Raising the *i*-factor: bridging parametric shape and parametric design

Ramesh Krishnamurti
Carnegie Mellon University, USA

Using the artifice of the *i*-device as an analogue, this paper examines issues in making design software more accessible through projects on computer-aided sustainable design, panelization for design and fabrication, and parametric shape grammar interpretation. In each case accessibility is improved by transitioning design from a representational concern to one that is more process oriented through the use of localized semantics.

The *i*-factor

I research software for computer-aided design. In this connection, whenever I talk to my students about design software I often use the *i*-device as the modern metaphor for inquiry, ideation, instruction, interaction, interchange, interface, and not least, information. It is the harbinger of change and quite possibly the symbol of present-day instant gratification. The *i*-device appears to allow people to work seamlessly and simultaneously with multiple technologies towards a single collective task. Prior to its advent, each individual technology was largely employed for specialized concerns; whereas now, seemingly disparate technologies work well together with what I term high *i-factor*. There are, of course, technologies that have and may always have a low *i*-factor; and others about which we remain unsure. Design research has many that fall into this last category.

Computer-aided design software has wide appeal. Much of this appeal has to do with the fact that most modeling software deal with geometry and shape manipulation, with some offering an added ability to create design information models. Design for use, fabrication, or construction requires renderings, performance analyses in a multitude of domains, for

example, lighting, energy, acoustics, fluid flow, materials and sustainability; or surface deconstruction for fabrication. They require software with specialized (that is, less) appeal. Combining the two has cost considerations—representation cost, interchange cost, process cost, integration cost and so on. Moreover, the costs would be disproportionately skewed towards higher costs designs. That is, performance software currently has a low *i*-factor if one considers a larger client (or designer) base. To work seamlessly together, to integrate multivariate requirements we would have to readdress issues pertaining to design representation, design information and/or design processes. Figure 1 is illustrative of agents and entities in a design process. It is also suggestive of the sorts of problems and models worth exploring, for instance, compositional explorations of geometry; formal and functional explorations via parametric design, design patterns and physics-based simulation; and physical manifestations through element decomposition, fabrication and/or assembly.

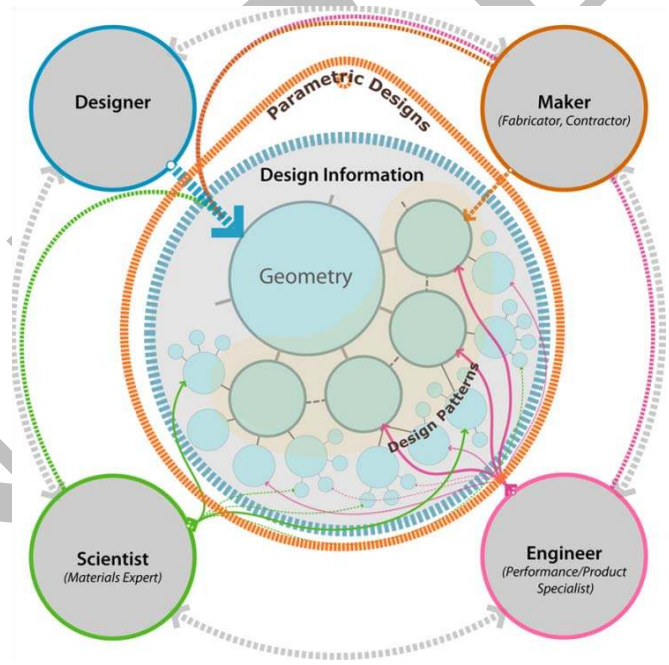

Fig1. Agents and entities in a design process

For the past few years I have been engaged in two distinct types of design research, one with a practical bent but requiring more *theoretical*

consideration; the other based much more solidly in theory but calling out for *real* practical application. Both types of research provide fresh insights. Interestingly, one arrives at quite similar conclusions to quite distinct problems. And, not least, judging by the results/products from earlier research in these areas, these appear to have low *i*-factor. The question is: *what does it take to raise the i-factor*?

There are a number of ways in which to explore this issue. One via cognition is on how users perceive and manipulate information and tools that are already provided by design software and in finding ways of making such information accessible, consequently affording new and perhaps novel uses. Note here that I am not questioning the nature and structure of information; these are taken as given, instead the focus is positioned on how the representation of different sorts of information allow for meaningful combinations that will assist the designer's intentions. As a simple recent example, interfaces now visually and dynamically allow users to interact with and change sun paths, rapidly calculating consequences on shades and shadows, thereby enabling users to make much informed, versatile on-the-fly design decisions. Sun path technology is established; newer models of interactions afford interesting and/or novel sun-tempered applications. Commercial design software is increasingly adopting this route towards redesigning the way known information is envisioned and manipulated.

Another, but equally important consideration is at the other extreme in looking at the underlying technology itself and seeing ways in which improved information access and use can be facilitated. This approach requires an investigation of alternative ways information can be structured, while taking into account any constraints in current design software and technologies. For instance, in digital fabrication there is a growing demand on how information that is not currently provided in the design software can be incorporated and accessed by the designer. This need does not involve simply finding ways to represent information but also questioning the current design tools and suggesting novel ways to structure information forming a rationalized process.

The idea underlying much of my work is simple, namely, if different sorts of information were made easily accessible to designers, they might then use them to solve a range of other problems that their computational design technology was not originally intended for. That is, raising the *i*-factor. In each case, it necessitates a design model and (designer friendly) programmable computational environment. In the sequel I highlight some of the design software projects that my students and I have worked on and suggest in each case possible ways of raising the *i*-factor.

Parametric sustainable building design

In a typical building design, evaluating performances of a design information model-based solution requires a heavy dose of human intervention and data interpretation, rendering analyses to be both costly and time consuming. Evaluating for sustainability according to a green building design standard not only requires different types of analyses but also requires evaluation of different aspects of the building. For this we used Revit® —with its advanced parametric and building modeling capabilities—as the design information model in an integrated database [1]. We used its environment to develop a prototype (Figure 2), which is focused on how users interact with design and evaluation with respect to a green building rating standard.

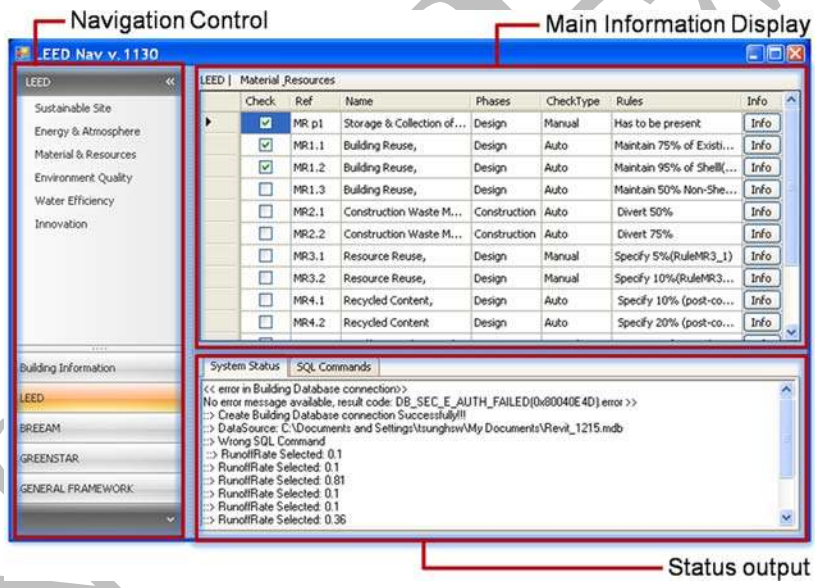

Fig2. Prototype to evaluate buildings with a green building rating standard

The purpose of the project is to provide real-time feedback on the status of a building project with respect to sustainability according to a chosen green building rating standard. Information required for evaluating the sustainability of the project is gathered from the building information model, external databases and simulation results. The prototype goes through three steps during an evaluation: i) check pre-requisite credits; ii) supply additional simulation results; and iii) evaluate credit.

