Generative Systems:

Intertwining Physical, Digital and Biological Processes, a case study

Gonçalo Castro Henriques[1], Ernesto Bueno[2], Daniel Lenz[3], Victor Sardenberg[4]
[1,3]Federal University Rio Janeiro, LAMO, PROURB - Brazil [2]Mackenzie University / Positivo University [4]Leibniz Universitaet Hannover, Germany
[1,3]{gch|daniel.lenz}@fau.ufrj.br [2,4]{ernestobueno|vsardenberg}@gmail.com

The fourth Industrial Revolution is characterised by the computational fusion of physical, digital and biological systems. Increasing information in terms of size, speed and scope exponentially. This fusion requires improved, if not new, tools and methods to deal with complexity and information processing. By opening Generative Systems to interact with the context, we believe that they can develop solutions that are more adequate for our time. This research began with a literature review about generative systems and their application to solve problems. We then selected the tools, Cellular Automata, L-Systems, Genetic Algorithms and Shape Grammar, and thought about how to translate these original mathematical tools to specific design situations. We tested the application of these tools and methods in a workshop, implementing recursive loops to open these techniques to interference. Analysing the empirical results made us revise our design thinking, relying on the study of complexity to understand how these techniques can be more context-aware, so we can make design evolve. Finally, we present a comparative framework analyses that interlaces techniques and methods, so in the future we can merge physical, digital and biological information.

Keywords: *generative systems, design thinking, complexity, context interaction, recursion*

RESEARCH APPROACH

To promote Generative Systems in architectural design, we developed a 5-stage process. First, we researched about generative systems and how to apply them. Secondly, we thought about the translation of abstract mathematical techniques to design context. Thirdly, we tested these techniques empirically to gain a better understanding of their capabilities and limitations in a workshop. Fourth, we analysed the tools and methods used and the results for each technique. Results recommended deepening our knowledge about design cognition and complex systems theory. Finally, we synthesised and compared the techniques and design methods.

GENERATIVE SYSTEMS AND COMPLEXITY

The use of generative systems in design, unlike traditional methodology, implies an indirect relation with the final product. Production, rather than being done directly with the "designer's own hands," is mediated by a "generative system" (Fisher and Herr, 2000). To define a generative system, it is necessary to define the abstract set of rules and proceedings that will create a set of objects. If we want this object to make any sense in a certain environment, these rules must be about the object-environment relation. Otherwise, the meaning of the object in a specific context will be a matter of chance.

Generative systems, exploring computational power, can provide unimaginable solutions, expanding creativity. According to Gero (1996), the most common concept of creativity does not consider the ability to develop possibilities during the generation of solutions; only the ability to improve end-product quality. To create multiple solutions, it is necessary to work with methods that explore and record the possibilities of solving a problem, aggregating unexpected and less familiar results usually discarded. Taylor (1972) categorises five types of creativity: expressive, productive, inventive, innovative and emerging. In this context, it is important to emphasise beyond the "emergent creativity", the "productive creativity", that is, the creativity contextualised in the domain of a technique, which allows controlling the project in the environment generated. Architects rely on Design Thinking (Simon, 1969) to help solve ill-defined problems, using mainly implicit design processes instead of linear thinking and with strict criteria. They value the hands-on experience with the tools and their application to find solutions, evaluate and develop them, seldom making any explicit algorithms. We often leave optimisation till the final stages, after defining the form, to improve its performance. However, mathematical optimisation and processes, such as generative systems, have a precise definition and strict procedures to solve problems. So, to understand generative systems, architects must enter the algorithmic process. This requires opening the algorithm "black box", making the mechanisms to produce results explicit, unlike the traditional "hands-on processes", where the final results are felt more directly. In a sense, architects need to cope with extensive processes to expand their abilities.

Composition and Complex Systems

By their education, architects are able to deal with complex problems related with the nature of their work. Beyond "Complexity and Contradiction" (Venturi, 1966), the study of complexity in science has developed to consider complex systems, supported by the evolution of mathematics and computation, a conceptual and technological evolution, intensified now with the fourth industrial revolution disruptive logic (Schwab, 2017).

