pages: 11 - 38

GENERATIVE AND EVOLUTIONARY MODELS IN THE DESIGN OF ARCHITECTURAL FORM - INSIGHTS FROM HISTORY

Krystyna Januszkiewicz*, Natalia Paszkowska-Kaczmarek**

West Pomeranian University of Technology, Faculty of Architecture, ul. Żołnierska 50, 71-210 Szczecin, Poland
* E-mail: krystyna_januszkiewicz@wp.pl, ORCID: 0000-0001-6880-0862
** E-mail: npaszkow@gmail.com, ORCID: 0000-0002-3161-5417

DOI: 10.24427/aea-2023-vol15-07

Abstract
This paper explores the possibilities of the use of computer-aided design models focused on imitating the works of Nature, its form-forming processes and behaviors. Tracking the development of the cybernetic models aimed at architects, the achievements of John H. Frazer and his team of scientists are presented. These are the first working morphogenetic models addressed to architects that use generative and evolutionary tools in search of new architectural forms. Models and design strategies developed between 1968 and 1995, including the Reptile System, the Interactivator and the Janssen Model, are presented. The IT solutions used in them provided the basis for the creation of modern computational tools coupled with digital technology.

Keywords: architecture; design; computer; generative; evolutionary algorithms; form-forming process

INTRODUCTION

The creative potential of information media has opened up a new dimension in the design of architecture, especially oriented to imitate the works of Nature, its formative processes and behavior through generative and evolutionary design tools. Today, the instrumentalization of these processes is changing the approach to the design of buildings by bringing designers closer to creating architecture that is compatible with the natural environment, not only visually, but also in terms of acting like a living organism. The growing current interest in imitating natural processes through computer tools rocks the analysis of primitive scientific experiments, making it possible to synthesize information to construct a coherent description of events that took place in the 1970s and set the directions of modern research. Thus, the cognitive scope of a specific slice of reality is expanded, which is the primary goal of scientific research, thus filling a gap in the theory and history of architecture in the second half of the 20th century.

Generative design tools are what are often called morphogenetic tools. They originate from the sciences and are used to produce 2D and 3D patterns and forms with complex geometry. They are mathematical models that describe states or phenomena that occur in the natural world, although they can only be a mathematical operation. Their name (Latin: generare - to give birth) refers to such methods of applying mathematical symbols and relations, which are used to produce states of increasing complexity corresponding to established rules*.

Synthetic evolutionary algorithms, on the other hand, mimic the mechanisms of evolution, the same

*A mathematical model is a finite set of mathematical symbols and relationships and strict rules for operating with them. They refer to specific elements of the modeled fragment of reality. Modeling is used to learn about a given process by replacing it with a simplified layout that reflects selected characteristics of the process. The mathematical description of the model is presented in the form of a system of algebraic or differential equations. See: Jakub Gutenbaum, Modelowanie matematyczne systemów, wyd. III, Instytut Badań Naukowych PAN, Warszawa, 2003.

ones found in biology. It is a set of methods and techniques that includes not only genetic algorithms, but also genetic programming and evolutionary strategies. Evolution, is now an indisputable scientific fact, documented by evidence derived from many fields of science. The essence of evolution is the combination of random (undirected) changes in genotype with strictly directed environmental pressures.

It proceeds according to the following general principles [10 AE 2009]:

- The genotype of an individual undergoes modifications during reproduction. These changes can result either from small, random mutations, or from mixing (crossing) traits of parental individuals;
- Changes in the genotype cause changes in the phenotype of offspring individuals, which affects the degree of their adaptation to the environment (subject to evaluation by means of a goal function);
- Changes in the genotype are random in nature. Changes favorable to an individual occur as often as unfavorable or indifferent ones;
- Individuals are evaluated by comparing their adaptation to a given environment. Those that are better adapted are more likely to reproduce;
- Less-adapted individuals succumb during competition for limited environmental resources and perish;
- Changes (mutation, crossover) are subject to the genotype of the individual, while selection is subject to phenotypes.

Biological evolution drives morphological diversity through genetic variation and results in high levels of adaptation, efficiency and resource efficiency. Therefore, a synthetic evolutionary algorithm modeled on biological evolution is applied to optimization tasks and modeling

Generative design tools can be divided into two groups [N. Paszkowska-Kaczmarek 2022, p.117]:

- Tools that mimic the formative processes found in Nature: Cellular Automata, L-systems, evolutionary and genetic algorithms;
- Mathematical tools-objects (specific type of sets) such as: Fractals, Voronoi Diagrams,
- Grammars of of shape, can describe the geometric results of natural formative processes.

In the design of architectural forms, the tools of the two groups are often complementary. They include techniques, both digital and computational models that are used to derive and transform form, raising it to a higher level of complexity, both formally and structurally, in an effort to obtain high environmental performance. The use of generative tools and computational models that mimic natural formation processes in design practice still requires the integration of methods and techniques and tools that work together associatively in a CAD/CAM system.

Although the basic computational models of natural formation processes were developed before the computer revolution, their implementation into computer systems was gradual, with the development of computational capabilities and computer memory.

Tracing the development of generative tools aimed at the architect, it is difficult to overlook the achievements of John Fresher and the team of scientists he led. Thanks to them, the second half of the 20th century saw the development of the first action models that use generative and evolutionary tools that can be used in the design of architectural form.

1. SEEDING AN IDEA

Between 1968 and 1995, the first design models were developed using generative tools and evolutionary tools. Each model defines a set of tasks to be performed by the design team, and in each case one of the tasks requires generative or evolutionary design tools. These models matched the capabilities of the hardware and the state of the art in computer science at the time. They can be described as follows:

- Generative design models are used to generate a large number of design alternatives that vary widely. The computer systems developed for them define a complex growth process that transforms the coded seed into a design. By making slight modifications to the transformation process or just the shape of the seed, alternative designs can be generated
- Evolutional design models were used to develop designs adapted to their environments. These models relate to the neo-Darwinian model of evolution by natural selection. The computer system then enables a cyclic process in which populations of design shapes are constantly manipulated to ensure that the population as a whole gradually evolves and adapts.
- Generative - evolu tional design models take the evolutionary process of nature as the process of creating architectural form. Such computer-based evolutionary systems consist of a cyclic process that manipulates the entire population of design shapes. Here, the generative system uses code scripts of instructions to create computer models of alternative designs. These designs are used to simulate the creation of prototype forms, which are then evaluated based on their performance in the simulated environment. By mutating and

manipulating the code scripts, new forms are generated. There can be multiple evolutionary stages processed in a short period of time, and the emerging forms are often unexpected.

The aforementioned design models and their corresponding generating systems were based on a common strategy called "seeding an idea" (for a project), or initial configuration, which John Frazer explained as follows [J.H. Frazer 1995]:

"If architectural concepts developed by designers could be captured and codified in a generic form, the generating system could invoke them to generate designs that embody those concepts. This approach to capturing and codifying architectural concepts is referred to as idea seeding." [J.H. Frazer 1974, p. 231].

When adopting an "idea seeding" strategy in a design model, three tasks should be defined:

- codification of generative concepts where a set of generative rules is defined that can transform the seed of a concept into a design;
- codification of architectural concepts where the seed of an idea is defined, which already contains certain architectural concepts;
- project generation where projects are generated in response to the project environment (which includes both context and criteria), which requires a system and tools that generate.

It should not have been assumed that the tasks identified by such a design model were mutually independent. Nevertheless, in most cases they developed in parallel.

The above diagrams summarize the "idea seeding" strategies for the design model and for the generating system. For the design model, the diagram (Fig.1a) identifies specific tasks to be performed by the project team, with each task requiring specific inputs and resulting in specific outputs. Input and Output outputs are shown in oval frames, while tasks are shown without frames. In contrast, the diagram (Fig.1b) shows a schematic of the generating system required by a design model based on an idea seeding strategy. In this case, information defined independently of the system is shown in frames, while information contained in the system is displayed without such frames [J.H. Frazer, J.M. Connor 1979].

A generating system based on the initial configuration or "idea seeding" is not itself a cyclic system. The system generates a single form proposal from a single seed in response to the design environment. However, the premise was that the designer explores a range of design possibilities, making small generative modifications to either the germ of the concept or the generative rules. The result is a cyclical process led by the designer. The first attempt to implement this approach was the Reptile System developed, in its first version in 1968 [J.H. Frazer 1974].

1.1. REPTILE 1968-1974 Generative System

The Reptile generating program and system, developed between 1968 and 1974 by British architect and scientist John H. Frazer, made a breakthrough in computer design methodology and building thinking. The Reptile generating system was able to create a wide range of multi-space roofs from just two basic structural units. These units could be positioned in 18 different ways relative to each other, providing more than three hundred combinatorial possibilities.

Manual drawing of objects, especially those composed of repeated components and their perspective views, necessitated the development of a computer program to facilitate the process. However, in 1967 the capabilities of computer hardware were limited, not only in terms of speed and memory, but also in terms of output graphic representation. In 1971, the Reptile System was enhanced with additional features, and the generating system was already capable of semi-

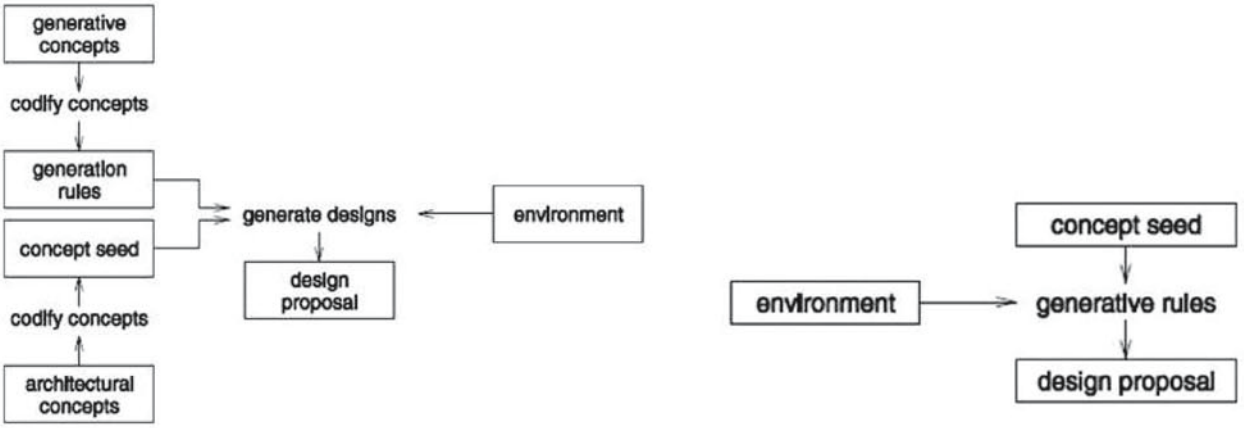


Fig.1a-b. "Seeding the idea" model, a) design model, b) computer generation system, diagrams; source: J.H. Frazer, P. Janssen.

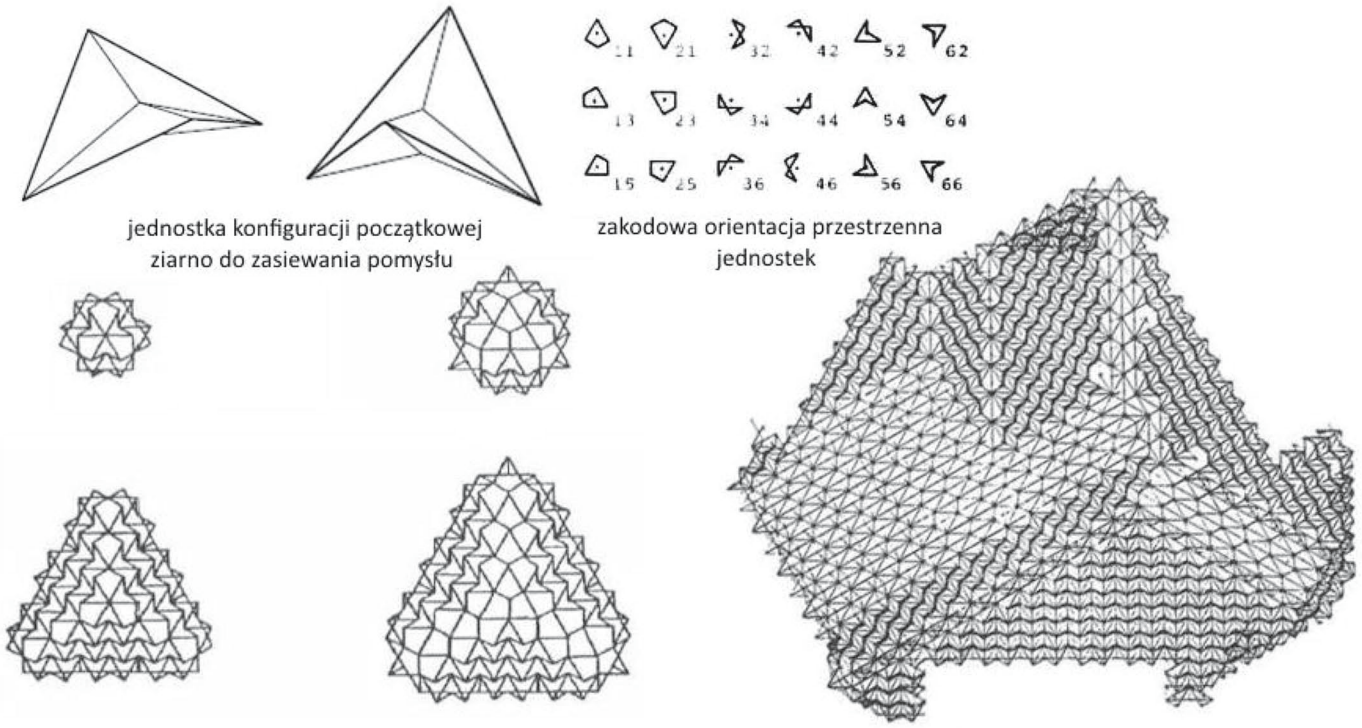


Fig. 2a-b. Generative Reptile program, 1967-1968, (a) overgrowth forms developed from two different seeds, (b) overgrowth of an object generated from a single seed unit (plotter printouts); source: J.H. Frazer, P. Janssen.

automatically producing the forms of complete architectural objects (Fig. 2).

The basic data structure was created and handled by a series of machine code subroutines and functions (written as binary numbers directly for execution by the processor). These allowed unit descriptions to be retrieved, deleted or updated, and additional units to be entered. The seed for seeding here was a minimal, closed configuration of units that included all possible orientations, but not necessarily in all possible combinations. Site development was initiated by spreading the data describing the structure of the units that make up the selected seed in chained units, from the top of the seed downward in a clockwise direction. The location and orientation of the units were determined by four integers. The first two numbers specified the location on a two-dimensional grid with axes at an angle of 60^0; the third integer specified the depth or level of the unit in the structure, and the fourth the orientation of the unit. Such a description of a unit, with an indication of the next unit in the chain, was defined as a genetic code script [J.H. Frazer 1974].

The configuration of units in the seed had a significant impact on the final, form of the object. In contrast, differences in information about the type and orientation of the initial unit and the units adjacent to it in the data chain affected the generating system's choice of procedures for filling the data chain with units. This is analogous to the difference between the transition rules in Cellular Automata and the initial (seed) configuration. The first limited version of the program used only two seeds, the node (containing 42 units), and the second version the star (containing 72 units) [J.H. Frazer 1974]. A generalized, component-based version of Reptile was later refined so that the program required two types of information:

- conceptual model of building information in its minimum coded configuration;
- description of the actual components and details of the output stage.

