Facilitating change: the design domain in prefabricated home design.

Author 1: Robert Doe Author 2: A/Prof Mathew Aitchison

Faculty of Architecture, Design and Planning,

The University of Sydney, Sydney, Australia

rdoe8535@uni.sydney.edu.au

Facilitating change: the design domain in prefabricated home design.

This paper examines extrinsic and intrinsic design domains which form topological associations between elements. Parametric modellers with declarative and imperative programming paradigms are used to explore this theme. A case study of a multi-storey prefabricated project provides focus for this research.

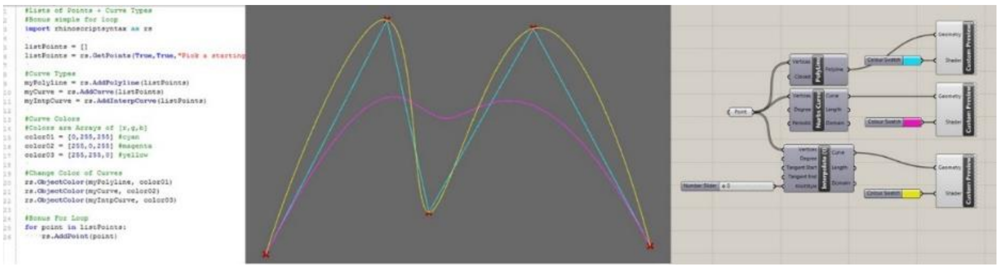
Keywords: design rules; design domain; declarative; imperative; parametric modelling; prefabricated homes

Introduction

It is clear that architects are challenged by the operation and conceptual organisation of computational design methods, and that more openly accessible and easily understood expressions of design rules are required if architects are to engage with its tools.

Therefore, to examine these prerequisites for engagement, this paper focusses primarily on design rules which define the ‘design domain’ [1] in relation to a case study. The design domain is the framework which creates better understanding of the structure of the artefact being designed, the topological space of relations and its abstract geometrical content. It has also been described as, “… the principal technical obstacle to further practical use” [2], but it includes significant benefits in allowing deferral of design decisions. It is further noted that the design domain is an alternative cognitive design space open for exploration by students and architects, requiring better understanding if authorial control is to be implemented and maintained.

This paper also explores the format in which these design rules are expressed. In the quest for more accessible and intelligible design rules it has been argued that they should be expressed in a declarative programming format [3]. Whereas imperative design rules describe the steps on the way to achieving an end result, declarative design rules describe what the user wants, but do not explain how it is achieved. These differences are illustrated by graphical user interfaces which represent these formats – textual programme and visual data flow graph - both of which implement the same NURBS and polyline curves [Figure 1].



```
#Lists of Points + Curve Types
#Bonus simple for loop
import rhinoscriptsyntax as rs

listPoints = []
listPoints = rs.GetPoints(True,True,"Pick a starting

#Curve Types
myPolyline = rs.AddPolyline(listPoints)
myCurve = rs.AddCurve(listPoints)
myIntpCurve = rs.AddInterpCurve(listPoints)

#Curve Colors
#Colors are Arrays of [r,g,b]
color01 = [0,255,255] #cyan
color02 = [255,0,255] #magenta
color03 = [255,255,0] #yellow

#Change Color of Curves
rs.ObjectColor(myPolyline, color01)
rs.ObjectColor(myCurve, color02)
rs.ObjectColor(myIntpCurve, color03)

#Bonus For Loop
for point in listPoints:
    rs.AddPoint(point)
```

Figure 1. Left - Imperative (object oriented), Right - Declarative (functional, dataflow)

In summary, architects want to understand and make best use of computers amid dynamic design processes, and are also aware that such computational design skills link design with better ways of fabricating and assembling buildings. This paper contributes to understanding and implementation of one of these skills, the set-up of the design domain - the portal between intuitive and abstract design processes. It also explores the extent to which this set-up may be realised in an accessible and flexible format, suitable for architects' use in everyday practice.