Complexity is not new in the Design Thinking process. Formerly, architecture was an art in the integrative sense, but, after the Renaissance, it separated into different branches of knowledge in a continuous specialisation process. In architecture, the segregation was not only among disciplines, but between composition and materialisation. Like the architect, the sculptor, the painter, the musician, the chemist and the physicist, all compose (Figure 1). What is the relationship between composition and complexity?

The relationship is closer than one might think. The architect, like other composers, invents the problem - there is no scientific law to describe the context of how an object emerges (Buchanan, 2000 and Stolterman, 2008). Then he formulates circumstantial laws to adapt a methodology that deals with problems beyond linear logic, dealing with implicit subjective techniques, where he intervenes directly. The artist, like the observer in second-order cybernetics, interferes and changes his environment (Dubberly, Pangaro, 2015). When we speak of composition, we use notions such as rhythm, weight, dynamics and static equilibrium, entailing a transition from a part to a whole relationship and vice versa. Therefore, in addition to quantities, the artistic process uses qualities that have implicit measure subjected to an individ-

Figure 1
Composition in different activities: composition as material search in sculpture (Chillida), expressive search in painting (Tapiés 1990), search in mathematics (Fuller 1948), and research in the study of the three-dimensional composition of the double helix (Watson and Crick 1954). Photos in the public domain.

ual learning process. Colquhoun (1989) refers about composition:

"Composition came to mean a creative procedure in which the artist created 'out of nothing' and arranged his material according to laws generated within the work itself ... Form was no longer thought of as a means of expressing a certain idea, but as indissoluble from, and coextensive with, the idea. Composition therefore was able to stand for an aesthetic of immanence in which art became an independent kind of knowledge of the world."

This article does not intend to argue about the dichotomy between art and science, but rather refer to a time when these two areas of knowledge were bound, forming a wholeness. This return to an integrated conception of different branches of knowledge was a necessity in the 50s, a reaction to scientific reductionism, as formulated in the "General Systems Theory". Thus, Integrating knowledge and tools in architecture, promised so much by the introduction of the digital in Architecture, is still to be fulfilled.

Observing the introduction of digital into design, the separation of materialisation and composition or the tool and process dichotomy, seems to persist (Terzidis, 2003). To bridge tooling and the processual use of digital, computerisation and computation, Terzidis proposed Algorithmic Design. The architect's process seems somehow incomplete when he intends to move into the unknown realm of mathematical tools, new "black boxes" he wants to open to incorporate technology and more information. For this, architects rely on exploratory search, a knowledge that has similarities with composition search using analogue or pre-digital generative systems. In this aspect, architects have the ability to invent new realities in other domains (Carpo, 2011). It is precisely because the architect can invent the new that he can contribute to refounding "the science of the artificial" (Simon, 1969). Simon refers to design as indispensable: "The proper study of mankind is the science of design, not only as the professional component of a technical education, but as a core discipline for every liberally educated man."

So, it is not a question of how to apply design to solve complex problems (Herr, 2002). Rather, it is a question of how Design Thinking can (re)invent approaches to Complex Systems. For this, we might remember the characteristics and nature of design problems (Buchanan, 2000). Also, how does artistic composition deal with these new tools? To compose is to establish an order among the parts - even if we do it according to top-down processes (architecture treatises) or bottom-up (rules of composition) - representing an understanding of how things come to be, that is, describing thing also as a process, or method, of design, unifying a set of elements based on the senses and previous experiences. Parallel to art, when science passed from deterministic logic to multi-causal and probabilistic reasoning, it required new tools. In both cases, one can say experimentation is a heuristic process to narrow the computational search space.

Complexity addresses relations among the parts and multiple levels of organisation as well as the relations (and behaviour) of these systems and their environment. Due to its comprehensiveness, Design Thinking can be useful in different areas: a) Visual and symbolic communication; b) Material objects such as products, tools, instruments and machines; c) Organisation of activities and services; d) Complex systems or environments for living, working, playing and learning. (Buchannan, 2000). Applying a synthetic approach to design thinking, in concatenating parts, the whole context is essential for complex systems.

Melanie Mitchell (2009) points out that to understand complex systems, it is necessary to address information, computation, dynamics and chaos, and evolution. In her book, she introduces these subjects in each one of the chapters to set a conceptual framework to define and measure complexity. She measures complexity as: Size, Algorithmic Information Content, Logical Depth, Thermodynamic Depth, Computational Capacity, Statistical Complexity, Fractal Dimension and Degree of Hierarchy. Finally, she concludes, "The diversity of measures that have been proposed indicates that the notions of complexity that we're trying to get at have many different interacting dimensions and probably can't be captured by a single measurement scale."

