Differentiating parametric design: Digital workflows in contemporary architecture and construction

Thomas Wortmann and *Bige Tunçer*, Singapore University of Technology and Design, 8 Somapah Rd, 487372, Singapore

This paper examines Parametric Design (PD) in contemporary architectural practice. It considers three case studies: The Future of Us pavilion, the Louvre Abu Dhabi and the Morpheus Hotel. The case studies illustrate how, compared to non-parametrically and older parametrically designed projects, PD is employed to generate, document and fabricate designs with a greater level of detail and differentiation, often at the level of individual building components. We argue that such differentiation cannot be achieved with conventional Building Information Modelling and without customizing existing software. We compare the case studies' PD approaches (objected-oriented programming, functional programming, visual programming and distributed visual programming) and decomposition, algorithms and data structures as crucial factors for the practical viability of complex parametric models and as key aspects of PD thinking.

Keywords: parametric design, design automation, architectural design, software design, parametric master model

This paper examines Parametric Design (PD) from the perspective of contemporary architectural practice. Parametric design has been applied in architectural practice for more than two decades, but continues to evolve as software and knowhow become available more widely. The paper presents an extended case study of developing a parametric, patterned envelope for tropical climates, which culminates in the realization of a 40-m span grid-shell clad with around 11 000 individual, perforated panels: The Future of Us (FoU) pavilion in Singapore by the Advanced Architecture Laboratory (AAL). It compares this case with two others, described more briefly: The Louvre Abu Dhabi (LAD) by Ateliers Jean Nouvel in the United Arab Emirates and the Morpheus Hotel by Zaha Hadid Architects in Macau, Special Administrative Region of the People's Republic of China.

Corresponding author:
T. Wortmann.
thomas_wortmann@mymail.sutd.edu.sg

The case studies illustrate how, compared to non-parametrically and older parametrically designed projects, PD is employed to generate, document and fabricate designs with a greater level of detail and differentiation, often at the level of individual building components. Such differentiation allows a wider spectrum of architectural form, richer architectural experiences and the integration of environmental, structural and buildability concerns. From this perspective, the case studies exemplify the role of PD in "supporting complexity" (Oxman, 2006).

Based on the case studies, we argue that such differentiation cannot be achieved with conventional Building Information Modelling (BIM) and without, to different degrees, customizing existing software through textual computer-programming. The paper examines the cases' digital workflows and identifies and compares four PD approaches: objected-oriented programming, functional programming, visual programming and distributed visual programming. Objected-oriented programming structures computers programs into objects that are composed of data and operations, while functional programming structures them into nested hierarchies of operations using higher-order functions (Abelson & Sussman, 1996). To increase the understandability and reusability of object-oriented programs, an object should hold only the data and functions that are relevant to it, with interactions between objects kept to a minimum. Compared to functional programming, the resulting constructs tend to be larger and more complex, but it can be advantageous to have the information about an object directly associated with it. Visual programming environments, such as Grasshopper, allow users to create a computer program by dragging-and-dropping predefined operations and connecting them into a directed, acyclic graph (Janssen & Chen, 2011). Distributed visual data flow programming combines several visual programs into a larger network. The paper identifies decomposition, algorithms and data structures as crucial factors for the practical viability of complex parametric models and as key aspects of PD thinking.

All case studies employ PD in conjunction with a master model that is shared between architects, consultants and, in two cases, contractors. In all cases, this master model is generated parametrically and serves as a basis for further parametric design development, automated documentation and numerically-controlled prefabrication. Before presenting the case studies, the paper reviews earlier applications and case studies of PD.

1 Background

The theoretical literature often understands PD in terms of design generation (Oxman, 2006) or exploration (Woodbury, 2010). Hudson (2010) identifies six (overlapping) applications of PD in architectural practice: (1) the translation of design ideas into parametric models, (2) the rationalization of designs into buildable shapes and components, (3) the control and setting-out of architectural forms, (4) the generation and testing of design variants based on various criteria and specialist input, i.e., efficiency-focused design exploration or optimization, (5) the sharing of information, which, for the case studies presented in this paper includes the automated generation of construction documents and (6) the capture of design knowledge from different stakeholders.

Whitehead (2003) describes early parametric models used by the architectural practice Foster and Partners to explore, control and rationalize building forms. The models are defined with computer-programming and based on geometric primitives such as cones and torus patches. Whitehead observes that "most designers already think programmatically, but having neither the time nor the inclination to learn programming skills, do not have the means to express or explore these patterns of thought" (Whitehead, 2003).

Hesselgren, Charitou, and Dritsas (2007) describe the PD of the (unrealized) 288-m tall Bishopsgate Tower in the City of London. They model the tower's form in an early version of Generative Components, a visual programming environment. A custom server programmed in C#, an objected-oriented programming language, stores design parameters and data. They validate parametric models with custom scripts that, using output data, recreate them in another 3D-modelling environment, Rhinoceros. They pay special attention to the configuration of the facade components, which are identical except for their opening angles. In summary, they use PD for formal exploration, the extraction of building data and the generation of design variants for testing (including structural and environmental analysis). Hesselgren et al. characterize PD as "a new mode of thinking" that allows designers "to impose implicit and explicit constraints in both sub-domains but also the entirety of the design space" (Hesselgren et al., 2007).

Park and Holt (2010) employ a computer-programmed, software-independent master model to control the shape and panelization of the Lotte Super Tower in Seoul. This master model anchors a "loosely organized network of scripts" used by the architects and structural consultants to develop the tower's structure and facade construction. Park and Holt apply PD for geometric control, rationalization and knowledge capture. For them, PD thinking is the abstraction of "a specific task into its operation and inputs" (Park & Holt, 2010).

Shepherd, Hudson, and Hines (2011) describe the PD process of the Aviva Stadium, Dublin, "the first stadium to be designed from start to finish using commercially available parametric modelling software", namely Generative Components. A parametric master model controls the stadium's envelope, which is further developed in separate parametric models for the stadium's structure and cladding. Both models serve for generate-and-test processes; the former to optimize the structure and the latter to adapt the rotating louvers of the cladding to wind and rain. The louvers have differentiated lengths and orientations, but identical profiles and connection details. The cladding model captures design knowledge from facade design specialists and the cladding contractor (Hudson, 2010).

2 Case studies

The selection of the case studies follows five criteria: (1) The projects are either built or under construction. (2) The projects are contemporary in that they either got completed recently or are under construction. The FoU pavilion opened at the end of 2015. The LAD is scheduled to open in 2017 and the Morpheus Hotel in 2018. (3) The projects use differentiated, i.e. non-repeated building elements, which in all cases are facade components. (4) PD was used to create, manage, document and prefabricate this differentiation. (5) Documentation of the projects' design processes is available; in the case of the FoU, from the first author's personal work and for the remaining two projects as online video lectures.

2.1 The Future of Us

The parametric model of the FoU grid-shell in Singapore was first developed for two smaller design projects, a speculative pavilion and a prototypical wall-screen. This section details the model's objects in terms of inputs, outputs, and the algorithms that connect them (see Figure 1).