Background

The design domain is the initial expression of the dynamic and hierarchical characteristics of an artificial system. A seminal explanation of these characteristics is von Bertalanffy's General Systems Theory [4] which captured a growing tendency in the science of its time to recognise the importance of dynamic interaction in all fields of reality. This marked a shift from a mechanistic and static understanding to one incorporating organic and dynamic tendencies, and its principles may be applied to all complex systems whether natural or artificial. Whilst disagreeing with von

Bertalanffy’s abstraction from particular to general, Simon (1969) [5] reiterated the need for hierarchy when dealing with complex systems and recommended decomposition of this complexity into its functional parts.

In seeking to understand and improve architectural design and building processes both these insights are useful and lie at the root of innovations which have implemented dynamic and hierarchical methods of organisation in computational design. A significant example is Sutherland’s Sketchpad of 1963 [6] which introduced a computer aided design system with parametric modelling capabilities. Thus, abstract design rules were now capable of generating form which, “… emerges from a complex of constraints on and interrelationships among parameters” [7].

The design domain represents the opening move in defining these abstract design rules, and it’s the format of these rules, their degree of implicit and explicit expression, and their integration into the design process which remains a serious challenge to architects and software designers.

Methodology

This exploration of the design domain is undertaken in relation to a case study, a 3 storey 26 apartment development in Brisbane, Australia [Figure 2].

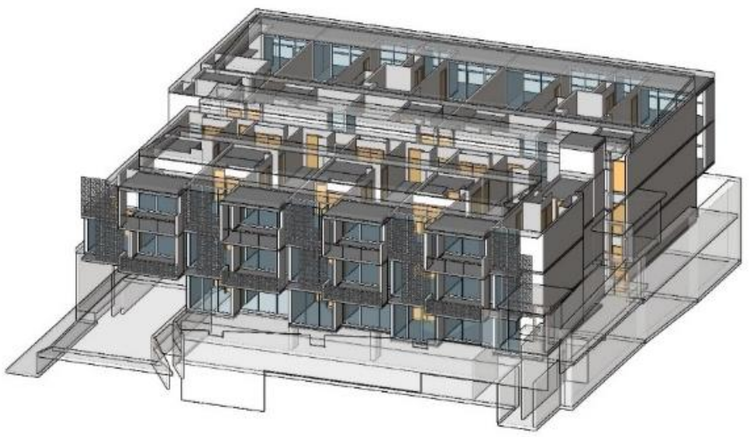


Figure 2. AP14 project, Brisbane, Australia

It is both a commercial development designed by Fairweather Jemmott Architects and a research project (AP14), funded by the Queensland Government Accelerate Partnerships program, and designed by Aquatonic, to compare prefabricated assembly with traditional building methods. A hybrid prefabrication strategy proposes the use of panellised floor, wall and roof elements, volumetric bathrooms pods, and clip-on balcony components.

Thus, the extrinsic design domain has been manipulated during negotiations between stakeholders. Horizontal planes representing height limits, and vertical planes representing set-backs from neighbours and boundaries, have been made formal, as is conventional. Internally, grids (more vertical planes) and levels (more horizontal planes) have been similarly conceptualised. The case study therefore considers elements bound to this extrinsic design domain – in particular, unit types and wall components.

The study is implemented using a 'component based modeller' (Autodesk Revit) mainly used in the construction industry, and a 'design development program' (Digital Project) [8] mainly used in manufacturing industries. These software systems include programming add-ins which may be declarative or imperative in format. Such programming paradigms have been described by Davis (2011) [9] and will be discussed in more detail later on relative to a simple automation design task exploring their degree of accessibility and intelligibility.

Design Domain

Morphology and topology

It has been stated above that definition of the design domain enables the generation of form using abstract design rules. However, despite this morphological outcome, we might be more enlightened by focussing on the invisible associations between, and the topological nature of, these elements. For example, topology has been noted as one of four design drivers [10] and is defined as the study of the connections between vertices, edges, surfaces and closed volumes, and of the properties preserved beyond deformations which might twist and stretch objects. Therefore the essence of the topological is change and fluidity, and yet recent architecture has come to represent the driving force of topology as a frozen object caught between transformations from one state to another [11]. However, rather than morphology, it's these invisible associations which more meaningfully define topological architecture and parametric modelling.