Complex Systems and Problem Types

According to Weaver (1948), science began by solving simple linear problems with few variables, during the XVII, XVIII and XIX centuries. Then it solved problems of disorganised complexity, with many variables, using statistics. Thus, it solved problems from the atoms to the stars resorting to computation using probability and statistical data. The next kinds of problems to grasp are those of organised complexity, of the mesoscale of everyday life, with more variables.

TRANSLATING GENERATIVE TECHNIQUES

We researched generative systems literature describing the application of these techniques to solve problems, identifying four significant techniques:

Cellular Automata (Neumann, 1951; Wolfram 2002), L-Systems (Lindenmayer, 1968), Genetic Algorithms (Holland, 1975) and Shape Grammar (Stiny, 1980). Each tutor studied one of these techniques along with a group of students, looking for examples and understanding of the type of problems each technique can solve.

We studied the technique's application in practical cases, resorting to the algorithms and descriptions available. For 3 months, a group of researchers identified types of problems, variables and applications, considering the potential and limitations of each technique. This helped to formulate the workshop design problem. References differed in each technique, but we focused on how to implement them in visual programming, testing available applications. Relying on this experience, we prepared a problem and tested it in the Workshop Form Finding and Generative Systems, LAMO, Rio de Janeiro 2017.

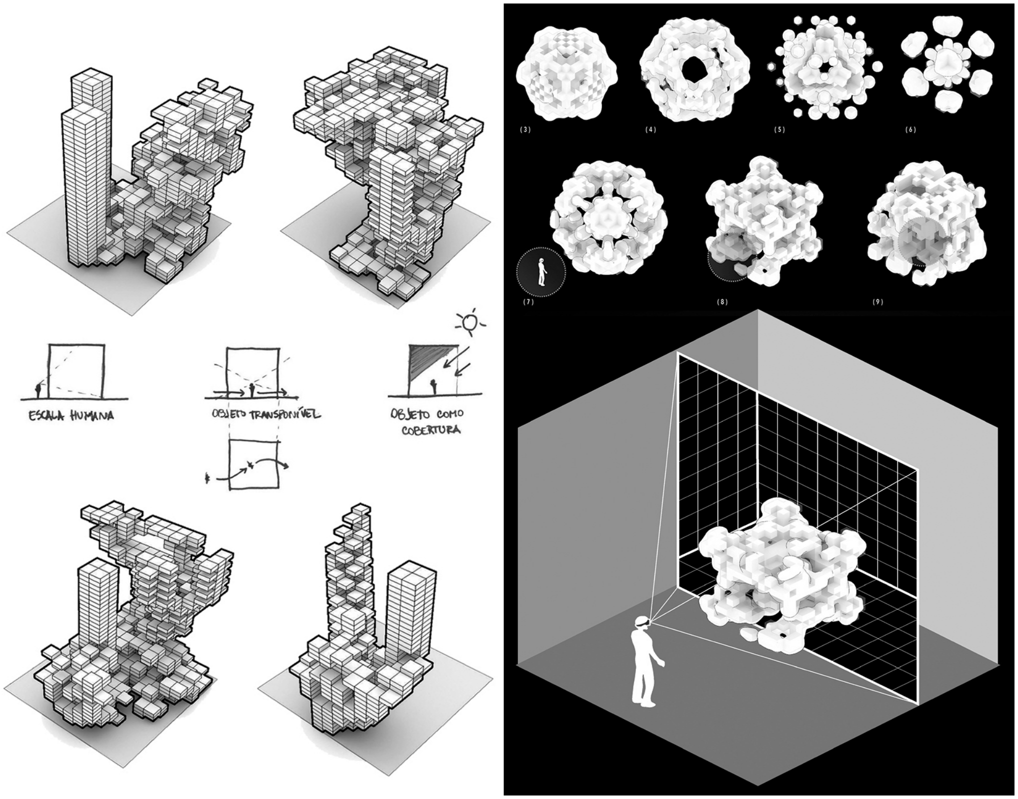
A Problem in Design Context

Preliminary research identified simple solutions to design problems on the mesoscale. We set the space limits between 3×3×3 m and 10×10×10 m, and thought about how to interact with the (few) variables during multiple generation through feedback. We outlined an interaction between the algorithm and the environment to open the "black box". Preliminary research identified similar situations among techniques, variables and contexts. We looked for applications of these techniques to public spaces, shelters, pavilions and constructive systems. By setting a boundary in the methods concerning the tools and the computational processes, we identified a search space and factors that interfered with this search, from mere generic to applied search.