The model's inputs are (1) a triangular mesh representing the form and panelization of the design, (2) pattern tiles defining the pattern for the design's envelope and (3) density requirements for the envelope represented as a point cloud. These inputs are integrated by labelling the triangular mesh. This labelled mesh encapsulates different kinds of data, such as the components' corner points, envelope density requirements (i.e. its opening percentage), curvature, and—in the case of the grid shell—structural member dimensions and orientations. These data stem from various participants in the design process and include simulation results as well as subjective design considerations. Based on these inputs, the model interprets every mesh face as a differentiated building component. The parametric model expresses the components as four output geometries: 2D for graphical visualization (1), solid for 3D printing (2), foldable resulting in cut sheets for sheet metal components (3) and foldable resulting in cut sheets and assembly drawings for sheet metal panels mounted on a substructure (4).

The inputs and outputs are implemented as classes in IronPython using RhinoScript, a scripting language for Rhinoceros. The classes define objects for, for example, the pattern tiles, the labelled mesh, and the four types of building components. This model allowed a broad range of highly differentiated outputs: a digital model that provides visual feedback, a 3D-printed scale model of a pavilion with a patterned envelope, two room-size, prototypical wall screens of varying density assembled from folded aluminium sheet components and a large grid shell, covered on top and bottom with around 11 000 unique, patterned aluminium panels. The decomposition of the inputs allowed the design teams to simultaneously work on the architectural appearance of

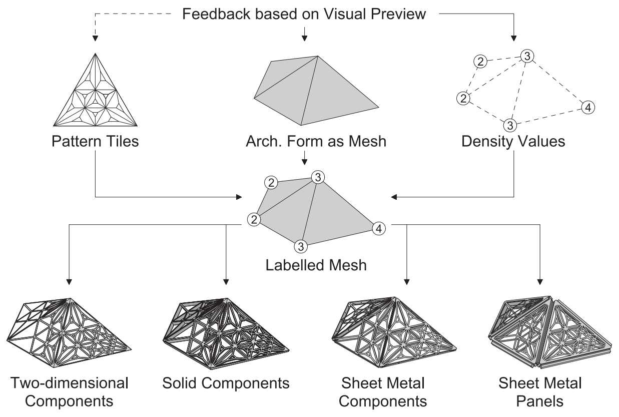


Figure 1 Overview of the parametric model of the Future of Us grid-shell.

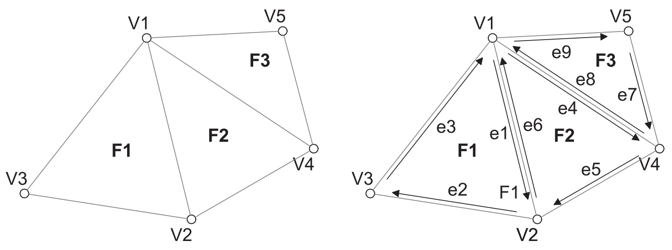
the designs, their performance and fabrication, while the organization into classes afforded rapid changes to the parametric model (such as integrating additional input data or output geometries).

2.1.1 Half-edge mesh representation

The first input to the model is a triangular mesh representing a design's form and panelization. A common representation in computer graphics, a mesh is an interconnected collection of vertices (points) that are connected by edges to form faces (polygons), in this case triangles. Every mesh face serves as the base geometry of one building component. To represent this mesh efficiently, we employ a half-edge-based data structure, which is the only polygonal mesh representation that supports queries such as finding the faces incident to an edge or the edges and faces incident to a vertex without additional computations (Botsch, Kobbelt, Pauly, Alliez, & Levy, 2010). Another advantage of the half-edge data structure is that one can associate its different elements (faces, half-edges, and vertices) with various kinds data. The parametric model exploits this feature to store and retrieve various inputs in a quick and geometrically organized manner.

A less efficient alternative is the face-based data structure that underlies common file formats for the exchange of three-dimensional information (e.g. STL and OBJ) and is available in scripting and visual programming interfaces. Figure 2 provides a visual comparison between face-based and half-edge-based data structures. Rhinoceros provides only face-based meshes which the model converts into a half-edge data structure in two straightforward steps:

(1) First, one traverses the sides of every face (according to the arrays of indices in the face-based face array), creating one half-edge per side and assigning previous and next half-edges according to the traversal of each face. The traversal and assignment ensure that the half-edges for each face form a closed ring, and that corresponding, i.e. opposite, half-



VERTEX LIST

	X	Y	Z
V1	10.0	8.7	6.0
V2	5.0	0.0	0.0
V3	0.0	8.7	0.0
V4	15.0	0.0	0.0
V5	20.0	12.0	0.0

FACE LIST

	Vertices		
F1	V1	V2	V3
F2	V1	V4	V2
F3	V1	V5	V4

VERTEX LIST

	X	Y	Z
V1	10.0	8.7	6.0
V2	5.0	0.0	0.0
V3	0.0	8.7	0.0
V4	15.0	0.0	0.0
V5	20.0	12.0	0.0

FACE LIST

	Start
F1	e1
F2	e4
F3	e7

HALF-EDGE LIST

	Start	End	Next	Prev	Twin
e1	V1	V2	e3	e2	e6
e2	V2	V3	e1	e3	-
e3	V3	V1	e2	e1	-
e4	V1	V4	e6	e5	e8
e5	V4	V2	e4	e6	-
e6	V2	V1	e5	e4	e1
e7	V5	V4	e9	e8	-
e8	V4	V1	e7	e9	e4
e9	V1	V5	e8	e7	-

Figure 2 Comparison between the face-based data structure on the left and the half-edge data structure on the right. Both data structures represent an example mesh with three triangles. Note that some of the half-edges are lacking a twin. Half-edges without a twin indicate "naked" half-edge on the boundaries of a mesh, which is one of the convenient features of this data structure.

edges are pointing in opposite directions. Simultaneously, one creates a new face array by storing the first half-edge of each face. The number of operations for this step increases linearly with the number of vertices, or O(*n*), since it requires one traversal per face.

(2) The second step pairs half-edges with their twins. Finding these pairs with brute force by comparing each half-edge against every other half-edges takes quadratic time, or O(n^2). Sorting the half-edges according to the Cartesian coordinates of their mid points reduces the required number of operations to linearithmic time, or O(*n log n*). Due to the sorting, twinned half-edges adjoin in the half-edge array, and thus can be referenced to each other in a single pass while the three sorts of the Cartesian coordinates require linearithmic time.

The above algorithm converts a face-based representation of a triangle mesh provided by a 3D-modelling environment into a half-edge data structure in linearithmic time, which in practice does not take more than a few seconds. The resulting half-edge data structure requires constant time O(*1*) to process topological queries about neighbouring elements and to compute vertex normals, edge normals and face normals.

2.1.2 Pattern tiles and envelope density requirements

The FoU uses a set of twenty parametrically-defined pattern tiles that achieve a gradient of densities. This gradient is defined by each triangle's corner points

(Figure 3). The pattern makes the regularity of individual panels and, in the case of the grid-shell, the underlying, triangulated structure less apparent visually. The pattern tiles are implemented as a set of nested functions that exploit the repetitions in the pattern. Given three corner points and three corresponding density values, these functions construct a set of closed polylines from the corner points, as defined by the appropriate pattern tile.

Modulating the pattern's density can address shading and programmatic requirements and create a differentiated play of light and shadow. A point cloud of density values defines these requirements, with each point corresponding to a mesh vertex. This point cloud can be created with various methods, for example through generating random patterns, performance simulations, or—in the case of programmatic requirements—direct assignment. Importantly, these density values are editable manually, allowing local adjustments by the designers. To transfer these values from the point cloud to the mesh vertices in linearithmic time, one sorts the density values according to their Cartesian coordinates.