The description of an object's initial (seed) configuration was not necessarily going to be a building component, but more often a set of components in a key configuration, such as a corner or a change of direction. As in the first version of Reptile, all the information needed for new components in the data chain came from the type, location and orientation of the seed-forming component and those with it in the data chain at the point where the move or change of direction occurred. A new feature was that the program took dimensional coordination into account only when it was relevant to the concept. It did not rely on modular coordination or a grid, but calculated the distances between the location points of elements in the data chain

to determine the modules appropriate for the design under consideration. In addition, the idea of "seeding an idea" to produce different types of buildings was extended to the concept of mutating "seed" details to generate variants that met the requirements of different types of environments, especially when it came to standard building structures. The individual mutations could be created interactively and stored as variants, which were referenced by an additional set of digits in the item description in the data chain. Thanks to these extensions, it was possible to develop large-scale bar structures and create new ones composed of multiple components [J.H. Frazer 1974].

Reptile elaborated - John H. Frazer (architect), Richard Prakins (programmer) and Francisco Guerra (research assistant) on The Cambridge University for Cambridge Atlas Titan computer working with plotter PDP7. Reptile being the first generative program aimed at architects constituted a new design model and methodology, which was developed in the following years [J.H. Frazer 1979].

2. THE EVOLUTIONARY MODEL OF NATURE IN THE GENERATION OF ARCHITECTURAL FORM 1980-1995

In the 1960s and 1970s, the evolutionary approach to design was more in the domain of engineering than architectural design. It allowed finding the optimal solution in response to computationally clearly defined selection criteria. Typically, evolutionary systems allowed for the initial definition of an already existing design, parameterizing those parts of the design that were thought to need improvement. In the second half of the 1980s, a Frazer-led research team at the University of Cambridge was already conducting fledgling experiments on improving hardware performance in terms of modeling and virtual representation, so that the computer would become the designer's "generative toolbox." In addition, there has already been research on communication modeled on the synapses of neural networks [J.H. Frazer 1995]. It may be recalled that in the late 1970s the first processor giving support to graphics systems was made available, and the first microprocessor for computer graphics was made available in 1977. This led to the development of the first graphics program in 1978 General Purpose Pattern Program (GPPP) along with its 3D version (GPPP3) [F.N. Krull 1994]. GPPP3 had functions that allowed the generation and transformation of geometric (rectilinear) objects, random selection and combinatorial operations, and animation. Curve fitting and spline-in curve modeling functions were soon added. The same year saw the development of Shape Processor Language (SPL), a graphical language with an extended data structure. Drawings generated by the Reptile system containing more than 2,000 structural units and 24,000 vectors could already be executed. However, the 8K Commodore Pet computer on sale at the time did not have adequate graphics drivers for plotters so that vector graphics could be plotted directly [J.H. Frazer 1995].

Computer modeling, in all cases, depends on the structure of the data, while graphical representation depends on how the data is transfomed. It is the process of transformation that determines the potential of the computer system model. In addition, the computing power of the computer, the structure of data storage and processing, and hardware solutions are important. It should be recalled that the data structures and interfaces of digital CAD systems were first designed to develop the geometry of the form, not the geometry of the relationships between components, which affected the integration of computational tools.

In the early 1980s, the first concept of "plastic" modeling was developed in opposition to the rigid implications of solid modeling. The idea was to extract higher-order information from the user about the relationships between elements, not just geometric coordinates. At the same time, links between computer graphics and means of production are also being developed.

Beginning in 1983, connections between graphical programs and computer-controlled equipment such as lathes and milling machines were developed. In 1985, the first educational solid modeling program with an easy-to-use graphical interface was also developed. Released in 1987, the program allowed interactive manipulation of three-dimensional forms and geometric modifications using a three-dimensional cursor. The user interface was called a "virtual workshop" because the program allowed the development of files for fabrication [J.H. Frazer 1995].

2.1. Synthetic evolutionary model and Universal Constructor system

John Holland's 1975 publication of a genetic algorithm in a high-level programming language (the syntax and keywords are intended to make the program code as easy as possible for humans to understand, such as Fortran) opened a new debate about imitating Nature through art, and especially about artificial life in architecture.

In the late 1980s, a growing pro-environmental consciousness supported the quest for new forms for the built environment based on patterns drawn from

Nature's creations. Research, in various fields, thus concentrated on biomimetic studies and experiments to find solutions that would make built forms as spatially, energetically and materially efficient as biological forms. In Nature, the information about the form that evolves is encoded only genetically, while selection involves the expression of this encoded information in the external form of the organism. Genetic codes produce instructions for the development of form, but the precise expression of these instructions is environmentally dependent. The model of a form with artificial life also involves a code that produces instructions that are dependent on environmental influences, just as in the real world, only that it is a code-script that evolves.

A typical evolutionary design model requires focusing on two tasks such as:

- codification of the parametric model and evolving projects, i.e., development of rules of mapping, which specify how parameters should be assigned to the parametric model and evaluation rules that determine how the model should be evaluated;
- projects evolve in response to this environment, and the generative system produces alternative designs [J.H. Frazer and J. Frazer 1996].

The evolution model developed in this way (Fig. 3) then required that the parametric architectural concept be written as a "genetic code." This code would be mutated and evolved by the generating system as a series of forms in response to the simulated environment. These forms would then be evaluated, and the code of selected models would be reused until a suitable form was selected for prototyping in the real world. Achieving such an evolutionary model required defining: the genetic code (script), rules for developing that code, mapping the code onto a virtual parametric model, as well as defining the model development environment and selection criteria [J.H. Frazer and J. Frazer 1996].

A typical evolutionary generating system, on the other hand, ensured that mapping rules would create forms from a coded set of parameter values by inserting those values into a parametric model. The evolutionary system evolves these parameter values. This is defined as convergent (convergent) evolution by natural selection (J.H. Frazer and J. Frazer 1996).

Convergent evolution by natural selection is not the only possibility. In a book titled *The Origin of Species* Darwin writes about the technique of artificial selection used by breeders of racehorses or dogs [Ch. Darwin 2003]. In this model, artificial selection by the designer or user opens up the possibility of demonstrating preferences. It can be useful as a way to deal with ill-defined selection criteria, particularly concerns about usage. It also provides an opportunity for the designer to use his or her experience and intuition to achieve faster results.

Nature also relies on divergence to keep a diverse gene pool active in order to cope with sudden changes, such as increasing predator success or changing environments. The model proposed by John Frazer's research team also provided a divergent evolutionary process for generating alternative ideas. This provided a matrix of four possible combinations of natural/artificial selection and divergent/convergent evolution.

Evolutionary models and methods (like Frazer's model) are often based on techniques such as the use of genetic algorithms developed by John Holland. Holland saw the genetic algorithm as a direct analogy

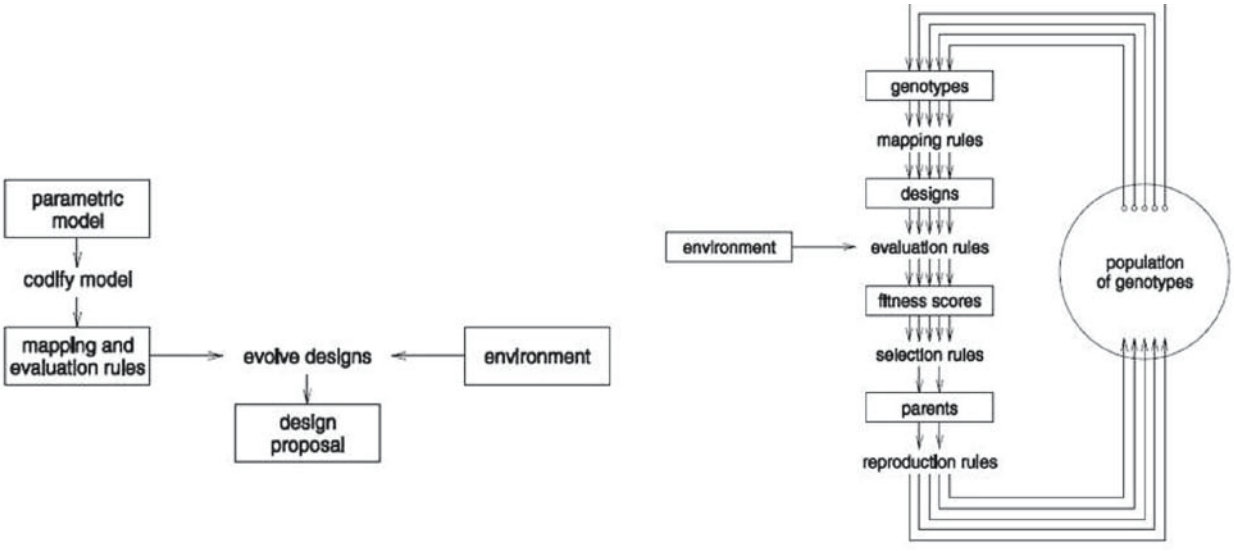


Fig. 3a-b. Typical evolutionary model, (a) design model, (b) computer generating system – diagrams; source: J.H. Frazer, P. Janssen.

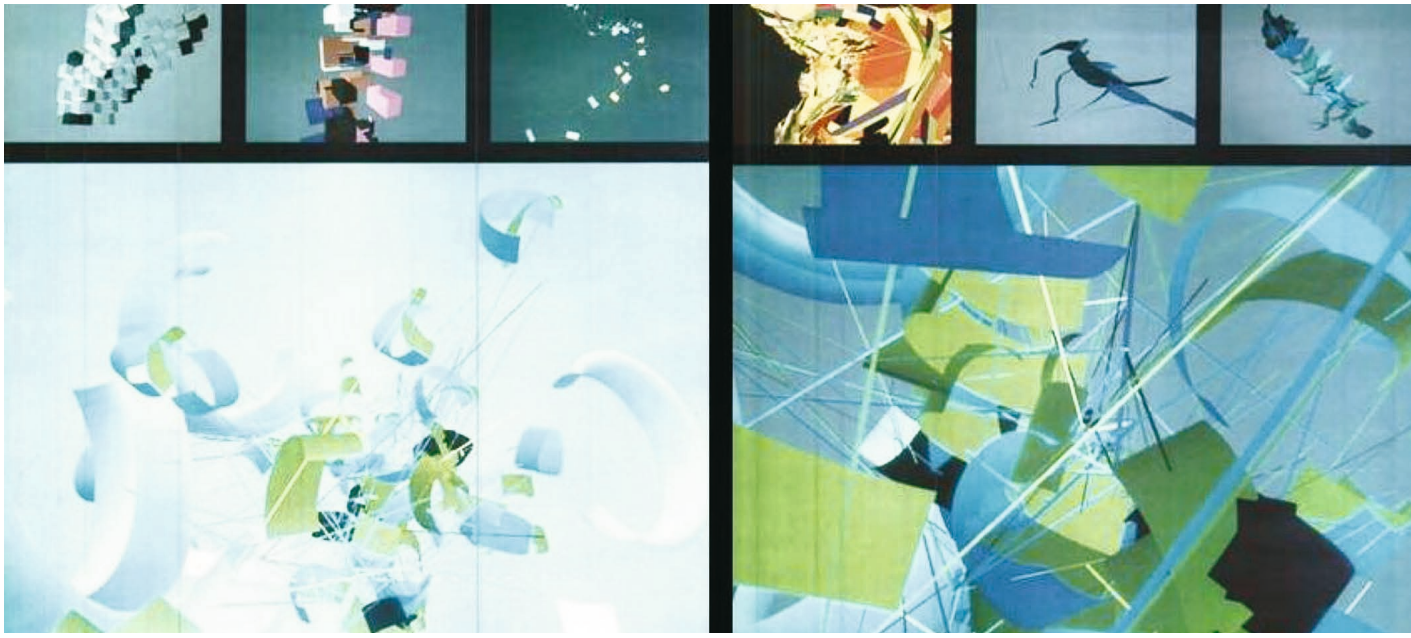


Fig. 4. Stefan Seemuller, Evolving sequences of geometric solids made with Universal Constructor software, 1991; source: J.H. Frazer 1995.

with the evolutionary processes of nature. It may be recalled that his first book was aptly titled *Adaptation in Natural and Artificial Systems* [J. Holland 1975]. The process of natural selection can generate a wealth of alternative experiments, and the better ones survive. There is no single solution, no optimal solution, but there is continuous experimentation. Therefore, Holland did not see the genetic algorithm as a convergent system.

In the early 1990s, John Frazer, began research on introducing genetic algorithms into generative design systems. Frazer was one of the first to attempt to extend the capabilities of CAD at the time to include computational tools for generating form. For this purpose, Frazer's team built the Universal Constructor in 1990, an electronic device designed to perform complex geometric operations and visualize them. This computer had 500 integrated circuits with 400 transistors each, and 6,400 diodes. The device was designed so that each circuit could operate as a whole on a common computer program. A separate application was created for each user to access the program and monitor screen. The user was encouraged to add environmental features to the set of problems being solved. The program allowed the inclusion of a variety of applications, resulting in the introduction of Cellular Automata computational models and making them responsible for graphically determining the location of design activities. Curve fitting was controlled by a series of Fibonacci numbers, while the generation of curves with complex geometry was based on *spline* curves [J.H. Frazer 1995].

The structure of natural data gives ease of input or change, and the amount of information affects the level of sophistication of modeling and simulation. Chromosomes in genetic algorithms are binary strings of fixed length (chromosomes can also be encoded with strings of integers or real numbers). This ease allowed the development of the idea, treating architecture as a form of artificial life, subject, like the natural world, to the principles of morphogenesis, genetic coding, replication and selection.

In 1991, the Universal Constructor was enhanced with an application based on an evolutionary computational model. It contained 256 available cell states that evolved as a series of chaotic loops, mimicking the "strings" of chromosomes. Each step of evolutionary development was continually read out and raised to accepted rules defining the pattern that the evolutionary process would follow expanding the complexity of the forms produced (Fig. 4). The application allowed the creation of sequential configurations of Euclidean solids (initiating) that evolved while the parameters were controlled by a graphical mapping program. It was also possible to intervene manually regarding making selections and stopping the process [J.H. Frazer 1995].

Imitating natural formative processes also required taking into account the influence of environmental factors on genetic code development, selection and mutation. Between 1991 and 1992, several approaches, such as data transmitters and simulation and environmental modelling techniques, were refined or developed from scratch.

During this period, analytical tools were developed for simulating the path of the sun for a specific latitude and determining the 'light and shadow' zones for a given development. These tools are now widely available in CAD packages. Based on the principles of the Cellular Automata self-replicating computer model (1979), simulations of turbulent flows that could be modelled were also developed. A system of classifiers was also developed, which received information from the environment and compared it with accepted classification rules to allow it to be fed into further operations. This linking of sensing and internal information processing to each other was thought to be analogous to an organism that processes information from its environment, 'thinks' about it and acts accordingly. If this action is successful, the organism is rewarded, and the modelling of the information implemented by the classifier system was a reinforcement of this success [J.H. Frazer 1995].

Frazer's research suggests treating Nature's evolutionary model as a generative tool to help, not only in the creation of new forms, but also in the study of the morphology of architecture, as exemplified by the evolution of the Tuscan column made in 1993.

the relationships between its parts were. The structure and logic of the rules were programmed into the computer, but a gene was substituted for each specified proportion. Populations were generated with random mutants (Fig. 5) [J.H. Frazer 1995].

The evolutionary design model presented was somewhat successful in its time, although it offered limited variability in the forms produced as only one 'seed' was evolved (the parametric model of the initial configuration). As a result, the programmes offered little scope for developing new ideas and forms.