An evaluation starts once pre-requisite credits have been checked. The prototype retrieves information to evaluate credits for the currently selected standard, which for the example in Figure 2 is LEED [2]. Any added data is then supplied, for example, simulation results. On aggregating the required data from the sources, namely, the building information model, simulation results and rating standard, results are updated to a data table.

To validate the implementation we chose an existing LEED Silver-certified building to run an evaluation based on a Revit model and real simulation results. We tried to maximize the possible credits achievable by automating the process. Figure 3 shows results for the LEED Material Resources category. The prototype can generate a graphical representation of the evaluation results in three different formats: as a table, an image, and as an html file. These results provide users with the current status of the project.

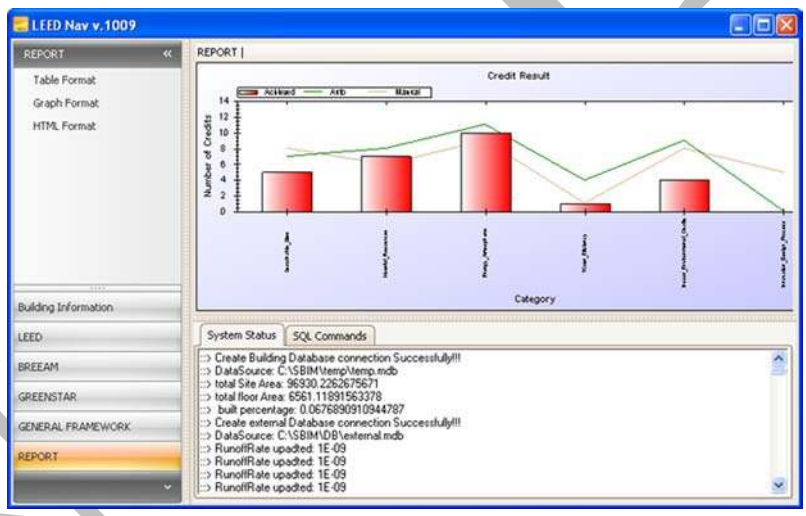

Fig3. LEED material and resources credit evaluation and report generation

On face value, all this appears straightforward: a design information model seemingly linked seamlessly to performance evaluation applications and a green standard. *So what's missing? Or rather, what does it take to accomplish this?* The answer lies quite literally in filling the gaps.

Filling the gaps for sustainable design

A design information model acts as a data container to hold design and other relevant information. However, currently, these models contain less than sufficient data to meet most rating system requirements. Some require data external to the design information model; these have to be accommodated in a cohesive manner [3]. The data might be electronically and geographically distributed; they may need to be salvaged; and they may even need to be certified. See Figure 4.

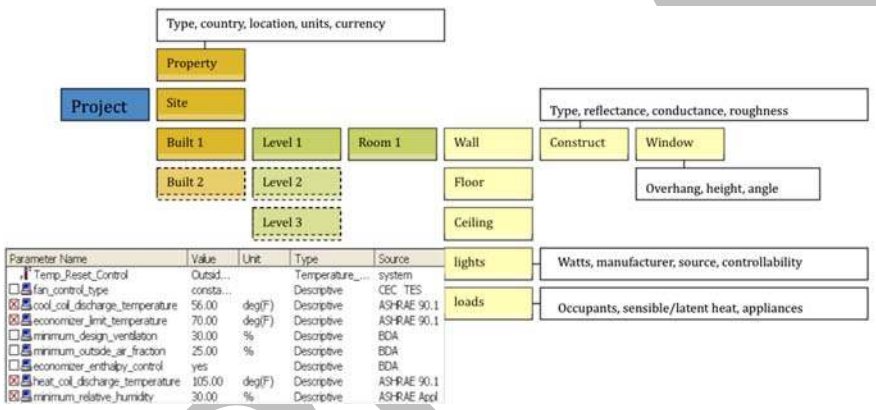

Fig4. Design elements for energy analysis not typically found in any design model

Our approach was to develop a framework for mapping elements and measures to evaluate each credit of a green building rating system to elements and measures in the design information model [4]. The framework also identifies information not accessible within the design model. We also developed a visualization tool to exemplify how the available information can be used to guide and create a design from a sustainability viewpoint. The functionalities of the visualization tool, shown in Figure 6, allow us to look at a chosen design, which has been evaluated by the prototype application, and show the nature of information used in evaluating chosen credits. In the window the top-level categories of the chosen rating system is shown on the left, here LEED; the next level down shows credits belonging to the category. Credits are color-coded to reflect the level of achievement when evaluated by the Revit application: red for a fully achieved credit, white when it has not, and pink when it almost has.

If missing design elements, supporting databases and simulation results are available, credits are clearly achievable. This tool focuses on required

design elements for evaluation rather than mirroring results from the prototype application. It can be used in two ways: as an extension to the prototype, or to invoke Revit directly for the current project. That is, the gap analysis tool can be used to template automated sustainability validation.