2.1.3 Digital and 3D-printed model

The parametric model automatically generates geometry from two sources of information: a triangular mesh representing both the envelope's shape and panelization and the user-editable point cloud. For the project for which the model was developed originally—a speculative design for a parametric, digitally fabricated pavilion in Singapore—these density values were assigned according two three considerations: Structural deflection (1), solar irradiance (2), and program and views (3). The designers derived the values for the structural and solar considerations from numerical simulations and manually painted a coloured mesh to define the desired density for the programmatic consideration. The structural consideration defined a lower bound for the envelope density, while the average between the solar and programmatic considerations defined the desired envelop density.

Aided by the parametric model, which rapidly interpreted the point cloud and mesh vertices as wireframe and surface geometries, the designers created a pavilion with a patterned envelope of varying density. This design was 3D-printed as a 1:50 scale model. Due to the object-oriented separation of pattern, pattern density and expression of envelope components, it was easy to extend the parametric model to produce solid, 3D-printable envelope components by extruding the base mesh along its vertex normals.

2.1.4 Prototypical wall screens

The designers further developed the concept of a patterned, gradated envelope in two room-size installations in Venice and Singapore (Figure 4). For these installations, the parametric model produced cut sheets for foldable

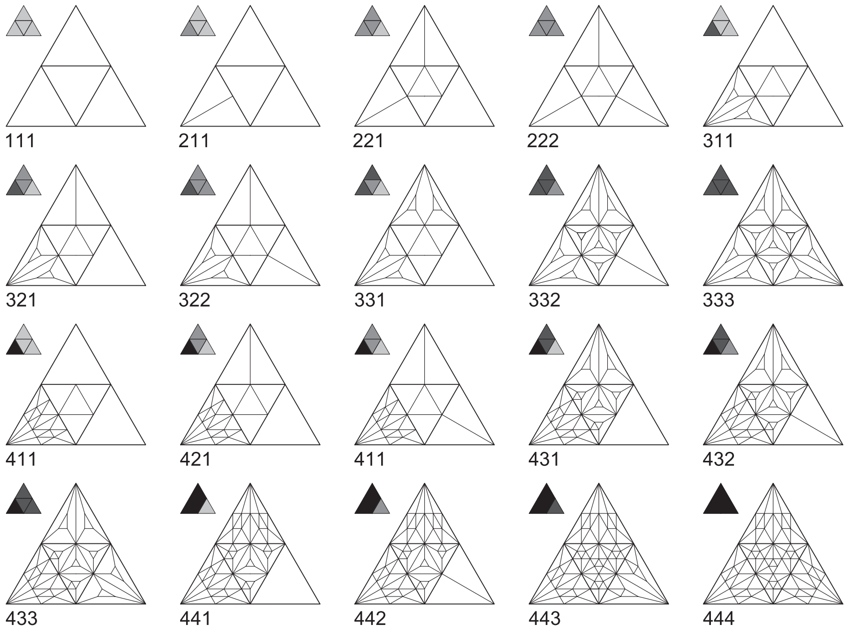


Figure 3 The twenty pattern tiles for the envelope with associated density values of their corner points.

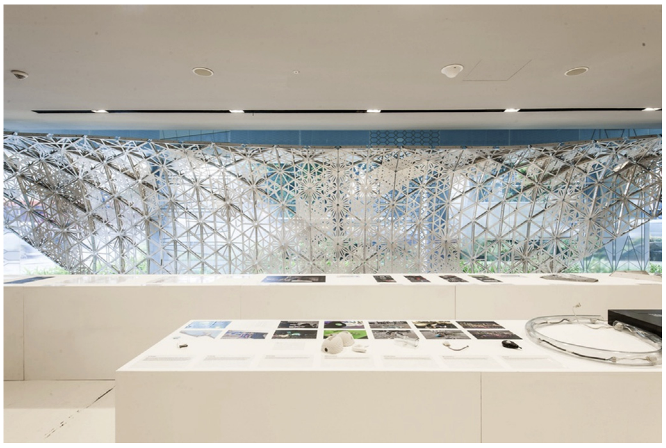


Figure 4 Prototypical wall screen installed at the National Design Centre in Singapore. Picture Credit: AAL

components that can be assembled easily into a double-curved wall. Each self-supporting, integrated component consists of two folded pieces of sheet aluminium, that when combined form a strong, triangular box.

Extruding the base mesh along its vertex normal, which was done for 3D-printing, leads to components with twisted sides. To achieve non-twisting sides for the folded aluminium components, the parametric model extruded them along their edge normals instead. This change was easy to achieve in terms of the model, but led to slight geometric inconsistencies at the corner points of the modules. These inconsistencies are solved by filleting the module corners.

The aluminium pieces were cut on an industrial laser cutter and folded by hand. The automatically generated cut sheets took material thicknesses, folding radii, and tolerances into account. It was unnecessary to fold the pieces to specific angles, because each triangular component could only be assembled in exactly one way and no measurements were necessary to assemble the wall screens.

2.1.5 *Patterned grid shell cladding*

A more practical application was the realization of a free-form, 40 meter-span grid shell that was the architectural centrepiece of an important national exhibition for Singapore's 50-year jubilee (Figure 5). The architectural brief asked for a connection between four prefabricated, standardized domes that housed the exhibits. Since these completely closed, air-conditioned domes were sufficient to protect the exhibits, the design team opted for a "tropical" envelope that would provide shade and natural ventilation, but only partial rain protection. Sitting between the two major domes and defining the main entrance, the free-form had three major openings and a tall central vault covering a third dome completely.

Clad with a gradated pattern on top and bottom, this grid shell was a further development of the wall screens discussed in the previous section. More stringent structural requirements and a short time span of eight months between the start of design development and the official opening, however, necessitated an approach of cladding a steel structure with non-structural panels, versus the self-supporting, integrated box components described in the previous section. Nevertheless, designing, digitally fabricating, and coordinating the assembly of the panels was by no means straightforward and lead to the development of new capabilities for the parametric model.

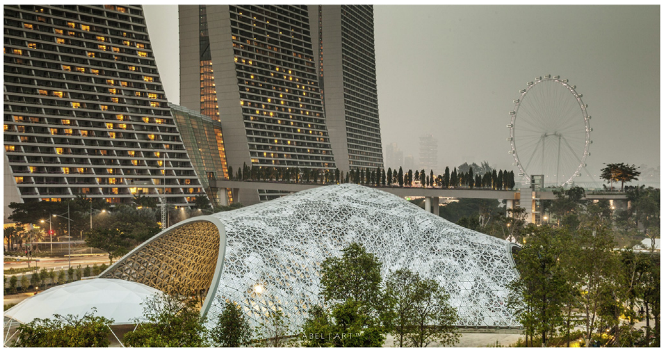


Figure 5 Overall view of the Future of Us pavilion. The dome on the left is still under construction. Picture Credit: Protag, Abel Art

2.1.6 Integrating the parametric model with a structural master model

A major difference with previous iterations was that the panels had to adapt to the substructure provided by the grid shell, which was drawn by the structural consultants. To facilitate coordination of the panels with the substructure, the structural consultants provided a master model with beam centrelines, types, and orientations indicated as lines perpendicular to the centrelines (Figure 6). The main arches of the grid shell are either vertical or rotate around a fixed axis, while the primary and secondary purlins align perpendicularly to the free-form envelope. To define a clear interface between structure and panels early in the process, the structural and design consultants agreed that the panels' supports on the beams would be located 300 milometers from each end of a centreline.