2.2. Generative-evolutionary design model and the Interactivator system

The next step in the development of design tools that mimic biological formative processes was a design model and computer system that combined previous experience and followed advances in hardware and computer knowledge. The generative-evolutionary approach to design was described by Richard Dawkins as early as 1986 when he presented the concept of a generative-evolutionary system that enables the evolution of two-dimensional insect-like structures through artificial selection [R. Dawkins 1986]. Dawkins'

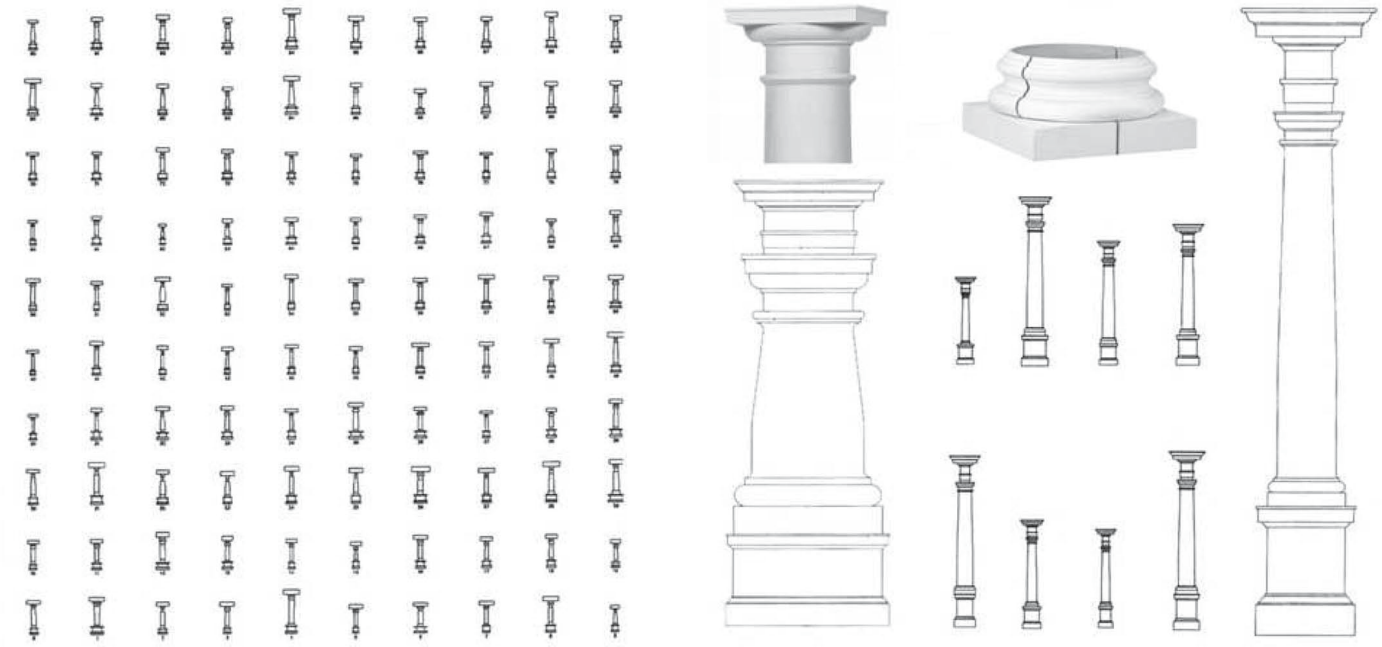


Fig. 5. John Frazer: evolution of the tuscan column by genetic algorithms, 1993; source: J.H. Frazer 1995.

Genetic algorithms were used to perform the evolution of the tuscan column, as part of an academic research programme, by means of which information on the proportions of the column was developed. Based on the Jemes Gibbs template, which defines the rules for drawing ancient orders, it was specified what

idea no longer required the embedded generative system to contain the idea of concept seeding. Instead, he proposed that the generative rules be described in a genetic code. The code would then be modified and developed into a series of design models in response to a simulated environment, and the models would

be evaluated in that particular simulation environment and a code of successful models selected. The selected code would then be used to repeat the cycle until a specific stage in the development of the model selected for prototyping in the real world.

The generative-evolutionary design model identifies two tasks:

- Codification of the generation concept, where rules and first principles of generation and evaluation are established. Generation rules generate projects from coded code scripts, and evaluation rules evaluate the generated project;
- Evolving forms that develop in response to the design environment. This second task requires a generative-evolutionary system.

4. selected code scripts are transformed by genetic operators such as crossing and mutation.

The above-mentioned four steps are then repeated, and this process can, at any time, be interrupted by the designer. Generative rules were usually developed to be very general and not intended to reflect specific architectural concepts. However, all such rules had to include preferences and constraints, resulting in the forms created having certain characteristics [J.H. Frazer 1995].

An important aspect of this approach is that the generative system produces forms in response to the environment. This environment consists of a design context and design criteria. Generative rules may require information about the context and criteria. In

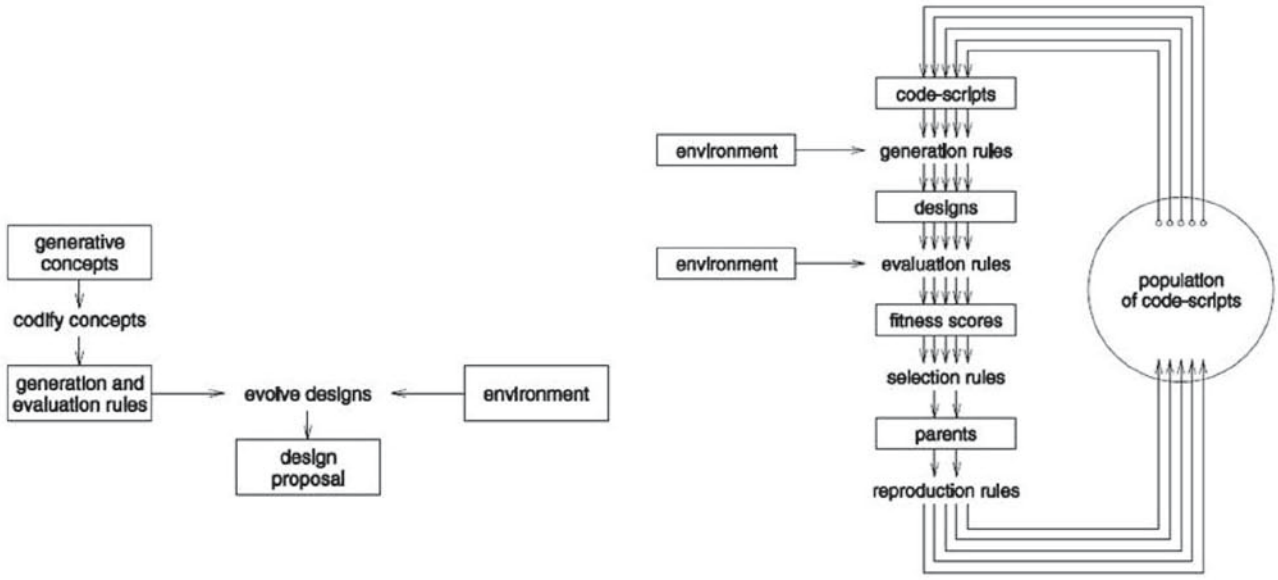


Fig. 6 a-b. Generative-evolutionary model (a) design model, (b) computer generation system, 1986, diagrams; source: J.H. Frazer, P. Janssen.

The generative-evolution model required by the generating system differs significantly from the previous system, as the mapping step has been replaced by a generative step (Fig. 6). This generative step has been built into the generative system, for which a population of code scripts is first created.

The evolutionary process therefore consists of four stages:

1. forms are generated from scripts of the genetic code mediated by the form in question, its epigenetic development in the environment;
2. the resulting forms are evaluated through simulation and analysis of their performance in their environment,
3. the most successful specimens are selected;

this way, the development process that 'develops' the design becomes an adaptive process. This adaptive process identifies subsequent structural modifications in response to the environment and measures the performance of different structures in the environment. The adaptive process creates a sequence of structures from a set of operators. Information extracted from the environment influences the choices made. This contrasts with systems that use a mapping process. Mapping processes perform simple, non-adaptive transformations that do not interact with the environment in any way.

Significant evolutionary-generative experiments were conducted by Frazer's team in the first half of the 1990s.

By 1993, it was already possible to demonstrate what a genetic architecture language was. Work on 'living' systems and artificial life led to the development of a set of scripts for the evolution of structure and form and a genetic architectural language for its encoding. A computer model was developed to the point where it was able to perform the form generation process in a simulated environment. Evolutionary space and time were compressed so that the model could evolve over a large number of iterative cycles. The theoretical framework was also developed so that it could be demonstrated interactively [J.H. Frazer, J.M. Silver 1994].

Such a display took place on 25 January 1995 at an exhibition entitled 'Evolutionary Architecture', the work of Frazer, his wife and their students from the Society of Architects and the School of Design and Communication at the University of Ulster. The demontrative model proposed a generative-evolutionary system that was accessible via the internet with the intention of encouraging widespread participation, and creating biodiversity in the pool of genetic forms on which the model depended. Janssen was involved in the development of a special demonstration version, known as The Interactivator. Although the theoretical system was simplified, all the key elements were presented. Participation in the development of the model could be achieved by being on display or in virtual form on the Internet.

Indeed, Interactivator had a system that was based on the sequential evolution of a family of cellular structures in the environment. Each cellular structure began to evolve from a single cell inheriting genetic information from its ancestors. In the same way, each cell in a cellular structure contained the same chromosomes that make up the genetic code. Cells divided and reproduced based on the script of the genetic code and the environment, with each new cell containing the same genetic information [J.H. Frazer 1995]. The data structure used to represent cellular growth was based on a universal state space or isospatial data structure, in which each cell in the world has up to 12 equally spaced neighbours and can exist in one of 4096 states, the state of a cell being the determined number of spatial arrangements of its neighbours. The local environment of a cell (in the world) could therefore be encoded in a 12-bit binary string. For example, a string type (110110000110) would spatially represent a cell with six neighbours in specific positions. Chromosomes control the growth and development of the cell structure. A typical chromosome consisted of the following 5 parts:

- Chromosome origin: Internet address from which the chromosome originated;
- Condition: local cell environment (* unattended situation);
- Action: the state of the cell in the next generation;
- Flag: whether the chromosome is dominant or recessive;
- Strength: Efficiency of the chromosome in relation to the environment.

The developmental process of each family member consisted of three parts - genetic search landscape, cellular growth and materialisation and was based on a simple Goldberg classification system. The genetic algorithm was used to ensure that future generations of the system would learn from previous generations, and to ensure biodiversity during the evolutionary process. Each of the three developmental processes consisted of cyclic processes.

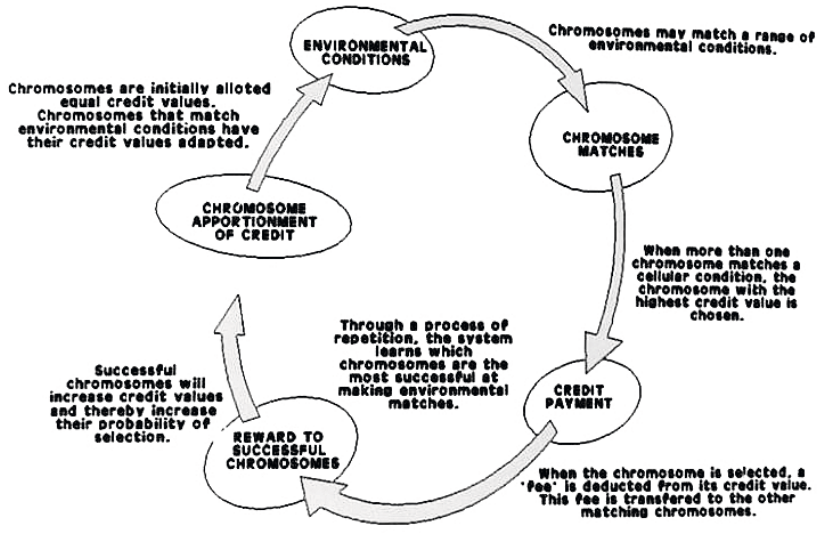


Fig. 7. Genetic search process – diagram; source: drawing by P. Janssen, J.H. Frazer et al. 2002.

GeneticSearchLandacape: Selection criteria are not defined, but are an emergent property of the evolutionary process and are based on the relationship between chromosomes, cellular structure and the environment over time. For each member, a genetic search landscape is generated that graphically displays the changing selection criteria. Form, or the logic of form, emerges as a result of traveling through this space of inquiry (Fig. 7a) [P. Janssen et al. 2002].

Cellurargrowth: Chromosomes are generated by sending them from a remote user, active site, or as a function of selection, crossover, and mutation within cellular activity, and are maintained in the main chromosome pool. The physical environment determines which part of the main chromosome pool will become dominant. Each cell's local environment then determines which part of the genetic code is turned on. The cell then reproduces and divides according to this genetic code (Fig. 8a-b) [P. Janssen et al. 2002].

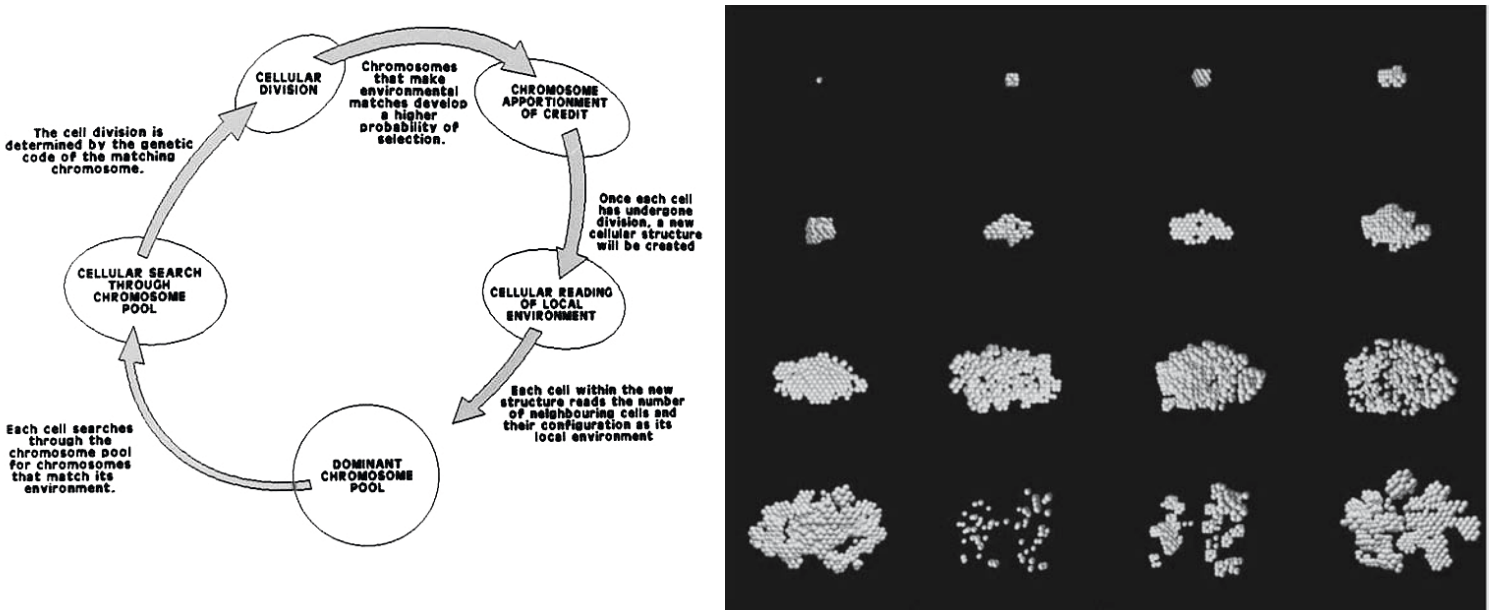


Fig. 8a-b. Interactivator, cellular growth process – diagram; source: drawing by J.H. Frazer, P. Janssen, b) cellular growth sequencies; source: J.H. Frazer, P. Janssen 1995.