We set up the workshop for undergraduate and postgraduate students, practitioners and professors, of different origins, in eight groups of three. They all learned to apply the different techniques, led by each tutor, and then we carefully selected 2 groups per technique. The participants' diversity of backgrounds and knowledge gave us the certainty to use visual programming with Grasshopper. This allowed implementation of generative algorithms without the drawbacks of learning a textual language, while its open policy enabled access to a great number of software add-ons for experimentation (Bueno, 2016).

CELLULAR AUTOMATA. Cellular Automata (CA) is a system that operates locally, as the state of the neighbours' cells, in each interaction, defines the state of each cell. John von Neumann and Stanislaw Ulam

Figure 2
Cellular Automata (CA) proposals, Estranging The Context, architectural elements that inhabit the movies films, group: Thatilane Loureiro, David Mendonça, Eugênio Moreira (Left), and The Virtual Cocoon a three-dimensional CA that produces a four-dimensional object to be explored in Virtual Reality, group: Nicolle Prado, Isadora Tebaldi, Emilio Marostega (right)



developed CA in the 40s, and, since then, have been applied in a wide range of fields, such as Computer Graphics or Cryptography. Despite CA's ability to create form, it has few applications in architecture. According to intuition, a system with simple inputs using simple rules would produce simple behaviour. CA defies this conception, as it can produce across the range from simplicity to emergent complexity, as demonstrated by Wolfram (2002). CA can produce emergent and complex results.

In the workshop, CA's high degree of unpredictability required introduction of the idea of Form-Making, instead of Form-Finding. This is an alternative to contemporary rational trends that argue that simple quantitative criteria - such as environmental performance, material use, and cost reduction - should determine the architectural form. (Carpo, 2012). We invited the CA participants to analyse the spatial and aesthetic qualities of each configuration, in each recursion, and how they interacted with the system. There is no feasible way to foresee exactly how to change a rule to affect the product. Therefore, designers interact in a ludic way with the system differing from Albertian total control of the drawing. Participants approach the form-making algorithm in a playful horizontal way, supported by previous research about CA and computational morphogenesis to handle complex phenomena (Sardenberg, 2013).

CA Results: The approach diverged from simulating physical reality to focus on design as cultural practice. The first team used CA form-making to produce architectural elements to inhabit cinematographic contexts. Each project was conceived according to a specific genre (i.e. science fiction, comedy and horror), as an installation that responds to the film and changes it simultaneously. The second team, "Virtual Cocoon", used three-dimensional CA to produce a four-dimensional object that interacts with Virtual Reality. Usually we use a series of cubes side-by-side to represent 3D-CA. However, VR enables the user to interact with virtual objects according to his/her movement, in an immersive cocoon.

L-SYSTEMS. LS research focuses on literature, but especially on its technical implementation. LS are symbolic systems capable of generating growth structures, based on the ability to rewrite rules recursively. The biologist, Aristid Lindenmayer (1925-89) invented LS to describe the growth of simple species, such as bacteria and algae. Three concepts are embodied in this technique: (i) the initial axiom or seed, which represents the initial state of the system; (ii) the production rules applied to transform the seed, through (iii) recursion, a repetitive computational method that, in each generation, recalls the previous one.