This abstract 3D-model of the grid shell's structure is integrated with the parametric model by matching the orientation lines to the beam centrelines, and the beam centrelines to the edges of the half-edge data structure. In this way, the model adapts individual panel to the supports provided by the underlying structure, whose locations depend on the depth of the beams and their rotations.

Since the layout of the substructure directly affects panel sizes and, indirectly, the appearance of the pattern, several iterations of this master model were exchanged between the design and structural consultants. In these iterations, the structural consultants used Grasshopper, a visual dataflow modeler, to generate new variations quickly, which the design consultants edited by hand (Poirriez, Wortmann, Hudson, & Bouzida, 2016). The parametric model supported these exchanges by allowing quick regenerations of the panel geometry.

The pattern of the grid shell was defined in a manner similar to the original pavilion design, with two inputs being solar irradiance and program. The third

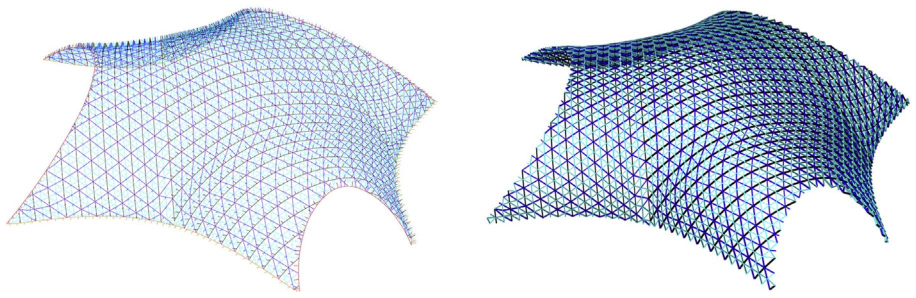


Figure 6 The structural master model on the left indicates centerlines, element types and element orientations, with the resulting 3D geometry on the right. The vertical main arches are colored black, the primary purlins dark blue, and the secondary purlins cyan. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

input, instead of being structural, consisted of a larger scale pattern that ensured visual variety from different distances (Figure 7).

2.1.7 Constructing the grid shell panels

The parametric model defines every panel with an abstract envelope composed of several geometric planes (Figure 8). The sides of the envelope depend on the planes defined by the orientations of the supporting structural elements, the bottom planes on the height of the supporting elements, and the top plane on a global thickness defined for the design. To achieve a homogenous depth and appearance for the complete package of top panel, structural steel and bottom panel, the depths of each panel's sides individually vary depending on the rotational orientations and cross sections of the underlying structural

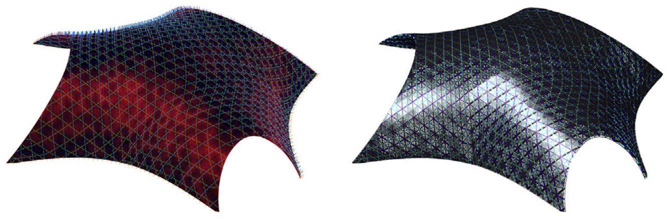


Figure 7 The integrated master model (colored according to the labels representing the required density) is on the left, while the final 3D model with structure and panels generated from the master model is on the right. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

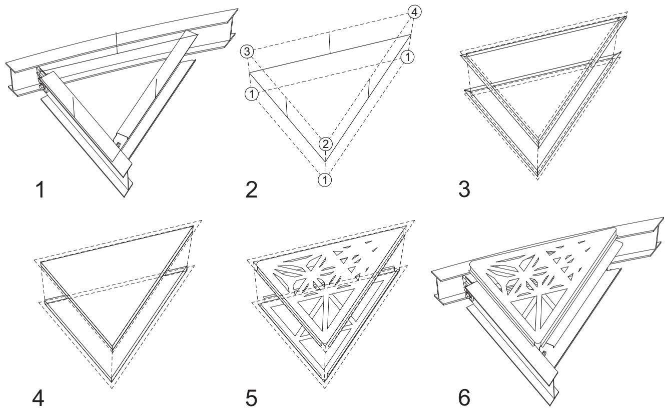


Figure 8 Diagram 1 shows the center lines and orientations for a single triangle with the substructure. Diagram 2 shows the panels' bounding planes and density values, diagram 3 the panel's mounting strips, diagram 4 the panel's flanges, and diagram 5 the completed panel. Diagram 6 shows the panel on the substructure.

elements. The panels' flanges are perpendicular to their tops and their mounting strips are perpendicular to their flanges. Accordingly, the panels could be fabricated with 90° bends only, but their geometry nevertheless adapts to the non-perpendicularities of the underlying structure.

The master model in combination with the parametric model ensured a tight coordination between the structural steel and the panels, which allowed the creation of working drawings for both to proceed in parallel. The parametric model generated not only the panel geometry, but also 1400 cut sheets for digital fabrication of the 11 000 panels and 53 assembly drawings (one per gridline). The coordination and automation provided by the parametric model allowed the realization of the project to specification and in just eight months.

2.2 Louvre Abu Dhabi

The design of the LAD consists of several exhibition buildings (pavilions) situated along an open-air circulation system and surrounded by pools of water. The arrangement of the pavilions evokes a vernacular Arabian urban morphology. A large, circular dome shades the pavilions, open-air circulation and pools. The dome is constructed as a steel space-frame and clad on both sides with 8750 aluminium facade elements, so called "stars", which form an Arabian-style pattern (Figure 9). With a structural depth of 5 m and only four supports (arranged in a square), the dome spans 165 m and is 26 m high (Imbert et al., 2013).

The pattern of the dome's cladding is not water-tight but modulates the sunlight into a "rain of light" that is brighter on the roofs of the exhibition buildings and darker on the circulation areas and pools. In this way, the dome creates an interesting play of light and shadow and a comfortable outdoor climate. The daylight requirements are translated into a perforation ratio, which define the width of individual stars. A map of perforation ratios was developed through several iterations of generating a perforation map with

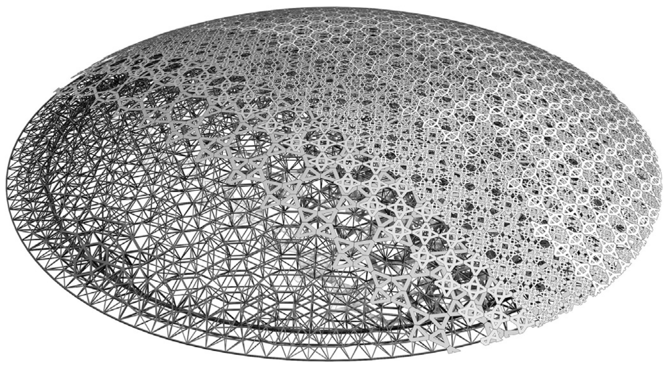


Figure 9 Cut-away diagram of the dome of the Louvre Museum Abu Dhabi indicating the spaceframe and the four layers of stars. Picture Credit: Goswin Rothenthal, Waagner Biro

an inverse lighting simulation and validating the result with forward daylight simulation (Tourre & Miguet, 2010). Unless indicated otherwise, the information in the below sections stems from Rothenthal (2016a).