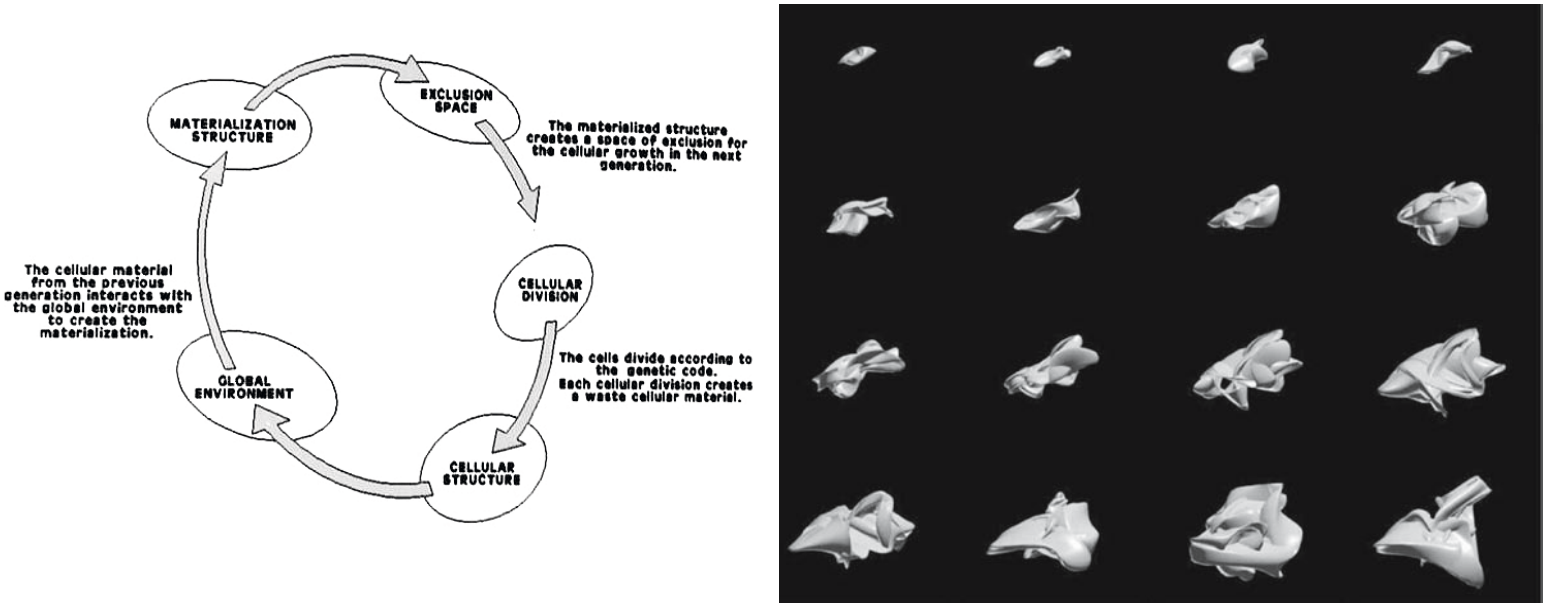


Fig. 9 a-b. Interactivator, materialization during cell division, a) materialization process; source: drawing by N. Paszkowska-Kaczmarek, based on Frazer, Janssen, b) materialization sequence; source: P. Janssen, J.H. Frazer et. al. 1995.

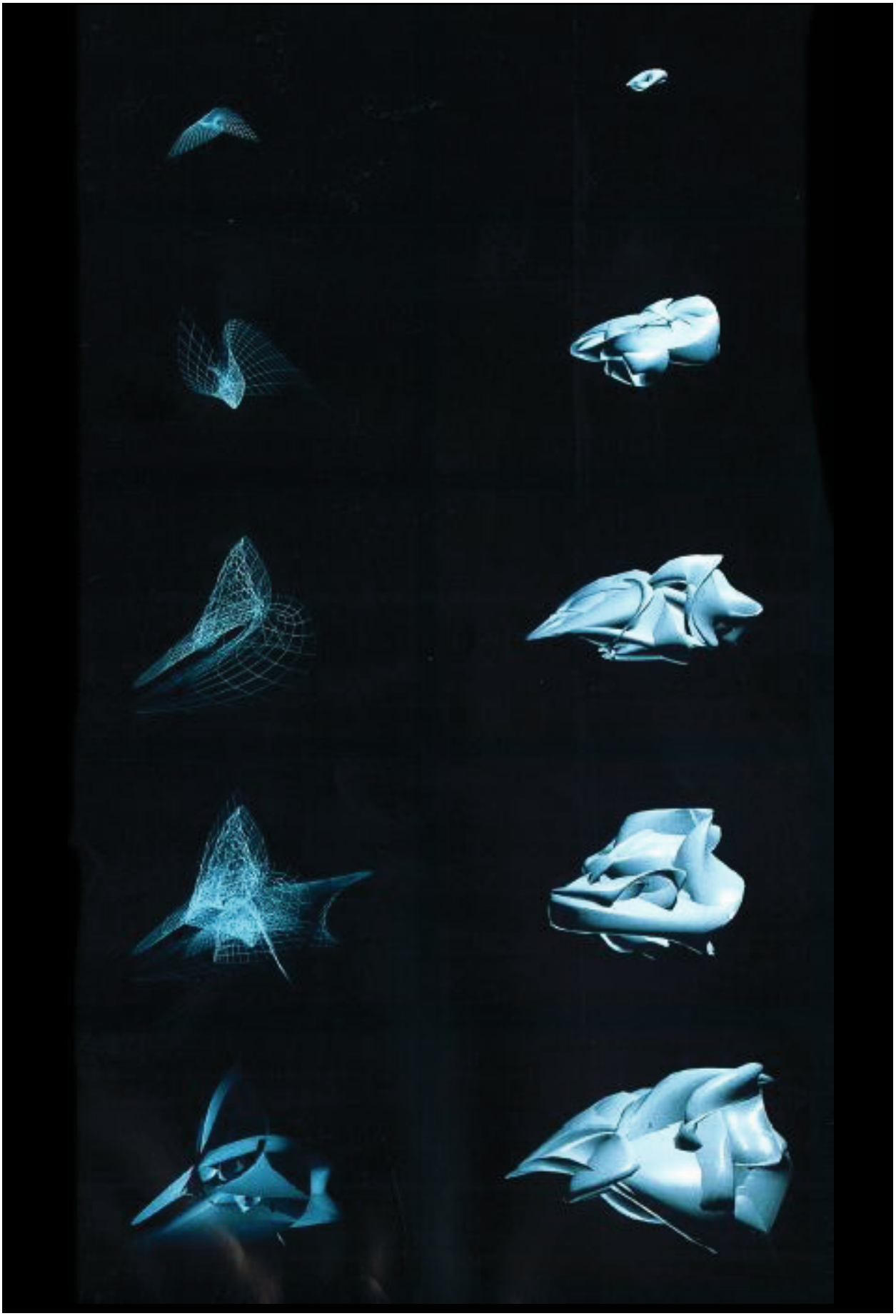


Fig. 10. Interactivator, the sequence of formation of the biometric structure of the form and its materialization; source: J.H. Frazer 1995.

Materialization: unstable cells are generated during cell division. In the next generation, this remaining material creates an exclusion space in the cellular space. This space of exclusion interacts with the physical environment to create materialization. Boundary layers are identified in unstable cells as part of their state information, and an optimized surface is generated to cover the structure. This material already exists during the evolution of the model and initially influences the growth of cells of future generations (Fig. 9a-b) [P. Janssen et al. 2002].

of concepts using the language of genetic design. Genetic algorithms were used to perform selection, and normal interbreeding and mutations were used to propagate the population. The model consisted of an infinite number of data points that together constitute the data space. Each point in the data space was intelligent in the sense that it knew where it was and why it was there, and had a clear awareness of the spatial relationships of its neighbors. The laws of symmetry and symmetry breaking were used to control the development of the model from the genetic code file. The

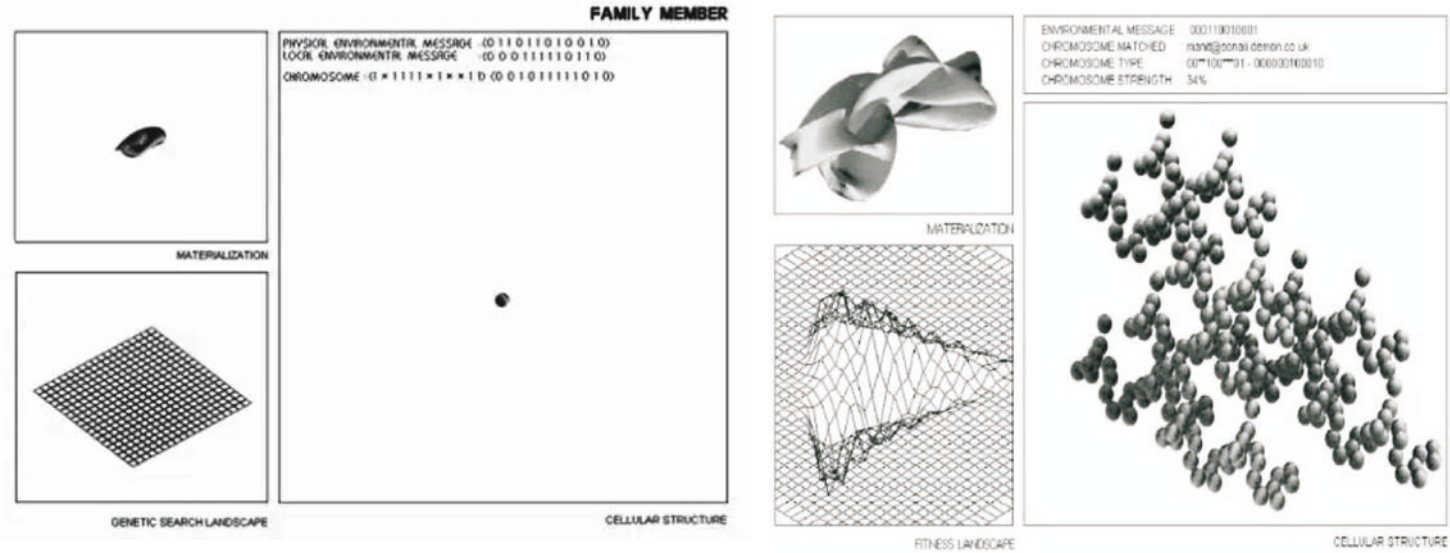


Fig. 11a-b. Interactivator: sequences of form formation by cell division; source: J.H. Frazer 1995.

In this context, a seed in a computer model transfers its genetic codes to other seeds through cell division and then disperses across all models (Fig. 11). According to the fitness value that is calculated for the computer medium, the success genes in genetic algorithms are selected as they occur in Nature. Then, these genes are subjected to crossover and mutation operations, and various architectural forms are created in the process of model operation.

The presented evolutionary model was organized taking into account a hierarchical approach to the model itself and the data structure, which is recursively self-similar. The simulated environment in which the assessment took place was modeled under exactly the same conditions as the evolving structures. Not only did the environment and structure evolve in the same data space, but they could evolve together. Moreover, competing structures could also evolve in the same space. The environment, in this case, involved user response and was modeled using virtual communities. It had a significant impact on the development flow of information through the model took the form of logical fields. The externalization of this data structure was a process driven by modeling the form generation process rather than the forms themselves [J.H. Frazer et al. 2002].

The selection criteria were not programmed into the model, but were an emergent feature of the evolution of the form model itself. For each element, a genetic search landscape was generated graphically depicting selection criteria changing over time based on the relationship between chromosomes, cellular structure and environment. The form, or the logic of form, was created as a result of traveling through this space of exploration [J.H. Frazer et al. 2002).

The Interactivator was an evolving environment that was intended to respond both to the interactions of exhibition visitors and to the atmosphere in the exhibition space (Fig. 12). Visitors were drawn into interaction by proposing genetic information that would influence the evolution of the model. Sensors in the exhibition space also influenced the evolution of the

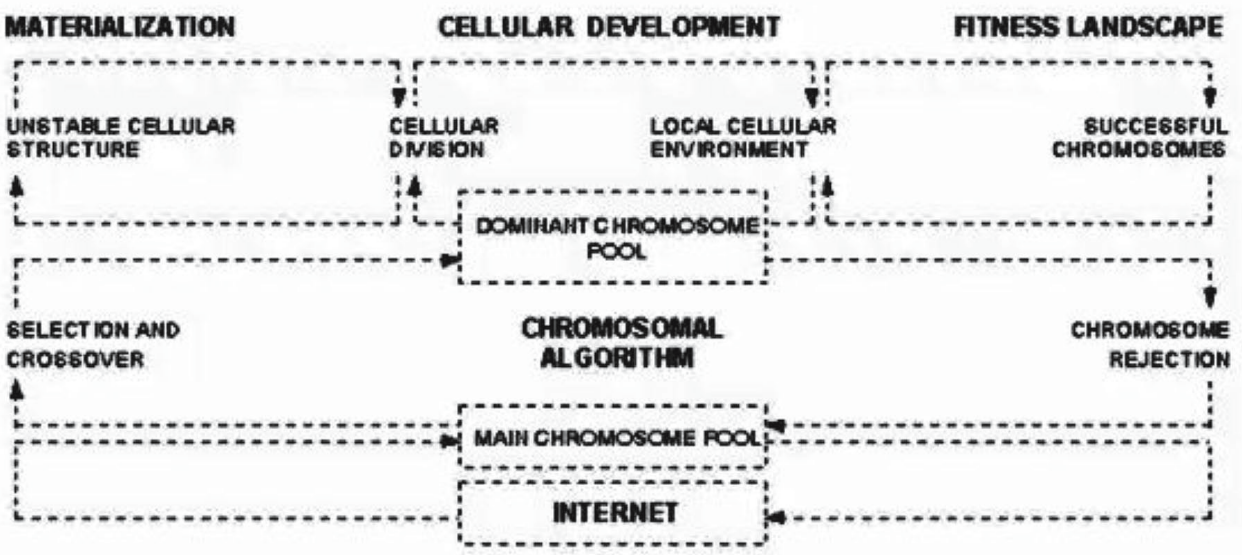


Fig. 12. Interactivator: action diagram; source: P. Janssen, J.H. Frazer et. al. 1995.

model, providing data on temperature, humidity, noise, smoke, and so on.

Collaboration via the Internet took place in three ways:

- the Internet was used to allow virtual visitors to enter genetic information into the model, as physically present at the exhibition;
- the model program allowed for remote data downloading, but in such a way that it could work on its own to replicate, and each replication followed a different evolutionary path, the results of which could also be fed back to the central model to contribute to the gene pool;
- online access to the exhibition and book via conventional means has been enabled through a website so that you can understand the context and observe the development stages of the evolving understanding model [J.H. Frazer et al. 2002].

Interactive experiences gained at the Ulster show were used to refine a generative computer model that could, with the participation of residents, determine the desired or expected development of the built environment. The evolutionary model was intended to explain the transition from the past to the present and predict trajectories of future possibilities, as a tool for examining and assessing existing and future urban and social phenomena.

2. 3. Interactivator: combined version

Combining the concept seeding model with the generative-evolutionary model resulted in a new type of model that integrates the advantages of both previous design models. This approach was first proposed by Frazer in 1990 [J.H. Frazer 1990] and later developed in the book *An Evolutionary Architecture* [J.H. Frazer 1995].

The concept seeding model enables the generation of design forms that incorporate specific architectural concepts. However, the model does not use an evolutionary system, and as a result, the designer must discover generative modifications that will achieve the most appropriate design. However, the generative-evolutionary model enables, in a complex way, the evolution of forms adapted to their environment.

However, these forms do not contain any architectural concepts and are therefore highly abstract. Janssen's combined model synthesizes previous models, thus enabling the evolution of designs that, in addition to adapting to their environment, also incorporate specific architectural concepts.

This combined design model generatively developed modifications that, in the case of the concept seeding model, the designer had to introduce manually. These modifications made minor changes either to the seed concepts or to the principles that transformed them. The representation of these modifications was a code-script, which allowed the evolutionary system to also evolve the modifications. Generative modifications encoded in the code-scripts were used to generate designs, which were then evaluated and, based on these evaluations, new populations of modifications were created. This approach required rigorous definition of the scope of important generative modifications. This rigorous definition was required to enable the development of rules that could autonomously create

new methods for generating modifications [J.H. Frazer and J. Frazer 1996].