Originally, LS are formal deterministic systems, as the rules are context-independent. Being deterministic, there is one, and only one, substitution rule for each letter of the alphabet. In order to extend this technique, we searched for methods in visual programming to develop context-sensitive LS, allowing them to evolve from generating a single artificial plant to different species over time. The reference for LS design is the book by Prusinkiewicz and Lindenmayer (1990), and the articles by Fisher & Herr (2000) and Agkathidis (2015). As practical experiments, we sought diverse research, but especially the practical experiments by Coates (2001). While working with Grasshopper, we evaluated different LS addons like Rabbit (Dimitrova, 2016), which has an interpreter that translates the code into turtle graphics. We found that this interpreter accepts only letter symbols as input and is deterministic. Therefore, we looked for recursive loop applications that enabled non-deterministic or stochastic LS. Hoopsnake, cited by Sedrez (2013), was not very intuitive and had limitations. We also tested Loop (Turiello, 2013), and, finally, Anemone (Zwierzycki, 2015). The latter is the most intuitive; and during loops maintains the data structure and has an expandable list of data paths, enabling parameter value change in each generation. Anemone enabled response to external factors in each generation, implemented with turtle movement.

LS Results: The first proposal developed in the workshop was "Menger Revisited", and the second, "Hugging Trees". The possibilities explored are complementary. The first revisits the Menger sponge, proposing interactive, recursive transformations: the fractal changes according to the user, in a virtual space. It went beyond the classical Menger sponge by breaking the symmetry in unpredictable, yet relational, ways, without losing self-similarity. The second proposal develops tree-like structures that reconfigure themselves according to human position, changing the branch angles and planes of rotation accordingly. The percentage of randomness that affects tree movement is associated with external factors. When the user reaches the centre, the trees embrace him/her. The first solution has potential for VR and can output data for augmented reality. The second, despite following tree stereotypes, possesses a feasible mechanism for a physical articulated structure.

GENETIC ALGORITHMS. A finite set of rules and operations defines GA that simulates the combination of individual characteristics of the same species, to select those with the best environmental fitness. The algorithm's structure considers the main mechanisms present in the evolution of species as genetic inheritance, random variation (crossover and mutation) and natural selection.

We relate GA with the broad use of engineering to optimise structures and components. The use of

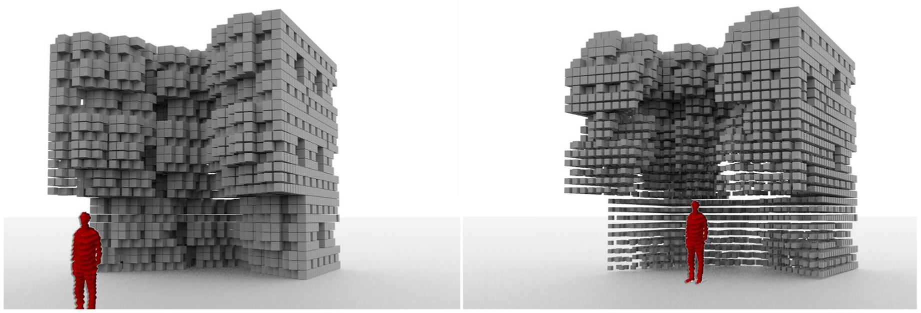
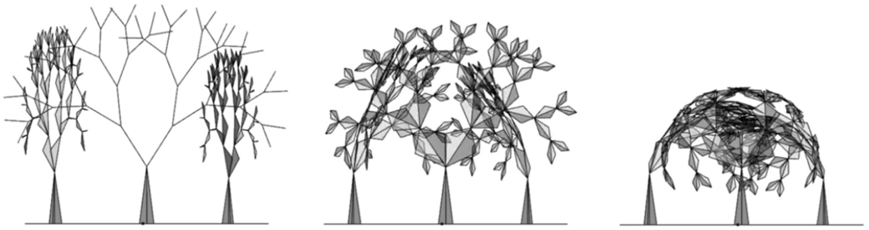



Figure 3
Menger Revisited, showing recursive system variations triggered by user proximity. Group: Fernando Lima, Aurélio Wijnands, Maria Eloisa. Hugging Trees, showing the fractal movement of trees as the user approaches. Group: Núbia Gremion, Erick Bromerschenckel, Daniel Wyllie.

GA in design is frequent in spatial planning optimisation and form generation. Although it is an optimisation method, we can use it as a generator mechanism to assist the designer in exploring the solution spaces, to obtain creative, emergent and unexpected results. In this context, the algorithm is set to obtain favourable solutions independently of the optimisation level attained, guaranteeing a flexible choice among the solutions generated. The main references are Holland (1995) Bentley (1999) and Mitchell (1999). Grasshopper has a genetic solver, Galapagos. However, the number of design problems it can tackle is limited (Rutten, 2013). We used the add-on, Biomorpher (Harding, Olsen, 2018) that allows the designer to interact with the algorithm during its execution. The designer selects the "parent individuals" from quantitative (optimal) and qualitative (aesthetic) criteria, directing the evolutionary process, as the "individual parents" will be crossed to generate "individual offspring".