But there remains a need for more intelligible representations of these associations which are also responsive to architects working methods [2]. For example, few representations of the design domain have been as conceptually immediate as Antoni Gaudi's analogue model of the structure of the Sagrada Familia church, Barcelona, Spain expressed by hanging weighted chains which reveal interactions between forces and forms.

Nevertheless, several metaphorical references to the design domain have illuminated its characteristics. Alluding to the process of making manufactured objects, the design domain has also been termed the 'jig', an abstract framework isolating structure and spatial location from geometric detail [12]. The jig functions to hold or support work and may guide machine tools to do the work [13]. In a similar vein these associations have been termed the 'control rig', a tool designed by the designer as a formalised and externalised design process [14]. With reference to the variability and interaction offered by the digital world, and in contrast to 'objects' of the previous era, the design domain has also been labelled the 'objectile' [15].

These terms define a tool for commencement of the act of creation of the design domain. The consequences which follow from this act will be discussed next.

Distance and deferral

Significantly, creation of the design domain places distance between designer and artefact representing one of, "… an increasing number of levels of indirection" [14] which architects face when working with computational design tools. Indirection in computer programming is the ability to reference something using a name instead of the value itself. Thus, the rigour of parametric manipulation of geometry interrupts the flow of the architectural design process.

Firstly, there are now two levels of authorship. There is the primary author of this structured framework, the design domain, and there is the secondary author specifying generic objects in an environment designed by others [15]. Therefore, if architects are to master these computational design tools and remain primary authors of their works rather than merely gamers and players, it is essential to understand and take control of the design domain.

Secondly, exploration during conceptual design might be hindered by this need to set up a design domain because, unlike the architect, the computer is unintuitive and therefore blind to ambiguous emergent forms [3]. For this reason it has even been questioned whether parametric modelling of the design domain is useful in the early stages, or whether it should be left until later when more is known about design problems [1]. However, this suggests that its set-up is singular, whereas it is argued here that set-up occurs multiple times, and that there are sub-design domains at different levels of detail and at different stages of the design problem.

Despite the challenges of authorship and set-up of the design domain, a significant advantage arises from creating distance between designer and artefact. It is precisely this distance which enables deferral of design decisions during the design process. This is because relationships between values and structure are made explicit and therefore they can be computed or re-computed as context and conditions change, and as more becomes known about the design problem. These retrospective changes may be made at less cost to design and construction than non-parametric modelling methods in which relationships between values and structure are unambiguous and are placed exactly in context - the inverse of parametric modelling processes.

Examination of the nature of the design domain, its set-up and need for redefinition, and the advantages of distance and deferral are explored next in the context of the AP14 project, its unit types and panellised wall components.

Case Study – AP14

Intrinsic design domain

The intrinsic design domain is bounded by horizontal (levels) and vertical planes (grids) which are the result of site constraints and negotiations between stakeholders in the project. Here we focus on grouped unit type B, placed by alignment to reference levels 1, 2 or 3 and in relation to perpendicular vertical planes [Figure 3].

