equipment</th>
<th>Dimensions (mm)</th>
<th>Material Cost (€)**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>width</td>
<td>length</td>
</tr>
<tr>
<td>Material extrusion</td>
<td>WASP (Clay Kit)</td>
<td>DeltaWASP 20 40</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>DeltaBots</td>
<td>PotterBot</td>
<td>432</td>
<td>356</td>
</tr>
<tr>
<td></td>
<td>VormVrij</td>
<td>LUTUM MXL</td>
<td>450</td>
<td>440</td>
</tr>
<tr>
<td>Material jetting</td>
<td>CeraDrop</td>
<td>CeraPrinter</td>
<td>305</td>
<td>305</td>
</tr>
<tr>
<td>Binder jetting</td>
<td>3D Systems</td>
<td>CeraJet</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Lamination</td>
<td>CAM-LEM</td>
<td>CL-100</td>
<td>150</td>
<td>150</td>
</tr>
<tr>
<td>Vat photopolymerization</td>
<td>Lithoz</td>
<td>CeraFab</td>
<td>76</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>3DCeram</td>
<td>CeraMaker</td>
<td>300</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>Digital Wax Systems</td>
<td>DWS Xfab</td>
<td>180</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td>Prodways</td>
<td>ProMaker V6000</td>
<td>120</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>FormLabs</td>
<td>Form 2</td>
<td>145</td>
<td>145</td>
</tr>
<tr>
<td>Powder bed fusion</td>
<td>3D Systems (Phenix)</td>
<td>ProX-200</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>Wuhan Binhu</td>
<td>HRPS-IV</td>
<td>500</td>
<td>500</td>
</tr>
</tbody>
</table>

* Reference for standard dimensions (dinner plate diameter) 270 270 160 (teapot height)

** Estimated cost values from quote requests to suppliers or press releases; values were normalized to euro (€)

The second limitation is set by the cost of printing tableware elements, namely of the printing materials. In fact, some of the contacted equipment suppliers actually considered it unsuitable for producing ceramic tableware elements, mainly because the ceramic materials used in the production of high-performance parts are mostly alumina-based. Although they could be used for manufacturing tableware, alumina-based materials are developed to meet demanding technical characteristics, and therefore expensive when compared to the ceramic materials typically used in tableware.
In general, we conclude that most of the commercially available solutions are unsuitable for producing mass customized tableware. Two processes, nevertheless, stand apart from these technologies: binder jetting and material extrusion.

Binder jetting is currently being established as the *de facto* standard for ceramic additive manufacturing. In fact, print-on-demand companies like iMaterialise\(^{53}\) and Figulo, now owned by 3DSystems, outsource this technology to supply ceramic “3D prints” to clients over the internet. This would establish upcoming binder jetting ceramic printers like CeraJet as an obvious choice for the production system in a mass customization initiative; however, the cost of those prints is still high. As with the ZPrinter, also supplied by 3DSystems, the materials to be used with the CeraJet are likely to be proprietary – and therefore relatively expensive, if the price range for material is similar to the original ZPrinter. Printer owners might try their own materials, but at the risk of damaging the printer and, if the materials used were not recommended by the equipment manufacturer, its warranty would become void. Therefore, using binder jetting for the production system represented a disadvantage for a ceramic tableware manufacturer, since they would depend on a single supplier and imply rather high material costs. Moreover, binder-jetting implies a phase of post-production in which the part is removed from the powder vat, and then airbrushed for removing the remaining powder, which corresponds to a significant cost in time.

On the other hand, in material extrusion processes, the extruded material is a ceramic paste mixed by the machine’s user, which lowers its price significantly. Although the material mix must be fine-tuned in terms of its properties, when the right mix is achieved it can be replicated. This “open-source” approach sets material extrusion apart from the other AM processes.

However, ceramic extrusion presents challenges that must be addressed. In all of the documented projects, we could identify two factors that could be improved. The first one is the expression of the layers in the manufactured objects, as shown in Figure 131. Such texture reveals the nature of the layering process used to make the object. Although in some cases this texture can be desired or even taken advantage of by the designer, it would be important to avoid it in other cases. The second factor relates to the shape of the produced object. In fact, ceramic objects manufactured by ceramic extrusion tend to have a vertical geometry. Although not stated by the authors, this

---

\(^{53}\) https://i.materialise.com/
can be identified as the result of avoiding the need for support structures. Consider that we wanted to manufacture an object with a more horizontal geometry, like a dinner plate for example, using the extrusion techniques mentioned above. Without using a supporting structure, the plate would likely collapse during its manufacturing process. However, supporting structures should be avoided, since they imply more material consumption and require time to be removed. Also, removing support structures usually leads to minor imperfections on the model (Smyth, 2014). Therefore, it would be desirable to overcome this limitation. In the 3D-printing community, the so-called 45-degree rule is often used to address this problem. This rule-of-thumb states that “overhangs that are greater than 45 degrees will need support material or you need to use clever modeling tricks to get the model to print.” (France, 2013). Figure 131 on the left illustrates a limit case of the 45-degree rule, which suggests that support structures can be avoided through geometry.

Figure 131 – Left: Limits of the 45 degree rule (Jokić et al., 2012). Right: the jiggering technique (Quinn, 2007)

The scope of the 45-degree rule is not limited to ceramic extrusion but it can be applicable to ceramic production in general. In fact, in both handcrafted (see Figure 131, right) and industrial production of horizontal shapes such as plates, a mold is typically used underneath the plate as it is being shaped, in order to support it. Nevertheless, we intend to look for strategies to minimize this limitation.

6.2 Experiments with ceramic extrusion

As a consequence of the findings from the documented survey, ceramic material extrusion was selected as the most suitable AM technique to implement the production system for mass
customized tableware. In order to illustrate a potential implementation of the production system, ceramic material extrusion was tested for manufacturing digital 3D models that were generated by the design system, using a ceramic material extrusion machine, which we will call ceramic printer for short. A number of customized tableware elements, belonging to a collection that had been generated by the design system (see Figure 132), were produced by the ceramic printer in its biscuit state, which could then follow the typical ceramic process of being fired in a kiln. Results suggest that selected AM technique is appropriate for developing the production system.

![Figure 132 – Elements of a tableware collection generated in Unity, some having been selected for production through ceramic extrusion](image)

The setup used to manufacture ceramic tableware elements is composed of both hardware and software. The hardware corresponds to equipment, namely the ceramic printer, a computer, and additional tools needed for loading the clay and cleaning the printer after its use, while the software corresponds to the applications used to operate the printer. The system receives two inputs: a) the ceramic material to 3D-print the element, which in these experiments is clay, and b) the digital 3D model that encodes information about the shape of the element (see Figure 133).

![Figure 133 – Setup for ceramic extrusion](image)
**Ceramic printer**

For the production of customized tableware elements in these experiments, two ceramic printers were used, both corresponding to prototypes of custom designs devised and assembled by Tom Lauerman (see Figure 134). The second prototype (henceforth referred to as **Printer Two**) is an evolution of the first one (henceforth referred to as **Printer One**), which in turn derives from the Prusa i3⁵⁴ design (Lauerman, 2016). Both printers comprise a similar set of components, which include a build plate, an extrusion head, and a clay cartridge. Besides these components, additional electronic and mechanical components, such as railings, motors and controllers, support the actions needed for producing an object.

![Figure 134 – Components of the two ceramic printer prototypes](image)

Although both printers correspond to Cartesian configurations, the main difference between the two is in the movements of their parts. In the first prototype, the build plate moves along one horizontal direction, namely along the Y axis, and the extrusion head moves horizontally, along

---

the X axis, and vertically, along the Z axis. In the second prototype, the build plate moves horizontally along both X and Y axes, while the movement of the extrusion head is constrained to vertical movements along the Z axis.

One of the advantages of the second prototype over the first one derives from having constrained the extrusion head’s movement, which permitted to attach the clay cartridge to the extrusion head. This solved one of the problems found in the first prototype, in which too much pressure would build up in the hose connecting the clay cartridge to the extrusion head. By having the cartridge moving along with the extrusion head, the clay is passed directly from the cartridge to the extrusion head, waiving the need for the hose.

**Clay**

The ceramic material used throughout these experiments was clay. As mentioned previously, one of the advantages of the ceramic material extrusion approach is the user’s ability to use their own paste, which can be customized to his or her needs. Throughout these experiments, both red and white clay were used, but besides color, no other significant difference between the two clays was detected. The suitable consistency of the clay had been determined empirically in experiments that preceded the experiments documented here. After being prepared, the clay is loaded into the clay cartridge, a cylindrical transparent PVC tube connected to the extrusion head. In our experiments, the clay was loaded manually, although a more efficient method, using a wall mounted extruder, has been tested successfully (Lauerman, 2017).

**Models**

The digital models used for the documented experiments correspond to a number of elements different types of a tableware collection, which were generated by the implemented design system. As identified previously, one of the challenges in ceramic extrusion is the production of overhanging parts such as handles and, therefore, in the first experiments we produced tableware elements of types without a handle, such as a teacup or a creamer.

