S. Chien, S. Choo, M. A. Schnabel, W. Nakapan, M. J. Kim, S. Roudavski (eds.), *Living Systems and Micro-Utopias: Towards Continuous Designing, Proceedings of the 21st International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA 2016,* 000–000.

AN ADVANCED PARAMETRIC MODELING LIBRARY FOR ARCHITECTURAL AND ENGINEERING DESIGN

STYLIANOS DRITSAS
Singapore University of Technology and Design, Singapore
dritsas@sutd.edu.sg

Abstract. This paper presents a design computation system supporting scientific computing methods relevant to architectural and engineering design under the paradigm of visual programming. The objective of this research work is to expand and advance the palette of methods employed in academic and professional design environments. The tools contain methods for linear algebra, non-linear solvers, network analysis and algorithms for classical operational research problems such as cutting and packing, clustering and routing. A few decades ago the idea that computing would become so pervasive in the realm of architecture and engineering as it is today was confronted with deep scepticism. The thesis of this paper is that while it may be equally implausible that such methods are relevant today it may be the next natural evolution in the direction of design computation. The current state of the presented software package is still in early alpha version and it is available online for evaluation.

Keywords. Design Computation; Parametric Modelling; Visual Programming.

1. Introduction

In the past decades the process of reinventing the traditional Computer Aided Design and Manufacturing (CAD/CAM) tool-chain moving away from design/drafting and towards design/computing gave rise to innovation by the adoption of computational methods for design such as the Processing by Casey Reas and Ben Fry (2014), which offered low overhead foundations for visual design using the Java programming language (2001); novel parametric modelling such as Generative Components and Design Script by Robert Aish (2005, 2013) introduced graph-based top-down parametric modelling better

aligned with architectural design thinking, in contrast to bottom-up part-assembly design; and visual programming systems such as Grasshopper by David Rutten lowered the steep learning curve barrier to entry into design computation offering an explicit graphical black box modelling approach to parametric process definition.

Even though there have been already at least two decades of research and practice within new generation digital media, ignoring for the moment the pioneering work in design computation from as early as the 60s (Rocha, 2004), our mental paradigm is still heavily form orientated. Design computation systems such as the aforementioned offer the means for conceptual design with computation but nevertheless the primary medium of design is graphics even though it is operated using computation constructs such as data types, containers and transformations. A positive yet perhaps modest direction forward is found in new systems for virtualizing design-for-construction practices which aim to push the building industry in the age of information, namely Building Information Modelling. The introduction of other units of measurement beyond spatial/meters offers the opportunity for insight in the other less popular dimensions spanned by design artefacts. Yet BIM is also heavily influenced by the notion of modelling as an improved version of rather traditional design processes and representations augmented by non-geometric metadata.

Those developments are nevertheless very valuable but one may inquire about the next steps in the evolution of architectural computing, for both the sake of advancing the state of the art in research and design as well as for addressing practice challenges. If computation is a valuable design thinking process rather than an infrastructural technology to support design actions such as automation, then perhaps it is useful to investigate how it can be further conceptually expanded.

2. Computational Modelling

A key notion identified in this work is that of modelling. But we need first to eliminate some preconceptions about the term itself. Instead of modelling for design generation purposes, constructing design intent through logical relationships and procedural representations or automated representation production as an improved medium compared to drawing, we may consider the idea of a model in the sense closer to what we find sciences. The distinction implied here is best captured by the notion of normative versus descriptive thinking by Peter Rowe (1986) extended to the realm of model making. A central concern in science models is foremost to understand complex phenomena, then to capture and organize their behaviours in functional constructs, only thereafter

to analytically evaluate results, derive predictions, support decisions and exercise control. A model is thus a thought construct of understanding as well as a medium for experimentation and operations/production rather than a mathematical or geometric representation itself. Nevertheless we need to be cognisant that design models are fundamentally normative as they objective is to prescribe rather than describe the world.