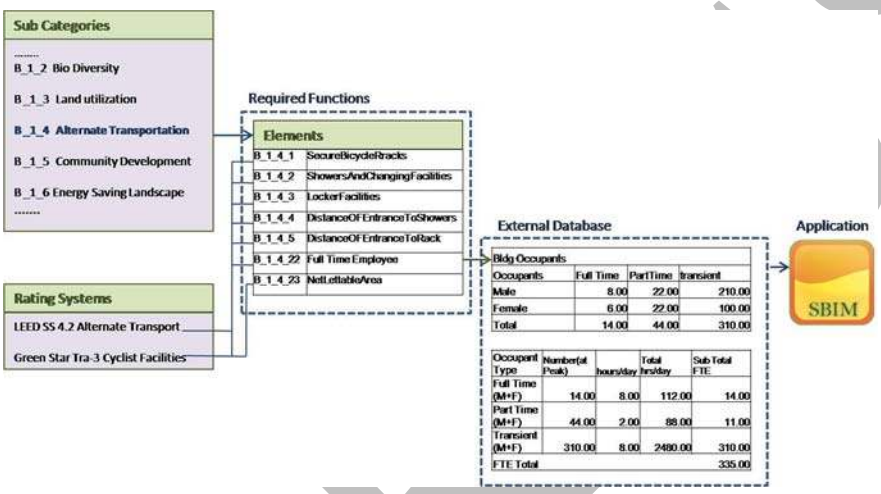

Fig5. Snapshot of mapping rating system requirements to elements in the model

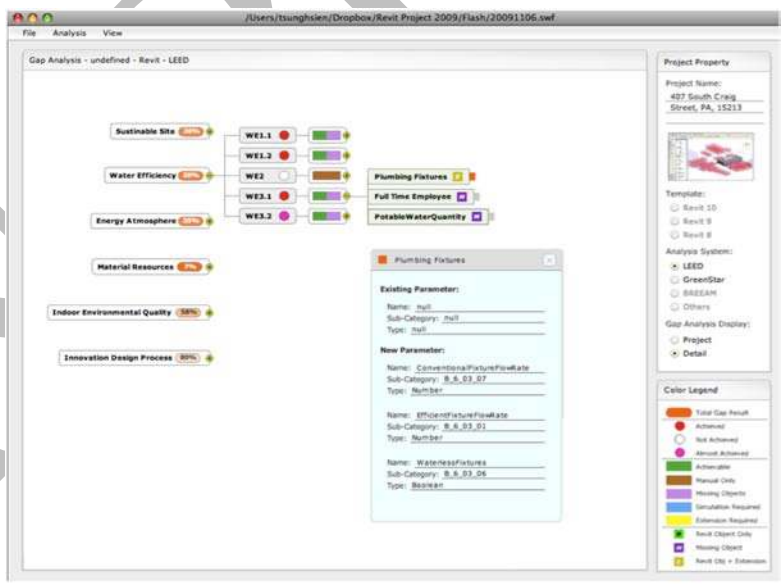

Fig6. Gap visualization of LEED WE3.1 illustrating extensions

In the spirit in which this paper is set out, the obvious next step is to ask how much further can one push the boundaries of integration. The next project does just that by exploring this single building design performance evaluation application at the urban scale.

Parametric sustainable urban water use

Calculating water use is straightforward. However, it can be problematical because of missing data when integrating sustainability requirements with a particular design model. Pertinent data for water use calculations include occupant numbers, fixture flow rates, fixture costs and materials. Designer usually supplies these. Other required data outwith the design information model include rainfall, plant water use, etc. Such data do not normally to fall within the designer's purview, yet these are all factors that have to be accounted for. For this project we focused on modeling sustainable water use on an urban scale. The introduction of scale leads us to consider two approaches aimed at fulfilling LEED requirements for water use reduction.

Approach 1. The first approach is based on the hypothesis that if water use can be calculated and evaluated for a single building it can be extended to multiple buildings and thus, to a larger scale. For this we modeled a single commercial building in Revit using our prototype application to retrieve relevant information, namely, building heights and occupants.

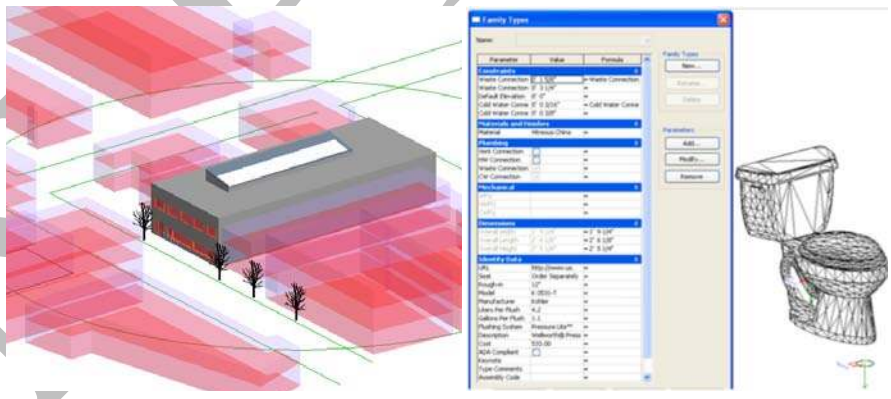

Fig7. (*Left*) A test case for modeling water; (*right*) fixture information

Calculations mainly follow the LEED method for water use, which is found by estimating occupant usage and fixture flow rates [5]. Occupant use corresponds to full time equivalent (FTE) occupancy of a building,

based on a standard 8-hour occupancy period, resulting in a value based on an eighth of the hours per day. For transient building populations such as students, visitors, or customers, hours are estimated on a representative daily average. Water use reduction for a building then corresponds to the difference between a design case and the baseline case [5].

A *design* case is obtained by totaling the annual volume of water use per fixture type and then, subtracting rainwater or gray water reuse. Actual flow rates and flush volumes for the installed fixtures are used in the calculation. For consistency, a balanced one-to-one ratio of male and female is assumed. To create the *baseline* case, the design case information is used to provide the number of male and female occupants, with fixture flush and flow rate values adjusted as per EPAct default specifications [5].

Water fixtures components are stored in the Revit library and can be queried for dimensions. Other element parameters such as fixture flow rate, fixture cost, have to be filled unless automatically generated from manufacturers specifications.

To calculate water efficiency, we implemented external databases for fixtures and landscapes. Additional materials and element parameters include material porosity and fixture flow rates. Fixture costs from manufacturers are used for comparison in water use and ultimate cost savings. In the prototype there are two tabs for water efficiency. See Figure 8. These contain the necessary tasks for evaluating water efficiency credits.

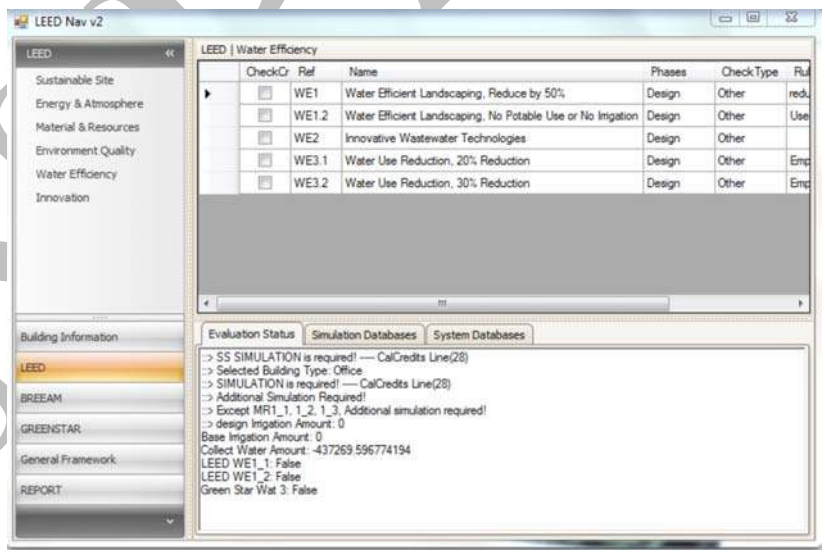

Fig8. Water efficiency tabs for rating credit and other water related calculations

The overall workflow for water use retrieves information about the numbers of male and female occupants, which are specified in the Revit model. Differences between the baseline and design cases are then compared to determine the number of credits earned at this stage.

