Generative and Evolutionary Models for Design

John FRAZER and Patrick JANSSEN
School of Design, Hong Kong Polytechnic University, Hong Kong

1 INTRODUCTION

This paper describes three design models that make use of generative and evolutionary systems. The models describe overall design methods and processes. Each model defines a set of tasks to be performed by the design team, and in each case one of the tasks requires a generative or evolutionary design systems. The architectures of these systems are also broadly described.

Generative design systems are used to generate large numbers of design alternatives that differ significantly from one another. These systems define a complex growth process that transforms an encoded seed into a design. By making small modifications to either the transformation process or the seed, alternative designs can be generated.

Evolutionary design systems are used to evolve designs that are adapted to their environment. These systems are loosely based on the neo-Darwinian model of evolution through natural selection. Such systems consist of a cyclical process whereby whole populations of designs are continuously being manipulated in order to ensure that the population as a whole gradually evolves and adapts.

The following three models are presented:

- The first model proposes the use of a generative design system that invokes a set of architectural concepts that have been captured and codified by the designer. The system generates designs in response to an environment. Capturing and codifying architectural concepts allows the system to generate designs that embody these concepts. The designer can generate alternative designs by making small generative modifications. This is called the *concept-seedin*g model.
- The second model proposes the use of an evolutionary design system that has a generative system embedded within it. The generative system generates alternative designs in response to an environment. However, in this case, the system does not use any codified architectural concepts. The generative system is used to generate designs, with the evolutionary system being used to manipulate and evolve the generative modifications. This is called the *generative-evolutionary* model
- The third model combines the previous two models. This model also proposes the use of a generative-evolutionary system. The embedded generative system however takes the concept-seeding approach and invokes a set of codified architectural concepts to generate designs. This allows the generative-evolutionary system to evolve alternative designs that all embody the codified concepts. This is called the *combined model*.

2 CONCEPT SEEDING MODEL

Most designers employ a methodology that is highly personalised. However, when the designer’s body of work is taken as a whole, this methodology can also be seen as being highly generic. It is part of their working method and hence characterises their ‘style’ by which they are known. With an artist this style maybe a clearly recognisable graphic technique, as with say the drawings of Beardsley, a painting technique, such as the impressionists, or perhaps a distinctive choice of palette, say Titian. With architects and designers, the same is true. In some cases the style is also immediately recognisable from visual clues such as the work of Gaudi or Mackintosh. But often the style is more organisational or procedural or concerned with more abstract space and form such as with the work of Frank Lloyd Wright, although even with Wright the large roofs and low eaves of his early houses are an immediate stylistic give away.

This personalised but generic methodology is an abstract conception of what is common to all designs. Inside the designer’s office, this methodology often becomes formalised. It is common to find sets of standard details in architects’ offices that serve to economise in time, ensure details are well tested, but also to ensure a consistency of detailing and to reinforce the house style. In many offices this extends to design procedures, approaches to organisation and so forth. The same is true of industrial designers where again stylistic characteristics such as details, colour, preferred materials give economy, consistency, quality control and identifiable house style. Again the design office may have other characteristics that make it identifiable such as concentration on certain building types or product markets, or it may be more abstract characteristics such as particular interest in social or user centred concerns. Even these more abstract concepts can make the designer immediately recognisable, at least to fellow professionals, and often to the interested public. These characteristics even extend into areas of fashion, jewellery and accessories where particular designer labels are immediately recognisable on the street. The identifying characteristics often go through changes during the development of the designers, sometimes with abrupt changes as with Le Corbusier, but usually a continuous progression can be seen. The stylistic characteristics can continue with an office, studio or company, long after the death of the original designer.

2.1 The concept seeding approach

If the architectural concepts developed by designers could be captured and codified in a generic form, then a generative system might be able to invoke them to generate designs that embody the architectural concepts. This approach of capturing and codifying architectural concepts is referred to as *concept seeding* (Frazer 1974, 1979).

2.1.1 The concept seeding model

Figure 1 summarises the concept seeing model. The model identifies certain tasks to be performed by the design team, with each task requiring certain inputs and resulting in certain outputs. The inputs and outputs are depicted in boxes, where the tasks are depicted without boxes.

The concept-seeding model defines three tasks: codifying generative concepts, codifying architectural concepts and generating designs. First, a set of generation rules are defined that can develop the concept seed into a design. Second, a concept seed is defined that captures certain architectural concepts. Third, designs are generated in response to the design environment (which includes both the context and the criteria). This last task requires a generative system.

The tasks that the model identifies should not be assumed to be mutually independent. In most cases, the generation rules and the concept seed would actually be developed in parallel.