The combined design model defines three tasks as:

- odification of generative concepts, i.e. first, as in the previous model, generation and evaluation rules are defined. However, in this case, the generation rules they must produce designs from seed, not from scripted code;
- codification of architectural concepts, i.e. defining the germ of the concept that codifies a set of architectural concepts;
- evolving designs, i.e. design alternatives evolve in response to the environment design, which requires a generative-evolutionary system that includes seeding.

In the area of the generative-evolutionary design system, the idea of seeding concepts was a significant departure from the adopted research direction. Most researchers developing divergent evolutionary systems viewed all constraints and inclinations as negative effects that should be minimized (though never completely eliminated). Admittedly, the concept seeding approach reinforces constraints and inclinations, but at the same time ensures that the generated forms reflect the design ideas. Therefore, limitations and prejudices are perceived as positive rather than negative phenomena.

2.4. Janssen's generative-evolutionary model

Patrick Janssen developed and refined Frazer's combined evolutionary-generative model, developing

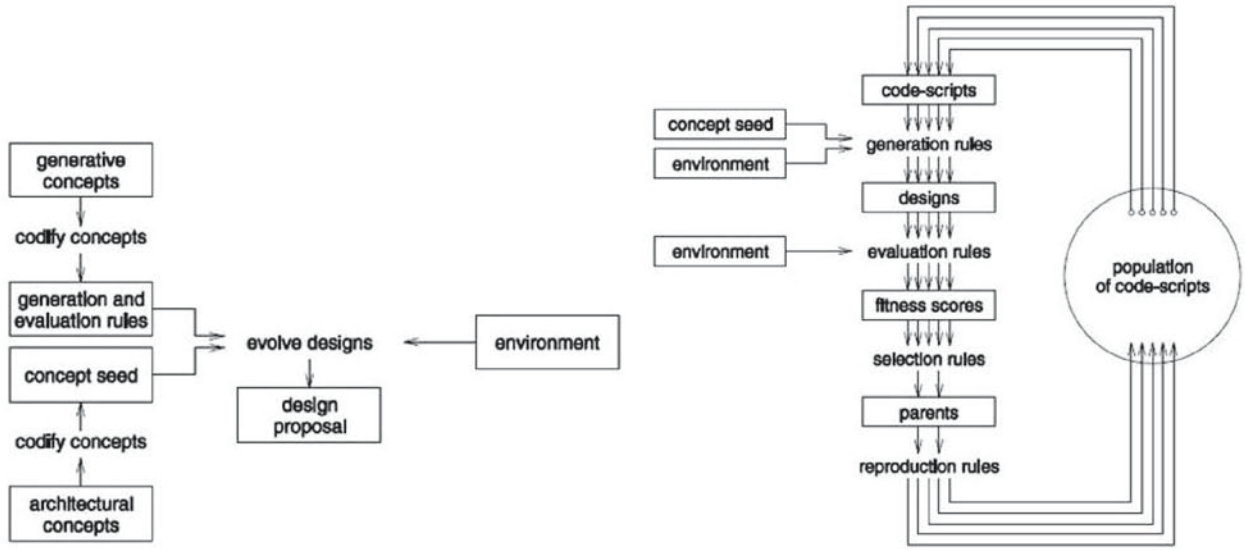


Fig. 13a-b. Combined generative-evolutionary model, a) design model, b) computer model generating system; source: P. Janssen, J.H. Frazer et al. 1995.

The generating system here is similar to the generative-evolutionary system described earlier. However, in this case the embedded generative system produces design forms based on concept seeds. The generated designs are therefore the embodiment of a set of architectural concepts codified by the seed. Coding scripts encode generative modifications. These modifications cause small changes either in the seed of the concept itself or in the generating rules that transform it. Generative modifications result in the production and evaluation of different designs. Generative modifications that result in designs with the highest efficiency scores are then selected. Genetic operators are then used to create a new population generating modifications that will be used to generate a new population of designs, and so on.

a wide range of generation systems using concept seeding. He also introduced significant modifications, especially when it comes to codifying the design concept and the overall structure of the model.

- The idea of codifying the initial architectural concept has been refined. The architectural concept must contain sufficient flexibility and adaptability to enable the creation of various designs. The architectural concept should therefore not predefine the overall organization and configuration of the designs, but should instead focus on defining the design parts and their interactions and overlaps. These interactions and overlays can be thought of as defining the nature of designs without specifying their overall form. Therefore, Janssen referred to a set of architectural concepts as a pattern of

form. However, the purpose of developing an architectural form is the ability to use its concepts to create various forms. Janssen defined "diversity by the distinction made in evolutionary biology between diversity and disparity [S.J. Gould 1989]. He referred to diversity as designs that differ in proportions and part dimensions, but have the same general organization and configuration of parts. The discrepancy concerned those projects that had a fundamentally different organization and configuration of their parts.

- The second area in which the combined Frazer model has been modified is its general structure. Two tasks from Frazer's combined model have been changed, i.e. concept codification and evolving form have been introduced between the two new tasks. At the beginning, Janssen placed a task that focuses on the development of a form diagram; at the end, he added a task that focuses on developing a detailed project proposal. These four tasks were grouped in pairs to create two phases: the blueprint development phase and the design development phase (Fig. 13). This division reflects two different levels of environment. On the one hand, the schema creation phase will produce a schema that is specific to a general category of environment, called a niche schema environment. On the other hand, the design development phase creates a design that is specific to one particular environment, called the design environment. In both cases, the environment includes both the criteria that the project must meet and the context in which the project will exist [P. Janssen et al. 2003].

The codification of the mold scheme also differs from the Frazer model. In Frazer's model, architectural concepts are codified into a concept seed, but it is unclear to what extent this architectural concept influences the coding of other rules, such as development and mapping rules. In Janssen's model, codification of the character schema affects all evolutionary rules and data structures, including development and mapping rules. The set of evolutionary rules and data structures is collectively called an evolution schema. This evolutionary pattern infuses the generative-evolutionary design system with inclinations and constraints that reflect the ideas developed by the design team.

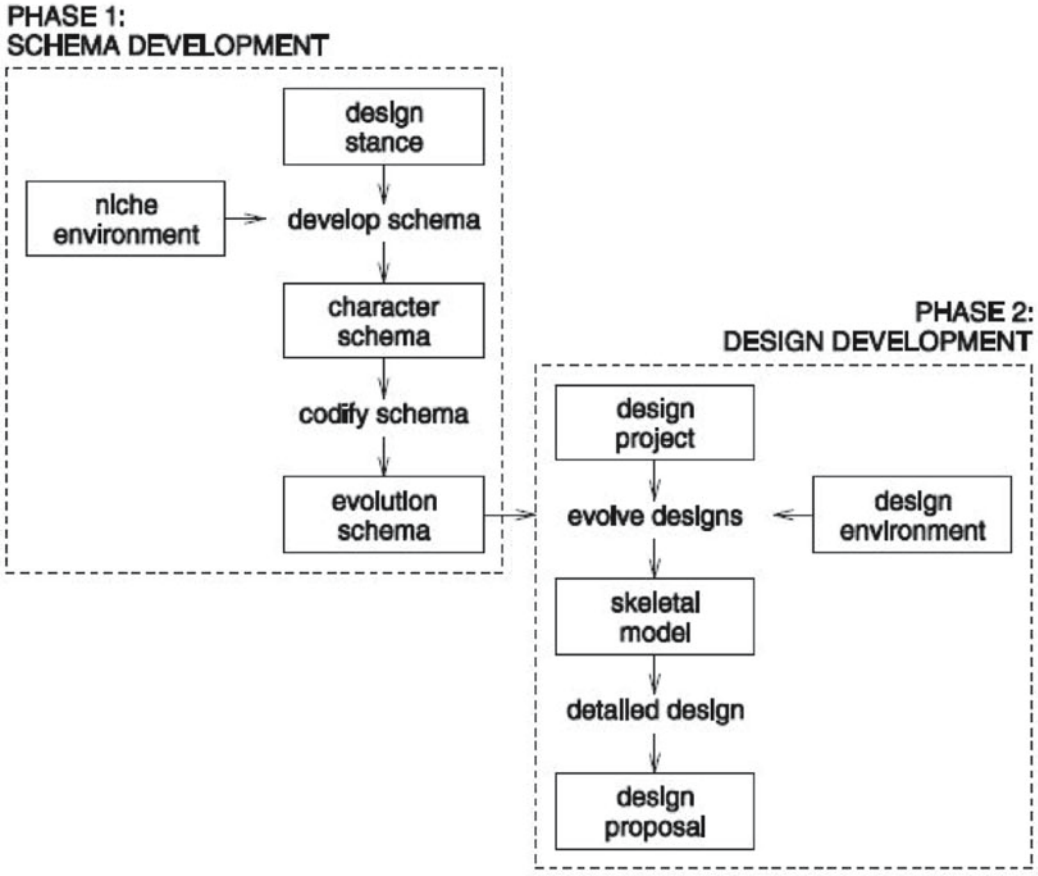


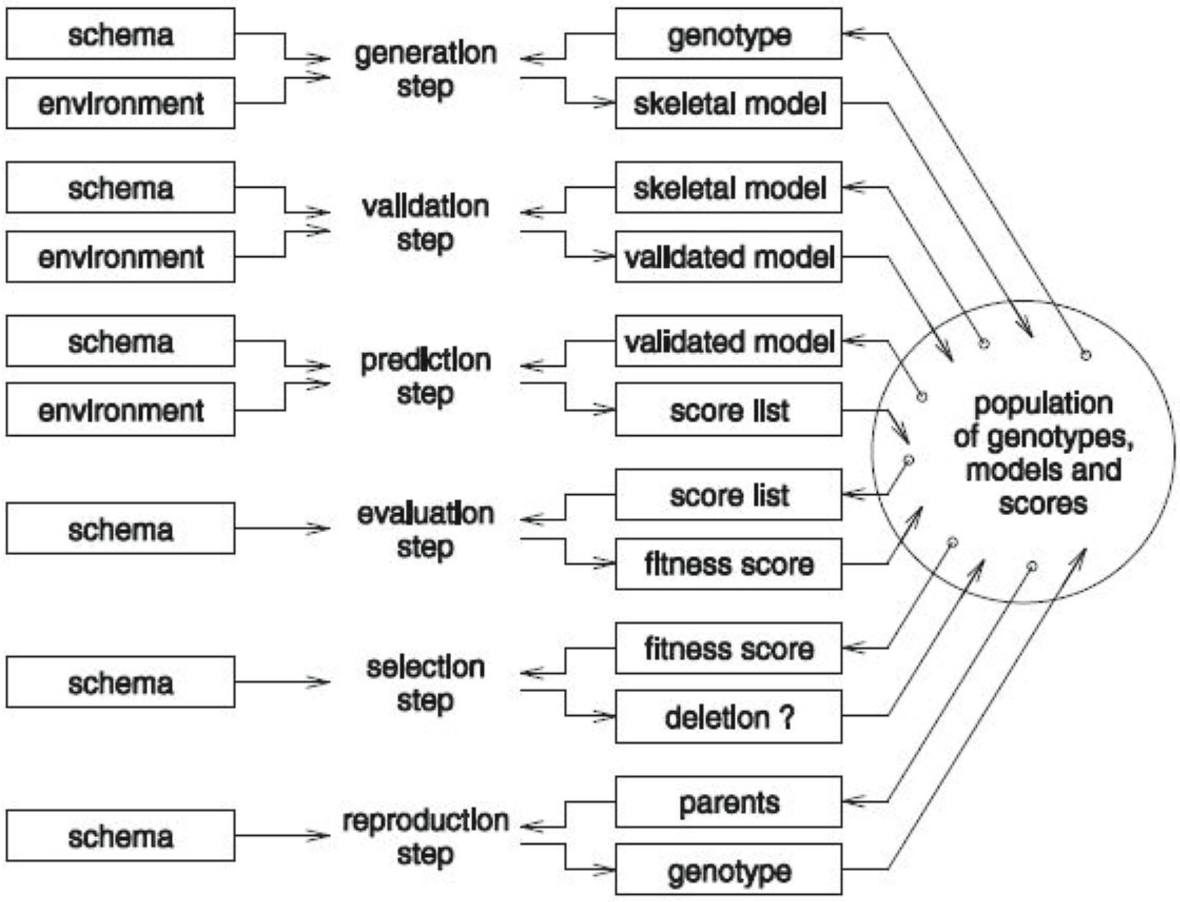
Fig. 14. Two-phase generative-evolutionary design model – diagram 1996; source: P. Janssen, J.H. Frazer et. al.

Dividing the design procedure into two phases made it possible for these phases to be implemented in different ways:

- The first phase develops and codifies the nature of the project. The architectural form will then reflect the beliefs and preferences of the design team (called the designer's position) and will be developed in response to the niche environment. This niche environment can be defined before finding a specific development environment. As a result, this phase creates a general design unit (evolution diagram) that can be reused across different projects;
- The second phase is the process of evolution and details the form proposals for a specific design task. The selected proposal will be tailored to the project environment and assigned tasks that will cover aspects such as site, space requirements, performance goals and budget. Hence, at this stage, a design element is created that cannot be reused.

The operation of the generating system in the generative-evolutionary model proposed by Janssen (Fig. 15) is based primarily on population data. Each evolutionary step takes specific data from the population, applies the rules specified in the evolution schema, and then transmits the newly created data. The computer hardware should be built to be able to identify a set of components of the generating system (software) and propose a network configuration for these components. These components were to use a number of technologies and software systems that existed at that time, such as a computer program acting as a client for services provided by the server.

The hardware configuration proposed by Janssen (Fig. 16) is divided in its architecture into a program operated by the server and a set of programs for the client. The server program stores a population of projects in a database as well as a set of client programs that communicate with the server. The file for each evolution step is executed separately by the client. Subsequent clients take over the seed client, which allowed



Ryc. 15. A generating system in the generative-evolutionary design model - diagram 1996; source: P. Janssen, J.H. Frazer et al.

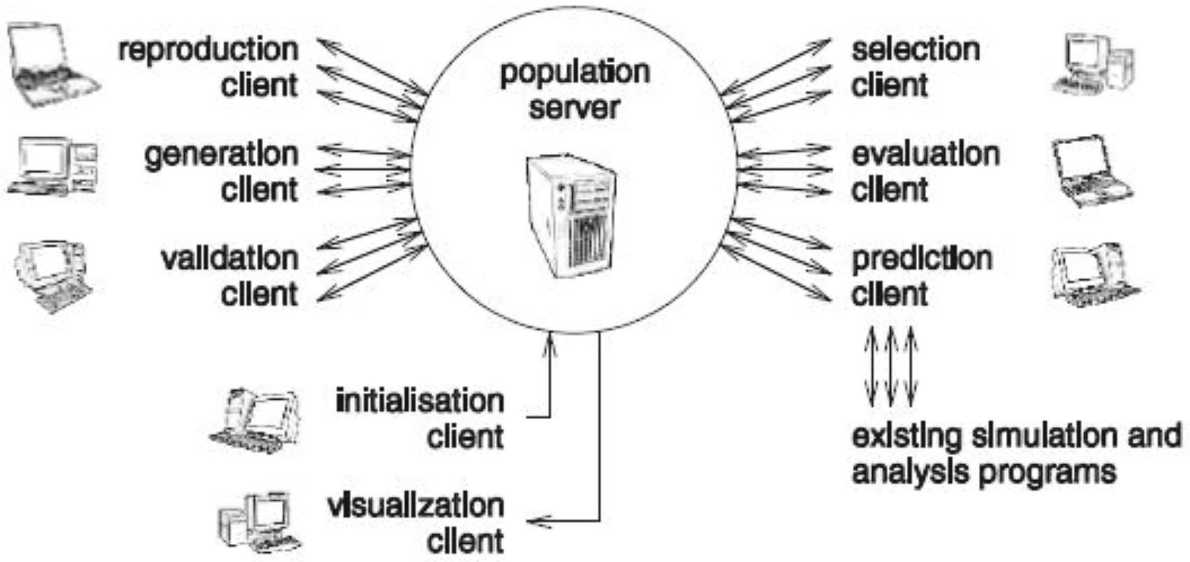


Fig. 16. Hardware configuration for the generative-evolutionary design model, 1998; source: P. Janssen, J.H. Frazer et al.

the population to be initialized, and the visualization client, which allows users to view a specific design in the population [P. Janssen et al. 2004].