2.2.1 Parametric model

The cladding of the dome's space-frame consists of eight layers (four top and four bottom). All layers are composed from stars that are arranged on a bidirectional grid. These grids are generated by intersecting a spherical cap with two sets of planes. One set of planes is rotated around the X-axis and the other around the Y-axis. These intersections yield more regular squares around the sphere's pole that gradually distort towards the edge. The grids' size, and thus the size of the stars, decreases with the layers' distance to the space-frame. Accordingly, most of the stars are different from each other. In addition, the grids of the layers are rotated with respect to each other. This approach allows a straightforward parametric definition of the stars' geometry and materialization as building elements, while achieving the desired visual complexity. In other words, unlike the models for the FoU and Morpheus Hotel, the LAD's model does not rely on an underlying geometry, but is strictly mathematical, which simplifies the setting out of the stars. Building on earlier studies developed by the architects and lighting consultants, this parametric model was defined by the dome's main contractor, Waagner Biro.

In contrast to the FoU and the Morpheus Hotel, the cladding adapts only minimally to the underlying structure, whose top and bottom chords form triangular grids. Instead, the pattern of the stars is intersected with the triangular grids to define the locations of supporting pins for the cladding. The underlying space-frame was defined as a parametric wireframe model that, early in the design process, was shared online between architect, engineer, and consultants (Imbert et al., 2013) and reconstructed by the contractor. Defining the connections between the layered stars and the stars and the space-frame was challenging. The quick regeneration of the model helped to overcome this challenge.

Geometrically, a single star consists of four triangles based on the sides of a single quadrilateral, with four stars resulting in an octagon in their centre. Each centre line of these shapes is materialized with several aluminium profiles, which are snapped together. In this way, the width of the stars can be varied for each centreline per the desired perforation ratio. This variation in width, together with the transformations of the grids, modulates the daylight and results in a specific geometry for every star. In short, the stars have differentiated sizes and densities but share the same parametric model.

As in the FoU, the cladding's geometry is stored in a half-edge data structure. This structure naturally lends itself to define the stars' geometry via offsets of the various faces (triangles, quadrilaterals and octagons) and allows the direct association of aluminium profiles with half-edges. The half-edges thus serve as a numbering system for the 459 260 aluminium profiles.

This parametric model can express the stars in at least five different ways: (1) As a "light" wireframe geometry of centrelines, (2) as a two-dimensional geometry where every centreline is expressed as two quadrilaterals, (3) a two-dimensional wireframe where the quadrilaterals are further subdivided into individual profiles, (4) a three-dimensional geometry of the quadrilaterals and (5) a three-dimensional geometry of the profiles. While (1) and (2) take a few seconds to generate and serve to "debug" the parametric model, (3) and (5) serve to visualize the resulting cladding. (3) can be generated in about 30 s and contains enough geometric information to define the CNC cut patterns for the aluminium profiles, while (5) can be generated in about 5 min and represents details such as the actual profiles, drill holes, slots and supporting pins. The parametric model provides construction documentation via a spread sheet that documents the required aluminium profiles and assembly documents for use in the factory and on site.

2.2.2 *Implementation of the parametric model*

The five expressions of the stars are defined as five higher-order functions in the F# programming language. The use of functional programming for generative architectural design has been advocated by Leitão and Proença (2014) due to its "expressive power", i.e. the ease of implementing complex ideas. Here, this ease is illustrated by the five expressions of the stars. The main logic, data structure and sphere geometry was implemented in Visual Studio—an integrated development environment—and Rhinoceros was used only to automatically draw geometry and save CAD files.

To achieve an interactive programming environment, Tsunami, an F# editor that can dynamically execute single lines of codes, was integrated with Rhinoceros via a custom plug-in. Combined with the possibility to quickly generate wireframe models of the cladding, this allowed the development and debugging of the parametric model in an intuitive and efficient manner. The developer of the parametric model explains that, for him, the expressiveness of F# and the ability to "live code" in Tsunami combined a "scripting feeling" with a better computational performance than other approaches to PD (Rothenthal, 2016b):

The sheer scale and complexity of the cladding on the dome required us to re-evaluate our parametric design approach. I developed an F# application to represent and organise all cladding elements of the dome. The switch to F# from dynamic scripting languages [such as Python] helped to reduce development time and execution time. The strongly typed environment, algebraic data types and immutable data [of F#] helped to avoid a whole range of bugs and fits well the domain of generating static 3d geometry.

In summary, for the LAD, PD was adopted to generate, develop and test the light-modulating geometry of the cladding elements and to automatically generate documentation for the digital fabrication and assembly of the cladding. The claddings' parametric model was implemented in a functional programming style using F#, which resulted in a flexible approach to programming and excellent performance.

2.3 Morpheus Hotel

The Morpheus Hotel, designed by Zaha Hadid Architects, is a 39-story hotel and casino currently under construction in Macau. It is 160 m high with a foot print of 52 by 99 m. While its overall shape is a rectangular block, it has a large, free-form opening in its centre, which is crossed by two footbridges (Figure 10). This free-form opening has a single symmetry axis. The free-form area around the central opening is single and double-curved, while the remaining facades are flat except for the single-curved edges of the block. The free-form facade's glazing is flat: Single-curved areas are approximated with skewed quadrilaterals and double-curved areas are triangulated. An exoskeleton with around 2500 steel members supports the facade. In the free-form area, the exoskeleton's structural steel is single-curved and clad with 24 577 aluminium panels, most of which are doubly curved. Unless indicated otherwise, the information in the below sections stems from (Muscettola et al., 2017).

2.3.1 Architects' and structural engineers' parametric models

To coordinate and rationalize the structural steel and its cladding, Zaha Hadid Architects developed a parametric model in Grasshopper, a plug-in of Rhinoceros. The Grasshopper model projects the lines of the exoskeleton onto the

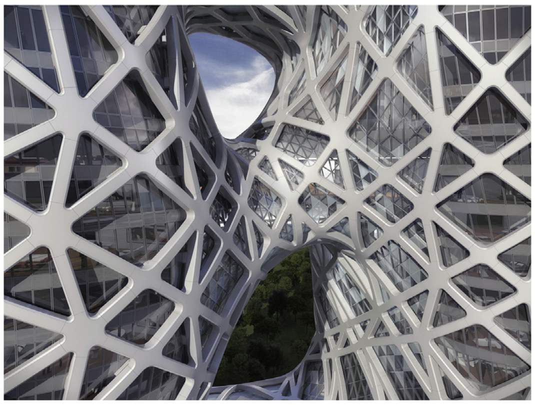


Figure 10 Free-form opening of the Morpheus Hotel. Note the curving exoskeleton and skewed and triangulated glazing. Picture Credit: Zaha Hadid Architects, Melco Crown Entertainment

free-form facade, converts the projected lines into a mesh, converts this mesh into a wireframe and generates the cladding's geometry. The line segments of the wireframe represent the centrelines of the exoskeleton. Several iterations of these centrelines were exchanged between the architects and the structural engineers (Buro Happold) to discover and embody design knowledge (e.g. that the structure can be only single-curved and that the structure's nodes need to align to the floors) and to optimize the structure. It might seem helpful to combine the information from the original mesh and the resulting wireframe into a half-edge data structure, but apparently, this data structure was not used. BIM was used only for standard structural elements such as floors, columns and cores.

