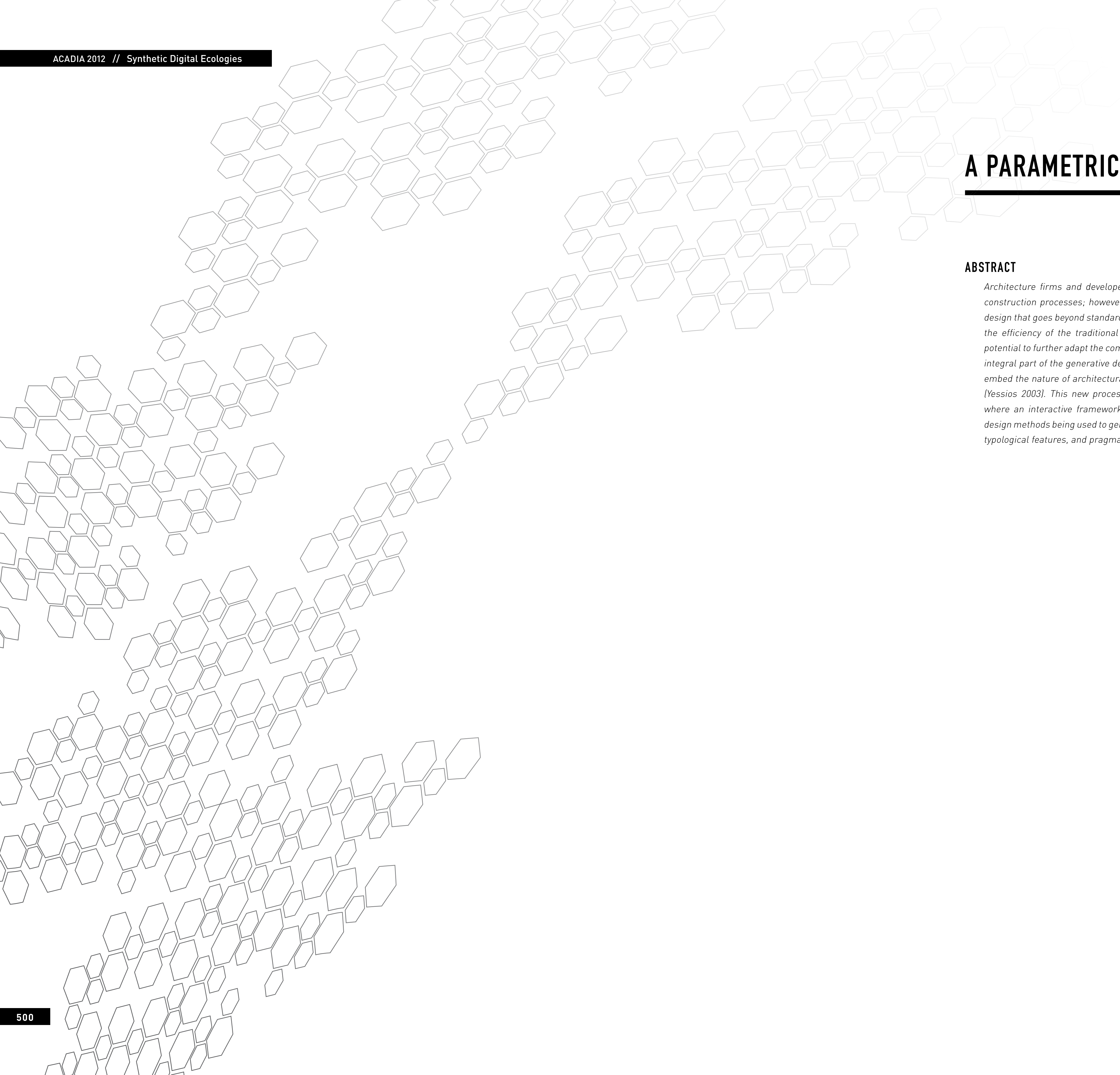

A PARAMETRIC MULTI-CRITERION HOUSING TYPOLOGY

ABSTRACT

Architecture firms and developers often use computers to streamline the documentation and construction processes; however, these firms often fail to engage computers with a process of design that goes beyond standard representation and documentation. Although this has increased the efficiency of the traditional approach to architecture, an alternative methodology has the potential to further adapt the computer's role in architecture, making it and its explicit logic a more integral part of the generative design process. New computational methodologies can be used to embed the nature of architectural design within a system of internal parametric representations (Yessios 2003). This new process allows for the creation of computationally designed systems where an interactive framework aids in the design process. This paper discusses parametric design methods being used to generate housing types based on multiple criteria of site constraints, typological features, and pragmatic housing functions and details.