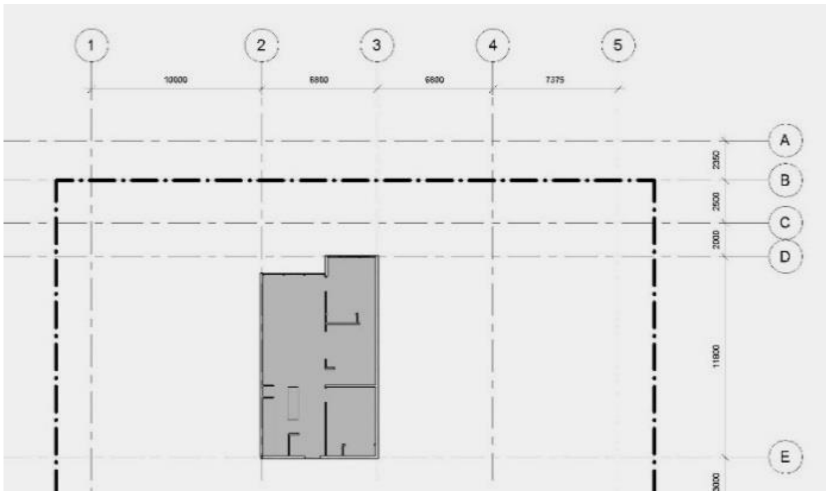


Figure 3. Unit B’s design domain (Autodesk Revit)

The component based modeller (Autodesk Revit) allows changes to propagate instantly between grouped unit types, whilst constraints and dimensions are parametrically defined. However, it should be noted that the ‘parametric’ behaviour of these types of modellers refers primarily to parameters associating data to elements in the model compatible with the requirements of Building Information Modelling (BIM).

Sub-design domains

Within the intrinsic design domain defined above are sub-design domains defining relationships between components’ elements. For example, the prefabricated wall panel, rather than a drawing, might instead be defined by a textual and graphical description of what its associations should do, how they affect each other, and to what purpose [1] as follows: “… instantiate the wall panel to any thickness, length, width or height, make it notched or un-notched at its ends, ensure it has fixing brackets automatically located at 600mm centres, set-in 300mm at its ends, and that all elements’ data can be automatically scheduled.” [Figure 4].

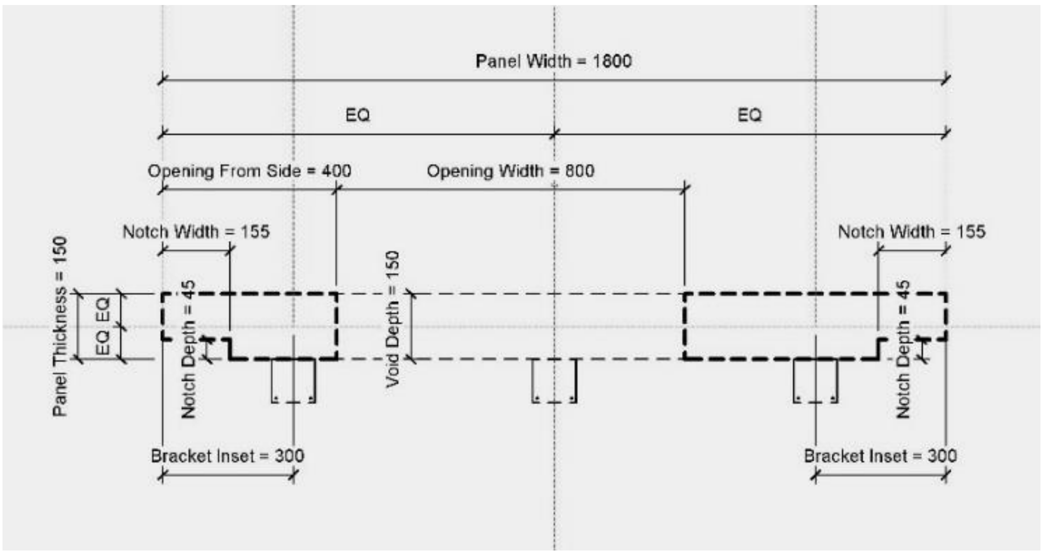
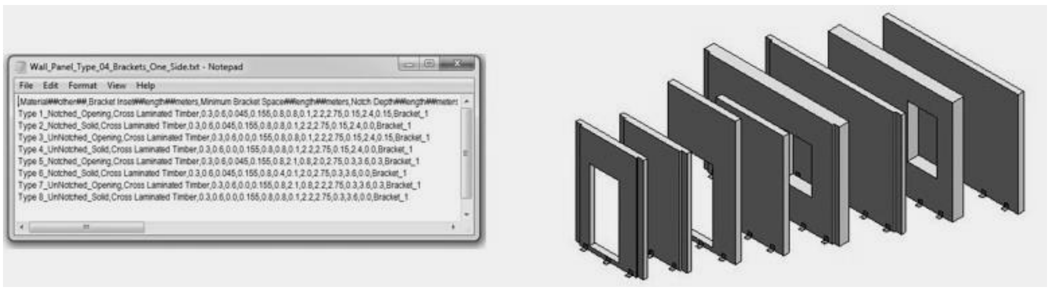