The presented work is motivated by the observation that there are certain recurring patterns in creating complex models for spatial design, namely (a) the escalation of complexity in terms of mental capacity to retain numerous interacting primitives and relationships, (b) the difficulty to retain the procedural representation aligned to mental understanding and construction mechanics of the model, and (c) recurrence of classical intractable type of problems such as combinatorial assignment which are found both in architectural studies of spatial configuration as well as in construction sizing and scheduling, known in computer science as non-polynomial hard (Hartmanis and Stearns, 1965; Garey and Johnson, 1979). The assumption in this work is that learning may occur either in forward manner from theory to practice or alternatively backwards on per-need basis moving from specific instances of practical problems to understanding the broader theoretical fabric.

There are thus two avenues available for investigation: either delve into the fundamentals notions of computational modelling, for instance questioning how computer languages or visual programing systems can be improved to support thinking by creating new interfaces to computation; and/or alternatively bring advanced methods of computational modelling already used in such fields such as mathematics, computer science, operational research in the design realm. The approach presented here pivot heavier in the later direction with intent to address complexity with the introduction of advanced computational methods compacted in black-box manner. Those may improve cognitive characteristics of modelling by introducing higher level notions, that is instead of black-boxing fundamental arithmetic and geometric operations, offer abstract modelling primitives such as user-defined data types, mathematical graphs, general matrices and operations such as computationally involved in both development and runtime analysis and optimization tools.

Many of those tools are already available and widely used in scientific computing packages such as MatLab, Mathematica and Sage as well as software libraries such as NumPy and SciPy. What is interesting about those systems is their ability to allow for rapid and terse model development by hiding computational nuances of advanced algorithms. In this spirit the presented work aims at offering a similar perspective to design modelling in a compatible mode of current practices based on visual programming.

3. Library Design

The library contains a large number of components organized externally in tool palettes (Figure 1) and internally in two packages: (a) Core Algorithms and (b) Presentation Engine. The model/view approach was employed as a design paradigm for separating computing logic of core algorithms from graphical presentation interface and visualization.

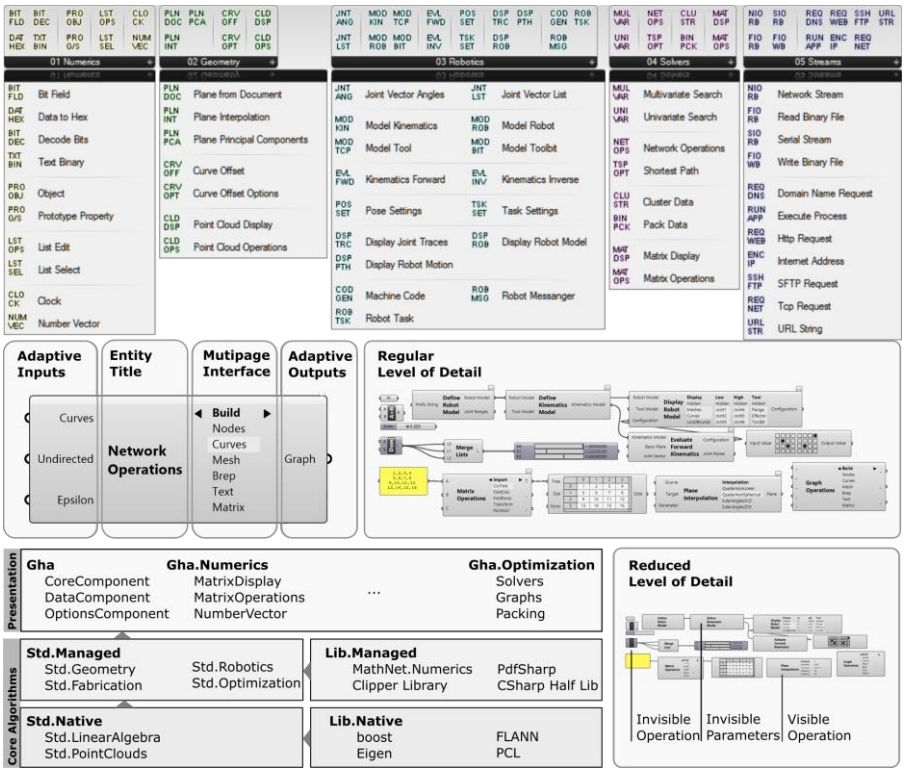

Figure 1. External structure of the library (top) and internal in algorithms and presentation.