Bryce Willis
University of Nebraska–Lincoln

Timothy Hemsath
University of Nebraska–Lincoln

Steven Hardy
University of Nebraska–Lincoln

1 INTRODUCTION

Developers who consider custom residential architecture as merely a buyer's choice from a list of interior finishes currently lead the home-building market. As a result, four or five floor plans populate a neighborhood. Architects currently account for a negligible portion of the residential architecture industry in the United States, being limited primarily to the design of rare, expensive custom homes. Although these homes often push the typology of a residential architecture, they are not an economical solution for most home design. In "Towards a Fully Associative Architecture," Bernard Cache showcases the Philibert de L'Orme Pavilion and his fully associative design and manufacturing process, which allowed him to produce everything from the initial form of the pavilion to the 100 percent CNC custom kit of parts.

Cache's projects elaborate on the traditional design methodology and production methods. His parametric methods are used to define greater complexity, balancing multiple criteria within his designs. Like most architect-designed custom homes, these methods are not widely affordable, and so the power of parametric methods cannot be captured by the average person, home buyer, or developer. This tool—the multi-criterion parametric housing modeler—optimizes multiple variables and arranges conflicting parameters into complete design solutions. In this instance, this complex process is realized by breaking down housing into discrete typological elements, which allows the parametric model to generate a complete, cost-effective, site-specific, and functional housing design from the ground up. By implementing new generative methodologies, these underutilized parametric systems can be leveraged to generate custom home solutions to both fully engage computers within an architectural design process and raise the quality of current housing practices.

2 RESEARCH

In Elements of Parametric Design, Robert Woodbury outlines various parametric-based concepts for modeling. Nathan Miller (2009) discusses parametric strategies in civil architectural design. In Michael Hodge's paper, he says generative algorithms demonstrate the professional applicability of parametric modeling. More recently, Hachem, Athienitis, and Fazio (2010) and Kampf et al. (2011) investigated the parametric optimization of various geometric housing units based on solar radiation as individual units and the aggregation of units in an urban block. These two projects limited geometric variation and did not consider other housing criteria critical to cost, site, function, or customization of housing. These examples begin to prompt the ability to apply algorithmic tools on a single, large-scale project. Previous experiments in generating architecture primarily have been reserved to housing design and have involved shape grammar by George Stiny and discursive grammar by Jose Duarte.

2.1 Shape Grammar

"Shape grammars perform computations with shapes in two steps: a recognition of a particular shape and its possible replacement. Rules are used to specify the particular shapes to be replaced and the manner in which they are replaced. Underlying the rules are transformations that allow one shape to be part of another" (Stiny and Gips 1972).

One example in "The Language of the Prairie: Frank Lloyd Wright's Prairie Houses" investigates ways of manually recreating a design style. For Koning and Eizenberg, recreations of Frank Lloyd Wright's prairie-style homes use a "parametric shape grammar that generates compositional forms and specifies the functional zones" (1981). Their system succeeds in its ability to lead an architecturally educated person through a series of steps or choices to achieve a similarly styled output each time. However, that same person could create a similar output more quickly by relying on his or her architectural abilities.

The advantages a shape grammar–based system could offer are vast. Although it is an inefficient model for architects to use within their own design processes, shape grammar's rule-based nature lends itself well to adaptation for use within computational systems and frameworks. The ability to apply different rules at different times within the design of a single solution allows shape grammars to be flexible within a large system and to create a wide range of final designs. It is fairly straightforward to parametrically create a simple shape grammar system because of the limited number of rules that can be applied at any given location. Attempts to program more complex shape grammar systems have been limited. Where an architect would know intuitively which rules to select for application and where to apply them, programmers have trouble getting the computer to identify the appropriate locations to apply the usable shape grammar rules. As a result, to computationally recreate interchangeable rule sets, a vast amount of computer coding is required.