This asynchronous evolutionary process means that any given client can be replicated many times. For example, a prediction client that runs a specific simulation program can be duplicated, thus allowing designs to be simulated in parallel. Additionally, clients can run on different operating systems, thus facilitating the integration of third-party CAAD simulation and analysis programs.

Generative and evolutionary design involves using the virtual space of a computer in a way analogous to the evolutionary processes occurring in nature. Although the techniques described could be achieved with relatively simple design problems, the architectural problems still required computing power that exceeded the hardware standards available at the time. The evolutionary model of nature proposed in the 1990s as a process of creating an architectural form was supposed to foster the achievement of symbiotic behaviors and metabolic balance characteristic of the natural environment by the built environment. Architecture came to be treated as a form of artificial life, subject, like the natural world, to the principles of morphogenesis, genetic coding, replication, and selection.

In 1971, chemist Tibor Ganti provided an important elaboration of the criteria of life in his seminal work Principles of Life, in which he distinguished between the criteria of actual life and the criteria of potential life. It is about the criteria that an organism must meet in order to be considered able to live and the criteria that are necessary for an organism to survive life on Earth [T. Ganti 1971]. The real criteria of life are: i) inherent unity - the system must be a coherent unit; ii) metabolism – a living system must have the ability to metabolize; iii) inherent stability – a living system must be inherently stable; iv) subsystem carries information - a living system must have a subsystem that carries information that is useful to the entire system; v) control program - processes in living systems must be regulated and controlled. The criteria for potential life are: growth and reproduction, ability to inherit changes and evolution, mortality. Research on synthetic life covers similar criteria, including (individual properties) metabolism, inheritance, and evolution [B. Holmes 2006]. Synthetic life must meet these criteria. They are derived from the intense self-organization capacity demonstrated by the articulation of biological materials at all size scales studied. These criteria are analyzed and applications discussed in terms of architectural application. At the end of the 1990s, there were even attempts to make a computer program “inhabit” the built environment, enter its structure, read it, understand its development principles and history, be able to capture its topography, latitude and climate, and model its society and economy. And then the computer program would start asking for suggestions and proposing possible functions and spatial solutions based on hardware standards [J.H. Frazer 1997].

The design techniques presented here implied significant changes in the architect’s working methods. First of all, they force us to rigorously define how an

architectural concept is expressed in the genetic code. Moreover, the architect must clearly define the criteria for assessing the idea and be prepared to accept Janssen's concept of client and user participation in the design process.

The use of generative-evolutionary tools in architectural design also increases the role of the architect in the design process, as it becomes possible to „sow" many more generations of new forms than could be individually supervised, and to achieve a higher level of sophistication and complexity far beyond the economics of normal office practice.

3. TOWARDS THE INTEGRATION OF METHODS AND TOOLS

Imitating the works of Nature also means striving for an artificial object to achieve properties that make it equally efficient and effective in its environment. In the context of natural morphogenesis, the formation process extracts chemical properties and physical material organizations. In the world of Nature, this is the result of the influence of environmental factors that influence morphogenetic movements externally and internally. Similar to natural morphogenesis, the process of physically finding form emphasizes the appreciation of material systems from the perspective of a bottom -up design approach. However, morphogenesis in IT spaces separates the process of materialization from the process of form creation; in this way, material systems are imposed on the generated forms. Rationalization and optimization methods are only an attempt to reconcile the materialization process with the formation process [A. Menges 2008].

Due to natural morphogenesis, it is expected that computational procedures will use the digital materialization method, in which materialization is encoded as an active controller within the digital formation. Otherwise, distinguishing form generation from materialization requires another phase to impose a digital form derivative on the materialization processes. Therefore, formation should treat materialization as an embedded process that consists of both material and production domains within the produced morphogenetic development.

For over a decade, research on computational morphogenetic processes has promoted design development by combining generative design techniques, manufacturing technologies, and analytical design strategies [K. Januszkiewicz 2016]. The introduction of these components means extending the linear design framework to include elements of linear design processes.

3.1. Genr8: Evolutionary Algorithms and Growth Algorithm in the System CAD/CAM

Morphogenetic and generative modeling tools were not yet available in the 1990s a compact package that would allow these tools to be used in a CAD/CAM system. One of the goals of the Genr8 project was to demonstrate that the combination of a growth algorithm and an evolutionary algorithm is useful for form exploration in the architectural design process. By implementing these algorithms in C++, a programming tool was developed that can be (and is) also used in educational practice.

In 1997, the interdisciplinary research team Emergent Design Group was established at MIT, which included computer scientists and architects. The idea was to explore the possibilities of synergy between architecture, artificial intelligence, artificial life, engineering and materials science in order to develop a prototype of a program providing new modeling tools in the CAD system. In 2001, Una-May O'Reilly and Martin Hemberg introduced the Genr8 program. This was supposed to be proof that the concept of generating surfaces by combining L-Systems and evolutionary algorithms is useful in modeling architectural formsj [U. Q'Reilly et al 2004].

It was intended that Genr8 would be treated as a sketching tool that should be used early in the design process. It was envisaged that this program would be useful in conceptualizing form, which would then be subjected to detailed definition and analysis in terms of structure or material. The lack of structural and material analyzes in Genr8 was expected to limit its specific value [M. Hemberg et al. 2007]. However, the Genr8 environment can be configured to somewhat reflect physical reality - adopting criteria and parameters that are geometric in nature. This forces the designer to interpret structural or material constraints (e.g. by introducing certain angles or distances between support points).

Genr8 consists of two main components: the HEMLS (Hemberg-Extended-Map¬ L-System) growth engine and the Evolutionary Algorithm (AE) (Fig. 17). The growth engine uses the HEMLS interpreter to analyze how you prescribe. It geometrically interprets the axiom and the set of rules for rewriting the analyzed system. The set of rewriting rules is a context-free grammar. The growth process encoded in the HEMLS engine is computationally linked to the simulated physical environment. This allows the architect to influence this abstract environment and the depicted elements that should interact with the growth process. It should be noted that AE as a tool can only be (and was) used with the growth algorithm.

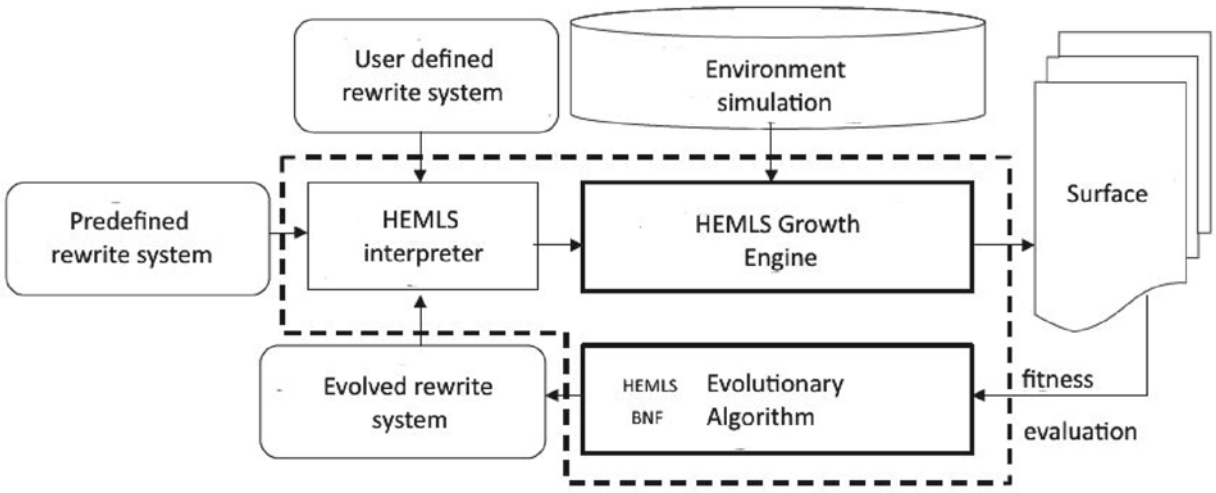


Fig. 17. Una-May O'Reilly i Martin Hemberg, Genr8 system model – diagram, 2001; source: U. Q'Reilly et al. 2004.

The growth algorithm, as the central element of Genr8, is used here to generate surfaces. This algorithm is based on L-Systems, which are used to model plant growth [P. Prusinkiewicz, A. Lindenmayer 1991]. The L-System is a grammar consisting of a seed and a set of production rules, and a rewriting process in which the production rules are repeatedly applied to the seed and its subsequent states. In simple terms, L-Systems can be considered as systems for rewriting sequences of symbols. Combined with the graphical interpretation of generated strings, they are a way of generating graphics. The most popular method of graphically representing L-systems is turtle graphics, in which symbols are interpreted as instructions for an imaginary turtle moving along patterns of lines in 3D space. The L-System should be understood here as a set of instructions on how to create a specific form, rather than a detailed plan detailing each building plan. The advantage of L-System is that at each stage of growth, the entire surface will be modified simultaneously, and not by sequential addition of components, and the form will acquire an organic appearance.

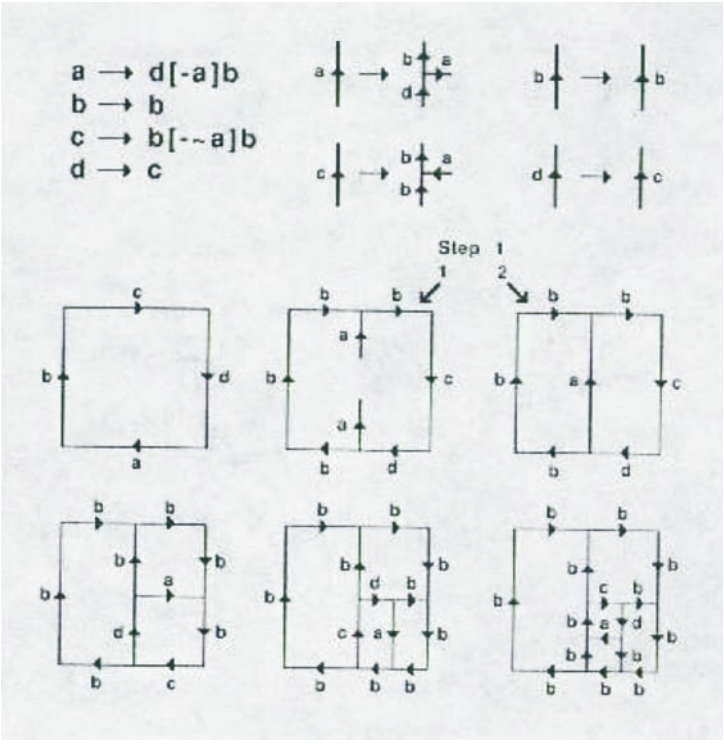


Fig. 18. Map L-Systems in biology and Gren8 - a model of cellular development – planar graphs, source: M. Hemberg et al. 2007

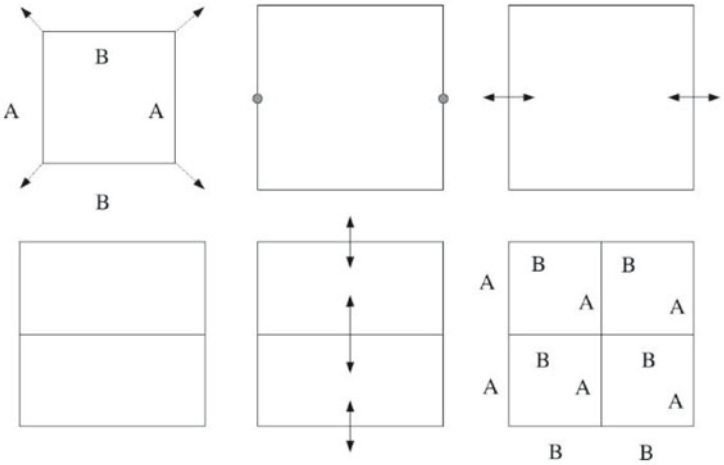


Fig. 19. Map L-Systems in biology and Gren8 - formation of squares derivedby the HEMLS rewriting system source: M. Hemberg et al. 2007

Map L-Systems were originally invented as a model for cellular development. This is a method of rewriting planar graphs using cycles (Fig. 18). With appropriate graphical interpretation, these graphs can be used in a biological context to represent cellular structures. While L-System generates arboreal structure, Map L-system generates graphs that can be interpreted as surfaces. The original L-system map model is designed for 2D only and to make it work in 3D, several additions have been made to this model. This is how the Hemberg Extended Map L-System was created (HEMLS).

In Genr8, each growth step has three phases (Fig. 19). They are illustrated starting from the seed (top left). In the first phase, the area size is increased by a simple scaling factor. Each vertex is moved away from the geometric center of the surface, as indicated by the arrows. In the second phase, the rewriting rules are applied to each edge of the surface. Here the edges of 'A' are split and the new vertices are marked with circles. In the final phase, the branches are drawn and connected. The same procedure is applied to the 'B' edge in the center panel down. The marking of each edge is only shown in the upper left corner and the lower right corner, which shows the surface with new markings after one iteration of the rewrite rules.

A limitation of the basic L-System model is that it can only create arboreal topologies. To generate surfaces, the Map L-systems Lindenmayer algorithm [P. Prusinkiewicz, A. Lindenmayer 1991] should be used.

In Genr8, the Map L-systems algorithm was further extended to create 3D surfaces and named Hemberg Extended Map L-systems (HEMLS). These surfaces grow in a reactive environment simulating the physical environment [M. Hemberg et al. 2007]. An example of a surface growing in an empty HEMLS environment is shown in Fig. 18-19. HEMLS requires a seed specification (or initial flat surface), a set of production rules, and two additional parameters, forming a prescribing system. The rewriting system shown in Fig. 19 is built into Genr8.

The Genr8 modeling environment has a significant impact on the result of the surface growth process. There are two types of elements in this environment: forces and boundaries. The forces may be point attractors or repellents that act like magnets, causing the surface to grow towards or away from their location (Fig. 20b). There is also a gravitational force that uniformly directs growth along one of the principal coordinate axes (Fig. 20a). Boundaries can be placed as obstacles or used as bounding boxes to cordon off a surface.