Recall that the design system implementation in Unity generates a polygonal mesh for each type in a tableware collection. In order to extract these meshes from Unity, a third party plug in was used, named ExportOBJ, which exports Unity meshes to files in the .OBJ format. This format can
be converted into .STL directly using Rhino. Nevertheless, the process between shape generation of the 3D models and their delivery to the production system as .STL files takes time, and could be greatly improved in future improvements of the design system, by providing the Unity implementation with functions that perform these steps automatically.

At the time of these experiments, some additional post-processing was necessary using Rhino, since the models generated by the Unity implementation were not watertight, due to an error in the implementation that has been corrected since then.

**Apps**

The ceramic printer is incapable of producing an object directly from its corresponding .STL model. Instead, it needs to be given specific instructions regarding for example the movement of the extrusion head or the extrusion rate. Like so many other CNC machines used for digital fabrication, the ceramic printer can only understand instructions written in a particular language, G-code. Therefore, the 3D model needs to be translated into G-code prior to its production, in a phase called pre-processing. There are a number of computer applications developed for this purpose that are usually referred to in digital fabrication jargon as slicers. Despite slicing being one of the steps in model pre-processing, these applications use algorithms in every step of a typical digital fabrication workflow, namely:

- decomposing the 3D model into horizontal layers, or slices;
- transforming slices into tool paths, including calculation of infills;
- translating toolpaths into g-code that can be understood by the ceramic printer.

Therefore, the input of a slicer is a polygonal 3D model, and the output is the corresponding g-code, which can be transmitted to and understood by the ceramic printer.

A number of slicers have been considered and tried for the purpose of running the experiments, namely Cura LulzBot 20.13, Cura Ultimaker 2.3.1, Cura Ultimaker 2.5.0, Slic3r MK2, and Pronterface. These applications have been tested empirically in terms of a) usability, b) control of g-code generation, which was evaluated by the number and pertinence of the available options for generating g-code, and c) capacity to control the operation of the ceramic printer.
Pronterface was the only tested slicers capable of controlling the operation of the ceramic printer. However, despite having some processing capabilities, Pronterface was surpassed by other tested slicers in terms of both usability and g-code control, the front-runner being Cura 2.5.0, which was thus selected as the actual slicer for the tests. Unfortunately, Cura 2.5.0 could not be used to control the operation of the custom-made ceramic printer, although it advertises that capability for other printers. Ultimately, because one integrated solution could not be found, two applications were used for the pre-processing phase: Cura 2.5.0 was used to generate the g-code and Pronterface was used to control the ceramic printer.

One of the challenges is using these applications is finding the right settings configurations in order to obtain satisfying results. These results can be evaluated using criteria such as structural integrity, surface texture and fidelity to the digital model. Remember that despite these applications being open-source, we have no access to the algorithms they use for decomposing 3D models into toolpaths, and therefore different configurations need to be experimented with in order to properly produce ceramic tableware elements. Configurations in the selected slicer Cura are fairly complex, comprising a large number of setting parameters. Nevertheless, having a large number of settings is useful for fine tuning the results. However, some of the settings are less self-explanatory, in the sense that the result of changing their value is neither the expected nor evident.

The tea cup

The first experiment aimed at producing a small and simple shape, namely a tea cup without its handle. This shape was expected to be fairly simple to produce using material extrusion, since it is predominantly vertical and has no overhanging parts. Four instances of the tea cup were produced in red clay using Printer One, from g-code generated in Cura version 2.3.1, from a model generated by the design system and adjusted in Rhino. One biscuit sample was then fired, glazed and fired again, so it could withhold liquids (see Figure 135).

The thickness of the cup’s walls is equal to the diameter of the material extruded from the nozzle. Therefore, it was decided that the cup would be printed using a particular option enabled in Cura, called Spiralize Outer Contour, and found in the Special Modes section. According to its tool tip in Cura 2.5.0, this option “[…] smoothes out the Z move of the outer edge. This will create a steady Z increase over the whole print. This feature turns a solid into a single walled print with a
solid bottom”. Enabling the Spiralize option thus results in producing the object’s walls as a spiral, rather than as a set of horizontal layers. In turn, this results in a continuous print, which is smoother than if produced in horizontal layers, since the vertical movement between layers is distributed along each layer.

Figure 135 – Consecutive models of a customized tea cup: 1. geometric model in Unity; 2. geometric model in Rhino; 3. Cura model showing toolpaths; 4. produced physical model, glazed and filled with water

Nevertheless, the use of the Spiralized option depends on specific conditions that limit the shapes that can be produced. One condition is that the wall thickness must be constant throughout the print job, and depends on the diameter of the extrusion nozzle. If the wall thickness is significantly larger than the nozzle diameter, then the wall layer must be decomposed into two parallel curves, for which the Spiralized option cannot be used. One solution is having a set of nozzles with a number of different diameters. Moreover, the thickness of the extruded material can be manipulated to a certain extent by changing the flow of material that is being extruded. However, the constant wall thickness imposed by the Spiralized option rules out producing walls with variable thickness, either along its profile, as well as along its contour.

The second condition for the use of the Spiralized option is that each horizontal layer in which the produced object is decomposed must correspond to a single curve. In more complex objects, horizontal sections might comprise more than one curve. In the example of the same tea cup but with a handle, a horizontal section through the middle of the cup results in two curves, one for the cup’s body and another for its handle. Therefore, it is impossible to use the Spiralized option for handled tableware elements. Given these two conditions, we can conclude that, despite providing a smoother surface finish, the Spiralized option significantly compromises the versatility of the production system.
The creamer

Following the teacup, we aimed at producing a larger tableware type, although with a similar simple shape, and which could be used to pour tea into the tea cups previously produced. Therefore a creamer was selected over a teapot, whose geometry would be too complex for this experiment. Since it also lacks a handle, this creamer can be considered a ceramic carafe. The creamer was produced in white clay using Printer Two, from g-code generated in Cura version 2.3.1, from a model generated by the design system and adjusted in Rhino. The biscuit sample was then fired, glazed and fired again, so it could withhold liquids.

Production of the creamer underwent a number of iterations. In the first iteration, the print was interrupted because there was not enough clay in the cartridge to build a whole creamer (see Figure 136, 1). The cartridge was refilled and the print job resumed from the point where it had previously stopped. However, the first layers from the new print job failed to attach to the previous one, and so the experiment was aborted. In the second iteration, the print job was repeated with a full cartridge at the start. However, the object collapsed a few layers before being completed (see Figure 136, 2). It was hypothesized that the collapse was caused by the creamer’s wall being too thin relative for its size, and compromising the wall’s ability to support its own weight. Also, the wall’s inclination next to the base might have aggravated that ability.

![Figure 136](image)

Figure 136 – Final results of each of the three iterations for producing the creamer

Therefore, a third print was attempted in which the relative thickness was increased. However, increasing the extrusion diameter would imply that more material would be needed for the print job, and the cartridge might not contain enough material for the whole print. Instead, the object
size was reduced to 85% of its original size. Reducing the object’s size while maintaining the extrusion diameter increased the relative thickness of the object’s walls.

Additionally, the scale reduction was performed only horizontally, meaning that the object’s height remained the same, and therefore the maximum inclination of the walls was reduced, particularly next to the base (see Figure 137). The compound effect of reducing the scale resulted in a successful print of the creamer (see Figure 136, 3).

![Figure 137 – Comparison between original (yellow) and reduced (cyan) models of the creamer](image)

This result reinforces the conclusion that the design system must be adapted according to the production system’s limitations, in this case being that the minimum buildable wall thickness depends on the tableware element’ size.

Results of this experiment suggested that the Spiralized option is limited and, while being a possible solution for particular objects, it should not be used in our production system, which must be flexible enough to be capable of producing any type of tableware. Also, results suggested that the printer should be equipped with a larger cartridge.

**The handled cup**

The previous experiments allowed for verifying that the ceramic printers could in fact produce tableware element of simple types. A subsequent set of experiments targeted the production of a more complex shape, namely a tea cup with a handle. The main objective of these experiments was to understand the ceramic extrusion technology’s constraints in producing overhanging parts, which is a challenge for material extrusion in general. In the following experiments, objects were printed without differentiated support material, since the used printers did not support this option
at the time. If any, supports were printed in the same material as the object itself, which is clay. In this case, it is important to minimize the amount of support material so that it can be easily removed in the post-processing phase, after production of the object itself.

Six prototypes of handled tea cups, numbered from 1 to 6 (see Figure 138), were produced in red clay using Printer Two, from g-code generated in Cura version 2.5.0. The prototypes correspond to two different designs, labeled A and B (see Figure 139), which were generated by the design system and adjusted in Rhino. The design of the tableware element was altered after production of the first two prototypes, according to its results. In both designs, the handle’s shape was modeled directly, as opposed to being generated by the design system, because implementation of the design rules for the handles were not developed.