2.2 Discursive Grammar

A discursive grammar is made of both programming and design grammars. "The programming grammar generates design briefs based on user data; the designing grammar provides the rules for generating designs in a particular style, and a set of heuristics guides the generation of designs towards a solution that matches the design brief" (Duarte 2001). Duarte provides the approach for creating custom homes for the masses, using the case of Siza's houses at Malagueria with the goal of computationally recreating architecture configuration. In his paper, Duarte defines a more computer-oriented process for recreating Siza's houses at Malagueira, but his system relies on a process of division. Starting from an initial perimeter, it arranges the same program within the perimeter in slightly different patterns. As a result, each home becomes a reflected or rotated version of the one before it. Although both shape and discursive grammars lend themselves to adaptation for computational re-creation, they both rely on a set of rules or steps that strictly predefine an output and limit themselves only to internal relationships.

2.3 Architect–Designed Homes vs. Home Builder Homes

There are two different design models currently employed in the housing market, one by home builders and the other by individually contracted architects. The home builders' strategy is to limit the number of predesigned homes within a couple of basic types, and then repeatedly build them based on their understanding of the market conditions. This business model finds value within the iteration of the same cost-optimized designs. With limited profit per each home-build, a home builder's goal is similar to that of a big box store: sell as much of a single low-cost product as possible and reduce costs within your structure through the optimization of the product and control over the manufacturing or construction. The downside to this model is that it encourages builders to reduce the quality of their final products based on a drive to increase profits. Conversely, the model is advantageous in its ability to rapidly proliferate a product at an affordable level. Because builders have such strict control over their processes to keep costs down, they become locked into a set of strategies that reap the best results. For example, home builders limit themselves to two different home types, ranches and split-levels, based on cutting costs to create the houses' foundations. To allow relatively untrained people to make these decisions, home builders develop standards based on their value. These standards, because of their rigidity, lend themselves well to being created within a computational system.

Architects primarily employ a second model. This business model finds value within the custom creation of a single design. For the business to remain solvent, the value of each design carries a much higher cost, and the home is much more slowly constructed because of the need to be individually developed. The limiting factor of this method then becomes one of cost and affordability, which, in a market that values, above all, time and money, limits that model's market share. The advantages of this method are, however, much more vast than those of the builder. Homes are designed specifically for their inhabitants and their location. As a result, these homes actually respond to their surroundings, are functionally superior for their inhabitants, have a much higher material finish, and are generally built with a superior quality and level of control.

2.4 Pattern Books

Each era of American history has provided us with manuals, known as pattern books, that served to coordinate the home-building process by providing designs suited to the technology and objectives at the time (Grindroz et al. 2004). Pattern books are to home builders as mom-and-pop stores are to Walmart. They are a past model, one which blended a few of the qualities of an architect-designed home with many qualities of current home builders. Pattern books were basically manuals that contractors used to build homes for individual clients. These books were generally produced by architects with sets of plans and details that could be used interchangeably for the final design and construction of the home. As a result, architects relinquished some control over the final products, but the system allowed architects to affect a greater number of homes, while letting contractors slightly adjust the design of a home to better fit its context. A contractor's clients received a greater variation between each home built in the area. Because clients worked directly with the contractor, they were forced to provide better products and services. As a result, each neighborhood built from a set of pattern books has a relative cohesiveness between each of the homes, while maintaining individuality.

These pattern books can be thought of as the analog form of a parametric model. Each iteration is distinct and varied, but is also strongly related to the iteration created before and after it. This methodology allows for translation into a digital framework much more easily than today's typical builder-developed home. Just as "The Language of the Prairie: Frank Lloyd Wright's Prairie Houses" demonstrated, a series of interrelated homes by the same architect have the ability to be analyzed and turned into a shape grammar model; a pattern book can be developed into a discursive grammar within a computational framework to produce individualized but similar homes.