This approach can be extended to accumulate information on multiple buildings and aggregate total water use. This approach works only when all pertinent information is available.

Approach 2. The second method uses a two-dimensional drawing to generate a mass model augmented with water related information. In modeling water use on an urban scale, where we have information on building area and height, we employ a combination of different software. As before, only fixture flow rates and occupant use are considered in water use calculations ignoring such quantities as gray water quantity and rainwater harvesting. With many buildings in an urban area, assigning occupants to each building is difficult; in this case, we employ Green Star's method, based on net usable space, for assigning occupants [6].

Fig9. CAD drawing of the test urban area

The sample case study covers an area nearly 17350m^2. Of this about 11700m^2 covers the building footprint; the remainders are roads, pavements and parking areas, which are assumed to have an impervious ground cover. There are open spaces with potential for planning for rainwater catchments and water management [7]. The mass model (Figure 10) is generated from a CAD drawing (Figure 9), from which total floor

areas of buildings can be calculated. We used Rhinoceros® [8] with Grasshopper™ [9] for the mass model.

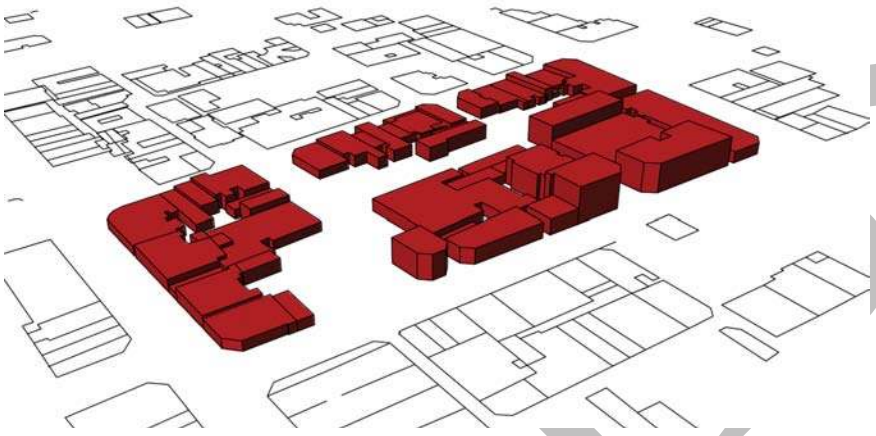

Fig10. Mass model of the urban area

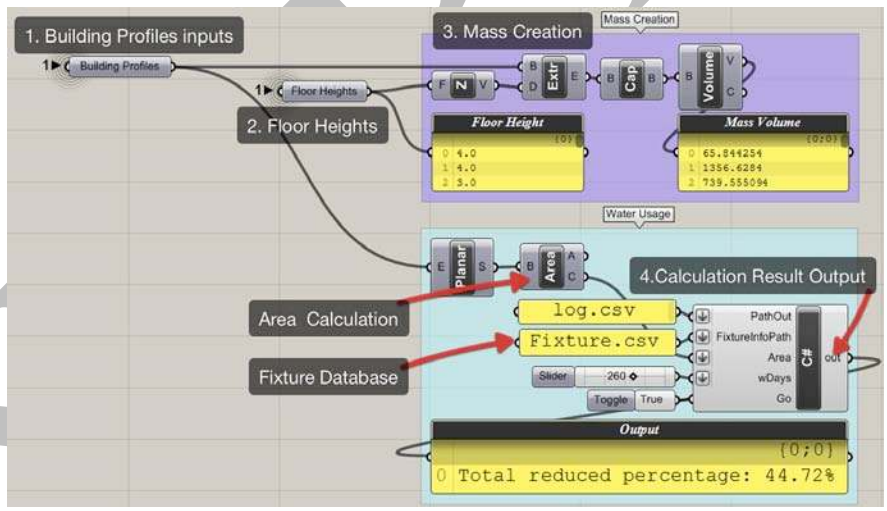

Fig11. Grasshopper definition file for urban water use model generation

This approach offers a way to easily create a parametric model with facility to calculate quantities and specify parameters for water use calculation on a larger scale with greater flexibility than Revit currently affords. As both Rhinoceros and Revit are built on top of the .NET framework [10], communicating between software is straightforward. The

Grasshopper definitions in Figure 11 shows the connection between model geometry and the fixtures database to generate an urban water use model.

Remarks

Both approaches are parametric allowing variations in fixture number and type, ratio of male to female occupants, and in allocating different design cases to different parts of the urban area in order to model various water use scenarios. The combination of two commercial software, external databases, and sustainability requirements illustrate how information can be gathered and processed on a larger scale. Despite differences in data structures and manipulations, it is possible to raise the *i*-factor by leveraging a programming environment to both pre-certify a single building for sustainability and on a larger scale, to project effects on environmental resources.

Parametric surface tessellation

The next project is inspired by a growing trend in contemporary architectural practice of exploiting freeform surfaces to design and model intricate geometries for projects which otherwise would be impossible to realize. In doing so, designers have liberally borrowed digital fabrication techniques developed in the automobile and aerospace industries [11-13].

To manifest a freeform surface, a discrete mesh model is utilized. Transforming a freeform surface into an appropriate mesh is computationally intensive; generally, it is not an easy for designers with no formal geometry training. To design, model, and, subsequently, fabricate intriguing, sometimes intricate, freeform shapes, we look at surface tessellation as an extension of meshing with the added consideration of incorporating constructible building components. There are close relationships and analogies between the elements of a mesh and the components of a freeform design.

Figure 12 shows two designs by Norman Foster and Partners: the Elephant House canopy and City Hall in London. Both demonstrate the value of the parametric approach to architectural applications. In the canopy design, the base geometry is a torus constructed by revolving a circle about an axis. The revolution gives the surface a good discretizing feature—namely, the planar quadrilateral patch [13] that can be derived directly from the principal curvature lines. This makes fabrication manageable, even for such type of doubly curved dome-like surfaces.

Fig12. Elephant House Canopy [15]; London City Hall [16]

The City Hall project demonstrates how parametric schemes used on a conical surface development can be realized for flat panel fabrications. The initial idea for the City Hall was a pebble-like form, which was later approximated by a collection of partial conical strips, which as a member of the family of cylindrical surfaces can be easily decomposed into planar quadrilateral faces.

The two designs clearly highlight the advantages of employing constructive geometry principles in the whole process from design to fabrication. The constructive geometry principle captures the underlying form of the target surface and in turn makes feasible its ultimate manifestation.

Interwoven patterns

In the first instance of surface tessellations I consider interwoven patterns. In principle given a quad(rilateral)-mesh we can construct patterns (panels) by trim operations and spatial transformations.

Consider Figure 13. The top row illustrates in top view the steps for creating a trimmed pattern. The middle row shows the corresponding surface manipulation in three-dimensional space. The bottom row illustrates transformation rules—in this case, rotation about a central axis and mirror operation via the planar quad boundary—being applied to generate the second half of this module.