![Six prototypes of the printed handled cup](image)

**Figure 138 – Six prototypes of the printed handled cup**

![Design A and Design B](image)

**Figure 139 – Two different designs for the handled cup**

In Design A (see Figure 139, left), the shape of tea cup’s body is the same as the teacup without handle that was produced previously, and its handle has a curved shape similar to its body. Prototypes 1 and 2 were produced using the ceramic printer according to Design A. Due to the
slenderness of its handle, it was decided that these prototypes should be printed with supports. While printing Prototype 1, an error caused the printer to extrude clay over the top of the cup, capping it (see Figure 140, left). It was hypothesized that the unwanted capping was caused by an option in Cura, namely the Top/Bottom Pattern in the Shell section, which was set to Linear. For a second test, this option was set to Concentric. However, the error persisted, but assumed a concentric configuration in Prototype 2 (see Figure 140, right).

Figure 140 – Failed prints of the handled cups

Despite the capping error, the two tests were useful to evaluate the ceramic printer’s performance in producing the handle. One conclusion was that, when compared to the body, the layers of clay in the handle were considerably less regular, to the point of causing its structural frailty. This irregularity is particularly present in the handle’s upper section, whose shape is more horizontal, thus challenging the 45 degree rule.

These results suggested that a new, more robust handle needed to be designed, which resulted in Design B (see Figure 139, right). The handle in the new design is broader and predominantly vertical, whereas the connections between the handle and the body are horizontal. The cup’s body was also redesigned. Its shape was generated by the design system, in which the value of the \( m \) parameter in the superellipse, related to its \( n \)-fold symmetry, was increased to five. The whole model was also scaled uniformly up to 150% of its original size, in order to produce a better surface finish, but also to test the capacity of the clay cartridge. Consequently, the walls were made thicker. Prototypes 3 through 6 were produced using the ceramic printer according to Design B.
In Prototypes 3 and 4 (see Figure 141), it was attempted to print the handled cup with limited use of supports. In fact, supports were only printed between the handle’s lower connection to the body and the build plate by setting the Cura value of the Support Placement option in the Support section to Touching Buildplate. The objective of using this option was to assess if the upper connection could be built without supports, given that its length is short. Production of Prototype 3 was interrupted due to insufficient clay in the cartridge, which was not completely full at the beginning of the print job. Consequently, production of Prototype 4 started with a full clay cartridge.

Figure 141 – Intermediate prints of the handled cup

The results of printing the handled cup with partial support were not satisfactory. As observed in both Prototypes 3 and 4 (see Figure 141), the handle’s unsupported shaft bent outwards because of its own weight, although it did not break. This bending effect causes the handle’s shaft to be shorter than the corresponding 3D model. Therefore, although the ceramic printer was able to produce the upper connection between the handle and the cup’s body, it was printed incorrectly, since the material for the extrusion head to deposit clay on was lower than expected.

For the subsequent Prototype 5 (see Figure 141, 5), it was decided to experiment with full support material, including between the handle’s lower and upper connections. In Cura, the value of the Support Placement option was updated from Touching Buildplate to Everywhere. As a result, the handle’s shaft in Prototype 5 bent less when compared with the previous prototype, which resulted in the upper connection being printed correctly. However, the bending shaft was not satisfactory.
In a last attempt to resolve the tea cup’s bending shaft, two changes were made to the Cura settings. First, the model was scaled again to be 130% larger than the previous models, since it was hypothesized that the size of the extrusion might compromise print quality in a smaller tableware element. The second change was related to the orientation of the support material, in order to better support the printing of the handle.

As an option in the Support section, Cura provides a number of Support Patterns, which determine and automatically calculate the pattern according to which support material is printed. Of the provided patterns, the Lines pattern produces the least support material by laying it along parallel lines, hence being the most suitable option for our experiments. Cura does not provide a direct way of controlling the line direction of the Linear Support Pattern relative to the printed object. However, since the lines are always parallel to the Y axis of the Cartesian referential, the object to be printed can be rotated in order to control the relative direction. In Prototype 5, supports were oriented transversally to the handle (see Figure 142, left), in an attempt to minimize contact between support material and object material, so that it would be easier to remove later. However, too little contact resulted in too little support for the handle’s shaft, allowing it to bend. Therefore, Prototype 6 was rotated 90 degrees thus orienting the Linear Pattern longitudinally along the handle’s connections (see Figure 142, right).

Figure 142 – Different orientations of linear support patterns according to object's rotation

Changes applied the printing options of Prototype 6 proved successful in producing an unbent handle for the tea cup, as well as the upper and lower connections. After a day of letting the clay dry, the support material was removed from between the handle and the body. The clay was still humid, although much less than when the object was printed, and therefore the support material was adhering to the object. The option of printing the support material longitudinally required the person removing the support to distinguish between support and object. Despite these limitations,
it was possible to remove the support material using a knife with minimum damage to the object (see Figure 143). However, this process can hardly be automatized, which suggests that an alternative is necessary.

Figure 143 – Prototype 6: production with longitudinal support (6a) and detail views of the handle (6b, 6c, 6d)

**The saucer**

A final set of printing experiments focused on the saucer, the smallest type within the plate metatype. The interest in printing a plate derives from its predominantly horizontal shape, which challenges the 45 degree rule. As seen earlier in the survey on AM applied to ceramics, objects produced through material extrusion are typically vertical in shape, in order to avoid the need for support material.

The saucer was produced in red clay using Printer Two, from g-code generated in Cura version 2.5.0 from a model generated by the design system and adjusted in Rhino. The design of the saucer derives from the same collection as the handled teacup, featuring a curved 5-pointed star shape. Due to its predominantly horizontal shape, printing the saucer could not waive the need for supports. Continuing on the strategy of minimizing support material, two different Support Patterns were tested with: Lines and Concentric.

The saucer printed using the Lines pattern was not satisfactory. In order to minimize support material, the spacing between parallel lines was too large, which resulted in insufficient support in-between support lines. This lack of support was particularly evident in artifacts near the endpoints of the star shaped plate (see Figure 144, left). Therefore, Concentric pattern was used in
a second print of the saucer, so that support would be evenly distributed along the plate’s contour. However, this option showed to originate too much support, since almost all of the space below the plate was filled with clay. In fact, it was impossible to distinguish object material from support material, as it was shown by sectioning the resulting print.

![Figure 144 – Printing the saucer with different Support Patterns: Linear (left) and Concentric (Right)](image)

In general, the results of printing the saucer were disappointing. Although additional experiments can be performed with different settings for support material, a potential solution for printing predominantly horizontal shapes in clay can be found by improving the ceramic printer with an additional component, namely a rotating platform.

### 6.3 Suggested improvements to the ceramic printer

From the performed experiments, we reiterate that the main difficulties in using ceramic extrusion for manufacturing ceramic tableware are related to the need for support material. For example, printing the saucer required a large amount of support material due to its predominantly horizontal shape, which would be difficult to remove being the same as the modeling material, clay, with which the tableware element was printed. In the case of the handled cups, despite having been easier when compared to the saucer, removing the support material left some imperfections on the object. In any case, it seems difficult to automatize the removal of clay as a support material. Such limitation would compromise the efficiency of the production system, should it be based on clay extrusion.
We propose two alterations to the ceramic printer, towards resolving these issues. The first alteration is implementing dual extrusion, which allows using two different materials, one for modeling and one for support. The second alteration is implementing a rotating build plate, in order to maximize the verticality of the object’s wall as it is being built.

**Dual extrusion**

Using a different support material implies equipping the ceramic printer with dual extrusion capability. For a material extrusion machine to be able to print objects with two different materials, it needs to be equipped with two parallel circuits, one for each material. Each circuit comprises a material storage cartridge, a pumping mechanism and an extrusion nozzle. In the case of functional graded materials, the two materials converge in only one nozzle, but an additional component is necessary to mix the materials (Craveiro et al., 2016). Therefore, assigning a machine with dual extrusion capability increases its cost, which emphasizes the need for assessing the benefits of such assignment. The benefits of using a different material for printing supporting structures depend on whether or not the material is easily removable from the printed object.

For example, Stratasys\(^{55}\) supplies material extrusion equipment that produces objects through Fused Deposition Modeling (FDM), in which the dual extrusion strategy is used. The material used for building the object itself is Acrylonitrile butadiene styrene (ABS), as well as Polycarbonate and Polyphenylsulfone in higher end machines, whereas the support material is a more brittle polymer, meaning it can be physically removed by the user. Also, the support material that is removed directly by the user can be dissolved by dipping the object into an water solution (Chua et al., 2010).

This approach can be replicated in ceramic extrusion by selecting a support material that can be easily removed in post-processing. Suggested materials include investment casting wax, paper paste or even mashed potato\(^{56}\). These materials have in common the property of being eliminated in the kiln-firing process, which makes each of them a good option for a support material. In fact,

\(^{55}\)http://www.stratasys.com/

\(^{56}\)https://plus.google.com/+TomLauerman/posts/FpV6j8Qt54A
the time and energy needed for removing the support material in a post-processing phase, which is usually mentioned as a disadvantage of using support structures (Strano et al., 2013) are both available during the biscuit firing process, which is a necessary step in the production process in order for ceramic objects to transit from plastic to solid state (see Figure 13). Biscuit firing can thus replace the post-processing phase in which support material is removed, reducing the cost of using a support material.