3 ANALYSIS

For the purpose of analyzing the computational potential of these ideas, they were mainly tested with Grasshopper. Grasshopper is a graphical scripting program that plugs into Rhino's 3D modeling and visualization tools. This software is advantageous because of its forgiving learning curve, and unlike other scripting programs, knowledge of scripting or programming languages is not needed. Four main experiments were initially conducted: a grammar-based American foursquare home; a home builder's interpretation of site and its resulting typological shift; a simplified recursive cut-and-fill calculation; and a solver-based bubble diagram floor plan generator.

3.1 Grammar-Based American Foursquare Home

For the creation of a grammar-based home, several different home typologies were analyzed. Most homes built by home builders today are of two general typologies: one- or two-story ranch homes and split-level homes. Although there is a clear type delineation, home builder homes tend to be poorly laid out in plan, resulting in a difficult translation into a grammar-based set of rules or divisions.

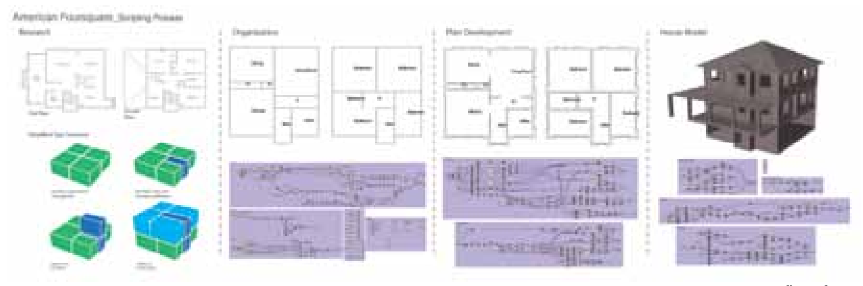

figure 1

figure 1

Diagram of American foursquare scripting process, which shows the four stages: first, the understanding of the typology; second, the script that organizes the program; third, the script that converts that organization into floor plans; fourth, the script that creates a massing model from those plans.

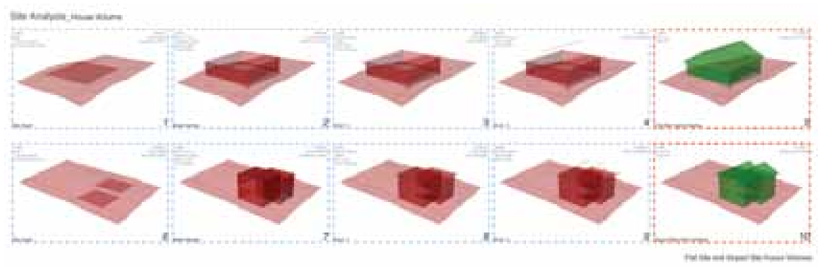

figure 2

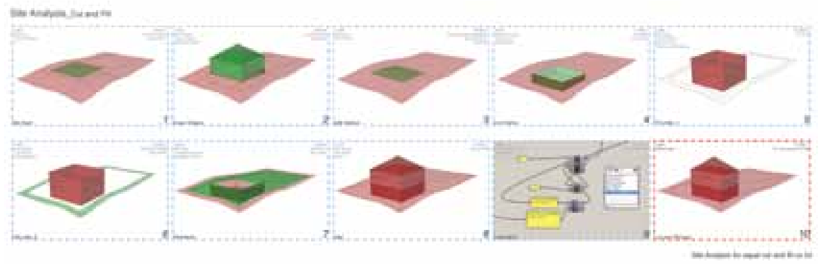

figure 3

For this reason, traditional American homes and pattern books were researched. The American foursquare was selected due to its strong parti, which lent itself to more easily be developed into a simple grammar based on quadrants to reproduce its plans. The script was developed in three stages: the plan-based organization, plan development, and house-massing model.

The plan-based organization of the home uses the grammar-based logic to create the basic arrangement based on an American foursquare parti diagram. By scripting it parametrically, a user can change values and decisions that work within the grammar to produce a series of different American foursquare homes. The grammar allows changes on several scales, from manipulation of the internal organization to the size of individual rooms. Due to the coding of the grammar, the overall form, or perimeter bounds, is similar, whereas the plans can vary significantly, which is consistent with the American foursquare pattern books.

The plan development step takes the initial plan organization developed in the first step and continues its process to create a typical floor plan. Because everything is parametrically based, changes made to the previous step result in changes within this step as well. For simplicity, only doors and windows have parametric controls, and they allow for the manipulation of elements within walls and resulting circulation paths.