The architects' Grasshopper model (Figure 11) generates two representations of the cladding: A non-rationalized representation for visualization and a rationalized one for defining the cladding's geometry for the structural engineers and facade contractor. The non-rationalized cladding follows the orientation of the structural steel, which means that, for the free-form facade, it deviates from the orientation of the underlying glazing. To absorb this deviation, the rationalized cladding's front and back of the are double-curved, following the glazing, while its top and bottom are flat, following the structural steel. This parametric model requires a complex Grasshopper definition (Figure 11). Generating the cladding geometry takes around 5 min for the non-rationalized cladding and 5 h for the rationalized one. (Probably because it is generated from offsets of and intersections with the original design surface of the free-form facade.) Such long generation times limit the interactivity of visual programming, which is one of its main advantages over computer programming.

Buro Happold employed PD to generate, optimize and document the around 2500 connections of the exoskeleton's structural steel (Piermarini, Nuttall, May, & Janssens, 2016). They also had to ensure that the connections fitted

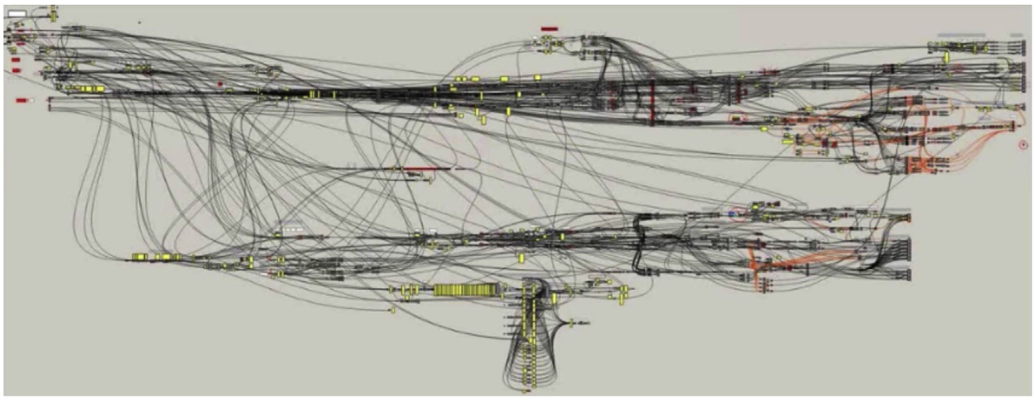


Figure 11 Grasshopper model developed by Zaha Hadid Architects to generate the exoskeleton's rationalized cladding geometry. Picture Credit: Zaha Hadid Architects

inside the envelope of the structural steel's cladding. Grasshopper models generated individual connections (sorted into around 400 types) and were linked to Finite Element software via custom scripts. In this process, Buro Happold used "the software in ways that had not been done before, sometimes working at the limits of the products' capabilities" (ibid.).

2.3.2 *Contractor's parametric models*

Specialist consultants (Front Inc.) hired by the facade contractor also prepared parametric models of the cladding with Grasshopper, albeit for a different purpose and with a different strategy. They used the architects' rationalized geometry (consisting of 24 577 surfaces) as a starting point for the detailed design of the free-form area's cladding, which included the definition of joints and the placement of stiffeners and connectors to the structural steel. Front Inc. automatically generated 350 000 fabrication and 150 000 on- and off-site assembly drawings as well as spreadsheets for construction documentation.

Instead of creating the cladding's geometry and documentation with a single parametric model, Front Inc. adopted a "distributed data model" consisting of a network of hundreds of parametric models and geometry files (van der Heijden, Levelle, & Riese, 2015). Each parametric model performs only a single operation and stores the result of this operation as geometry, which becomes the input for another parametric model (Figure 12). (Some of these models include small textual programs.) In this way, information and more detailed geometry are added as the data flow through the network.

To facilitate this flow, Front Inc. have developed a plug-in (Elefront) that augments geometry with user-defined properties. This plug-in reads, creates and modifies these properties in Grasshopper. These properties turn geometry into custom, BIM-like objects that contain non-geometric data. Front Inc.

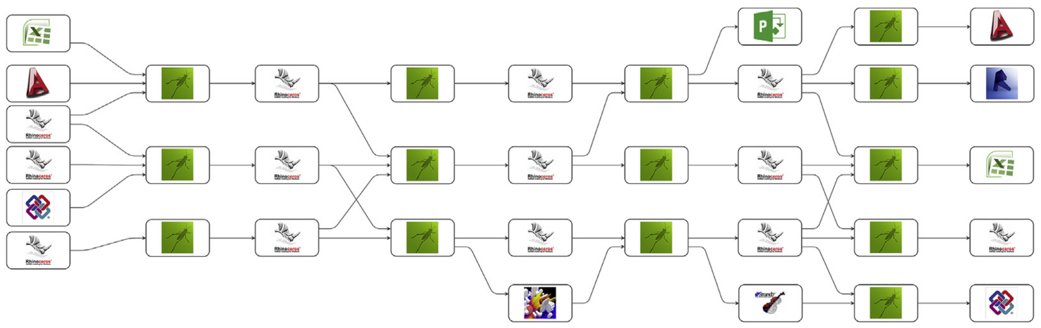


Figure 12 Diagram of the "distributed data model". White boxes indicate data and geometry files and green boxes indicate parametric models. Picture Credit: Ramon van der Heijden, Front Inc. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

term this approach “Building Information Generation” and explain that “when dealing with complex geometry and high degrees of variation [...] traditional BIM-modelling is not a viable option”.

Front Inc. identify four advantages of the distributed data model: Flexibility, scalability, verification and collaboration. (1) Flexibility lies in the ability to easily generate different outputs based, in this case, on the needs of the fabricator. (2) Scalability refers to the ability to deal with large amounts of information. In this case, the panels of the cladding consisted of 1 668 301 unique parts. (3) Verification refers to the ease of checking small, individual geometry files and parametric models—both automatically and manually—compared to checking complex integrated models. (4) Collaboration refers to the ability to work on the distributed data model simultaneously (in this case, with up to five developers). The distributed data model also facilitates collaboration because it requires a smaller skillset than more complex models. This last advantage appears especially relevant in that parametric models typically have only one or two expert authors. (Although, presumably, the design of the distributed data flow model itself still requires an expert.)

In summary, for the Morpheus Hotel, PD was used to generate, evaluate, rationalize and embody design knowledge about its exoskeleton geometry and to automatically generate documentation for the digital fabrication and assembly of the exoskeleton connections and cladding. Two approaches to PD were adopted: Single visual dataflow models for generating and rationalizing the overall geometries for the structural steel and cladding and the structure’s joints, and a distributed data model for the detailed design and documentation of the cladding.

3 Discussion

This section discusses similarities between the case studies in terms of their digital workflows and compares their approaches to PD, using the criteria of flexibility, scalability, verification, collaboration and ease-of-learning.

3.1 Similarities between the case studies

Although the LAD and the Morpheus Hotel are larger and more complex than the FoU, the case studies nevertheless exhibit similarities that indicate the current state-of-the-art in terms of parametric workflows for architecture, engineering and construction (AEC) and offer hints for the future development of AEC software:

- All case studies have a parametrically designed facade composed of differentiated, digitally prefabricated components. For the FoU and the LAD, these elements modulate daylight to create a richer architectural experience and to meet environmental performance requirements.