GA results: The projects developed were "Evolutionary Aggregation" (EA) and "Dwell Debris" (DD). They adopted as criteria: shade, contact with the ground and formal arrangement. As a strategy for structuring the GA, EA distributed the components in a regular 3D grid, ensuring modularity and orthogonality. The DD authors opted to anchor components in a random cloud of points, creating an irregular arrangement. In both situations, the teams subtracted the original grids with additional voids, but using different components. EA defined a single component with freedom of rotation on the central axis. DD defined four components that rotate on their axis, all generating a diversity of solutions. EA programmed the GA to find solutions that had the largest contact among components and the largest projected shadow. The DD team searched for the greatest number of intersections among components, the greatest volume on the ground and the least shade. The participants used the add-on, Biomorpher to cause evolution of formal solutions with human intervention.

SHAPE GRAMMARS. SG uses encoded abstract formalisms that limit its use to specialists. These formalisms also limit free interaction in the form of de-

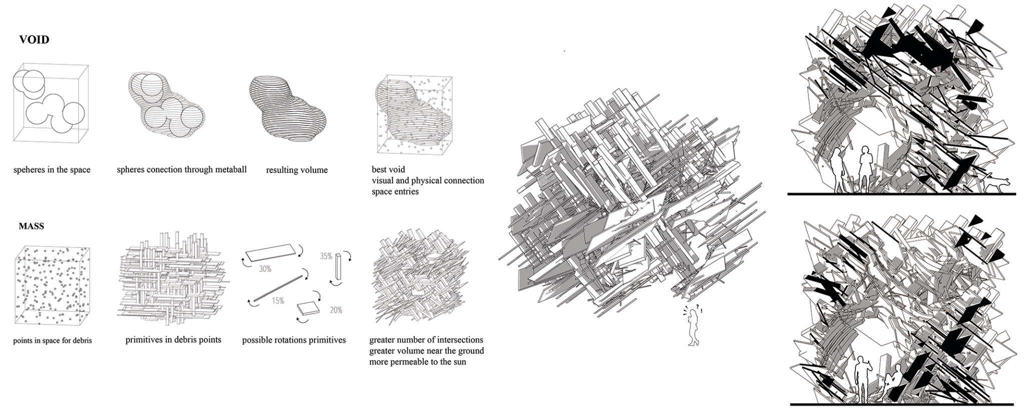

VOID
speheres in the space
spheres conection through metaball
resulting volume
best void
visual and physical connection
space entries
MASS
points in space for debris
primitives in debris points
30%
35%
15%
20%
possible rotations primitives
greater number of intersections
greater volume near the ground
more permeable to the sun

Figure 4
Group DD strategy and solutions. Group: Anael Alves, Loan Tammela, Felipe Lannes. Group EA strategy and solutions. Group: Luciana Gronda, Igor Machado, Monique Cunha, Wellida Coelho.

velopment. SGs have different natures as descriptive and generative grammars (Garcia, 2016). To bridge this gap, we developed a more intuitive interface in visual programming, relying on earlier Stiny texts to structure the definition. We identified a recurrent description that we organised into the triple ordinate:

$$G = (v, r, d)(1) \qquad (1)$$

where G is grammar, v vocabulary, r rules set, and d derivation. The algorithm concatenates three sets, maintaining each variable's internal independence and the set's co-ordination. The loop uses v as input, applying the r rule indicated in d, accumulating the result in v. While v and d are simple lists, r is more complex. The initial axiom has two shapes, setting the initial and final form, memorising the Euclidean transformations, and then applying them as input shape. As holdbacks, the definition operation cannot recognise the shapes used. The next step would be recognising emergent forms. The algorithm's simplicity encouraged participants to develop their own definitions.