Figure 14 shows the resulting interwoven pattern ED_03. The pattern is inspired by the Erwin Hauer's continuous screen, *Design 03* (circa 1952), for the Church in Leising, Vienna, Austria [17]. On the left is shown the interweave module consisting of two parts—an upper and lower module component colored in shades of green and blue respectively. The thickness

of the component is derived from an offset operation in a direction normal to the base quadrilateral face.

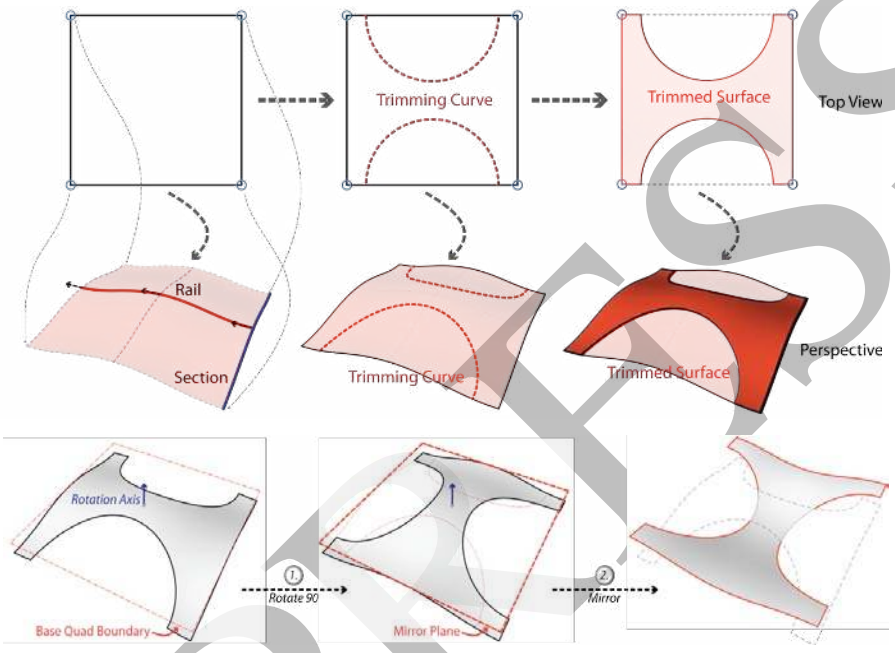

Fig13. Creating a trimmed surface for pattern ED_03

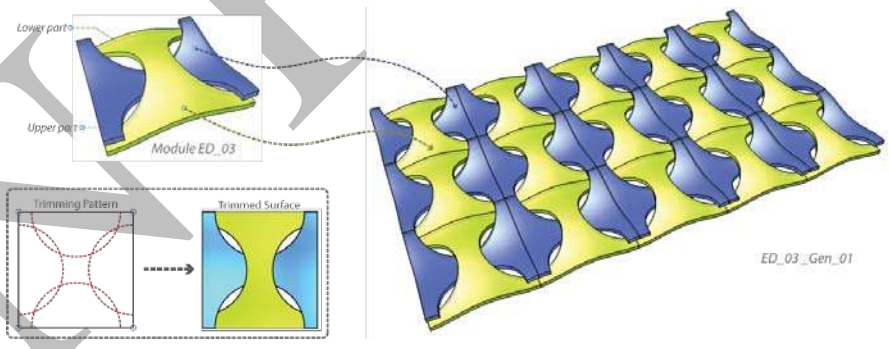

Fig14. The interweave pattern ED_03

Figure 15 illustrates two other Hauer inspired patterns ED_04 and ED_05 again based on simple trim and spatial transformation rules. Notice that the two patterns employ the same trim patterns—two ellipses

intersecting orthogonally, they generate distinct results by variation of the corresponding location to the base boundary.

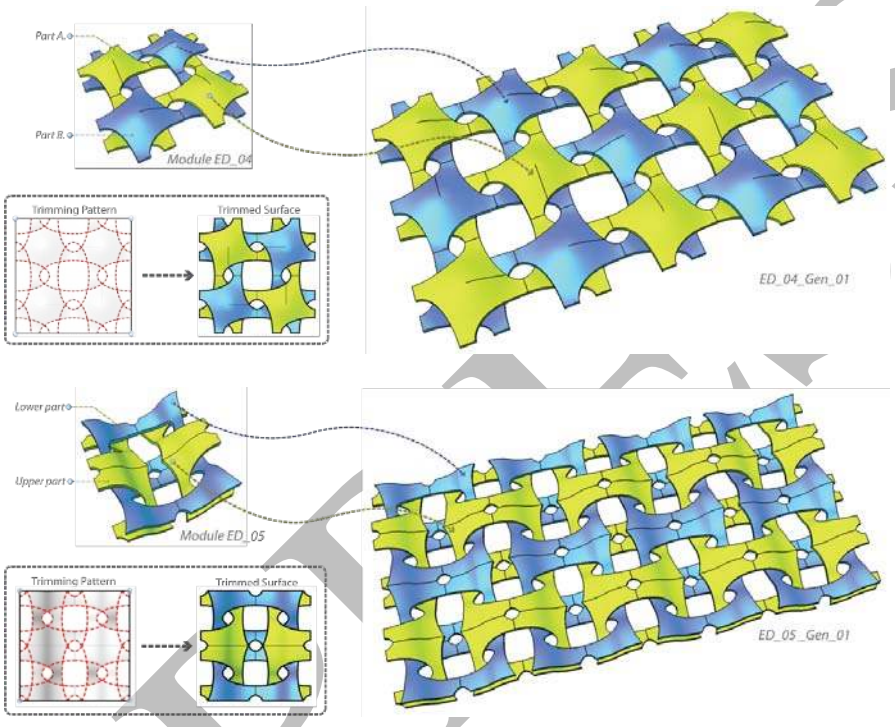

Fig15. Interweave patterns ED_04 and ED_05

The next logical step would be to consider trim rules that ensure that the basic module is self-continuous. Instead of treating a module with two separate parts, joined only at external edges with adjacent modules, this type of pattern joins parts internally. This characteristic creates a more intricate continuous movement from local module to entire modular propagation. Figure 16 illustrates such a trimming operation applied on a target surface with customized trim curves at four corners, which can be exploited to create a self-interlocking pattern.

Interwoven patterns show the application of procedure-based (or rule-based) approaches to design exploration. These patterns generalize to a parametric framework, identifying generative rules that can further manipulated, thus providing strategies for designers to develop their own parametric modeling toolkits.