The house-massing model references the data from the previous plan definition script to develop the massing of the house. Compared to the two previous steps, relatively little information is needed that has not already been defined. The porch and roof elements in this step are combined with the developed plans to create a representation of a traditional American foursquare house.

These stages are linked together to allow any input or value to be edited or changed to update the resulting massing model. Since the process is parametrically driven without placing limits on the inputs, the grammar and coding of the script will allow the creation of both functioning and nonfunctioning floor plans.

figure 2

Diagram that shows in a series of steps the process the script goes through to replicate the home builders' response to topography.

figure 3

Diagram showing the steps the script goes through to create an equal amount of cut and fill on a site.

figure 4

Diagram that shows a few of the parametric controls in the first portion of the framework.

3.2 Site-Based Analysis and Type Massing

From observation, typical home builders construct two different types of homes, and they base which type to use on their initial analysis of the site. The main difference between these homes is the number of finished floors. The first type, ranch, works on flat sites and generally has a single finished floor on each level of the home that matches the area for the footprint of the home. The second type, split-level, is used when the site is sloped across its front elevation. In this situation, the ranch type is difficult to use due to the extensive earthwork needed to level the site. Instead, the split-level creates multiple finished floors in the form of half-levels. This basic difference in design is how home builders place their home types on a specific lot. This process can be recreated parametrically, allowing the computer to automatically respond to the difference in site conditions. The resulting Grasshopper script is an interpretation of the home builder process. Inputs into the script range from city-regulated site setbacks to the manipulation of the elements of the home—for example, square footage and number of levels to be placed on the site. The information is then used to visualize the volume of a typical home builder home on a given site. This script demonstrates the basic ways in which a home builder interacts with a residential lot.

3.3 Cut-and-Fill Parametric Recursive Loop

Another way home builders control cost is by reducing the amount of dirt that needs to be taken away from or added to a site. When it is possible, sites are graded so that the dirt removed from the site to create the basement is equal to the fill put onto the site to create a level driveway and to mitigate water flows on the site by creating a sloped grade away from the house. This process also lends itself to computation. Its complexity requires a different computational strategy, however. The initial Grasshopper scripts were parametrically driven, and the data flowed linearly through a series of steps. This script uses a plug-in called Hoopsnake, which adds a recursive loop to allow a calculation to be made by stepping through the process several times until a solution can be found. This script first calculates the volume of material for removal for the basement and foundations, and then uses that dirt as fill material to grade the site.