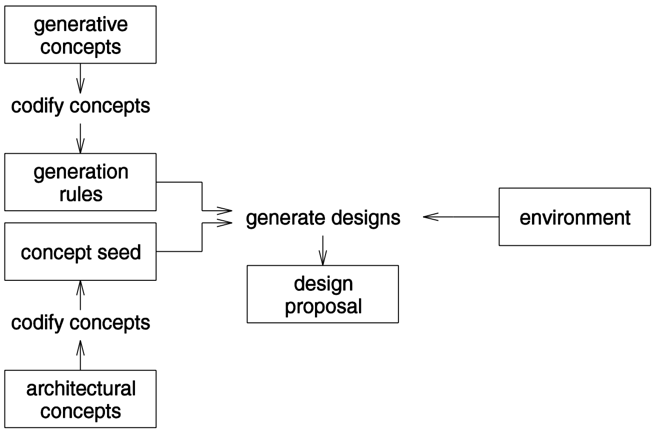

Figure 1: The concept-seeding model.

Codifying the architectural concepts can be done historically by analysis as manifest by endless attempts to recapture the nature of the paintings of Mondrian or the villas of Palladio. The success varies depending on the sensitivity of the re-creator. Often the results are crass particularly in the case of trying to create new works by dead artists. Mondrian would be horrified to see what some programmers might think would pass as his work. Often the help of the living designer can be involved. In the case of our work with Cedric Price and Walter Segal, the architects were alive and were involved in the capture process. Walter Segal is now dead but we still have the concept seed for his timber housing system captured for perpetuity in the software (Frazer 1982).

2.1.2 The concept seeding system

Figure 2 summarises the system required by the concept seeding model. In this case, information that is defined separately from the system is depicted in boxes, whereas information that is embedded within the system is shown without boxes.

The concept seeding system is, in itself, not a cyclical system. The system generates a single design proposal from a single seed in response to the design environment. However, the assumption is that the designer will explore a wide range of design possibilities by making small generative modifications to either the concept seed or the generative rules. A cyclical process guided by the designer therefore results.

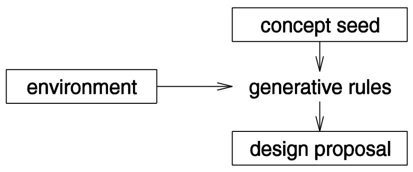

Figure 2: The concept seeding system

2.2 1968: Reptile System

The first attempted to realise this approach was the *Reptile System*, developed by Frazer from 1968 onwards (Frazer 1974). The Reptile system was developed as a folded plate space-frame system that was capable of creating a wide variety of enclosures from just two basic structural units. These units can be orientated in 18 different ways relative to each other resulting in over three hundred combinatorial possibilities.

Drawing a Reptile enclosure by hand was very tedious, and as a result a computer program was developed. Initially in 1967, a program was developed to aid in the process of drawing the enclosures and creating perspective views. At this stage, the computer hardware, in terms of speed and memory, but also in terms of graphical output capabilities, was extremely limited. Much of the programming effort therefore focused on optimising how these enclosures were stored and manipulated. However, the program was soon enhanced with additional features, and by 1971 a generative program was developed that was able to semi-automatically generate complete Reptile enclosures.

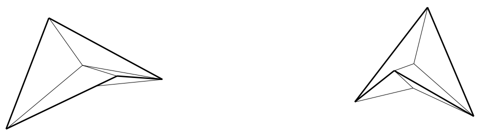

Figure 3: The two structural units.

The location and orientation of units are specified by four integers. The first two integers specify the location on a two dimensional grid with 60° axes; the third integer describes depth or level of the unit in the structure, and the fourth integer specifies the orientation of the unit. This whole description of the unit in space was originally packed into one word of the Cambridge Atlas Titan with a pointer to the next unit in the chain. This is define as the genetic *code script*.

The basic data structure is set up and manipulated by a series of machine code subroutines and functions. These allow the descriptions of the units to be retrieved, deleted or updated, and additional units to be inserted into the chain.

The seed is a minimal closed configuration of the units which includes units in all possible orientations but not necessarily all possible combinations. The development of a building is initialised by seeding the data structure with a description of the units making up the selected seeds with the units chained from the top of the seed downwards in a clockwise direction.

This initial seeded data structure is then manipulated by a series of Fortran subroutines which enable the seed to be grown, stretched, sheared, until the required building from is produced. Figure 4 (Frazer 1974).

The particular configuration of the units in the seed has a significant effect on the final over all building form. The information about the type and orientation of the starting unit and those adjacent to it in the data chain will vary from seed to seed and this will affect the choice in the program of procedure for infilling units in the data chain. This is analogous to the difference between the transition rules in Cellular Automata and the initial configuration (often referred to as the seed). In this early and restricted version of the program only two seeds were ever used, the knot (containing 42 units) and the other, the star (containing 72 units).