One future improvement to the ceramic printer could then be the implementation of dual extrusion capability, which would allow performing tests with the different burnable support materials.

**Rotating build plate**

Alternatively, we suggest that the need for supporting structures can be eliminated in some cases by keeping the extrusion head vertically aligned with a vector that is tangent to the part’s surface. In order for the extrusion head to be tangent to the object’s surface, it should be rotated in space. However, in this scenario, the clay could bend downwards while being extruded from the nozzle. Therefore, in order to make use of gravity, we propose to rotate the object itself (see Figure 145), keeping it aligned with a vector that is both tangent to the part’s surface, and perpendicular to the tool path’s curve (see Figure 146, left). The use of NURBS surfaces to describe ceramic tableware, proposed in the Tableware Design System, facilitates calculating that tangent vector.

![Figure 145 – Build sequence of a teapot's body according to the proposed printing process](image)

Because of the 45 degree rule, rotating of the platform can be constrained to a maximum of 45 degrees, since compounding the two effects should enable the ceramic printer to produce surfaces parallel to the build plate (see Figure 146, right), reducing the need of supporting structures for simple types such as plates. Nevertheless, for more complex cases, such as building the handle onto a mug for example, supports would still be needed.
Gravity can be both a benefit and a hazard in the implementation of the rotating build plate. In fact, although the rotation of the build plate compensates the inclination of the wall on which the ceramic material is being extruded, at the same time it aggravates the inclination of the wall in the diametrically opposed part of the printed object, potentially causing it that part to deform. Therefore, implementing the rotating build plate depends on an additional improvement of the ceramic printer, at the material level.

In fact, this concept implies that the extruded clay dries quickly enough so it does not deform when the extrusion head reaches the opposed part. Such implication might defeat the point of rotating the build plate if, by itself, reducing the clay’s drying time compensates for the deformation associated with the wall inclination. In turn, interfering with the drying time might produce unexpected results in terms of shrinking and generate subsequent alignment issues. Nevertheless, it is an option that might be worth exploring in future improvements of the ceramic printer.

For rotating the object in 3D, we suggest implementing a rotating platform on which the object is fixed. For the kinematic configuration of the rotating platform, two solutions are suggested. The first solution is the Gough-Stewart platform, or hexapod. This parallel kinematic configuration corresponds to a set of six linear actuators that enable moving a platform along 6 degrees of freedom (see Figure 147, left). This is a tested configuration, having been implemented in the 1950’s by Gough (Merlet, 2006). More recently, the Gough-Stewart platform was implemented in a material extrusion machine enabling movement in six degrees of motion (Song et al., 2015). An alternative approach can be borrowed from more modern 5-axis metal machining equipment such
as the DMG MORI LASERTEC 45 Shape\textsuperscript{57}, which complements the XYZ motion of a tool head with an additional two degrees of freedom provided by a swivel rotary platform. A circular platform can rotate around a normal axis in its center, but also around its diameter via a swivel mechanism (see Figure 147, right). Given that the Gough-Stewart platform can be built with simple linear actuators, it might be an easier option than building a swivel rotary platform.

![Figure 147 - Left: Gough-Stewart platform (Stewart, 1965). Right: DMG MORI LASERTEC 45 Shape (industryarena.com\textsuperscript{58})](image)

The implementation of the rotating platform implies significant changes to the existing equipment in terms of both software and hardware. On the software side, a custom slicer application should be developed, which should be able to calculate not only toolpath points but also orientation according to the surface’s tangent plane. An option is to develop it in Grasshopper, since it works with Rhino, which is particularly efficient in working with NURBS surfaces, namely in calculating auxiliary shapes such as tangent planes. Also, a plug-in for Grasshopper called Silkworm allows users to generate custom g-code. On the hardware side, the first challenge is building the rotating platform. Also, the printer’s controller should be made capable of reading

\textsuperscript{57}http://us.dmrmori.com/products/lasertec/lasertec-shape/lasertec-45-shape

the upgraded g-code and of transmitting the corresponding commands to the elements that change the platform’s position.

6.4 Chapter conclusion

This chapter illustrates the challenges and possible solutions of using digital fabrication technology to produce customized ceramic tableware. From a survey on additive manufacturing applied to ceramics we have selected material extrusion as the most suitable digital fabrication technique for manufacturing tableware, identifying support structures and layer expression as its main shortcomings. Subsequent experiments realized with ceramic printers shed light on additional issues that challenge the success of using material extrusion for our purposes. These issues include more pragmatic problems such as the capacity required for the clay cartridge, the most effective setting configuration for the slicer, or the potential of equipping a ceramic printer with a rotating platform.

The production system documented in this chapter is a work in progress, and in this section we point out the necessary steps to improve it. It serves the purpose of showing that the design system developed in the scope of this dissertation can output objects than can be produced, yielding real ceramic tableware elements.

Future work should include systematic testing of the prototypes produced through ceramic extrusion, assessing factors such as structural integrity. The integrity of these elements should be tested at the end of the production process, which in the case of tableware necessarily includes glazing. In terms of structural integrity, the prototypes should be tested while holding food or liquids, which add to the overall weight. Another factor worth analyzing is the approximation of the printed prototype to the corresponding 3D model. Fidelity can be partially assessed by comparing the printed object with a laser-cut section of the 3D model. A more sophisticated option would be to laser-scan the resulting object, and compare it with the actual 3D model used for its production.

Another interesting research path would be to reverse calculate deformations inherent to the production process. It is expected that the shape of the produced object changes during and after its production process. For example, a ceramic object shrinks during biscuit-firing. The amount of
shrinkage depends on the ceramic material used. For instance, the red clay used in our experiments is estimated to shrink about 17%. Integrating the effects of these phenomena would be a useful feature in the design system.

Another aspect that should be further researched is the implementation of the suggested improvements into an existing ceramic printer, namely alternative support materials and the rotating platform, in order to verify if indeed it waives the need for supporting structures altogether. Research on alternative support material should be carried out with specialists from Material Science, while the implementation of the rotating platform requires work both on the software side, such as adapting an existing slicer or creating a new one, and on the hardware side, namely to adapt an existing machine to add the platform.
Chapter 7
Conclusion

In this final chapter, we reflect back on the research developed on mass customization of ceramic tableware presented in this dissertation. We summarize the developed work, discuss the obtained results of the performed tasks and of the research as a whole and suggest paths for future work.

7.1 Discussion of results

In our research we have dealt with a number of different issues regarding the design challenges in mass customization, which are reflected on the various tasks that have been performed. Nevertheless, the main focus was to improve the mass customization experience to both designers and end-users by reducing complexity of the design process.

The Design Participation Model as a tool for managing complexity

As mentioned in Chapter 1, in a typical mass customization solution, the end-user interacts with a configurator that is able to generate and present design solutions according to the end-user’s preferences. In this scenario, the role of designers is restricted to the development of a generative design system that enables the configurator to generate valid design solutions. Compared to traditional mass customization systems, the proposed Design Participation Model extends the role of designers beyond the conception and development of the design system, allowing them to create customizable design solutions after the design system is implemented. This novel approach grants more flexibility to the design system, while granting designers a more direct quality control of design solutions.

As explained in the section about the evolution of the Design Participation Model, the initial concept for the design system corresponded to a design machine in which the design logic of the customized product would be encoded. However, such approach was considered unsuitable for
the customization of ceramic tableware, in part due to the complexity of designing tableware to the non-expert user, but also because it was considered that designers should be integrated in the design process. In order to reduce the complexity of the design task that end-users would be exposed to, it was decided that end-users would be limited to manipulating parametric models, rather than shape grammars. Moreover, it was decided that designers would define those parametric models, using shape grammars to some extent. Our approach was therefore to split the design system into two components, one oriented to designers and another to non-expert end-users, called Modeler and Navigator, respectively.

Splitting the design system allowed us to overcome the complex design logic so as to present a simplified interface to the end-user. Without the designer as an intermediary agent between the design system and the end-user, the design system would be either too complex, carrying the risk of overwhelming the end-user, or too simple, limiting the design space. The designer thus acts as a mediator, charged with using a more complex design system via the Modeler component, to create customized collections, then presented in a simpler fashion to the end-user via the Navigator component. This mediation does not place the whole burden on the designer, since the implemented Modeler component facilitates the creation of parametric models without resorting to more complex tools, such as programming.

The split design system was tested with a group of its first target users, designers, for whom the Modeler application was designed, as explained in Section 4.4. Their feedback was positive and complemented with a number of suggestions for improvement, as expected in user-centered design processes. When compared to the CAD tool that they most commonly use, designers considered Modeler to be quicker in producing results, easier in visualizing those results, and more efficient for generating diverse designs. Their main suggestions pointed towards including evaluation tools, such as highlighting actions that compromise design guidelines. Additionally, a number of features usually expected of a finished CAD tool was also suggested, such as an ‘undo’ command or additional flexibility in visualization. All the feedback from the designers is to be taken into account in subsequent development of the design system.