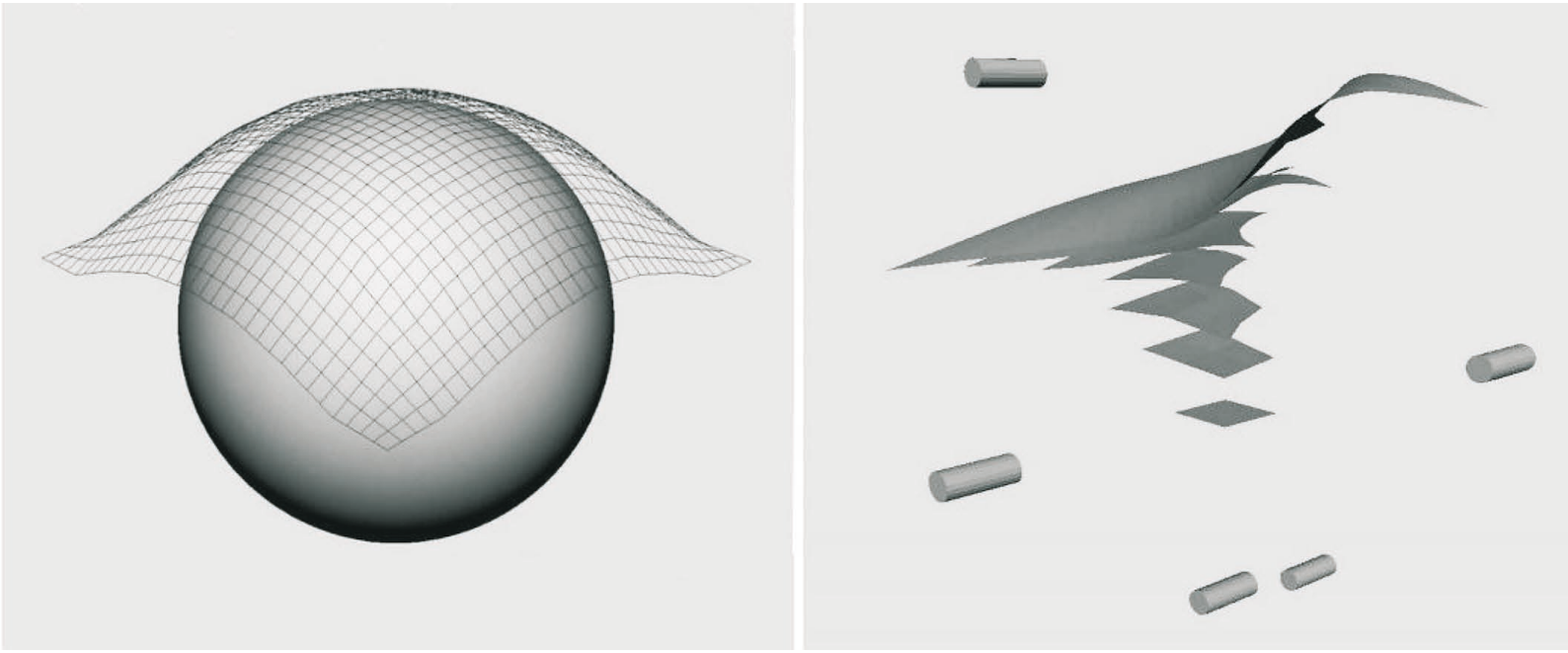


Fig. 20a-b. Square surface modeling environments a) modeling in a simulated gravity field, b) modeling in the force field of repellents; source: M. Hemberg et al. 2007.

A complex task would be to create a prescription system that would allow manual management of the growth process of a given surface. This is mainly due to the difficulty of imagining what a given rewriting system will look like after repeated iterations. Environmental influences only exacerbate this problem. Furthermore, additional complications would arise from the difficulty of ensuring that the rewriting system is syntactically correct. With this in mind, the Evolutionary Algorithm (AE) has been incorporated into Genr8. AE automatically generates selectively adaptive and syntactically correct rewriting rules. The designer exercises high-level control over the process by defining the fitness function and environment [M. Hemberg et al. 2007].

Genr8 uses Evolutionary algorithms invented by O'Neill and Ryan in 1997 called Grammatical Evolution (GE) (Ryan, O'Neill 1998). EG is based on the Genetic Algorithm (GA), and its advantage is that it combines the features of Genetic Algorithms (GA) and Genetic Programming (GP) [M. Mitchell 1996]. It applies to standard genetic operations on a vector of fixed length expressed as integers. These numbers are then used to generate a specification of Backus-Naur Form (BNF) language structures, which provides the basis for implementation (numerically controlled devices). This grammar is represented by a set of production rules. This requires additional mapping that does not exist in traditional GAs. Grammatical Evolution provides genetic degeneracy. This means that there are multiple gene encodings that, once mapped, can be individually decoded. In the first step of this two-step genetic mapping process, BNF (in conjunction with Map L-System) Genr8 allows access to different HEMLS universes. The application of Grammatical Evolution (GE) routines in the Genr8 system allows a linear genome, as in the Genetic Algorithm (GA), to be mapped to a tree structure, as in GP genetic programming. This is achieved by mapping a set of integers to the desired language using a Backus grammar representation Naur Form (BNF). This technique can be applied to any language that can be represented by a context-free grammar. All language constraints are handled by BNF and GE, which ensures strict separation between the genome representation and the target language. An important part of Evolutionary Algorithms (EA) is fitness assessment, which guides the search for better solutions. In design, there is no general way to algorithmically determine a "good" surface. Developing a useful framework for assessing suitability for design applications is still an open research question [S.S.Y. Wong, K.C.C. Chan 2009].

Genr8 uses a fitness assessment scheme that gives the designer control over the evolutionary search at a high level. This was implemented as a multi-parameter fitness function. Each parameter represents a specific surface feature. The designer can set target values for each parameter, as well as weights to determine the importance of each criterion. Important criteria are: size (range in the x and y directions), symmetry, soft boundaries (wall interference is allowed but may be penalized as part of the fitness function), subdivisions (a measure of surface quality), smoothness and waviness (local and global measures of variability in Z direction) [M. Hemberg et al. 2007].

The challenge for the designer is to understand the abstract parameters and the behavior and results of the evolutionary algorithm together with the growth algorithm, and then to combine these activities with the geometric and spatial arrangement of the designed form. The designer gradually learns to recognize how tools negotiate between various constraints and performance criteria in the form development process. The key, then, is to understand how the available settings (including the environment) are related to this specific design goal.

Integration with Maya is seamless. GENR8 is implemented as a MEL (Maya Embedded scripting Language) command and is used in the same way as any other function in Maya.

However, MEL is based on a command line interface, and for simplicity, a GUI has been implemented. This allows you to set all operating parameters. The GUI also prevents the user from entering incorrect combinations. HTML help files are also available. Boundaries are set using regular Maya surfaces. You can draw arbitrarily convex surfaces and they will be treated as faces by GENR8. Attractors and repellents are placed using a special command. You can draw a curve and this curve will be used as the starting point for growth (it will replace the genotypically encoded axiom, which is always a regular polygon). Surfaces are drawn in separate layers. It is also possible to save both the grammar, the genome and the actual surface of Maja.

Genr8 is a collaborative creative tool, where the designer's role is highly personalized and the design process follows a unique trajectory. Since 2003, Genr8 has been used in student projects as part of the Emergent Design and Technologies (EmTech) program at Architecture Associate in London.

The goal of the Butterfly Machines project was to develop a chair (Fig. 21a-b). First, several sets of surfaces were generated using Hemberg Extended Map L-Systems (HEMLS) to intersect themselves. By gradually changing the parameters and the environment, a whole set of self-intersecting shapes was obtained. Each set is a parametric variation of the system resul-

Fig. 21a-b. Butterfly Machines project, a) modeling a self-intersecting surface in response on the assumed position of attractors and repellents in the force field, b) two selected variants; source: Steven Fuchs, 2005.

ting from rewriting the square (Fig. 21a). Each set had a different location of attractors and repellents. Their arrangement was adopted to promote the chance of self-intersection of the surfaces during the growth process. Each surface "grew" along the vertical axis (time axis). Two variants were chosen due to aesthetic considerations in relation to the amount of self-intersecting surfaces and ergonomic considerations (Fig. 21b). Surface ergonomics were assessed using a script in Digital Projects/Catia [U. Q'Reilly et al. 2004].

3.2. Genr8: 3D surface modeling and fabrication

Genr8 allows you to model and produce fundamentally different objects by using different methods of interpreting a given surface introduced as a "grain". These can be both flat 2D and bi-curvaceous 3D surfaces. The program is a collaborative creative tool where the designer's role is highly personalized and the design process follows a unique trajectory.

Three different Genr8 surfaces with the same values but with different degrees of complexity in their articulation (Fig. 22). These surfaces are triangulated and unfolded to create a pattern that can be used for laser cutting and scoring. The first column shows the contours obtained from the Genr8 surface and the second the triangulation obtained with Maya. The third column shows the developed version, and the last one is photos of the completed surfaces U. [Q'Reilly i inni 2004].

Genr8, having semi-automatic spatial sketching tools, can propose practical "design solutions" in a specific environment, as well as explore exploitation possibilities, which is inherent in the AE evolutionary algorithm. The results of the combined possibilities and limitations of the manufacturing and assembly processes included directly in the process of generating computational forms will be presented in an experiment with a variable curvature contoured surface (Fig. 23).

The experiment begins by describing the geometry of a surface with variable curvature as a system of tangent and perpendicular construction planes (Fig. 23). These flat-sided elements will then be used as input to the manufacturing process, which involves computer-aided laser cutting of sheet material. Genr8 was used to initiate the coevolution of two interlocking surfaces with increasingly complex geometric articulation. Therefore, a number of geometric constraints were used to select the parameters of the efficiency function, thus ensuring that the elements were properly distributed planarly [M. Hemberg et al. 2007].

Evolutionary tools were used to initiate a process in which two curved surfaces joined together according to geometric performance criteria. The experiment was based on geometric data obtained from several surfaces with different curvature. They are described as a system of perpendiculars and tangents. The geometric constraints concerned only the local curvature and were applied to the entire surface. In an environment defined by attractive and repulsive forces, many generations have been grown from two connected surfaces. Geometric features, such as local changes in curvature and the adopted surface direction (Normal), redefined the position and number of planes, crossing multiple populations (Fig. 24a). The Normal surface is defined as a vector unit for the local surface

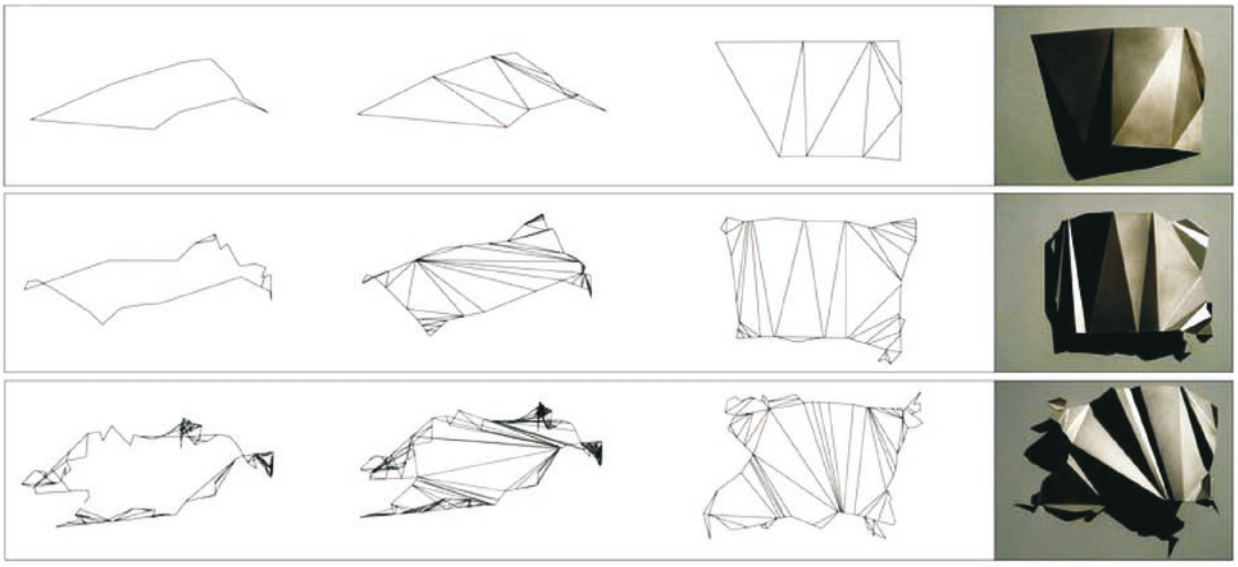


Fig. 22. 3D surface modeling and fabrication; source: C. Goncalves, 2004.

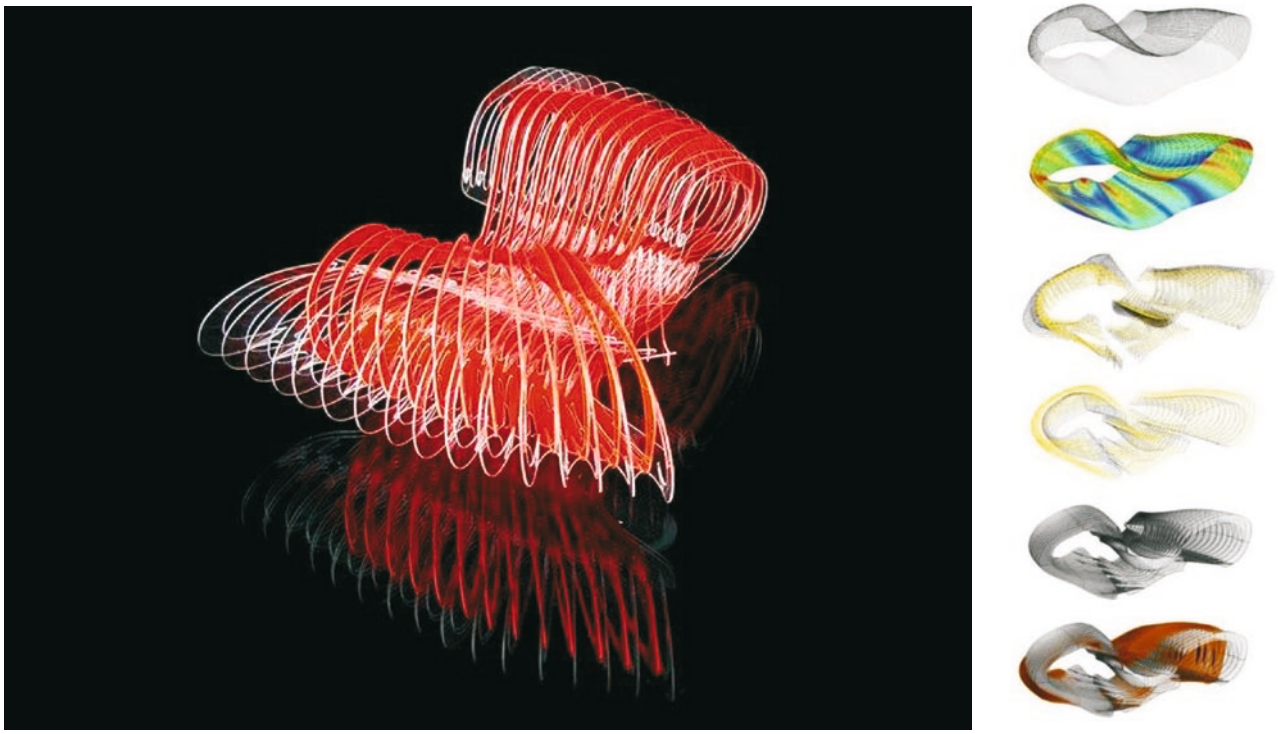


Fig. 23. A screenshot of the surface divided into production sections and the stages of the division and aggregation process; source: A. Menges, 2003.

and constitutes the first derivative of the position by indicating the actual dimensions for the designed surface. The basic geometric relationships were relatively simple, but through nonlinear evolution the surface articulation became more complex. The morphological process and common geometric performance criteria made it possible to maintain the logic of the material system so as to directly proceed to laser cutting of the necessary elements (Fig. 24b). The result of this experiment demonstrates a level of complexity and consistency that is difficult to achieve in conventional design approaches.