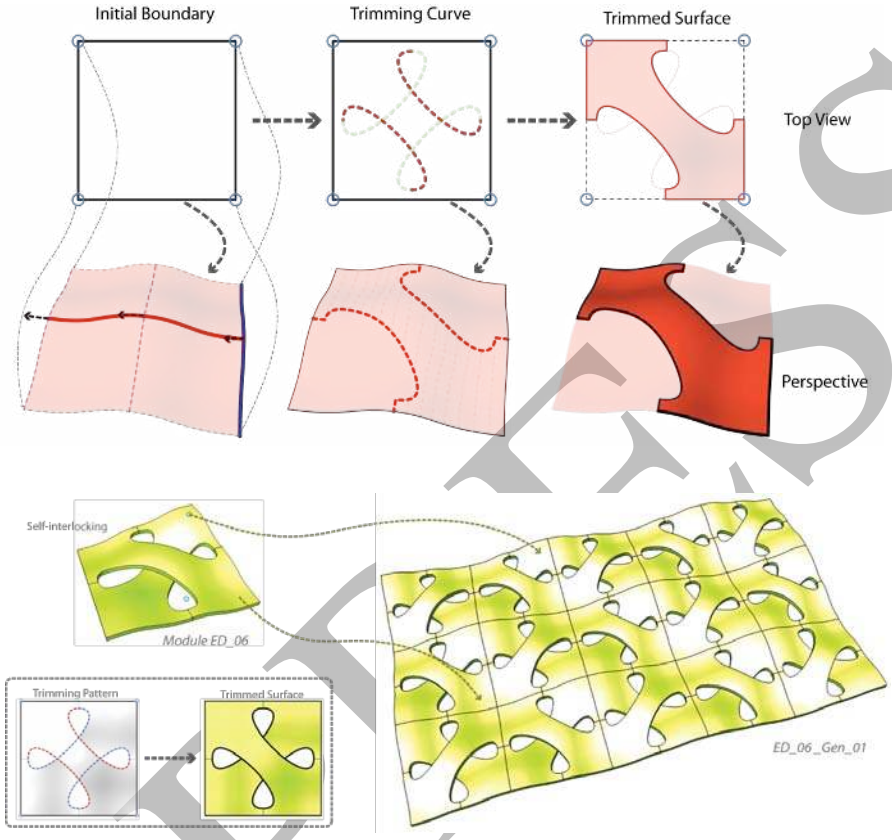

Fig16. Constructing the self-interlocking interwoven pattern ED_06

Surfaces with irregular boundary conditions

Real freeform designs tend not to be regular. Rather surfaces have openings introduced to meet specific design intentions, for example, lighting, viewing or circulation, etc.

Figure 17 shows the west elevation of a surface trimmed for skylight, natural view and entrance. Problems immediately occur when new boundaries are introduced to an originally untrimmed surface. The trimmed edges, for example, cut through the uniform shapes of certain quad dominant surface panels. Irregular panels surrounding the trim edges are generated and these, in turn, affect the overall aesthetic appearance of the surface manifestation as well as final fabrication.

Fig17. Trimmed surface boundaries inspired by a Zaha Hadid design [19]

To address such issues we need to explore how the global boundary conditions, which primarily determine the final freeform surface, can be used to affect or tune the surface tessellation process and be directed towards a more balanced solution. For example, directions of panelizing could be modified (or better instructed) to avoid, or reduce, the irregular panels as the boundary conditions evolve. This is ongoing work. Boundaries define the ultimate appearance; given that panelization can take all boundaries into account parametrically and algorithmically, our contention is that a coherent surface tessellation can always be achieved, and amenable to the application of design patterns [18].

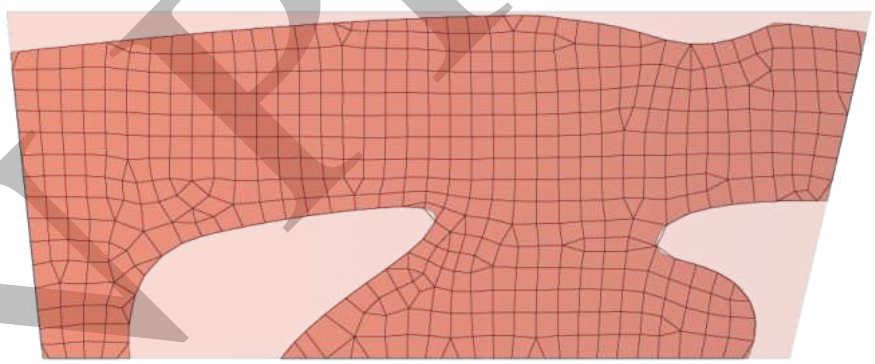

Fig 18. A bubble meshed [20] quad-dominant tessellation

Remarks

Firstly, it should be noted that there is distinction to be drawn between this and the first project. Parametric sustainable design relies on an established technology, namely a CAD system, which by its very nature is *i*-amenable. On the other hand, parametric surface tessellation has its application in

digital fabrication, which is less established, its techniques and methods are, at best, ad hoc [11, 12, 21]. Raising the *i*-factor is tantamount to providing designers with easier ways to find alternative fabrication strategies. At the very least, this is a two-stage development. From an information processing perspective, freeform surface tessellation has to become a rationalized process; only then can we consider ways of leveraging information inherent in the process.

Parametric shape grammars

The last project deals with shape grammars, a largely theoretical construct geared towards analyses and idea formation [22]. Shapes are created by the application of shape rules, each made up of a left and right part. Under rule application, the left part of the rule 'found' in the shape under some transformation (including parameter assignment) is replaced by the right part of the rule under the same transformation to produce a new shape. Shape grammars have been widely applied for analyzing designs [23].

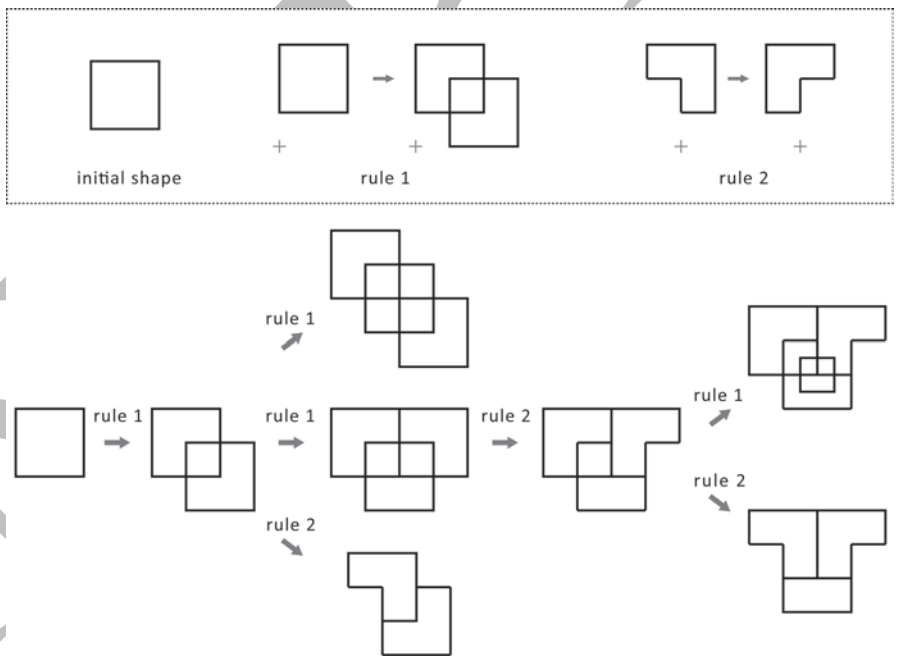

Fig 19. An example of a shape grammar

Although the basic formalism of a shape grammar has remained largely the same over time, there have been changes both in definition and development. Factors that have influenced these changes relate to the scope of permissible shape elements and possible augmentations. The first formal definition, *SG1971,* was given in the seminal article by Stiny and Gips [24] More definitions have since appeared in the literature, each reflecting either the understanding at a particular time, or reflecting a specific research flavor. Definitions fall into two stages: marker-driven and subshape-driven. Definitions *SG1971*, *SG1974*, *SG1975* and *SG1977* are marker-driven [24-27]. Definitions *SG1980, SG1991, SG1992* and *SG2006* are subshape driven [22, 29, 29, 23]. Chronologically, the definitions are backwards compatible. That is *SG1971* << [*SG1974*, *SG1975*] << *SG1977* << *SG1980* << *SG1991* << *SG1992* << *SG2006*, where the right side of << is more general than the left side.