Even though there are was effort made for non-overlapping dependencies there are unavoidable influences in the design of algorithms to afford interactivity. Numerical solvers by driving input parameters violate the directed acyclic graph models by introducing feedback relationships. Similar requirements are encountered in simulating kinematics as well as network communications where asynchronous events counteract the steady-state logic of the modeling graph. Using graph based modelers for dynamic event data handling is not uncommon as it is already a paradigm employed in signal processing software

products such as LabView (National Instruments, 2015) and SimuLink (Mathworks, 2015).

3.1. CORE ALGORITHMS

The data structures and algorithms were developed by the author over many years of research targeting applied design problems. Those we generalized for wider use in architecture, engineering and construction applications. The system incorporates open source libraries for numerical computation, advanced data structure modelling, computational geometry and file transactions. Algorithms are implementations of published research in computer science, graphics, engineering and operations research, acknowledgements thereof are included in the documentation.

3.2. PRESENTATION FRAMEWORK

Visual programming is less succinct and inefficient compared to document-based programming. However, it is much more accessible to design audiences. The end-user's goal is often not in developing production grade software but design experimentations and as such visual programming is an excellent medium. A key challenge in visual programming is the unavoidable clutter produced by complex directed acyclic graphs of numerous nodes and edges. In developing libraries for visual programming this problem is greatly exaggerated with the potential clutter of the component palette with numerous nodes of similar functionality only with slightly different input or output parameters.

To maximize functionality in the smallest possible visual footprint: (a) we developed a visual style such that overall design logic remains accessible at different graph viewing scales, such that when node/edge saturation inevitably occurs it is possible to parse high-level logical relationships; (b) created custom representation components for such non-visual concepts as bit-fields and data maps; (c) integrated functional and visual options (enumerated types and bit flags) within node types as well as multipage nodes with optional input and output fields to compress numerous functions (Figure 1: bottom right).

4. Functional Component Organization

4.1 PRIMITIVES

Primitives aim for data-type definition beyond the currently supported entities. The category contains components for intrinsic data types and containers thereof. The Prototype Object component offers a thought experiment on user-defined data type definition, closely related to record types of pre-object-oriented programming languages (Figure 2). This is highly desirable as there are

no options for modelling structured heteronomous data arrangements beyond lists and trees which are formally containers rather than types. There are conceptual challenges to achieve method encapsulation using this approach because those are contained in the document space. Nevertheless, the available capability for object augmentation with properties offers prototype inheritance commonly found in programming languages such as javascript. The Prototype Property component enables field selection, even with heterogeneous types, which is symmetrical to the Prototype Object's compositional logic. The direction of this experimental modelling process aims towards a declarative and/or functional approach to visual programming where complex high-level queries can be performed on sequences of data.

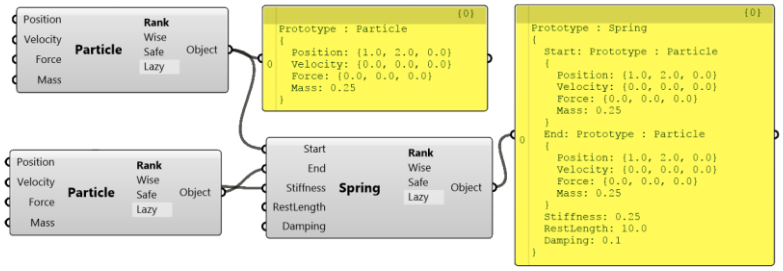

Figure 2. Structured user-defined data type definition using the Prototype Object.