- These differentiated components also resolve geometric misalignments between structure and facade that, for the FoU and the Morpheus Hotel, arise from free-form geometry.
- All case studies use a simplified, parametrically-generated master model to coordinate between architects, consultants and contractors. Each master model represents specific aspects that are relevant to the individual designs: structure for the FoU, the layering of the stars for the LAD and the exoskeleton's cladding envelope for the Morpheus Hotel. The models are examples of the easy-to-change, "lighter data-sets and models" recommended by Holzer (2007) as an alternative to mainstream BIM.
- The generative master models' employment of geometric primitives such as lines and surfaces circumvents problems of data exchange and interoperability between AEC software. Compared to the FoU, the LAD and the Morpheus Hotel required more complex design teams with multiple consultants and contractors. But in all cases, the geometric, easily exchangeable data generated by the models aided the collaboration between architects, consultants and contractors.
- The generative master models allow rapid iterations and thus faster collaboration between architects, consultants and contractors, which enhances the integration of performance aspects into the designs. For example, the models facilitated the coordination between façade and the underlying structure for the three projects. In the cases of the FoU and the LAD, these models could be regenerated within minutes, and within hours for the Morpheus Hotel. The FoU and the LAD also employed generative master models to integrate daylight considerations into the facades' density. For the FoU, this was done directly based on solar irradiation, while the density of the LAD's dome was studied extensively by specialised consultants. Buro Happold developed smaller parametric models to automatically generate and analyse the structural connections of the Morpheus Hotel based on its generative master model.
- None of the cases employ mainstream BIM (except for major structural elements). Instead, they develop customized digital workflows to manage design, prefabrication and assembly. These workflows illustrate the "user-controlled and process-oriented approach to integration and interoperability" proposed by Toth, Janssen, Stouffs, Chaszar, and Boeykens (2012).
- To ensure flexibility and scalability, these customized workflows apply decompositions of the parametric model and efficient algorithms and data structures. The FoU and the LAD employ the half-edge data structure, which indicates its suitability for the efficient generation and coordination of differentiated building components. Both cases clearly separate the definition of the designs' geometric shape from its materialization. For the Morpheus Hotel, the cladding envelope for the exoskeleton serves to both separate and coordinate the design's overall shape, structure and facade.

3.2 Comparing the case studies' parametric design approaches

The FoU implements a mostly objected-oriented programming style in IronPython, while the LAD implements a functional one in F#. (These styles are not mutually exclusive. IronPython and F# support both programming styles, albeit with different emphases.) The Morpheus Hotel employs both visual programming in Grasshopper and distributed visual programming with Grasshopper and Elefront (van der Heijden et al., 2015). This section compares these approaches under the four criteria discussed in section 2.3.2 (flexibility, scalability, verification and collaboration) and adds ease of learning as a fifth.

3.2.1 Object-oriented programming in IronPython

The FoU's objected-oriented style utilizes classes and objects that emphasize the decomposition of the design into form, panelization, pattern and the expression of building components (using Rhinoceros and IronPython, a Python variant). This approach ensures flexibility by allowing each of these aspects to be changed separately. Decomposing the pattern tiles and the half-edge data structure of the FoU into separate objects allows the swapping of one set of pattern tiles for another, the easy inclusion of new data, such as—in the case of the grid-shell described in section 2.1.6—information about the underlying steel structure and the straightforward addition of different expressions of the triangular building components, such as 3D-printed components, folded components and simplified components on a substructure.

The scalability of this approach is demonstrated by the successful generation of the 11 000 panels. Since the parametric model's logic is largely separate from actual 3D geometry, the biggest scale limitation is the size of the generated files. For the FoU, this limitation was addressed by generating the final geometry and cut sheets separately for each grid line, which also aided in visually verifying the results. Developing the parametric model on various iterations of the complete design ensured the correctness of these partial models, which followed the same parametric logic but contained more geometry such as assembly drawings and cut sheets.

An objected-oriented style enables collaboration by decomposing a programming task into objects with clearly defined responsibilities. This approach is most commonly adopted in the development of commercial software applications, but requires programming skills from all participants. For example, in theory, the decomposition of the FoU's parametric model would have allowed one person to develop the model that defines the pattern tiles and another to develop the model that generates a façade component from a pattern tile and a mesh triangle. In practice, the FoU's decomposition allowed the design of its shape and overall façade density to proceed largely independently from its materialization in terms of structure and building components.

In terms of the scalability of the programming language itself one should note that Python is dynamically typed. In other words, in Python, data types are not assigned by the programmer but inferred at runtime. This type inference can make Python slower than statically typed languages (such as C# and F#), but, in practice, the quality of the employed algorithms and the amount of created geometry tend to be more critical for scalability. Python's type inference makes it easier to learn and apply, but can make it harder to verify the correctness of a program.

3.2.2 Functional programming in F#

The LAD employs functional programming (using Rhinoceros and F#), which conceptualizes the parametric model as a hierarchy of nested operations instead of a decomposition into objects. For functional programming, flexibility consists of the ability to call different combinations of functions to generate different parts of the geometry at varying levels of detail and for different purposes. New capabilities can be added by adding functions to the model. More complex functions describing the LAD's stars include progressively more details, while simpler functions maintain the possibility for quick regenerations of the model. Like an object-oriented style, a functional style decomposes programming tasks into functions which can be implemented by different developers. Compared to Python, F# is slightly more scalable and verifiable because F# is statically typed. However, functional programming and statically typed languages can be harder to learn and slower to write.

3.2.3 Visual programming in grasshopper

Of the four approaches, the single visual programming model employed for the Morpheus Hotel likely is the least flexible, since, instead of the explicit decomposition of the objected-oriented, functional and distributed approaches, it employs only a loose grouping of elements (Figure 11). Explicit decomposition is possible in Grasshopper, but was not employed in this case. Davis, Burry, and Burry (2011) show that decomposing Grasshopper models improves their understandability and thus their ease of verification and collaboration. Note that, in contrast to the distributed data model discussed below, this decomposition is applied within a single file. Compared to textual programming, explicit decomposition is less frequently applied in visual programming, possibly because it is less integral to visual programming languages and more cumbersome to apply.

Scalability can also be a problem for this approach, as is demonstrated by the long generation times for the Morpheus Hotel's master model. From an experiment with computational designers, Leitão, Santos, and Lopes (2012) conclude that "learning a [textual programming language] takes more time and effort than learning a [visual programming language], but this effort is quickly recovered when the complexity of the problems becomes sufficiently large".

3.2.4 Distributed data model with Grasshopper and Elefront

The distributed data model of the Morpheus hotel consists of hundreds of Rhinoceros files and Grasshopper models. The model is decomposed both into BIM-like objects (in this case, three-dimensional geometric elements with attributes created with Elefront) and operations (the Grasshopper models). In this case, flexibility means the ability to make both very small and big changes to the system (depending on whether the changes are applied "upstream" and "downstream"). While the other parametric models have this ability as well, it becomes more pronounced by decomposing the model into separate files.

The distributed data model displays affinities not only with BIM, but also, in a limited sense, with object-oriented and functional programming. The combination of properties with geometry and the transfer of properties from simpler to more complex geometry evokes the object-oriented concepts of encapsulation and inheritance, while the limitation of Grasshopper models to a single operation evokes a functional programming principle.

A major difference between the distributed data model and other approaches discussed in this paper is that it creates geometry not only at the end of the computation, but in numerous in-between steps. In other words, the distributed data model presents an alternative to avoiding geometry creation as an approach to scalability. Distributed geometry creation affords collaboration and easier verification, but is more time-consuming and, as Front Inc. point out, presents its own challenges and skill requirements for designing and managing the data flow through the different files (Muscettola et al., 2017).