Despite the identified limitations from the usability perspective, the positive feedback on the Modeler application demonstrates its capability of generating parametric models that correspond to customizable collections, or templates, without resorting to programming. This reflects the
premise of the proposed Design Participation Model of reducing complexity for its participants, in this case designers, contributing to its validity.

The Tableware Shape Grammar

Recall that the generative design system is supported by a shape grammar that encodes rules for the design of ceramic tableware. This Tableware Shape Grammar (TGS) was developed as an intermediate tool towards computer applications that implement the proposed mass customization design system. Despite being developed with the intent of encoding a particular design language, namely that of the Manufacturer, the TSG embeds general knowledge about ceramic tableware design that extends beyond that style. For example, the total and partial dimensions for the different tableware types, although having been inferred from examples produced by the Manufacturer, are likely similar to those used by other manufacturers. The types, parts and elements included in the proposed taxonomy of ceramic tableware are also likely to be common to other brands or companies, thus being also part of the embedded general knowledge. This knowledge is common across different styles, rather than pertaining to a particular style. Therefore, the TSG can be used as a generative tool to create new designs of ceramic tableware outside the design language of the Manufacturer, for instance if subjected to transformation towards a new style, following the methodology proposed by Knight (1994).

As explained in Chapter 3, development of the Tableware Shape Grammar (TSG) was based on research about generic shape grammars, which explores the concept of combining rules from different specific grammars into a generic grammar that encodes a broader design domain or style. The first objective of generalizing the rules into a generic grammar was to have a smaller number of rules, so that such rules could be more efficiently implemented, namely in the Grasshopper prototype. Having fewer rules simplifies the task of editing the code used in the implementation of each rule. Particularly in the case of systematic changes to the design system that have to be applied to all rules, the probability of error during the editing process is proportional to the number of rules. The process for inferring the generic grammar included decomposing rules to make them more similar, and then generalize them. We consider that the intermediate step of decomposing the rules into simpler components also contributed to a more efficient implementation. Despite temporarily increasing their number, the simpler rules were common to more specific grammars, minimizing the occurrence of one-off rules that existed in
only one grammar. Although we explain the decomposition and generalization processes using the shape grammar, they were applied during its implementation, which shows how the processes of developing the grammar and implementing the rules were intertwined.

One unavoidable shortcoming of the generic shape grammar is that it conceals the specificity of the original rules, some of which may be particularly distinctive in terms of style, such as Rule 11b in the Romantica Collection (see Figure 148). This way, the generic grammar does not replace the specific grammars, but rather complements them. Even though specific rules can be reproduced through a specific sequence of generic rules and a specific set of parameters, it might be useful to create explicit associations between rules from the specific grammars and their counterpart in the generic grammar.

![Figure 148](image-url)

**Figure 148** – Left: Rule 11b. Right: partial derivation of a soup plate, Romantica collection

There seems to be a point of equilibrium for the generalization process. Rules of the TSG were allowed to be as generic as possible, for the purpose of a more efficient implementation of the rules, and because researchers could easily use these general rules to recreate the specific rules due to their tacit knowledge of their specificities. However, the generic grammar might need to be less generic in order to be used by a third party, either a human, like other researchers or designers, or a machine. For example, a set of similar specific rules can be generalized into more than one generic rule, thus keeping important specific design features apparent.

The development of the TSG offered the chance of refining the methodology for obtaining a generic grammar proposed by Duarte (2011). In fact, by mapping the Design Participation Model (DPM) onto Duarte’s methodology we find correspondences between the two (see Figure 149). In DPM, the role of the Researcher is to develop a generic shape grammar, which corresponds to inferring existing designs into specific shape grammars and subsequently into a generic grammar. The role of the designer is to use the generic grammar to extract a new specific grammar, which
in the DPM is simplified to a parametric design. Finally, the role of the end-user is to generate a new specific design.

![Diagram](image)

Figure 149 – Roles of design participants in the process of developing a generic shape grammar (adapted from Duarte, 2011)

This correspondence was demonstrated in the development of the design system, suggesting that the inference process of a generic shape grammar systematized in this dissertation can be used to support a mass customization design system. Given the use of Duarte’s methodology in other design domains, such as architecture and furniture design, we believe that the presented correspondence contributes to the validation of using the Design Participation Model in design domains other than ceramic tableware. Moreover, development of the TSG illustrated that rules can be imported from external design domains, namely from classical architectural elements to tableware.

**The Modeler component**

As discussed in Chapter 4, in the scope of implementing the proposed design system, a number of computer applications were developed with various degrees of operability. One of the initial tasks was to select an appropriate tool for implementing the design system. As we have demonstrated, there is not one tool that can satisfy all implementation requirements. While Grasshopper provided a flexible CAD environment with a visual interface that could eventually be used by more advanced designers for manipulating shape grammar rules, it lacked methods to articulate the developed models with a web-based interface. On the other hand, although the Unity implementation of the design system can be deployed as a web-based interface with the potential
to be tested and used by a large number of end-users, such implementation entailed a number of additional tasks, such as learning a new programming language and implementing geometric modeling functions, like lofting. Also, it implied a simplification of the TSG into a modifiable parametric model. The difficulties in implementing the whole design system into a single programming platform should not be surprising, given the many different aspects underlying a complex system such as the one sought for the mass customization of tableware.

An alternative approach would be using different platforms to implement the various modules of the design system. For example, the Modeler module could be implemented in Grasshopper, taking advantage of design-oriented features such as CAD functions for enabling designers to create customizable tableware collections in the form of parametric models. Likewise, Navigator could be implemented in Unity, enabling it to be deployed as a web application and accessed by a large number of end-users, who would manipulate the parametric models into customized tableware collections. Although a mixed platform approach might respond better to the needs of each target user group, it entails a new challenge: how to exchange parametric models between applications?

The scale of this challenge increases as we consider that current exchange formats for a number of other domains are static. Think about the STL format used in digital fabrication, which is composed of static vertex coordinates, or the PDF format used in written documents. A suitable exchange format for parametric models would have to encode both lower level information pertaining to geometry, which in our case would necessarily be NURBS geometry, and higher level information pertaining to design intent. Developing such exchange format was beyond the scope of our research, thereby justifying the use of a single platform for the Modeler and Navigator modules.

Despite the obstacles and limitations found during the implementation of the design system, the main purpose of such implementation was to test the design system and consequently to experiment with the strategy of subdividing design spaces in order to reduce complexity in the design process. Such purpose was accomplished and the resulting applications serve as more tangible evidence of the results of this research.
Linking the design system to user interfaces and production system

Even though the main focus of this study was the design system, we sought to illustrate its relevance and capacity to be integrated in a complete mass customization system by exploring user interfaces and the production system.

Exploration of user interfaces occurred in the implementation of the Navigator component as explained in Chapter 5. As mentioned previously, in the proposed mass customization system, designers create customizable collections encoded in parametric models. Despite being simplified design tools when compared to shape grammars, parametric models can still be overly complex for non-expert users, proportionally to the number of parameters. Therefore, two user interfaces were developed that further simplify the effort of manipulating the parametric model, shielding end-users from being overwhelmed by the design process. The two developed interfaces differ from each other significantly in terms of complexity, as suggested by the number of tasks and information needed to implement each one. While Interpolator could be implemented directly based on existing collections, the implementation of Qualifier implied surveying users and analyzing gathered data. Nevertheless, both interfaces aimed at providing a more comfortable customization experience than existing parametric configurators.

The Interpolator interface simplifies the customization process by reducing the number of choices to a single slider, whereas parametric configurators with multiple interface elements offer a number of design choices that might instill mass confusion in a less design-driven end-user. One possible criticism is that such an over-simplification might limit the available design space, compromising the motivation for the customization process. However, since the user can replace one or all of the initial collections that serve as limit solutions, the extent of the design space is proportional to the amount of available collections, not to the number of interface elements or model parameters. Interpolator served as a stepping stone for Qualifier. In fact, the code developed for interpolating collections is used in both interfaces. In Qualifier, new solutions are generated by adding a quality vector to a selected initial collection and determining a magnitude for that vector, which is the same as interpolating between the initial collection and another collection resulting from the addition.

Exploration of the production system is documented in Chapter 6, as prototypes of design solutions generated by the design system were produced using digital fabrication technology. The
selected digital fabrication technique was additive manufacturing through material extrusion, as a consequence of the survey on additive manufacturing applied to ceramics that was conducted. Experiments with the clay printer allowed the production of tableware elements whose shape was generated in the Unity implementation of the design system. In fact, models generated by the design system could be produced almost directly, with just minor adjustments that can be incorporated in the design system. Continued work with the clay printer should elaborate on the use of differentiated support material or, alternatively, on the modification or addition of components with potential to minimize the need for supports, such as the proposed rotating platform.