4.2 GEOMETRY

The current set of geometric extensions is limited adding components that are useful in digital fabrication such as plane interpolation for kinematic model definition and pathing; parallel curve offset, polygonization and clipping for machine pathing; bi-arc curve interpolation for geometry rationalization such as singly-bent steelwork fabrication; and principal components plane fitting for bounding box alignment in logistical operations such as part packing and point cloud fitting. There are also tools for 3D scanning data visualization and editing such as large meshes and point clouds loaded from user-defined ASCII databases as well as proprietary formats.

4.3 DIGITAL FABRICATION

The digital fabrication package includes tools for kinematics modelling in terms of both the mathematical and visual representation and for simulation of linkages in typical CNC and industrial 6-axis robotic systems (Figure 3). Relevant work in architectural literature of digital fabrication include (Brau-

mann and Brell-Cokcan, 2012; Pigram and McGee, 2011; Payne, 2012). Models are generated by loading visualization geometries, axial configuration data stored as construction planes, and angle limit vectors stored as annotation from the CAD document. Two kinematic families are currently supported for ABB, KUKA, Denso and Universal robots; and perhaps other robots but have not been yet tested. In addition there are tools for defining end-effectors such as grippers, welding torches and spindles.

Motion planning captures concepts of machine path generation and overall operation modelling. The main mode of modelling is based on an offline paradigm, where the fabrication process is planned, modelled, simulated, compiled and executed with target equipment. The abstraction model of this is based on Tasks which contain one or more machine Operations, such as motions types or input/output signals. Tasks can be simulated individually and merged into Programs. Advanced task such as pick and place shortcut the process of defining explicitly individual operations.

In online mode the model-graph contains communication components that exchange live information with the machinery via the network. This is to enable use of sensors and actuators to dynamically program, operate and control machines. The goal for online operation is twofold: (a) to unify machine programming instead of using multiple incompatible offline formats, such as UrScript, KRL, Rapid, and G-Code; and (b) to enable experimentation with non-prescriptive modes of operation using vision systems and bespoke effectors. The challenge with dynamic coding is in the static and synchronous or rather strongly stateful logic of parametric models expressed as directed graphs which is conceptually distorted by transient event-driven logic.

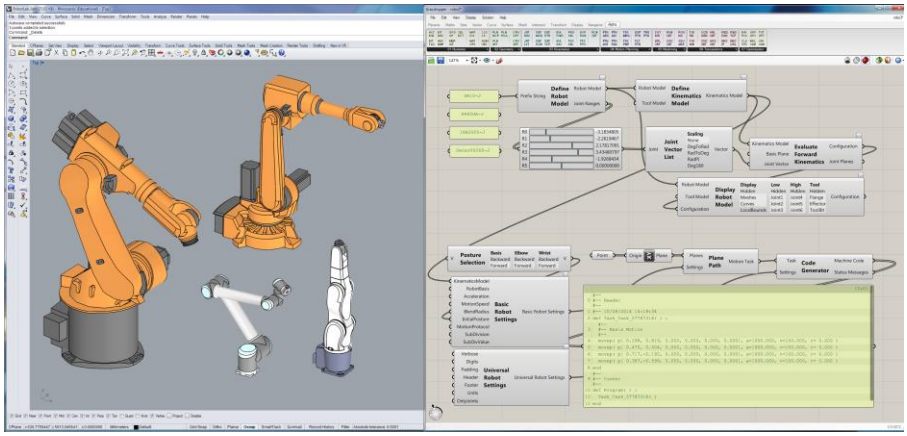

Figure 3. Overview of robotics models and task-oriented machine programming logic.