Markers play a pivotal role in determining the shape rule applicability and their corresponding transformation. Markers can ensure that rule application can be directly computed. By comparison, there are harder computational issues involved with subshape-driven grammars, particularly, parametric subshape recognition and indeterminacy.

The evolutionary development exhibits a trend from 'rigid' to 'soft'. 'Rigid' here means that the shape grammars were defined to be closer to phrase structure grammars [30]. Such shape grammars are more machine-bound in the sense that they are easier to compute, but harder to use to create novel designs. A recent series of notable shape grammar implementations fall within the rigid category of shape grammars [31-34].

On the other hand, the 'softer' development is more human-centered, showing more concern and consideration on how to use shape grammars to generate novel designs. This explains, in part, the importance of subshape-driven grammars, concepts of indeterminacy and shape emergence, and the support for ambiguity in shape grammar research. Humans have little trouble handling such concepts. Moreover, human designers actually benefit from them. However, these concepts are problematical when considering computer implementation, especially when directed at parametric shapes defined by open terms [35], though there have been notable attempts [36]. Computer implementation is vital to both research and application. The central difficulty lies in parametric subshape recognition.

In this regard, a parametric shape may have an indeterminate number, k, of open terms, that is, points with no fixed coordinates, matched against n possible points. Then, choices are combinatorial; as k approaches $n/2$, time complexity for subshape recognition is a super-polynomial.

Shape grammars fall into two distinct categories [37]. The first handles special shapes; the second is more general with potentially super-polynomial time complexity, which is only practical for shapes of smaller sizes. The implication for practice is that the best we can achieve is implementing shape grammar interpreters, each capable of handling a subset. This leads to a paradigm for practical, 'general' parametric shape grammar interpreters. See Figure 20.

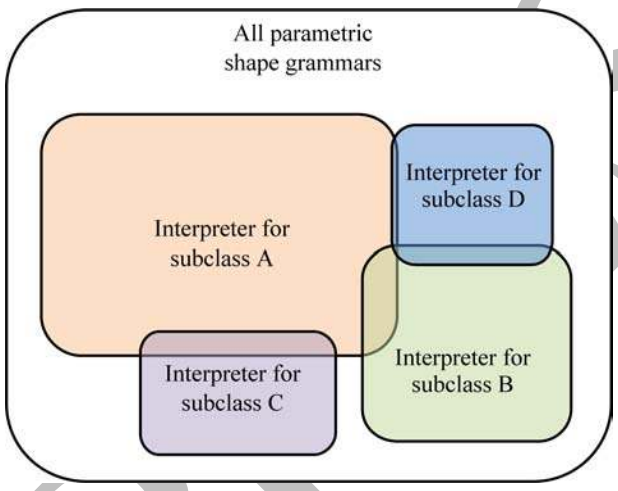

Fig20. A paradigm for practical "general" parametric interpreters

We make the assumption that interpreters for shape grammars belonging to different subclasses will collectively cover most parametric shape grammars. The classification can be considered "better" when the number of subclasses is smaller, and when, simultaneously and collectively, the scope covered is larger. Possible ways of classifying shape grammars needs further research.

We consider categories of shape grammars whose implementation is tractable. Shape grammars, which capture certain building styles, generally fall into this category. These are normally parametric shape grammars, in which rule application does not depend on emergent shapes. Markers drive rule application, and configurations are rectangular or approximated as such. Parameterization is typically limited to the height, width or room ratios. Shape rules typically relate to adding a room, subdividing a room, or refinements such as adding windows, doors, etc.

Apart from such internal characteristics, there are other factors that influence computational tractability, that is, in how shape grammars are

designed and described. Normally, a shape grammar is designed to simply and succinctly describe an underlying building style, with little consideration on how it can be implemented. As a result, in order to translate this into programming code, shape rules have to be quantitatively specified with sufficient precision to disallow the generation of ill-dimensioned configurations.

In practice there may be several ways to describing a particular shape rule; it is possible that one way is easier to compute, and another, computationally intractable. As a result, it is desirable to design an application program interface as the framework to support the design of shape grammars; then, such shape grammars are guaranteed to be computationally tractable. Such a framework requires an underlying data structure, and basic manipulation algorithms. Moreover, for ease of code translation, a meta-language built on top of the basic manipulation algorithms should be developed. As grammars in different classes typically have differing underlying structures, the appropriate underlying data structure for the framework is different. Ideally, the interpreter for a subclass of shape grammars can be supported on a single framework. Consequently, the framework for parametric shape grammars comprises a series of sub-frameworks, one for each subclass of shape grammars. Overall, as the framework is capable of ensuring tractability, we term such shape grammars as computation-friendly.

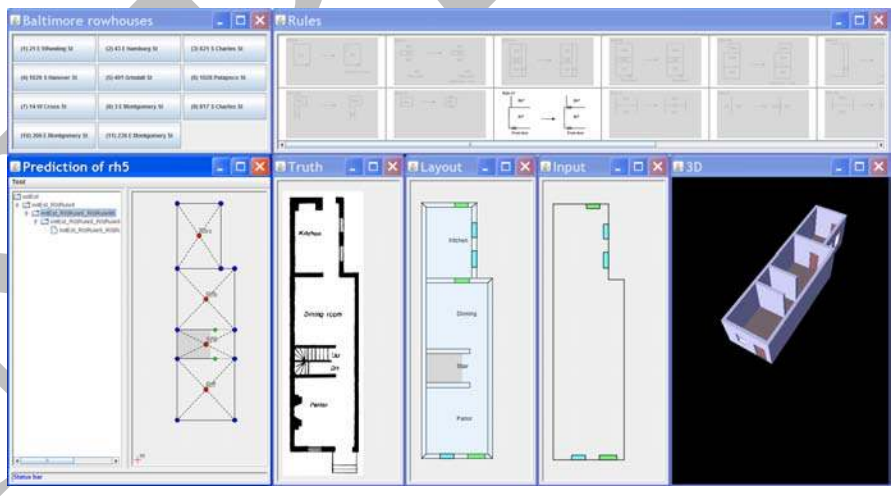

Fig 21. Screenshot of layout determination of the Baltimore Rowhouse

We have developed three sub-frameworks each specifying a way of implementing a subclass of shape grammars: rectangular, polygonal and graph [37]. Figure 21 illustrates a screenshot of a shape grammar interpreter based on the rectangular framework. The interface shown is an adaptation of shape grammar application in another context [38].