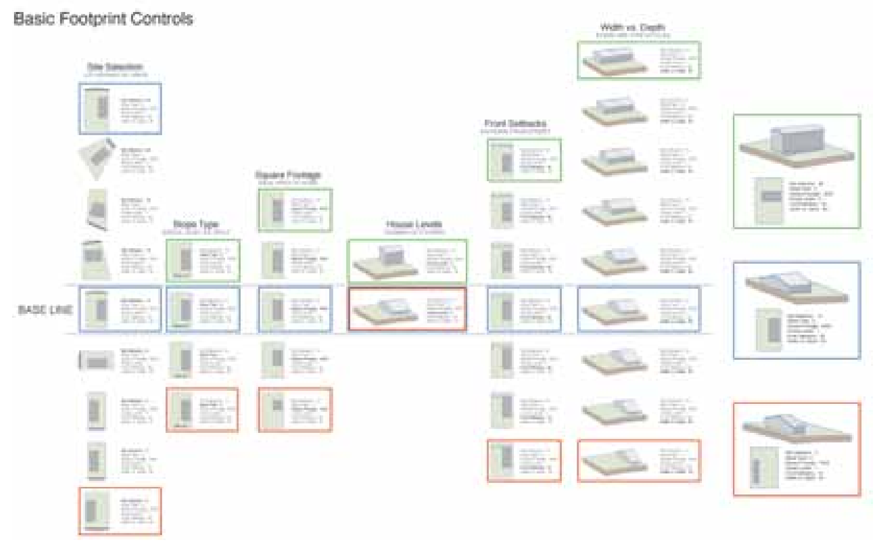

figure 4

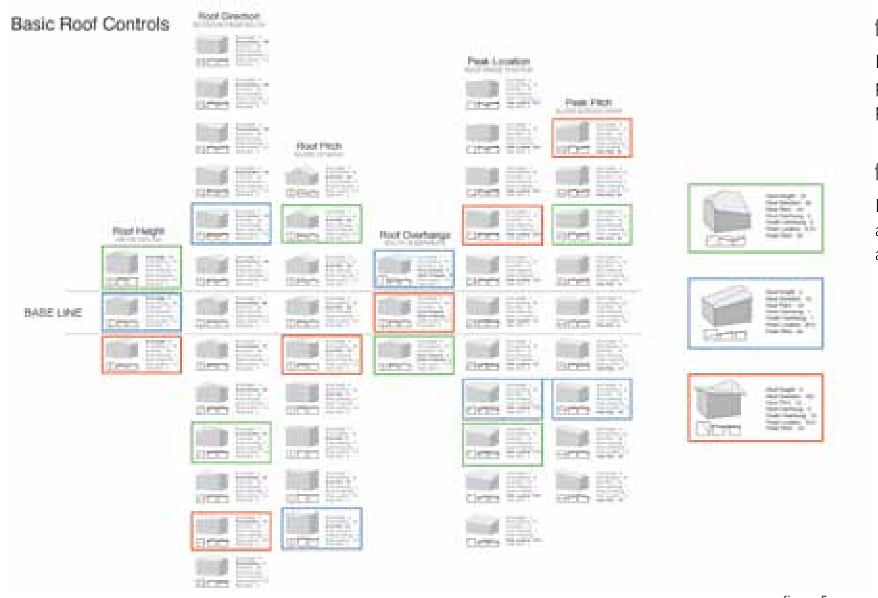

figure 5

figure 5

Diagram that shows a few of the parametric controls in the second portion of the framework.

figure 6

Diagram that shows the process of applying a typology-based grammar to an initial mass.

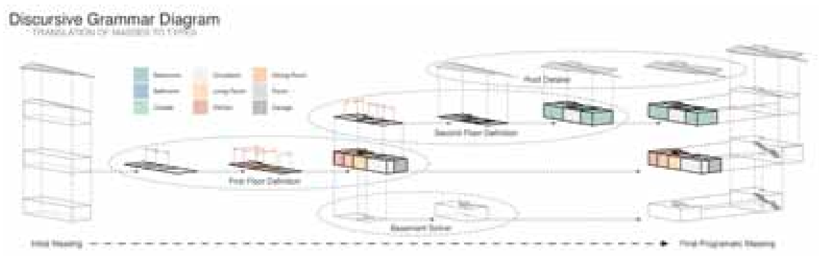

figure 6

4 A COMPUTATIONAL FRAMEWORK

From the initial site- and type-based analysis, a more robust computational model can be produced. The goal is a computational framework incorporating the advantages of the quality offered in architect-designed homes, while applying the cost-effective home builder production system. The resulting computational framework has a few basic steps: the initial site analysis; the basic massing of the home; the typological selector; and the application of a specific parametrically driven grammar. The resulting framework has the ability to respond to a wide range of sites with a wide range of housing types. Although it does not go as far computationally as the American foursquare analysis did, the resulting floor plan diagrams are quickly extracted from the parametric framework and developed into functioning floor plans. For a further understanding of the fundamental step the framework goes through, it will be described in five criteria: site, roof and massing, type selector, discursive grammars, and floor plan diagrams.

figure 7

Diagram that shows the application of each of the typology-based grammars.

figure 8

Render showing a neighborhood of homes developed from the framework.

figure 9

Diagram that documents the steps and results of the framework.