Although implementing a production system was outside of the scope of our research, the work developed about that topic generated information that allowed identifying problems in the design system, as well as define suitable configuration settings in the used equipment. Such information might be useful for an actual implementation in a mass customization context, as well as for other applications in ceramic manufacturing. In particular, the results of testing digital fabrication technology, both successes and failures, can prove useful for other practitioners that make use of similar equipment.

We consider the prototypes for the two interfaces and the production system as work in progress, given that some steps still need to be taken for those prototypes to generate more detailed results. For example, usability tests should be performed on both interfaces in the scope of future research. The Qualifier interface in particular can benefit from further work, namely in obtaining additional qualities in order to improve the number of possible transformations to existing collections. Nevertheless, the main value of the work developed towards user interfaces and the production system is having demonstrated the possibility of linking the design system with other components necessary to a mass customization system.

### 7.2 Future work

The breadth of this research opens the way for a number of research paths for future work. We begin by addressing the Design Participation Model. As this was being implemented, an additional type of user was considered: the power-user. In computing jargon, a power-user is a knowledgeable and sophisticated user of computers. In the context of our design system, a power-
user is a user situated in-between the expert designer and the non-expert end-user, who would be interested in having more control over the design process than what is provided by the Navigator. For such power-user, it might be beneficial to grant access to the Modeler application, or perhaps to an additional application also situated in-between the Modeler and the Navigator. Yet another possibility would be to implement one single application with changeable levels of control. Therefore, a possible path for future work might be to integrate the figure of the power-user in the Design Participation Model.

Another possible path for future work is the improvement of the Tableware Shape Grammar (TSG). In future work, we suggest adding a number of features that have been deemed interesting during its development and use. In general, these improvements would provide greater control over the grammar, making the encoded knowledge explicit rather than tacit, as explained below. This tacit knowledge was implicitly embedded in the system by the researchers, while developing the grammar.

Rules should be fine-tuned by restraining its application to parts of a particular type by specifying the part’s type in the left hand side of the rule. Such improvement would provide the control needed for applying rules to parts according to type. For example, a future rule that attaches a handle to a part should only be applied to parts belonging to types such as cup or mug. Likewise, rule application should be restrained to a part with a specific function by specifying that function in the left hand side of the rule. In the current state of the TSG, Rule 7 is the only case where such restraint is explicitly defined, in which an envelope of a lying part is replaced by a straight surface, generating the base of the tableware element. In the current TSG, restraints of this sort are determined implicitly by the user of the grammar, for example when applying Rules 2a and 2b. Rule 2a creates a part for holding and should be applied to an envelope of a shallow type, whereas Rule 2b should be applied to deep types, in which the holding function is assigned to handles. Currently, these rules can be applied interchangeably and it is up to the user to apply them to the right type. However, such a restraint could be specified in the rule through appropriate labelling.

As a result of using the TSG as an intermediate tool for the implementation of the design system, some features were integrated in the implemented applications but failed to be formalized in shape grammar rules. For example, inclusion of parts such as handles and lids to specific types in the collection was implemented in the final Unity prototype, but not contemplated in the shape
grammar. Likewise, contour definition, which is determined in the implemented Modeler by the superellipse equation, is not reflected in the grammar. A number of features that have been identified during interaction with designers while testing Modeler should also be integrated in the TSG, such as an option to verify whether tableware elements geometry allows them to be stacked.

Possible research paths in order to validate the generalizability of the proposed methodology for obtaining a generic grammar include performing further comparative analysis with existing generic shape grammars (Beirão et al., 2011; Benrós et al., 2014; Garcia & Barros, 2015), or developing further applications to different design domains. One particularly interesting possibility is to confront the presented methodology with the shape grammars developed for Digital Alberti, given their influence on this research.

Regarding the implementation of the design system, a possible future research path would be to explore a multi-platform solution in line with what was discussed in the previous section, trying to find a solution for the current lack of an exchange format for parametric models. Such a research initiative would imply close collaboration with computer scientists, although some of the main issues to be addressed are related to design, such as geometry and design intent.

Regarding improvement on the interfaces, one available path is to verify whether all the interpolated solutions are valid from the point of view of mass customization, including design, production and price, assuming that limit solutions are valid themselves. Future work should first and foremost include user tests for both of the proposed interfaces. Focusing on the Qualifier interface, its current implementation is based on the variation of parametric configurations and the association of such variation to qualitative descriptors, which constrains design space exploration to topologically similar design solutions. Extending this approach to the level of shape grammar rules would open new possibilities, since the Qualifier interface could be used to explore topologically different collections. In such an extended Qualifier, qualitative descriptors could be associated to differences in rule application, namely which rules were applied and, in a hypothetical more sophisticated model, in which order those rules were applied. Transformations in design (Knight, 1994) could be used to operate these differentiation processes, considering that each collection is a specific grammar, capable of generating a number of types.

Finally, we suggest research paths for the production system. Despite the selection of material extrusion for the experiments, alternative methods were not discarded, and can be pursued in
future research. In the additive manufacturing survey, other techniques were shown to be possible alternatives to material extrusion. Despite its higher costs, binder jetting was deemed as the *de facto* technique for digital fabrication of ceramic objects when the survey was conducted. A number of different research projects were studying this technique, and companies like 3DSystems showed signs of investing in the technology, through acquisition of companies like ZCorporation and Figulo and the announcement of CeraJet. However, since then, binder jetting appears to have been losing relevance relatively to other techniques such as material extrusion and vat photopolymerization.

Vat photopolymerization shows potential as an alternative additive manufacturing technique for producing customized tableware. US-based company Tethon3D, who supplies materials for additive manufacturing of ceramic objects through binder jetting, now provide a photo-curing ceramic composite resin for SLA and DLP equipment (Tethon3D, 2017). Also, the cost of SLA equipment has been decreasing due to expiration of patents. In the survey we conducted, besides material extrusion and binder jetting, the only machines with a cost below €10,000 were Digital Wax Systems and FormLabs, who apply this technology in their printers Xfab and Form 2, respectively. The remaining machines show prices in the order of hundreds of thousands of dollars. Although these printers were discarded due to the reduced dimensions of their building envelopes, we should expect those dimensions to increase in the near future.

If we look beyond additive manufacturing, we find additional alternatives for future production of customized tableware. While additive manufacturing enables the direct production of ceramic elements, formative methods such as the use of molds provide an indirect alternative. In our research, additive methods of digital fabrication were selected over formative methods because our contacts with companies showed that profitability of forming techniques like slipcasting begins in the order of hundreds of items. Nevertheless, formative methods are being used commercially, as shown by Shapeways, who uses SLS additive manufacturing technology to produce one-off molds of models ordered by customers (Shapeways, 2017), thereby suggesting commercial viability of these methods too. Also, by being practiced by these companies, who have relatively large exposure to general audiences, the technique is also exposed to the wider public, which might result in decreased costs in the future.

A final alternative is to analyze the costs of using third party production service suppliers such as Shapeways and iMaterialise. Although a preliminary analysis reveals that costs are high, it might
be worth to keep track of service prices since they might decrease with time, as technology is embraced by the community.

7.3 Final note

The results discussed previously reflect the breadth of this research. The research focus was primarily on the design system, but other components of the proposed mass customization system, such as the user interface and the production system, were also addressed to demonstrate the feasibility of the framework. The implementation of a proper mass customization system for product design depends also on other components not covered in our research, such as logistics (Barros, 2015). Nevertheless, we believe that the research presented in this dissertation makes a contribution towards proper implementation of mass customization. Despite the focus on ceramic tableware, we believe that results can inform mass customization processes in other domains of design, such as product design or architecture.

More than contributing to the implementation of a mass customization system, this research contributes to the resolution of a number of challenges that arise in that context, through innovative strategies such as the proposed Design Participation Model, which grants designers a more preponderant role in a mass customization process, or the adaptation of existing design systems to new design domains, exemplified by the reuse of code from Digital Alberti and in the proposed methodology for inferring generic grammars. Experimentation with such strategies provided results and insights that we hope to be useful for further research in design for mass customization but also in the field of digital design.
Reference list


Alberti, L. B. (1485). De Re Aedificatoria. Florence, Italy: [publisher not identified].


https://doi.org/10.1145/971300.971420


Lauerman, T. (2017, July 4). Loading clay into the tube for printing by hand was leading to lots of air b... Retrieved September 5, 2017, from https://plus.google.com/+TomLauerman/posts/VdYPbdyjyKC


Appendix A:
Tableware Shape Grammar

We hereby present the elements that compose the Tableware Shape Grammar, which was analyzed in Chapter 3. We begin by presenting the rules that were inferred from analyzing six existing collections: Romantica, Bisel, Catarina de Bragança, Cotelé, Cuisine Provençale, and Flor. Then we present the derivations for generating each of those collections. Remember that the Single Collection Shape Grammar corresponds to a specific grammar for generating the Romantica collection. Also, remember that the focus of the comparative analysis was on decoration, and therefore the derivations are applied to the decoration phase for all but the Romantica collection. Finally, we present the tables that illustrate the process of inferring the generic grammar from the specific grammars for generating each of the six collections.
Specific shape grammars: Rules for generating the base shape

Initialization

Subdivision

Distortion

Substitution
Specific shape grammars: Rules for generating the decoration

**U-subdivision**

1.  