4.4 OPTIMIZATION ROUTINES

Mathematical modelling primitives and numerical optimization solvers offer the opportunity to embed advanced computation methods as first class entities that is at the same level of abstraction as with regular components rather than meta-processes. There is a lack of numerical methods in computer aided design tools and while they are extensively used to perform environmental and structural simulations they are seldom exposed as general purpose tools. The motivation is not to replace production certified analysis tools but offer programmable interfaces to the first principles for the purpose of research, rapid prototyping new computation methods and education. The methods available address common design and construction topics organized into:

Iterative numerical solvers for univariate root-finding and multivariate minimization problems; envisioned to be integrated in the parametric logic to offer alternatives to purely geometric constructions. Mathematical models expressed by numerics often offer simpler formulations in addressing both linear and non-linear problems. The matrix component embeds a linear algebra kernel of approximately forty different functions including: matrix generation and construction from data sources, geometric transformations, and algebraic operations, including factorizations, analysis and matrix characterization.

Classical information processing methods such as routing, namely solvers for the traveling salesman problem useful for machine planning optimization (Laporte and Udatta, 2002) such as reducing the time required for CNC machines to travel between contours (Castelino et al, 2003) additive manufacturing (Chou et al, 2008); Data clustering methods for analysis (Jain and Stearns, 1965; MacQueen, 1967; Gonzalez, 1985), such as understanding the range of different parts and dimensional variance in a design and optimizing part typologies (Dritsas 2012); Cutting and packing methods (Dyckhoff, 1990; Martello and Toth, 1990) relevant to digital fabrication for control of material use in machining processes (Dritsas, 2012) and efficient arrangement of 3d prints into build volumes.

Graph or network analysis: contains a range of methods for construction of graphs, analysis of such metrics as various centralities and computation of path finding and tree generation. Graph theory offers advanced set of procedural machinery for such as applications as digital design, such as topology modelling and constraint resolution as well as digital fabrication, such machine pathing but also prominently in urban and circulatory network analysis studies (Turner el al, 2005; Porta et al, 2006; Sevtsuk and Makonnen, 2012) and general signal processing.

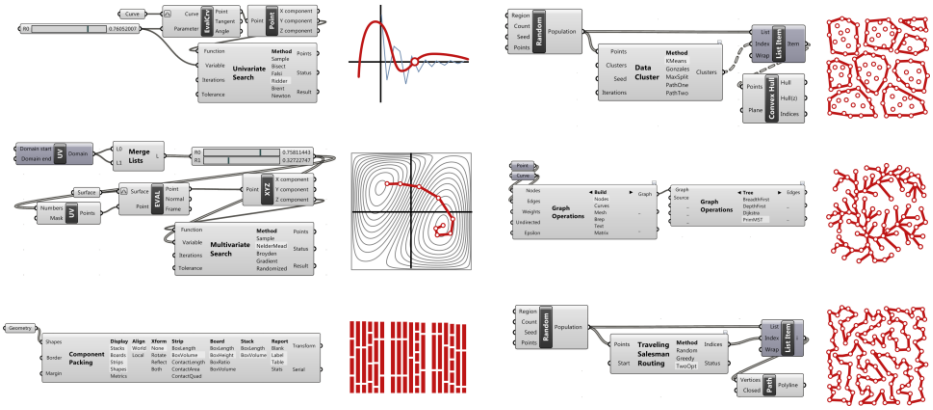

Figure 4. Overview of supported solvers for mathematical modelling and optimization.

5. Conclusions

This paper presented an overview of a new design computation library for advanced modelling aiming to integrate scientific computing methods within parametric modelling systems. The topics raised for broader discourse are: (a) The notion of a model as not only virtual machinery of design production but closer to a science ideas and methods may potentially enrich the current modes of thinking and understanding of complex systems; (b) The need of first class analysis and optimization methods is certainly a topic that requires broader debate as it may address efficient allocation of human and material resources; (c) Embedding advanced methods of analysis and optimization, such as graphs analysis, data clustering, pathing, routing and packing offers some powerful tools for understanding the implications of early digital design actions to fabrication and construction rather more holistically. Finally, we find that while black-box modelling is inhibiting direct exposure to the theory of the underlying method, the availability of advanced methods can motive learning and understanding and it can effectively solve real world design problems.