SG Results: The first team used SG that relates a façade composition with music notation, while the other designed a shelter. They adapted the initial algorithm, exploring the generation of diverse designs. If we know the result of a rule application, it is easier to predict the emerging phenomena. Thus, in the SG definition, we knew the initial and final form, which facilitated the interaction. The derivation played a fundamental role in SG. However, SG teaching focuses on shapes and rules, how to change the derivation that needs to incorporate uncertainty that remains obscure. Full randomness probably leads to meaningless design, and some randomness can still generate meaningful results. This requires encoding the derivation data, organisation (data tree) and the random derivation paths.

RECURSIVE LOOPS

Beings evolve interacting with the environment in a continuous process in time. For generative systems to evolve, in addition to interaction and selection mechanisms, we need recursion. The creation of such repetitive cycles was limited to those mastering formal computation. By introducing recursion in different design situations, using visual programming, we pointed out how systemic interaction can generate evolution.

TECHNIQUES AND INTERACTION

The interaction between CA rules and the context happens after the form generation, when the user can interact in virtual reality with the CA. In LS, we open the system boundaries to accept interference

Table 1 Generative Techniques in Design (CA, LS,GA,SG): Purpose, Scope, Challenges, Outcomes and Interference. Interference is based on the results of the empirical tests of the workshop, envisioning future possibilities.

Generative Design Techniques

	Purpose	Scope	Challenges	Outcomes	Interference
CA	Creation of multi-dimensional objects and a (large) variety of results that can be context-sensitive.	Form-finding and form-making; Growth patterns; Self-replicating machines; Social application, data models.	Relation between the local rules and global results. Unknown or undefined system boundaries. Predictability.	Emergent spatial arrangements and typologies that can be context-sensitive. Undesired, strange or chaotic results.	Interference achieved in virtual reality. Possible Interference with growth processes that relates local-global results.
LS	Generation of form, structure and patterns. Spatial layouts. Morphological designs.	Symbolic nature (abstract rules) Closed set-rules: deterministic. Open set-rules probabilistic or stochastic.	Generates fractal patterns, of a graphical nature that are in a closed system. Difficult preview /change global form.	Spatial arrangements, roads, networks and terrains; Evolutionary systems through interaction.	Achieved recursion in-cycles. Control dynamic structures (Digital, physical, but can be also biological).
GA	Optimisation; Improve design; Multiple design alternatives. Meeting fitness criteria.	Combinatorial and morphological designs. Disruptive innovation.	Difficulty to translate problem into representation. Fitness function, genotypes phenotypes, requires experience.	Optimised, solutions. Multiple design alternatives. Component-based combinatory solutions. Might cope CA,LS,SG	Interference with each generation, associating quantitative (numerical) and qualitative criteria using Biomorpher.
SG	Generation or description of form using rules.	Shape nature (shape rules) Rules to construct a family of forms.	Formal language difficult to master. Self-centred form definition. Difficult use non-probabilistic reasoning form-finding	Exploratory geometrical design; Solutions for complex combinatory problems. Ruled based design.	Initial and final form set, so interference only rules derivation. Possible interference using cumulative rules and evaluation.

between the user and the context, in every generation, using scholastic algorithms. In GA, we overcame the limitation of a blind system of blind optimisation by allowing the user to interfere in each generation, introducing qualitative criteria that change evolution. In SG, we broke the closed system, allowing the user to affect, not only initial and final form, but also interfere with the derivation rules.

Visual programming contributions

- A cellular automaton including a library of context-specific objects;
- 4D-CA with VR visualisation;
- Responsive stochastic L-systems;
- Multi-objective evolutionary optimisations (geometric max shade optimisation), possibility of qualitative decisions during iterations;
- Automation shape grammars (Grasshopper);
- A 3D graphical music grammar.

However scarce, the workshop results reveal the gap between tools and processes. Singh Gu (2012) compared generative tools to integrate them in a "Computational Design Framework", a valuable contribution to research. However, besides the literature survey, is necessary to test them in context. We used empirical knowledge to fuel design thinking about both tools and design processes. Above we present a synthesis, explaining how we interfere with the tools rules, envisioning future possibilities.