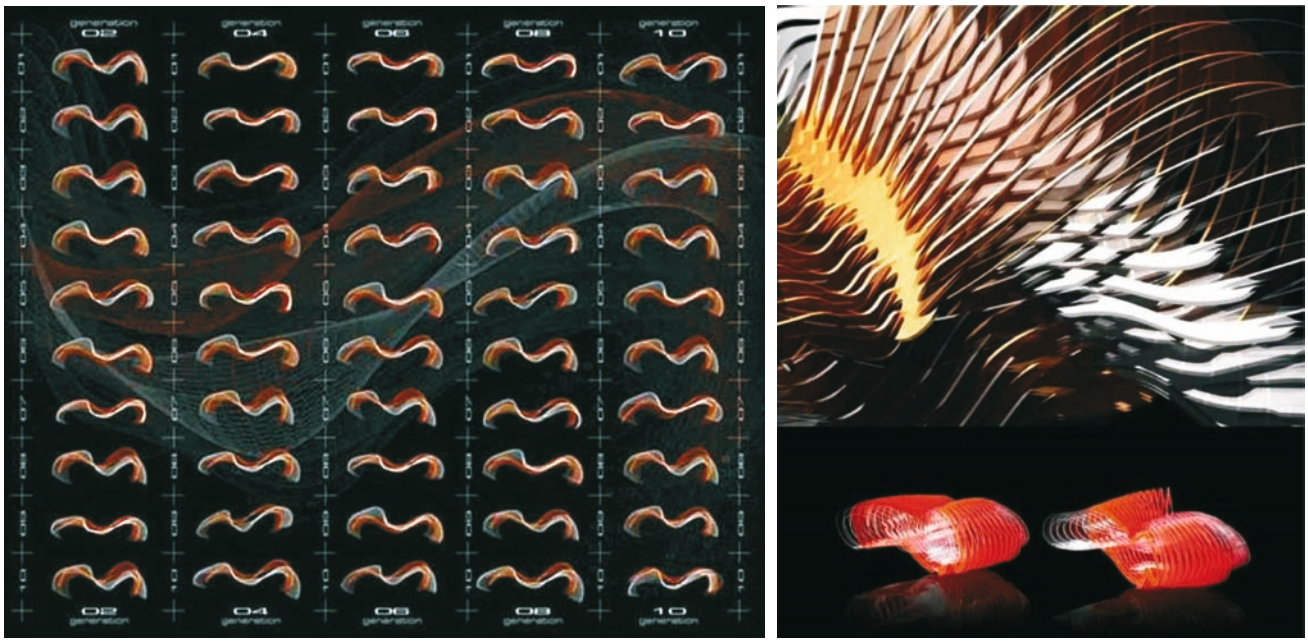


Fig. 24a-b. A population grown from connected surfaces and a fragment of the surface composed of cut elements cut from sheet metalin CNC technology; source: A. Menges 2003.

The design of the pneumatic strawberry bar cover was intended to exploit the evolutionary dynamics of reproduction, mutation, competition, and selection as design strategies (Fig. 24). The possibilities and limitations of form creation were examined from the initial stage of its generation to the actual production process. Performative patterns were sought that evolve as a species in populations and subsequent generations while maintaining structural load-bearing capacity and geometric features [M. Hemberg et al. 2007].

The starting point for the Genr8-based growth process was a relatively simple pneumatic component, geometrically defined to cut out two trapezoidal surfaces that will be sewn together during fabrication. Once filled with air, each component acquires a three-dimensional form determined by the length of the surface in relation to the points defining it. These simple geometric relationships, defined as an overall 3D cutting pattern, formed the basis for the subsequent evolutionary process. Instead of developing only one surface, a scheme based on the coevolution of three subpopulations was used. A feedback loop was initiated in which the most recently evolved surface was used as a bounding box for the current surface. This method preserved the properties of the pneumatic element in the larger system, but removed distinctions between environmental constraints and individual response. The next feedback loop used digital forming in dedicated membrane engineering software, and in addition, test results on physical models also influenced the evolutionary process and its evaluation.Genr8 generated over 600 generations in which 144 pattern species with appropriate geometric features were identified and cataloged. Since the structural behavior of the designed pneumatic system was based primarily on specific geometric relationships, individuals were selected that shared these geometric features. The genotype of these individuals contained the genomes of the three geometry-defining surfaces, establishing a degree of phenotypic plasticity that allowed the resulting pneumatic system to adapt to the limitations of digital pattern cutting and computer-aided manufacturing processes [M. Hemberg et al. 2007].

The examples presented show how two cooperating algorithms can be used, i.e. the Evolutionary Algorithm (AE) and the growth algorithm based on L-Systems. This also poses a significant challenge in understanding and applying the tools: how to best use Genr8 to achieve your goals. The designer's task, therefore, is to come up with a way to express his design through criteria in a way that is applicable within Genr8. A designer using Genr8 does not need to understand the algorithmic details of

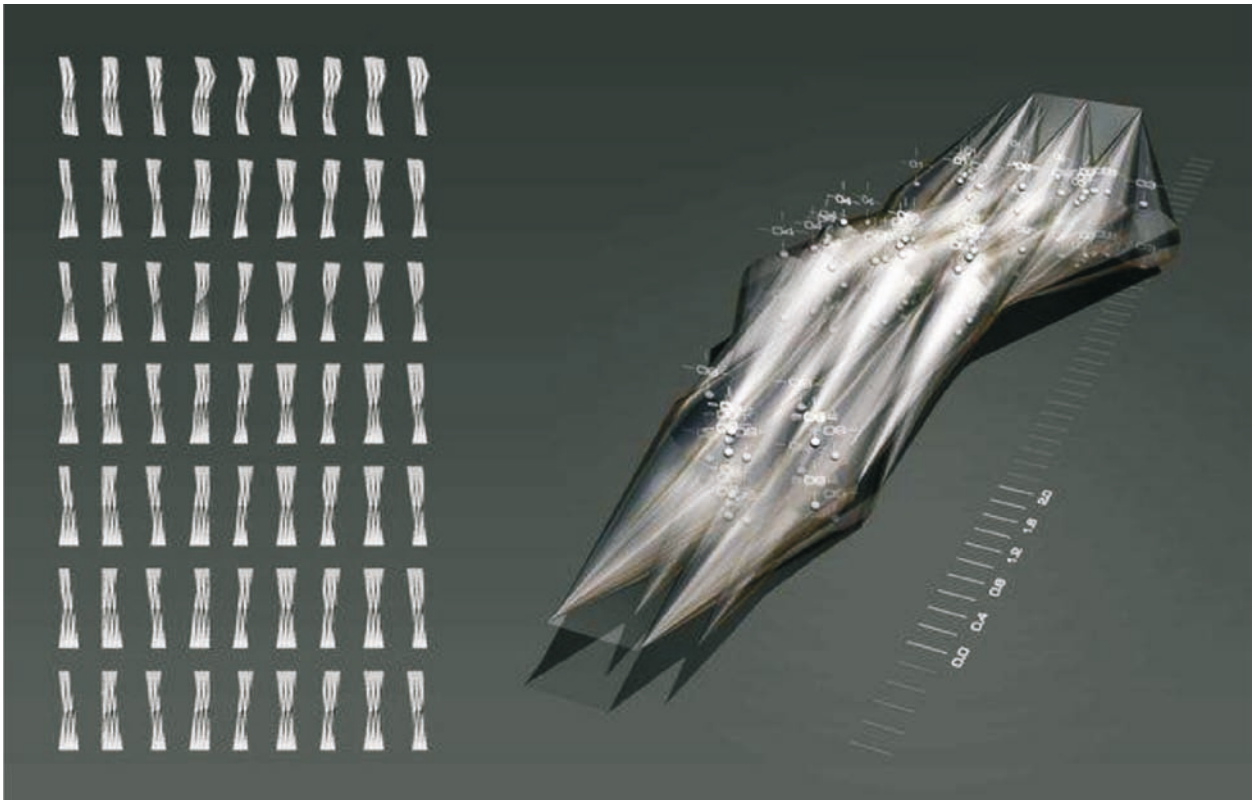


Fig. 25. Covering the strawberry bar; source: A. Menges, 2003.

the program, but must have a coherent and sufficiently accurate mental model of how the tools will behave.
Genr 8 is the first program aimed at architects that combines geometry modeling with a computer-aided fabrication process. Integrating evolutionary and generative computing methods and physical environment modeling techniques, this system is an essential tool for architects to creatively model surfaces and structures. For the first time in history, architects can model the shape of the form in correlation with the material and means of production at any scale.

CONCLUSIONS

The development of tools imitating form-forming morphogenetic processes presented here demonstrates that their applications in architectural design require integration and ease of use in CAD. Indeed, the implementation of generative and evolutionary tools to create design assumptions enables architects and engineers to use computational morphogenesis in the creation of new architectural forms and structures.

Morphogenetic design thus appears as the use of algorithmic processes or rules and principles to obtain design solutions. Rules for generative morphogenetic systems can be specified in various ways, e.g. by verbal grammars, diagrams, geometric transformations or command scenarios. Generative systems have varying degrees of control from automated to step-by-step manual. Using these methods requires the architect to approach the creative process differently than before. This is a significant change, because before the IT revolution, the theory and practice of architecture focused primarily on form, whose shape or relationships of parts were supposed to imitate the creations of Nature. New aspects of the creation of form were studied in Poland by Adam M. Szymski. Drawing on his knowledge of the human sciences, he conducted comparative analyses of the creative process and the processes of systemic design, and demonstrated the topological nature of the geometry of their systemic relationships, impossible to depict at the then level of development of computer technology [A. Szymski 1997].

Nowadays, architects are turning to computer-based morphogenetic tools and generative systems to study the influence of various factors on form. They borrow them from other disciplines and use them to design buildings and materials. The most widely used are: cellular automata, L-Systems, fractals, Voronoi diagrams, shape grammars and genetic algorithms. This is because the development of computational morphogenesis follows the behaviour of natural systems resulting from internal interactions between different layers of information. This method supports a computational framework for creating layers of information for modelling morphogenesis. The complexity of this framework cannot be simplified to a top-down system due to emergent behaviours that result from interactions between low-level elements. Therefore, computational morphogenesis explores the links between the two levels of micro and macro interactions to ensure self-organisation in the internal components of the form.

The development and dissemination of morphogenetic design tools in CAD integration is a promising alternative for future climate-change and sustainability-oriented architecture.

LITERATURE

1. **10 AE 2009,** only available on-line: https://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad10/w10.htm
2. **Darwin Ch. (2009),** *The Origin of Species*, Signet, Reprint, 150th Anniversary Edition, London.
3. **Dawkins R. (1986),** *The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe Without Design*, Norton & Company, Inc, London.
4. **Frazer J. H. (1974),** *Reptiles*, "Architectural Design"*,* April, 231-239.
5. **Frazer J. H. (1990),** *a Genetic Approach to Design – Towards an Intelligent Teacup,* in: *The Many Faces of Design*, Nottingham.
6. **Frazer J. H. (1995),** *The Evolutionary Architecture,* AA publication. London.
7. **Frazer J. H. (1997),** *The Groningen Experiment. Global Co-operation in the Electronic Evolution of Cities*, CAADRIA'97, Proceedings of the Second Conference on Computer Aided Architectural Design Research in Asia, April, 345-353.
8. **Frazer J. H. (2002),** *Creative design and the generative evolutionary paradigm* in: *Creative Evolutionary Systems*, red. P. Bentley and D. Corne, Morgan&Kaufmann Publishers, San Francisco, 253-257.
9. **Frazer J. H.**, **Connor J. M. (1979),** *a conceptual Seeding Technique for Architectural Design*, PArK 79, Proceedings of the International Conference on the Application of Computers in Architectural Design, Berlin (Online Conferences with AMK), 425-434.
10. **Frazer J. H., Frazer J. M. (1996),** *The Evolutionary Model of Design*, in: Asanowicz A., Jakimowicz A. (ed.) *Approaches to Computer Aided Architectural Composition*, Technical University of Bialystok, Białystok, 105-117.
11. **Frazer J. H., Frazer J., Liu X., Tang M., Janssen P. (2002),** *Generative and Evolutionary Techniques for Building,* Generative Art 2002: 5th International Generative Art Conference GA 2002, 11-13 December, Italy, Milan, 301-316.
12. **Frazer J. and Janssen P.,** *Generative and Evolutionary Models for Design*, School of Design,

Hong Kong Polytechnic University, Hong Kong (only available on-line: https://www.google.com/search?client=firefox-b-e&sca_esv=44922cbe319e9a1e&q=Frazer+Interactivator&tbm=isch&source=lnms&sa=X&ved=2ahUKEwioq6PAxIyEAxXQHBAIHTduBfoQ0pQJegQIDBAB&biw=1819&bih=761&dpr=1#imgrc=VINyWSTy94L0qM [access: 05.12.2023].

13. **Frazer J.H., Silver J. M. (1994),** *The genetic language of architecture – from living systems to artificial life*, "Projects Review", Vol. July 1994, 198-205.
14. **Ganti T. (1971),** *Azelet prince-piuma*, Gondolat, Budapest; also: Ganli T. (2003) *The Principles of Life*, Oxford University Press, Oxford.
15. **Gould S.J. (1989),** *Wonderful Life: The burgess shale and the nature of history*, Vintage, London.
16. **Hemberg M.**, **Menges A., Q'Reilly U-M, Jonas K. (2007),** *Genr8: Architects' Experience with an Emergent Design Tool*, in: J. Romero, P. Machado (ed.), *The Art of Artificial Evolution*, Springer Berlin-Heidelberg, 167-188.
17. **Holland J.H. (1975),** *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI, USA.
18. **Holmes B. (2005),** *Alive, The race to create life from scratch*, "New Scientist" Iss. 2486, 12 February 2005, 28-34.
19. **Janssen P., Frazer J.H., Tang M-xi. (2002),** *Evolutionary Design Systems and Generative Processes*, *"Applied Intelligence"*, Vol. 16**,** 119-128.
20. **Janssen P., Frazer J. H., Liu Xi., Tang M. (2003),** *Evolution Aided Architectural Design: An Internet based evolutionary design system,* Proceedings of the 9th Europe IA International Conference, 157-188.
21. **Janssen P., Frazer J. H., Liu Xi., Tang M. (2004),** *Evolutionary Design Exploration Systems*, Proceedings of the International Conference on Construction Information Technology (INCITE 2004), 123-132.
22. **Januszkiewicz K. (2010),** O projektowaniu architektury w dobie narzędzi cyfrowych. Stan aktualny i perspektywy rozwoju, Oficyna Wyd. PWr, Wrocław.
23. **Januszkiewicz K. (2016),** *Projektowanie parametryczne oraz parametryczne narzędzia cyfrowe w projektowaniu architektonicznym*, „Architecturae et Artibus", Vol. 8, No 3, 43-60.
24. **Krull F.N. (1994),** *The Origin of Computer Graphics within General Motors,* "IEEE Annals of the History of Computing", Vol. 16, No. 3, 40-55.
25. **Menges A. (2008),** *Integral formation and materialization: computational form and material gestalt*, in: *Manufacturing Material Effects*, Kolarevic B. and Klinger K. R. (ed.), Routledge, New York, 195-210.
26. **Mitchell M. (1996),** *An introduction to genetic algorithms*, MIT Press, Cambridge MA, USA.
27. **Paszkowska-Kaczmarek N. (2022),** *Problem mimesis w architekturze w dobie morfogenetycznych narzędzi projektowania*, rozprawa doktorska niepublikowana, promotor K. Januszkiewicz, ZUT, Szczecin.
28. **Prusinkiewicz P., Lindenmayer A. (1990),** *The algorithmic beauty of plants*, Springer-Verlag New York Inc., New York.
29. **Q'Reilly U-M., Hemberg M., Menges A. (2004),** *Evolutionary Computation and Artificial Life in Architecture: Exploring Potential Generative and genetic Algorithms as Operative Design Tools*, *"Architectural Design"*, Vol. 74, No. 3, 48-53.
30. **Simms K. (1991),** *Artificial Evolution for Computer Graphics*, "Computer Graphics", Vol. 25, No. 4, 319 -328.
31. **Szymski A.M. (1997),** *Twórczość architektoniczna. Wstęp do teorii projektowania systemowego* (*elements of system designing theory*), Prace Naukowe Politechniki Szczecińskiej, nr. 101, Instytut Architektury i Planowania Przestrzennego, Szczecin.
32. **Wong S.S.Y., Chan K.C.C. (2009),** *EvoArch: An evolutionary algorithm for architectural layout design*, "Computer-Aided Design", Vol. 41, No 9, 649-667.