2.  

3.  

**V-subdivision**

1.  

2.  

3.  

**Curved subdivision**

1.  

2.  

**Shift**

1.  

**Contour**

1.  

2.  

**Relief**

1.  

2.  

3.  
Derivation: soup plate, Romantica collection – base shape
Derivation: soup plate, Romantica collection – decoration
Derivation: cereal bowl, Romantica collection – base shape
Derivation: cereal bowl, Romantica collection – decoration
Derivation: dinner plate, Catarina de Bragança collection
Derivation: dinner plate, Cuisine de Provençale collection
Derivation: dinner plate, Flor collection
Derivation: sugar bowl, Bisel collection
Derivation: sugar bowl, Cotelé collection
Shape rules of the six encoded collections

<table>
<thead>
<tr>
<th>u-subdivision</th>
<th>R11a</th>
<th>□ → □</th>
<th>□ → □</th>
<th>□ → □</th>
<th>□ → □</th>
<th>□ → □</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R11b</td>
<td>□ → □</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>R11c</td>
<td>□ → □</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>v-subdivision</td>
<td>R12a</td>
<td>□ → □</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>R12b</td>
<td>□ → □</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>R12c</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
<tr>
<td>curved subdivision</td>
<td>R13a</td>
<td>□ → □</td>
<td></td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
<tr>
<td></td>
<td>R13b</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
<tr>
<td>shift</td>
<td>R14</td>
<td>□ &gt; □</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contour</td>
<td>R16a</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
<tr>
<td></td>
<td>R16b</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
<tr>
<td></td>
<td>R16c</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
<tr>
<td></td>
<td>R16d</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
<td>□ → □</td>
</tr>
<tr>
<td>relief</td>
<td>R18a</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
</tr>
<tr>
<td></td>
<td>R18b</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
</tr>
<tr>
<td></td>
<td>R18c</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
<td>□ → □ (□)</td>
</tr>
</tbody>
</table>
Shape rules after revision of the specific grammars

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
<th>Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>R11a</td>
<td></td>
<td><img src="image1.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R11b</td>
<td></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R11c</td>
<td></td>
<td><img src="image3.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R12a</td>
<td></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R12b</td>
<td></td>
<td><img src="image5.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R12c</td>
<td></td>
<td><img src="image6.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R12d</td>
<td></td>
<td><img src="image7.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R13a</td>
<td></td>
<td><img src="image8.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R13b</td>
<td></td>
<td><img src="image9.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R13c</td>
<td></td>
<td><img src="image10.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R14a</td>
<td></td>
<td><img src="image11.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R14b</td>
<td></td>
<td><img src="image12.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R14c</td>
<td></td>
<td><img src="image13.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R15a</td>
<td></td>
<td><img src="image14.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R15b</td>
<td></td>
<td><img src="image15.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R15c</td>
<td></td>
<td><img src="image16.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R16a</td>
<td></td>
<td><img src="image17.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R16b</td>
<td></td>
<td><img src="image18.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R16c</td>
<td></td>
<td><img src="image19.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R16d</td>
<td></td>
<td><img src="image20.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R17a</td>
<td></td>
<td><img src="image21.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R17b</td>
<td></td>
<td><img src="image22.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R18a</td>
<td></td>
<td><img src="image23.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R18b</td>
<td></td>
<td><img src="image24.png" alt="Diagram" /></td>
</tr>
<tr>
<td>R18c</td>
<td></td>
<td><img src="image25.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Shaded**: added rules

**Dotted**: deleted rules
Rule generalization through grouping and parameterization

R11a

R11b

R11c

GRule 11

\[ u = \sum_{i=1}^{n} u(i) \mid n \in \{2,3,...,n\} \]

\[ u(i) \in [0,u] \]

R15a

R15b

GRule 15

R12a

R12b

R12c

R12d

GRule 12

\[ v = \sum_{i=1}^{n} v(i) \mid n \in \{2,3,...,n\} \]

\[ v(i) \in [0,v] \]

R17a

R17b

GRule 17

R13a

R13b

GRule 13

\[ v = f(u) \cdot u, v \in [0,1]\]

\text{curve generated from input function};

R18a

R18b

R18c

GRule 18

\[ u = 2 \cdot u1 + u2 \]

\[ u1, u2 \in [0,1] \]

\[ v = 2 \cdot v1 + v2 \]

\[ v1, v2 \in [0,v] \]

\[ w \in [-2v1, v2] \]

\[ x = \min(u, v) \]
Generic grammar rules and parameters used to encode the six collections

<table>
<thead>
<tr>
<th>Rule</th>
<th>Diagram</th>
<th>Romanica</th>
<th>Bisel</th>
<th>Catarina...</th>
<th>Cotelé</th>
<th>Cuisine...</th>
<th>Flor</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR11</td>
<td>![Diagram](u \rightarrow u' u_2)</td>
<td>1:1</td>
<td>1:2:1</td>
<td>1:1</td>
<td>1:1</td>
<td>1:1</td>
<td>1:1</td>
</tr>
<tr>
<td>GR12</td>
<td>![Diagram](v \rightarrow v_1 v_2)</td>
<td>1:2:1</td>
<td>1:1</td>
<td>1:1:2</td>
<td>1:2:1</td>
<td>1:1:2</td>
<td>1:2:1</td>
</tr>
<tr>
<td>GR13</td>
<td>![Diagram](u \rightarrow f_{s_0})</td>
<td>(gauss)</td>
<td>(gauss)</td>
<td>(gauss)</td>
<td>(gauss)</td>
<td>(gauss)</td>
<td>(gauss)</td>
</tr>
<tr>
<td>R14a</td>
<td>![Diagram](_ \rightarrow _ )</td>
<td></td>
<td></td>
<td>USED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R14b</td>
<td>![Diagram](_ \rightarrow _ )</td>
<td></td>
<td></td>
<td>USED</td>
<td>USED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R14c</td>
<td>![Diagram](_ \rightarrow _ )</td>
<td></td>
<td></td>
<td>USED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GR15</td>
<td>![Diagram](_ \rightarrow _ )</td>
<td>(top)</td>
<td>(top)</td>
<td>(bottom)</td>
<td>(bottom)</td>
<td>(top)</td>
<td>(bottom)</td>
</tr>
<tr>
<td>GR17</td>
<td>![Diagram](_ \rightarrow _ )</td>
<td>(horiz)</td>
<td>(horiz)</td>
<td>(horiz)</td>
<td>(horiz)</td>
<td>(horiz)</td>
<td></td>
</tr>
<tr>
<td>GR18</td>
<td>![Diagram](u \rightarrow v_1 v_2 v_1)</td>
<td>-U:-V</td>
<td>-U:-V</td>
<td>+U:+V</td>
<td>-U:-V</td>
<td>-U:-V</td>
<td>+V</td>
</tr>
</tbody>
</table>
Appendix B: Insights gathered within Contextual Inquiry

Observations and information gathered was registered by the research team during inquiry days in notebooks, and typed into word processing in a freeform report in the end of the day. Analysis of this report resulted in a series of important insights provided by the design team members, which either confirmed hypotheses that had been previously considered by the research team, or revealed new, unforeseen aspects of the process of designing new tableware collections. In this section, we will present each of these INSIGHTS, accompanied by its INTERPRETATION by the research team. Interpretation of these insights resulted in a set of internal recommendations that ought to be implemented in the design system. These recommendations range from broader design strategies that result in high-level decisions in the design system’s design, indicated as IMPLICATIONS, to more detailed issues that lead to particular FEATURES in its implementation.

It should be noted that the inquiry subjects were aware that such CI was part of an effort to develop a computational application for tableware design. Therefore, along with the insights about the design process itself, the Manufacturer’s design team also provided direct input about features that they would like to see present in the eventual application. Such direct recommendations have also been registered in a list of features that ought to be implemented either in the final prototype of the present research, or in a future iteration of the implemented design system. Such list of features, which we call Wishlist, will be referenced further afield, when we talk about User Tests. Moreover, the Wishlist that results from direct recommendations from the surveyed designers will add up to the Features inferred from the Insights.
**INSIGHT** - The process of designing a new collection can start from the ground up, or be based on a previously existing collection.

**INTERPRETATION** – Such approach enables the design team to ensure that some design qualities of the previous collection are passed on to the new collection. Besides the transmission of design qualities, such approach also saves time in the design process. In the mass production scenario in which the observed design team operates, it is useful to use previous designs, thus minimizing the need for changes in the production process.

**IMPLICATION** – The process of creating a new collection that is based on a pre-existing one can be formalized within the shape grammar as a grammar transformation (Knight, 1994).

**FEATURE** – It should be possible to save design solutions, as well as to retrieve them afterwards, and more importantly, it should be possible to continue editing a retrieved solution.