Figure 4. Wall panel sub-design domain (Autodesk Revit)

Thus, reference planes are the frame by which wall geometry is parametrically controlled, enabling explicit associations between elements of the wall panel component and its rules for locating fixing brackets to be amended at a later date. Such parametric variation enables instantiation of many different wall panel types - notched, un-notched, solid or with openings – and the creation of instances using a text defined type catalogue [Figure 5], allowing efficient modelling of unit type B [Figure 6].


Wall_Panel_Type_04_Brackets_One_Side.txt - Notepad
File Edit Format View Help

```
Material##other##,Bracket Inset##length##meters,Minimum Bracket Space##length##meters,Notch Depth##length##meters
Type 1_Notched_Opening,Cross Laminated Timber,0.3,0.6,0.045,0.155,0.8,0.8,0.1,2.2,2.75,0.15,2.4,0.15,Bracket_1
Type 2_Notched_Solid,Cross Laminated Timber,0.3,0.6,0.045,0.155,0.8,0.8,0.1,2.2,2.75,0.15,2.4,0.0,Bracket_1
Type 3_UnNotched_Opening,Cross Laminated Timber,0.3,0.6,0.0,0.155,0.8,0.8,0.1,2.2,2.75,0.15,2.4,0.15,Bracket_1
Type 4_UnNotched_Solid,Cross Laminated Timber,0.3,0.6,0.0,0.155,0.8,0.8,0.1,2.2,2.75,0.15,2.4,0.0,Bracket_1
Type 5_Notched_Opening,Cross Laminated Timber,0.3,0.6,0.045,0.155,0.8,2.1,0.8,2.0,2.75,0.3,3.6,0.3,Bracket_1
Type 6_Notched_Solid,Cross Laminated Timber,0.3,0.6,0.045,0.155,0.8,0.4,0.1,2.0,2.75,0.3,3.6,0.0,Bracket_1
Type 7_UnNotched_Opening,Cross Laminated Timber,0.3,0.6,0.0,0.155,0.8,2.1,0.8,2.2,2.75,0.3,3.6,0.3,Bracket_1
Type 8_UnNotched_Solid,Cross Laminated Timber,0.3,0.6,0.0,0.155,0.8,0.8,0.1,2.2,2.75,0.3,3.6,0.0,Bracket_1
```

Figure 5. Text defined type catalogue and wall panel family types (Autodesk Revit)

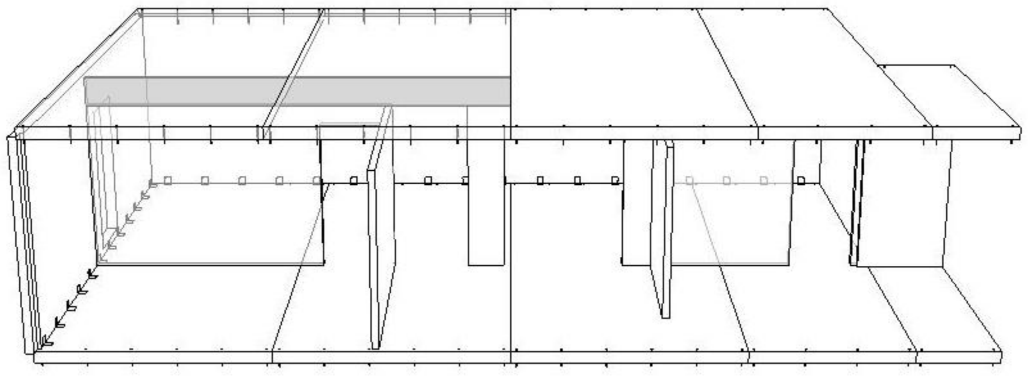


Figure 6. Unit type B - wall panels & automated bracket placement (Autodesk Revit)