GENERATIVE TECHNIQUES AND DESIGN RESULTS DISCUSSION

The techniques described in this article have existed since the 80s, but their implementation is slow. Previous formal approaches failed to translate the tools into design methods. We proposed a framework grounded on Design Thinking and complexity, to implement contextual interference with recursive cycles. Finding an alternative to formal computation, using loops in visual programming, we show how to interact with tools using recursion. Anemone proved adequate for recursive processes, in both data replacement and incremental growth, allowing designers to interfere between the cycles. Biomorpher demonstrated the value of human interaction with an evolutionary process, working as input of qualitative criteria. We found recursion and selection mechanisms have potential to develop systems that may evolve in the desired direction using Visual Programming. However, this requires technical improvement of the loop processes and further development of the solution selection mechanisms.

With the increase in Information. the problem space augments. With the introduction of interaction in time, with a selection mechanism, and enabling the system to learn in the future, we expect to enlarge the solution space. This would follow the Von Neumann self-replicating machines that can extend AI. Finally, the generative tools are several algorithmic methods, with their specificities. We must tame these algorithms and go beyond the dichotomy between techniques and methods (tools and processes). Only then can these algorithms be absorbed, both by the design thinking and by the maker culture.

Facing the fourth Industrial Revolution, this fusion of techniques reinterprets old methods: by introducing direct interaction and an increasing number of context-related variables, generative systems can deliver responsive designs. Generative methods can offer a symbiotic partnership between the designer and the system to expand architectural abilities skills. In the light of the fourth revolution, they can intertwine information, whether digital, physical or, in the future, biological.

Form Finding and Generative Systems

LAMO Seminar/Workshop Form Finding and Generative systems, Rio de Janeiro Federal University, 28 Aug to 7 Set 2017, Coordination Gonçalo Castro Henriques, Andres Passaro and Elisa Vianna. Tutors generative tools: Victor Sardenberg, Ernesto Bueno, Gonçalo Castro Henriques, Jarryer de Martino and Daniel Lenz. Our gratitude to PROURB for supporting the event and to all participants and collaborators, we cannot name here due to space limitations.

REFERENCES

Agkathidis, A 2015 'Generative design methods implementing computational techniques in undergraduate architectural education', *eCAADe 2015*, pp. 47-55
Buchanan, R 2006, 'Wicked Problems in Design Thinking', *Design Issues*, 8, pp. 5-21
Carpo, M 2011, *The Alphabet and the Algorithm*, Cambridge, Massachusetts, MIT Press
Fischer, T and Herr, CM 2001 'Teaching Generative Design', *Conference on Generative Art*, Milan, pp. 1–11
Garcia, s 2017, 'Classifications of Shape Grammars', in Gero, J (eds) 2017, *DCC*, Springer, pp. 229-248
Gero, JS 1996, 'Creativity, emergence and evolution in design', *Knowledge-Based Systems*, 9, p. 435 448
Harding, J and Brandt-Olsen, C 2018, 'Biomorpher: Interactive evolution for parametric design', *International Journal Architectural Computing*, 16(2), pp. 144-163
Holland, JH 1995, *Hidden Order: How Adaptation Builds Complexity*, Basic Books
Martino, JA 2015, *Algoritmos evolutivos como método para desenvolvimento de projetos de arquitetura*, Ph.D. Thesis, Universidade Estadual de Campinas
Mitchell, M 2009, *Complexity: A Guided Tour*, Oxford University Press
Rutten, D 2013, 'On the Logic and Limitations of Generic Solvers', *Architectural Design*, 83, pp. 132-135
Sardenberg, V 2013, *Processos Emergentes em Territórios Informais*, Master's Thesis, Mackenzie University
Schwab, K 2017, *The Fourth Industrial Revolution*, Penguin Random House.
Sedrez, M and Martino, J 2018, 'Sistemas generativos', in Celani, G (eds) 2018, *Arquitetura contemporânea e automação: Prática e reflexão*, Probooks, pp. 25-28
Simon, H 1969, *The sciences of the artificial*, MIT Press
Singh, V and Gu, N 2012, 'Towards an integrated generative design framework', *Design Studies*, 33, pp. 185–207
Stiny, G 1980, 'Introduction to shape and shape grammars', *Environment and Planning B*, 7, pp. 343-351
Terzidis, K 2006, *Algorithmic Architecture*, Routledge
Venturi, R 1996, *Complexity and Contradiction in Architecture*, The Museum of Modern Art, New York
Weaver, W 1948, 'Science and complexity.', *American scientist*, 35, pp. 1-12
Wolfram, S 2002, *A New Kind of Science.*, Wolfram Media