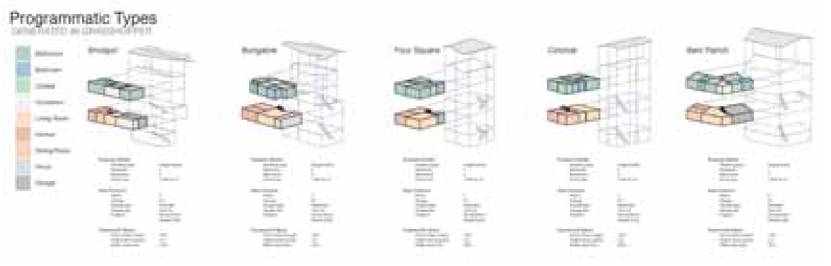

figure 7

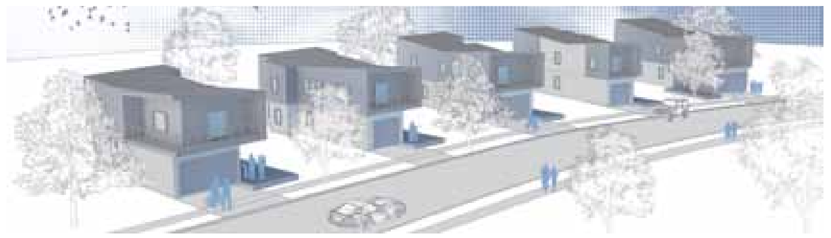

figure 8

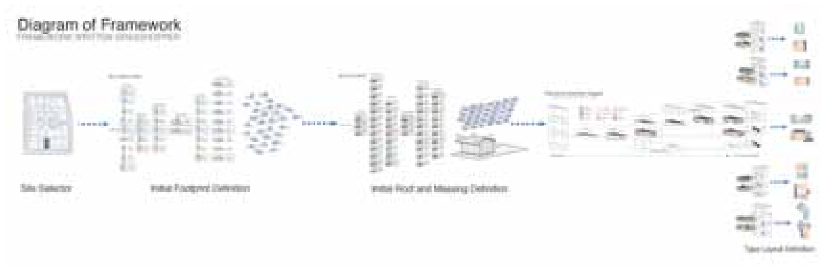
Diagram of Framework
Site Selector
Initial Footprint Definition
Initial Roof and Massing Definition
Type Layout Definition

figure 9

4.1 Site

The framework begins with the input of the site information. This information is used throughout the script to affect the massing and the type. Basic setback and code-driven inputs are applied first, and they are followed by additional user-defined inputs. These parametric controls are designed to constrain the following solutions to what is affordable and appropriate to the site and its context. Their main purpose is to understand the slope and topography of the site, determine the buildable area of the site, and place an initial footprint within that region. At this stage, there is an understanding of the site constraints and building proportions. To determine the footprint, program data and basic house information has already been input. At this stage, the footprint of the house is constrained to a rectangle shape. This both simplifies the initial perimeter and helps control the buildability and cost of the final solution.

4.2 Roof and Massing

The previous steps' analysis and understanding are passes into the roof and massing portion of the script. Several factors are used to understand the sectional dimensions of the mass and the nature of the roof. Although these factors are not as limited, a secondary optimization script can be used to analyze and optimize each option. For example, the roof is allowed to have any orientation, but with the application of solar panels to the roof, you could optimize the roof pitch and orientation, as well as add additional surface to the southern face by relocating the ridgeline further to the north. These moves have a large effect on the character of the house and open up a large solutions space. At this stage of the script, visualization can be made of a fairly generic mass, responding to the site with a more highly articulated roof plane.

4.3 Type Selector

The footprint and the volumetric massing are analyzed in the type selector portion of the script. In this step, the overall proportions of the mass are determined. Five different grammars based on typologies were then developed. These typologies are updated to resemble current housing needs, including a shotgun, a bungalow, a foursquare, a colonial, and a site-adaption ranch. In the next stage one of these types is plugged into the house masses.

4.4 Discursive Grammars

Each of these parametric typologies has a wide variety of controls and customizations. Each grammar was developed in a similar way to the American foursquare home from the earlier study. To reduce the amount of scripting needed, each of these grammars was limited to a predefined program, although when it was possible flexibility was added to each script. Some of these include changing the size of the garage, attaching or detaching the garage from the house, and bending the plan of the house to respond to orientation. All the grammars have the same abilities, from the global reorganization of the floor plan relationships to the individual expansion and contraction of each room in the house.