A design development program (Digital Project) is used to further examine creation of the design domain for the wall component. Due to a hierarchical set-up conceptualisation of the workflow is clear and follows a logical sequence, familiar to manufactured design processes. For example, a wall panel and bracket part are assembled to make a unit type product. The wall panel part consists of three sketches which define a rectangular plan of any dimensions, the location of notches of any size or depth to the ends of the wall, and an opening of any size [Figure 7]. These sketches' parameters and constraints drive the model's geometry - the pad (wall panel) and its pockets (notches, opening) in this case - and are associated with levels and grids in the unit type B assembly [Figure 8].

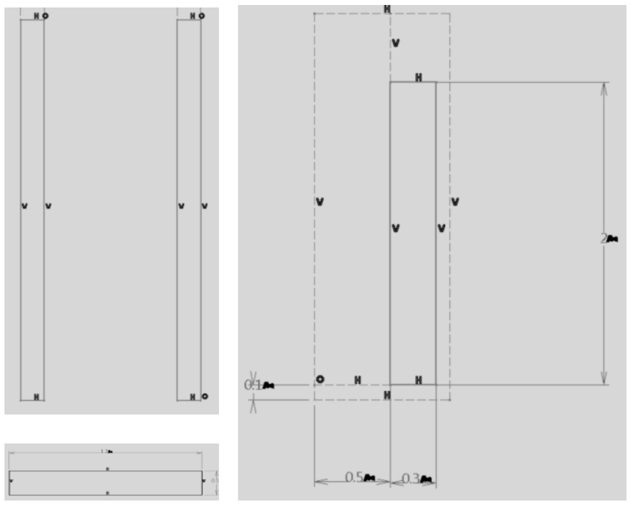


Figure 7. Three sketches for wall panel part (Digital Project)

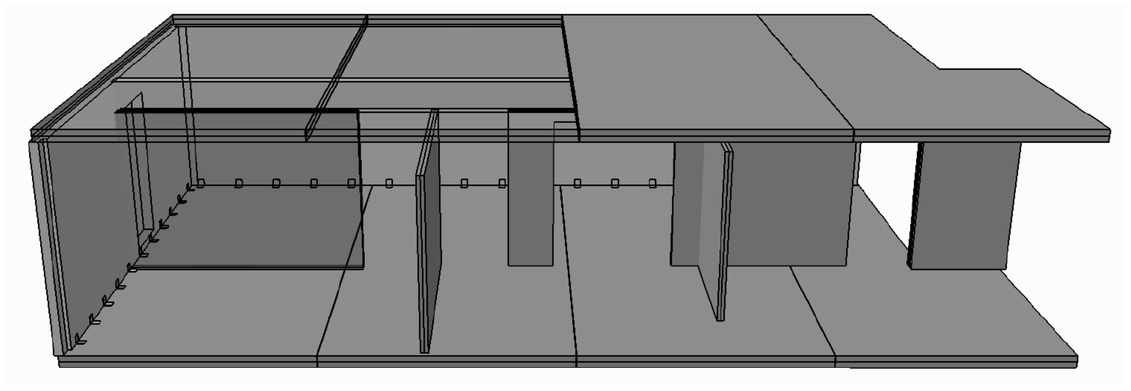


Figure 8. Unit type B – wall panels with brackets multi-instantiated (Digital Project)

The fully parametric and automated design capabilities of such design development programs have been advocated as improvements to workflow in the production of prefabricated components [16]. Their hierarchical rigour enables the design domain to be analysed and interrogated with reference to assembly and relational trees revealing the ‘parents and children’ of parts created. The increased cognitive effort, that is the attention and understanding required to implement these abstract logical associations, is traded-off for better understanding of complexity and the ability to retrospectively affect these associations.

Declarative & Imperative

Automation may be extended using the programming add-ins to these software systems. For example, using a paradigm which is declarative in format, a collection of 3D Cartesian coordinates may define an abstract design domain for the placement of wall and floors panels [Figure 9].