4 Conclusion

Compared to the theoretical description of PD in terms of formal generation and exploration, the case studies emphasize more practical and specialised applications, such as differentiating the designs' building components, integrating performance aspects and generating design documentation. Here, the strength of PD does not lie in generating many design variants, but in realizing highly specific, differentiated, rule-based designs. In that sense, the case studies signify the "normalization" of PD and its increasing integration into different aspects of architectural and engineering practice.

PD thinking thus expands its role from being mostly focused on representing, generating and evaluating designs to also managing and materializing them, especially when these designs employ differentiated building components. This expansion implies that, increasingly, several participants in a design process—including architects, consultants and contractors—practice PD, which reinforces the need for software- and programming language-independent methods of data exchange.

In comparing the PD approaches of the FoU, the LAD and the Morpheus Hotel, we find that both textual and visual programming have been applied successfully to the realization of highly differentiated buildings, using objected-oriented, functional and distributed programming styles. However, it appears that, while visual programming has a flatter learning curve, in requires additional care and effort to achieve a flexible and scalable decomposition of the parametric model.

Skill is an important barrier for the wider adoption of textual programming and distributed (or decomposed) visual programming in practice, despite the advantages of these approaches. Leitão et al. (2012) present an attempt to overcome this barrier by combining different textual programming languages and AEC software into an interoperable framework called Rosetta.

The fact that all parametric models rely on the same software, Rhinoceros, for geometry generation and graphical representation but employ different, visual and textual, programming languages and styles emphasizes customizability as an important property for future AEC software. By allowing visual and textual programming, this customizability also addresses different levels of skill.

The creation of customized, attributed geometry in the case of the Morpheus Hotel is notable and could be combined with other, non-distributed approaches. The fact that the cases largely avoid the IFC classes that lie at the heart of BIM indicates that those classes are not necessary to ensure data exchange and likely insufficiently reflect the thinking of parametric designers.

Decomposition of design moves is a powerful method for creative parametric designers. It supports flexibility by allowing easier changes to parametric models and enables the exchange of design data between different software and stakeholders. Determining decompositions that ensure flexibility, designing workflows that harnesses such decompositions to enable coordination and collaboration in a design team and selecting appropriate algorithms and data structures that ensure scalability are aspects of PD thinking that transcend applied PD skills in using software or programming languages. This insight clarifies the necessity of more conceptual approaches to PD in, for example, architectural education, in addition to the need for and in support of more customizable and interoperable PD and BIM tools and workflows.

Acknowledgments
We thank the anonymous reviewers for fruitful interactions that were instrumental to shaping this paper. We also thank Goswin Rothenthal for his

clarifications on the parametric master model of the LAD and Ramon van der Heijden for his clarifications on the distributed data model of the Morpheus Hotel. The Future of Us and related projects were designed by the Advanced Architecture Laboratory at Singapore University of Technology and Design, directed by Professor Thomas Schroepfer.

References

Abelson, H., & Sussman, G. J. (1996). *Structure and interpretation of computer programs* (2nd ed.). Cambridge, Mass: The MIT Press.

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., & Levy, B. (2010). *Polygon mesh processing*. Natick, MA: A K Peters.

Davis, D., Burry, J., & Burry, M. (2011). Understanding visual scripts: Improving collaboration through modular programming. *International Journal of Architectural Computing, 9*(4), 361–376.

Hesselgren, L., Charitou, R., & Dritsas, S. (2007). The bishopsgate tower case study. *International Journal of Architectural Computing, 5*(1), 61–81.

Holzer, D. (2007). Are you talking to Me? Why BIM alone is not the answer. In *In proceedings of the fourth international conference of the association of architecture schools of Australasia*.

Hudson, R. (2010). *Strategies for parametric design in architecture: An application of practice led research*. (Ph.D. Dissertation). Bath, UK: University of Bath.

Imbert, F., Frost, K. S., Fisher, A., Witt, A., Tourre, V., & Koren, B. (2013). Concurrent geometric, structural and environmental Design: Louvre Abu Dhabi. In L. Hesselgren, S. Sharma, J. Wallner, N. Baldassini, P. Bompas, & J. Raynaud (Eds.), *Advances in architectural geometry 2012* (pp. 77–90). Springer Vienna.

Janssen, P., & Chen, K. W. (2011). Visual dataflow modelling: A comparison of three systems. In *Proceedings of the 14th international conference on computer aided architectural design futures* (pp. 801–816), (Liege, BE).

Leitão, A., & Proença, S. (2014). On the expressive power of programming languages for generative design. InThompson, E M. (Ed.). (2014). *Fusion – Proceedings of the 32nd eCAADe conference, Vol. 1* (pp. 257–266). Newcastle upon Tyne, UK: Northumbria University.

Leitão, A., Santos, L., & Lopes, J. (2012). Programming languages for generative design: A comparative study. *International Journal of Architectural Computing, 10*(1), 139–162.

Muscettola, V., Salvi, M., Mutyaba, M., van der Heijden, R., Tai, A., & Levelle, E. (2017). *The Morpheus Hotel: From design to production*. Retrieved from. www.rhino3d.com/go/morpheus.

Oxman, R. (2006). Theory and design in the first digital age. *Design Studies, 27*(3), 229–265.

Park, K., & Holt, N. (2010). Parametric design process of a complex building in practice using programmed code as master model. *International Journal of Architectural Computing, 8*(3), 359–376.

Piermarini, E., Nuttall, H., May, R., & Janssens, V. M. (2016). City of dreams, Macau: Making the vision viable. *The Structural Engineer, 94*(3), 56–67.

Poirriez, C., Wortmann, T., Hudson, R., & Bouzida, Y. (2016). From complex shape to simple construction: Fast track design of "the future of us" gridshell in Singapore. In K. Kawaguchi, M. Ohsaki, & T. Takeuchi (Eds.), *Proceedings of the IASS Annual symposium 2016 "spatial structures in the 21st century."*. Tokyo, JP: IASS.

Rothenthal, G. (2016a). The 3D geometry of Louvre Abu Dhabi. In *Presented at the fsharpConf 2016*. Retrieved from. channel9.msdn.com/Events/FSharp-Events/fsharpConf-2016/The-3D-Geometry-of-Louvre-Abu-Dhabi.

Rothenthal, G. (2016b). *We see great potential for F# to be used as a scripting language in CAD; it fits very well for computational design challenges in the construction industry*. Retrieved from. http://fsharp.org/testimonials/#goswin-1.

Shepherd, P., Hudson, R., & Hines, D. (2011). Aviva stadium: A parametric success. *International Journal of Architectural Computing, 9*(2), 167−185.

Toth, B., Janssen, P., Stouffs, R., Chaszar, A., & Boeykens, S. (2012). Custom digital workflows: A new framework for design analysis integration. *International Journal of Architectural Computing, 10*(4), 481−500.

Tourre, V., & Miguet, F. (2010). Lighting intention materialization with a light-based parametric design model. *International Journal of Architectural Computing, 8*(4), 507−524.

van der Heijden, R., Levelle, E., & Riese, M. (2015). Parametric building information generation for design and construction. In *ACADIA 2105: Computational Ecologies: Design in the Anthropocene* (pp. 417−429). ACADIA.

Whitehead, H. (2003). Laws of form. In B. Kolarevic (Ed.), *Architecture in the digital Age: Design and manufacturing* (pp. 89−113). Taylor & Francis.

Woodbury, R. F. (2010). *Elements of parametric design*. London; New York: Routledge.