---

**INSIGHT** – In Portugal, there is no actual regulation for standard sizes in ceramic tableware.

**INTERPRETATION** – The sizes for different types are therefore determined based on the designers’ previous knowledge.

**IMPLICATION** – The size of each type is an essential element of information for the design system. Instead of having the designers elaborate on what they think are the adequate sizes for each of the tableware types, such values will be inferred from the analyzed tableware collections.

**FEATURE** – It seems useful to allow for variations in the overall dimensions of each of a collection’s type.

---

**INSIGHT** – Some more “geometric” designs can feature straight lines.

**INTERPRETATION** – Such insight counters an initial intuition that the design style of the analyzed Manufacturer was characterized exclusively by curved shapes.

**IMPLICATION/FEATURE** – The design system should allow for straight lines both in profile and in contour, in addition to the curved lines that were already planned.
**INSIGHT** – Thickness should be consistent across the collection’s types.

**INTERPRETATION** – In the example of plates, we verify that the shape of a smaller plate, like for example a dessert plate, cannot be obtained by simply scaling down the design of a dinner plate, for such approach would also scale down the thickness of the dessert plate, eventually compromising its physical integrity.

**IMPLICATION / FEATURE** – [Accumulate with next insight.]

**INSIGHT** – Thickness might be reduced towards the rim of an element to render it more elegant.

**INTERPRETATION** – Thickness is not necessarily the same along the same collection type.

**IMPLICATION** – Distance between inner and outer surfaces of the tableware elements should be properly controlled, namely by rules that take its type into consideration, rather than relying on a general rule to govern thickness of all types as a whole.

**FEATURE** – The implementation should offer the user the possibility of different thicknesses.

**INSIGHT** – A design is copied on top of itself to assess if it is possible to create stacks of it. Moreover, it is useful to stack different types, such as a saucer and a tea cup, to assess their compatibility.

**INTERPRETATION** – Stackability is a desired design quality, at least for plates and other similarly shallow types such as trays, or between types that usually go together. It is useful to analyze a design’s stackability.

**IMPLICATION** – Originally, the main focus of the design system is generation of shapes, and therefore we have not focused on developing analytical tools. Nevertheless, it might be easier for the designer to visually assess the stackability property, rather than encoding analytical rules for analyzing the stackability property.
**Feature** – Simulating a tableware element’s stackability implies rendering copies of the element on top of each other. Moreover, it must be possible to view the designs in some sort of section view, with which the user can visually assess the desired stackability.

**Insight** – Two different types of the newly designed collection were based in different pre-existing collections.

**Interpretation** – Such insight seemed to go against the principle of coherence that had been established for the design system. Nevertheless, it seems to be useful for designers to select different types from different pre-existing collections as a starting point for a new collection.

**Implication** – Enabling the creation of a collection based on different partial collections implies greater complexity in the design system, which would have to be able to manage divergent parametric models for the same collection.

**Feature** – Having accommodated the design system to manage different partial collections, it should be made possible to the designer to select different types from different collections to begin creating a new collection, instead of departing from a complete pre-existing collection.

**Insight** – During development of a new type, some different versions are saved, so they can be later compared, and the best one selected.

**Interpretation** – This is a recurrent strategy in many design processes, including architectural design (Mateus, 2013) or software design (made possible by version control).

**Implications** – Saving different solutions can be useful either for serving as future basis for a new design, keeping a record of the design process, as well as for comparing alternative designs. Moreover, such strategy suggests an advantage in keeping a more formalized historic record of the design operations performed during the creation or transformation of a collection.
**Feature** – More than the ability to save design solutions for later retrieval, it should be made possible to open and display more than one collection at a time, enabling the user to compare them.

**Insight** – A stylistic trait of a particular collection can be the existence a part that features constant dimension in all types of the collection.

**Interpretation** – An example was witnessed of a collection in which the lips of all plates, which have different sizes according to their type, feature a lip with the same width – that is, the rim is unaffected by the scaling from one type to the other. In that case, dimensional variation due to scaling from one type to the other is absorbed by the remaining parts.

**Implication** – Such approach would imply implementing user-defined constraints in the design system, so that inter-type scaling would not affect constant-dimensioned parts, pointing toward a constraint-based modeling paradigm (Hoffmann & Joan-Arinyo, 2002), which falls outside the scope of this research.

**Geometry paradigms**

Among the various insights gathered in the scope of the Contextual Inquiry, two have been considered as main insights, mostly due to the impact that they had in the design system’s philosophy. The first one relates to the type of geometry used to describe tableware elements.

**Insight** – For describing the curved profile of tableware elements, designers use both circumference arcs and freeform curves.

**Interpretation** – The profile of a tableware element might be a relatively complex shape: typically it is a curved shape, along which its attributes might vary, such as curvature or concavity. Observation within the current CI showed that designers can describe tableware profiles using two different types of geometry: circumference arcs, and freeform curves. Due to
the relative complexity of those profiles, designers take some time to find a satisfactory shape using either one of the geometry types, for different reasons.

It is our interpretation that designing a profile using circumference arcs exclusively implies a relatively large number of design operations because the designer struggles to control continuity. Typically, CAD applications provide functions such as snapping tools that help the designer guaranteeing positional continuity (C0). However, less support is provided for guaranteeing tangential (C1) or curvature (C2) continuity. Some CAD applications provide operations that take tangential vectors as input for creating circumference arcs, enabling the designer to create arcs with tangential continuity. However, when these arcs are to be later edited, no support is provided to maintain such continuity. Moreover, curvature continuity between circumference arcs is only possible if they feature the same radius.

A suitable alternative to arcs for describing tableware profiles is using freeform curves, such as Bézier or NURBS curves. If a profile is represented by a single NURBS curve, both positional and tangential continuity are guaranteed, being inherent to the construction itself, and even curvature continuity is likely accounted for. Learning how to design freeform curves using CAD tools requires some experience from the designer. However, using freeform curves relieves the designer of the time-consuming task of manually guaranteeing geometric continuity, and should therefore justify the time invested in learning how to work with these tools.

**IMPLICATION** – The Contextual Inquiry shed some light on which type of geometry do tableware designers normally use. The initial iterations of the design system were based on a hybrid solution between circumference arcs and NURBS curves, in which simple Bézier curves are combined into more complex profiles. Since both approaches – circumference arcs and freeform curves – were used by the inquired designers, there was no preference of one over the other.

**FEATURE** - Due to its efficiency in dealing with continuity, the NURBS paradigm has ultimately been adopted in the computationally implemented design system.

**Designing a collection as a whole**

A second main insight relates to the strategy used by designers to maintain a collection’s style.
**INSIGHT** – Designers attempt to guarantee a stylistic coherence among the different types of a collection by first transforming the types which were created first.

**INTERPRETATION** – Good examples are when designers transform the profile of a cereal bowl into a salad bowl, or a dinner plate into a dessert plate. These particular transformations are carried by first scaling up or down the original profile, and then adapting its thickness. Although designers usually proceed by applying some additional changes to the resulting profile, it appears to be useful and possible to automatize the initial standardized operations that manage size and thickness of the tableware elements.

**IMPLICATION** – In order to automatize the mentioned standard operations, the design system should provide that changes in one type are replicated in all other types of a collection, to what we call collection-wise editing.

**FEATURE** – In the computational implementation, this meant that all types are generated as instances of an Archetype, which informs about the shape of all of the collection’s elements. Details on implementing such Archetype will be discussed in Chapter 4.
Appendix C: Publications


Eduardo Castro e Costa is a researcher in Design and Computation. Presently he is concluding his PhD in Architecture in the Pennsylvania State University, on the subject of 'Mass Customization of Ceramic Tableware'. He started this research as a PhD candidate in Design at the School of Architecture of the University of Lisbon (FAUL) in 2012, funded by Fundação para a Ciência e a Tecnologia with a Doctoral grant.

He graduated in Architecture from the University of Lisbon in 2006, and worked in Franken Architekten GmbH in Frankfurt, Germany and Hanoi, Vietnam from 2007 to 2009, specializing in areas such as 3D modeling and scripting. He collaborated as a researcher in the Digital Alberti project between 2011 and 2013, applying parametric modeling and digital fabrication to the Albertian column system, and completed M.Arch. at FAUL in 2012, with a dissertation on the work developed for the Digital Alberti project. He also collaborated as a researcher in the TECTON 3D project between 2013 and 2016, research on innovative interfaces for architectural design using multi-modal interaction techniques and immersive virtual reality environments.

Besides a PhD student, he was teaching assistant to the courses 'Digital Alberti' and 'Digital Design and Fabrication' and ‘Computer Programming for Artists and Designers’, and collaborator in laboratories dedicated to Digital Fabrication and Virtual Reality. His research interests dwell along the application of computation to Architecture and Design, namely generative design systems, from parametric and procedural modeling to shape grammars, along with mass customization, digital fabrication, user interfaces and 3D modeling, with a special interest in programming and scripting.