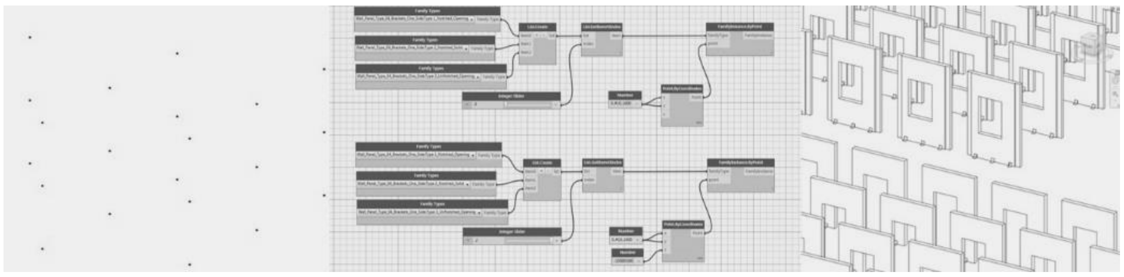
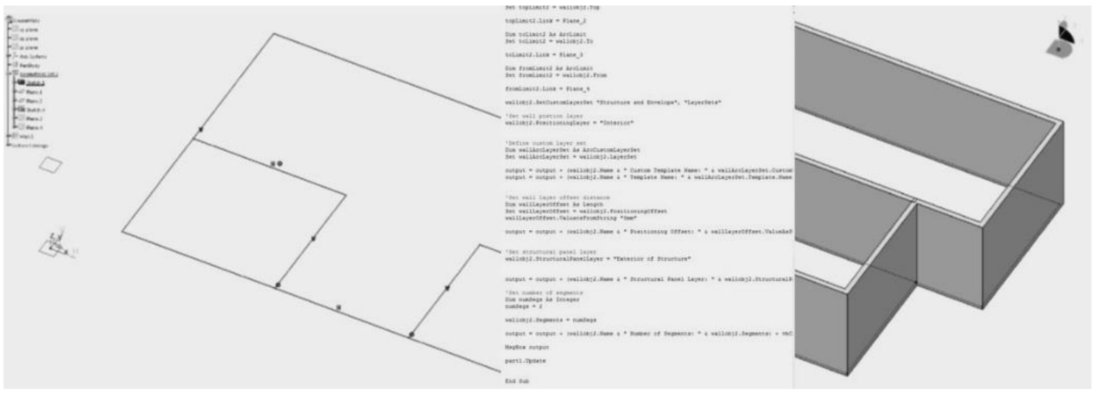


Figure 9. Declarative format (Autodesk Revit & Dynamo) – wall panels

Alternatively, using a paradigm which is imperative in format and text-based, curves and planes may define the design domain for the placement and generation of wall panels [Figure 10].



```
Set topLimit2 = wallobj2.Top

topLimit2.Link = Plane_2

Dim toLimit2 As ArcLimit
Set toLimit2 = wallobj2.To

toLimit2.Link = Plane_3

Dim fromLimit2 As ArcLimit
Set fromLimit2 = wallobj2.From

fromLimit2.Link = Plane_4

wallobj2.SetCustomLayerSet "Structure and Envelope", "LayerSets"

'Set wall position layer
wallobj2.PositioningLayer = "Interior"


'Define custom layer set
Dim wallArcLayerSet As ArcCustomLayerSet
Set wallArcLayerSet = wallobj2.LayerSet

output = output + (wallobj2.Name & " Custom Template Name: " & wallArcLayerSet.Custom
output = output + (wallobj2.Name & " Template Name: " & wallArcLayerSet.Template.Name


'Set wall layer offset distance
Dim wallLayerOffset As Length
Set wallLayerOffset = wallobj2.PositioningOffset
wallLayerOffset.ValuateFromString "0mm"

output = output + (wallobj2.Name & " Positioning Offset: " & wallLayerOffset.ValueAsS


'Set structural panel layer
wallobj2.StructuralPanelLayer = "Exterior of Structure"


output = output + (wallobj2.Name & " Structural Panel Layer: " & wallobj2.StructuralP

'Set number of segments
Dim numSegs As Integer
numSegs = 2

wallobj2.Segments = numSegs

output = output + (wallobj2.Name & " Number of Segments: " & wallobj2.Segments) + vbC

MsgBox output

part1.Update


End Sub
```