4.5 Script Output

The grammar information creates an articulated model of the plan elements, and the roof from the initial mass is reapplied. The roof planes are manipulated based on interaction with the plan massing. The final output from the script is an organizational model of the house with a semiresolved form and roof. This scripted process could continue to add detail using the same methods as the American foursquare. The final output is, however, easily extracted from the framework and manipulated by hand to quickly output schematic designs for several homes of different typologies. As a tool, this framework allows for iteration of several different typologies of houses or iterations of the same typology within a single type.

5 CONCLUSION

The value of the computational framework presented here extends beyond its usefulness to architects as an iterative design tool; it allows home builders or home buyers pathways to generate customized, site-specific housing. From the home builders' perspective, they could engage their clients in an entirely new way. Instead of offering them a list of five poorly designed homes, builders could create custom homes specifically tailored to each client and specific site. From the clients' perspective, these homes would be of superior design because the framework would have the ability to consider a greater number of factors with a higher degree of optimization and design specification than home builders can do on their own. The resulting homes are parametrically multi-criterion-optimized for site, roof, massing, solar orientation, and type, with future potential to optimize other criteria of construction via digital fabrication.

REFERENCES

Cache, B. (2003). Towards a Fully Associative Architecture. In Architecture in the Digital Age, ed. B. Kolarevic. New York: Spon Press.

Duarte, J. (2001). PhD dissertation, Massachusetts Institute of Technology. Customizing Mass Housing: A Discursive Grammar for Siza's Malagueria Houses. Accessed August 23, 2011. http://cumincad.scix.net/cgi-bin/works/Show?diss_duarte.

Gindroz, R., D. Csont, and K. Levine. (2004). The Architectural Pattern Book. New York: W. W. Norton,

Hachem, C., A. Athienitis, and P. Fazio. (2011). Investigation of Solar Potential of Housing Units in Different Neighborhood Designs. Energy and Buildings 43 (9): 2262–73.

Hodge, M. (2009). Automating Practice: Defining Use of Computation in the Architectural Design Workflow. Perkins Will Research Journal 1 (2).

Kampf, J. H., M. Montavon, J. Bunvesc, R. Bolliger, and D. Robinson. (2010). Optimisation of Buildings' Solar Irradiation Availability. Solar Energy 84: 596–603.

Koning, H., and J. Eizenberg. (1981). The Language of the Prairie: Frank Lloyd Wright's Prairie Houses. Environment and Planning B 8: 295–323.

Miller, N. (2009). Parametric Strategies in Civic Architecture Design. In ACADIA 09: reForm(): Building a Better Tomorrow, eds. T. Sterk and R. Loveridge. Proceedings of the 29th Annual Conference of the Association for Computer Aided Design in Architecture, Chicago.

Stiny, G., and Gips, J. (1972). Shape Grammars and the Generative Specification of Painting and Sculpture. Information Processing 71: 1460–65.

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

Yessios, C. (2003). The Construction Industry in an Age of Anxiety. In Architecture in the Digital Age, ed. B. Kolarevic. New York: Spon Press.

EDIBLE INFRASTRUCTURES: EMERGENT ORGANIZATIONAL PATTERNS FOR THE PRODUCTIVE CITY

ABSTRACT

Edible Infrastructures is an investigation into a projective mode of urbanism that considers food as an integral part of a city's metabolic infrastructure. Working with algorithms as design tools, we explore the generative potential of such a system to create an urban ecology that provides for its residents via local, multiscalar, distributed food production, reconnects urbanites with their food sources, and decouples food costs from fossil fuels by limiting transportation at all levels, from source to table.

The research is conducted through the building up of a sequence of algorithms, beginning with the "settlement simulation," which couples consumers to productive surface area within a cellular-automata-type computational model. Topological analysis informs generative operations, as each stage builds on the output of the last. In this way we explore the hierarchical components for a new Productive City, including the structure and programming of the urban circulatory network, an emergent urban morphology based around productive urban blocks, and opportunities for new architectural typologies. The resulting prototypical Productive City questions the underlying mechanisms that shape modern urban space and demonstrates the architectural potential of mathematical modeling and simulation to address complex urban spatial and programmatic challenges.

Darrick Borowski
Architectural Association
Emergent Technologies and Design

Nikoletta Poulimeni
Architectural Association,
Emergent Technologies and Design

Jeroen Janssen
Architectural Association, Emergent Technologies and Design