Acknowledgements

The author would like to express gratitude to the SUTD-MIT International Design Centre, the SUTD Digital Manufacturing and Design Centre and Singapore Ministry of Education for supporting this research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the aforementioned organizations.

References (sample)

Aish, R.:2005, From Intuition to Precision, *eCAADe*, Lisbon, 10-14.

Aish, R.:2013, DesignScript: Scalable Tools for Design Computation, *eCAADe*, Delft, 87-95.

Braumann, J and Brell-Cokcan, S.: 2011, Parametric Robot Control, Integrated CAD / CAM for Architectural Design, ACADIA, 242–251.

Castelino, K., D'Souza, R. and Wright. P.K.: 2003, Toolpath optimization for minimizing airtime during machining, Journal of Manufacturing Systems 22.3, 173-180.

Chou, C.C., Chen, Y.K. and Chou S.Y.: 2008: A fundamental tool path planning problem for circles in layered manufacturing, Integrated Computer-Aided Engineering 15.1, 37-52.

Dritsas, S.: 2012: Design-Built Rationalization Strategies and Applications, International Journal of Architectural Computing, 10-4, Doppler Press, Essex, UK.

Dritsas, S.: 2013: Packing Optimizations for Digital Fabrication, eCAADe, Delft, 655-664.

Dyckhoff, H.: 1990, A typology of cutting and packing problems, European Journal of Operational Research, 44, 145-159.

Garey, M. R and Johnson, D. S.: 1979, Computers and Intractability: A Guide to the Theory of NP-Completeness, San Francisco, CA.

Gonzalez, T. F.: 1985, Clustering to minimize the maximum intercluster distance, Theoretical Computer Science, 38, 2–3, 293–306.

Hartmanis, J. and Stearns, R. E.: 1965, On the Computational Complexity of Algorithms, in Transactions of the American Mathematical Society, 117:285-306.

Jain, A. K., Murty, M. N. and Flynn, P. J. 1999, Data clustering: A review. ACM Computing Surverys, 31(3), 264–323.

Laporte, G, and Udatta P.: 2002, Some applications of the clustered travelling salesman problem. Journal of the Operational Research Society 53.9, 972-976.

Mathworks: Simulink, Simulation and Model-Based Design, last accessed November 2015 http://www.mathworks.com/products/simulink/

Martello, S. and Toth, P.: 1990, Knapsack Problems: Algorithms and Computer Implementations, John Willey and Sons, England.

MacQueen, J.B.: 1967, Some methods for classification and analysis of multivariate observations. Fifth Symposium on Math, Statistics and Probability, Berkeley, PP281–297.

National Instruments: LabVIEW System Design Software, last accessed November 2015, http://www.ni.com/labview/

Payne, A.: 2012, A Five-axis Robotic Motion Controller for Designers, ACADIA 2012, 162–169.

Pigram, D and McGee, W.: 2011, Formation Embedded Design, A Methodology for the Integration of Fabrication Constraints into Architectural Design, ACADIA 2011, 122-131.

Porta S, Crucitti P, Latora V, 2006, The network analysis of urban streets: A primal approach. Physica A: Statistical Mechanics and its Applications, 369-2, 853–866.

Reas, C. and Fry, B.: 2014, *Processing: A Programming Handbook for Visual Designers,*

Second Edition, MIT Press.

Rowe, P.: 1986: Design Thinking, MIT Press.

Rocha, A.J.M., 2004, Architecture Theory, 1960-1980: Emergence of a Computational Perspective, PhD Thesis, Massachusetts Institute of Technology.

Sevtsuk A, Mekonnen M, 2012, Urban Network Analysis Toolbox, International Journal of Geomatics and Spatial Analysis, 22-2, 287–305.

Turner A, Penn A, Hillier B 2005, An Algorithmic Definition of Axial Map, Environment and Planning B, 32, 425-444.