Figure 10. Imperative format (Digital Project & VBA) – Unit type B

Conclusion

This paper has examined a computational design method, set-up of the design domain - the opening move in defining abstract geometrical associations between design elements. Intriguingly it creates distance between architect and artefact, but rather than a hindrance this network of relations greatly improves architects understanding of the fundamental nature of their designs. Perhaps more importantly this distance allows a breathing space in the design process by enabling deferral of design decisions and retrospective reviews along the way.

It might be observed that it takes more cognitive effort for architects to set-up the design domain because abstract logical operations are not part of most architects training. Parallel paths developing architects algorithmic thinking skills alongside their intuitive capacities is one way forward [14]. The incentive is that, by doing so, architects are more likely to maintain authorial and conceptual control.

This paper also briefly explored whether design rules expressed in declarative format rather than imperative format are more easily understandable and modifiable when setting up the design domain. Clearly declarative formats allow a more intuitive engagement by the architect, but the need for algorithmic thinking still applies.

Future research will examine in more detail the ease of use and intelligibility of the expression of design rules in both declarative and modular format, as urged by Mitchell (1989) [3], whilst exploring their synergies relative to a live case study of a prefabricated housing project.

References

[1] J. Burry, "Mindful spaces: computational geometry and the conceptual spaces in which designers operate," *International Journal of Architectural Computing,* vol. 5, pp. 611-624, 2007.

[2] R. Aish and R. Woodbury, "Multi-level interaction in parametric design," presented at the Smart Graphics, 5th International Symposium, Germany, 2005.

[3] W. J. Mitchell, "A new agenda for computer-aided design.," presented at the CAAD Futures, Vienna, Austria, 1989.

[4] L. von Bertalanffy, "An outline of General System Theory," *British Journal of the Philosophy of Science,* vol. 1, pp. 134-165, 1950.

[5] H. A. Simon, *The Sciences of the Artificial*. Cambridge, MA: MIT Press, 1969.

[6] I. E. Sutherland, "Sketchpad: a man-machine graphical communication system," PhD dissertation, MIT, 1963.

[7] D. R. Scheer, *The Death of Drawing: Architecture in the Age of Simulation*. London, UK: Routledge, 2014.

[8] D. Schodek, M. Bechthold, K. Griggs, K. M. Kao, and M. Steinberg, *Digital Design and Manufacturing: CAD/CAM Applications in Architecture and Design*. NJ, USA: John Wiley & Sons Inc., 2005.

[9] D. Davis, "Modelled on software engineering: flexible parametric models in the practice of architecture," PhD dissertation, School of Architecture and Design, RMIT University, Melbourne, VIC, Australia, 2013.

[10] A. Kilian, "Design innovation through constraint modelling," *International Journal of Architectural Computing,* vol. 04, pp. 88-105, 2006.

[11] J. Burry and M. Burry, *The New Mathematics of Architecture*. London: Thames and Hudson, 2010.

[12] R. Woodbury, R. Aish, and A. Kilian, "Some patterns for parametric modelling," presented at the ACADIA, Halifax, Nova Scotia, 2007.

[13] J. Garrison and A. Tweedle, *Modular Architecture Manual*. USA: Kullman Buildings Corporation, 2008.

[14] R. Aish, "From Intuition to Precision," presented at the eCAADe23, Lisbon, Portugal, 2005.

[15] M. Carpo, *The Alphabet and the Algorithm*. Cambridge, MA: MIT Press, 2011.

[16] P. Jensen, T. Olofsson, and H. Johnsson, "Configuration through the parameterisation of building components," *Automation in Construction,* vol. 23, pp. 1-8, 2012.