Remarks

Implementing parametric, indeed, non-parameterized shape grammars fall into an altogether distinct category from the first two kinds of projects. Here, theory and algorithms are well established, however, practical demonstrations have proven to be exceedingly hard. Although there are implementations of grammars [39], most cannot claim to be anything beyond a toy. Part of the difficulty involves the technicalities of shape rule application, of harnessing the power of emergent shapes to the designer's advantage. Resolving such issues will impact the *i*-factor. The approach suggested is one of classifying shape grammars in terms of local data structures, each with its own set of manipulation tools. In this way we can recast shape rule application to a limited set of functions.

Discussion

The work presented is motivated in part by wanting to extend the capabilities of the design software beyond its original intent, that is, to raise its *i*-factor. We have attempted to do so not by trying to be novel. Instead, in each project we have shied away from any global representation to more localized semantics by empowering the process. That is, the information in any rule is of greater significance than the rule (i.e., its structure and nature) itself. In the case of computer-aided sustainable design, the framework for sustainability is merely a pool from which specific elements and measures can be drawn to specify a specific relationship between a rating standard and a design model. This relationship constitutes a process of evaluation. As rating systems or design models evolve new relationships similarly evolve. Moreover, the processes can be extended to the urban scale in an unlikely fashion using software that is not geared to urban scale modeling. Likewise, in the case of surface tessellations, it is local panelization, a rule-based process, that ultimately specifies the overall design and fabrication. In the case of parametric shape grammars simplification through localized data structures, manipulations and rules in the form of 'code' is the first step to making grammars

practically accessible to users. In short, part way to raising the *i*-factor is to transition design from a 'modeling' to a 'programming' exercise through local data structures and local manipulations. The next and perhaps more difficult step is to make the transition seamless.

Acknowledgement

The work reported here would be impossible without the contributions of my graduate students Tajin Biswas, Tsung-hsien Wang, Peng-hui Wan, Kui Yue and Varvara Toulkeridou.

References

1. Autodesk. (2003) Building Information Modeling for Sustainable Design. www.federalnewsradio.com/pdfs/BuildingInformationModelingforSubstainableDesign-white%20paper.pdf (accessed May 10, 2010)
2. http://www.usgbc.org/LEED (accessed May 10, 2010)
3. Krygiel E, Nies B (2008) Green BIM Successful Sustainable Design with Building Information Modeling. Wiley Publishing, Indiana.
4. Biswas T, Krishnamurti R (2009) Framework for supporting sustainable design. Innovations for Building and Construction, 373-386, Europia, Paris
5. USGBC (2006) New Construction and Renovation Guide V2.2, Washington.
6. www.gbca.org.au/green-star/rating-tools/green-star-office-design-v3-green-star-office-as-built-v3/1710.htm (accessed May 12, 2010)
7. Echols S (2007) Artful Rainwater Design in the Urban Landscape. J of Green Building, 2(4):103-122.
8. www.rhino3d.com/ (accessed May 12, 2010)
9. www.grasshopper3d.com/ (accessed May 12, 2010)
10. www.microsoft.com/NET/ (accessed May 12, 2010)
11. Kolarevic B (2005) Architecture in the Digital Age; Design and Manufacturing. Taylor & Francis.
12. Kolarevic B, Klinger K (2008) Manufacturing Material Effects: Rethinking Design and Making in Architecture. Routledge.
13. Pottmann H, Asperl A, Hofer M, Kilian A, Bentley D (2007) Architectural Geometry. Bentley Institute Press.
14. Pottmann H, Liu Y, Wallner J, Bobenko A, Wang W (2007) Geometry of multi-layer freeform structures for architecture. ACM Trans Graph 26:1-11.
15. www.arcspace.com/architects/foster/elephant/ (accessed April 5, 2010)
16. www.fosterandpartners.com/Projects/1027/Default.aspx (accessed April 5, 2010)
17. Hauer E (2007) Erwin Hauer: Continua-Architectural Screen and Walls. Princeton Architectural Press.
18. Woodbury R (2010) Elements of Parametric Design. Taylor & Francis.

19. Hadid Z (2008) Schematic Design Report for Next-Gene Architecture Museum. National Chiao-Tung University, Hsinchu, Taiwan
20. Shimada K, Gossard DC (1995) Bubble Mesh: Automated Triangular Meshing of Non-Manifold Geometry by Sphere Packing. ACM Third Symposium on Solid Modeling and Applications, 409-419.
21. Kolarevic B (2005) Performative Architecture: Beyond Instrumentality. Routledge.
22. Stiny G (1980) Introduction to shape and shape grammars. Env & Plan B 7**:**343-351.
23. Stiny G (2006) Shape: Talking about seeing and doing. MIT Press.
24. Stiny G, Gips J (1972) Shape grammars and the generative specification of painting and sculpture. The Best Computer Papers of 1971. 125-35, Auerbach, Philadelphia.
25. Gips J (1975) Shape Grammars and Their Uses: Artificial Perception, Shape Generation and Computer Aesthetics. Birkhaüser, Basel.
26. Stiny G (1975) Pictorial and formal aspects of shape and shape grammars and aesthetic systems. Birkhaüser, Basel.
27. Stiny G (1977) Ice-ray: a note on Chinese lattice designs. Env & Plan B 4:89-98.
28. Stiny G (1991) The algebras of design. Res in Eng Design 2:171-181.
29. Stiny G (1992) Weights. Env & Plan B 19: 413-430.
30. Taylor RG (1998) Models of Computation and Formal Languages. Oxford University Press.
31. Müller P, Wonka P, Haegler S, Ulmer A, van Gool L (2006) Procedural modeling of buildings. ACM Trans Graph 25(3): 614-623.
32. Müller P, Zeng G, Wonka P, van Gool L (2007) Image-based procedural modeling of facades. ACM Trans Graph 26(3): article 85, 9 pg.
33. Watson B, Müller P, Veryovka O, Fuller A, Wonka P, Sexton, C (2008) Procedural Urban Modeling in Practice. IEEE Comput Graph Appl 28: 18-26.
34. Weber B, Müller P, Wonka P, GROSS M (2009) Interactive Geometric Simulation of 4D Cities. Computer Graphics Forum 28(2): 481-492.
35. Chau HH, Chen X, McKay A, de Pennington A (2004) Evaluation of a 3D shape grammar implementation. Design Computing and Cognition '04. 357-376, Kluwer Academic, Dordrecht.
36. McCormack JP, Cagan J (2002) Supporting designers' hierarchies through parametric shape recognition. Env & Plan B 29: 913-931
37. Yue K (2009) Computation-friendly shape grammars: with application to the determination of interior layout of buildings from image data. PhD Thesis. School of Architecture, Carnegie Mellon University.
38. Aksamija A, Yue K, Kim H, F Grobler F, Krishnamurti R (2010) Integration of knowledge-based and generative systems for building characterization and prediction. AIEDAM 24: 3–16
39. Shape grammar implementation: from theory to useable software http://www2.mech-eng.leeds.ac.uk/users/men6am/DCC10-SG-Implementation-Workshop.htm (accessed Nov 4, 2010)