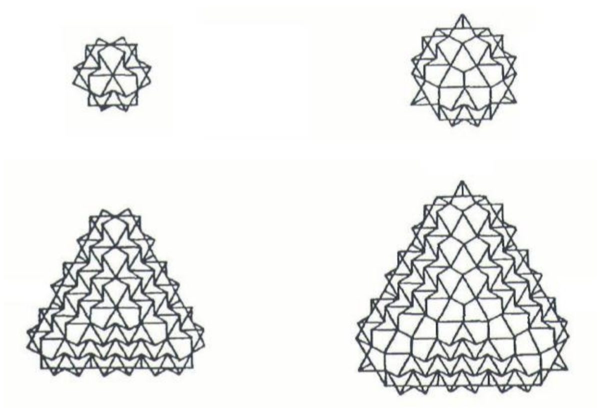

Figure 4: Building forms growing from two different seeds during the epigenetic process.

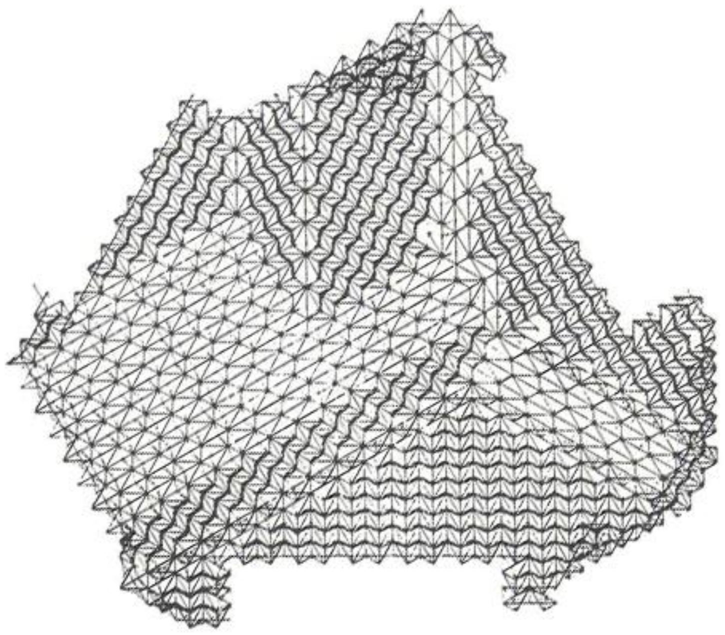

Figure 5: Plan of building generated from star seed.

2.2.1 Generalised version

A generalised, but still component based version of the program was later developed. The program required two kinds of information, first, the conceptual model of the building information in its minimal coded configuration and second, a description of the actual components and details for the output stage.

The item description in the seed is not necessarily a building component but is more usually an assembly of components in a key configuration such as a corner or change of direction. The item is again located by just one point and this point is usually taken as a change in surface direction to facilitate daylight and similar calculations.

2.2.2 Extended version

The concept of cultivating the seed to produce different buildings has been extended to the concept of mutating details of the seed to produce variants to overcome the great increase in environmental variety encountered with more general purpose building construction. Individual mutations of the assemblies can be developed interactively and stored as variants which are referred to by an additional set of digits in the item description on the data chain.

As with the Reptile program all the necessary information for the creation of new items in the data chain is derived from the type, location and orientation of the item in the seed and those adjacent to it in the chain at the point where a move or change is to be made. An important feature of the program is that it only takes account of dimensional co-ordination if this is essential to the concept. It does not rely on modular co-ordination or a grid but computes the distances between locating points of items in the chain, to determine modules appropriate to the construction being considered.

In operation the configuration of the minimal construction or seed is compared with the user's requirements for a particular building, and the seed is grown, stretched, deformed and pruned, until it conforms to these requirements. For the same set of building requirements many forms of building may be possible even from the same seed and also the same form of building can be produced by several different cultivation routines. The seed is compared with the brief for the building and the data structure is modified in two ways. First the quantifiable and specific parts of the brief are dealt with as far as possible in the form of optimisation routines. Alternative strategies for cultivating the seed are automatically evaluated and the program adjusts itself on a simple heuristic basis to adopt those tactics that have proved most successful in previous attempts. Second, the user evaluates a series of solutions produced by the first technique in terms of non-quantifiable criteria, including aesthetic judgements. The program can either learn to adapt to the aesthetic prejudices of a particular user, or the types of solution can be classified in terms of formal or aesthetic characteristics, which could then be requested by the user (Frazer 1979).

3 GENERATIVE-EVOLUTIONARY MODEL

The generative-evolutionary approach is described in *An Evolutionary Architecture* (Frazer 1995) as an attempt to achieve in the built environment the symbiotic behaviour and metabolic balance that are characteristic of the natural environment. The generative-evolutionary model therefore proposes the evolutionary process of nature as the generating process for architectural form. The profligate prototyping and awesome creative power of natural evolution are emulated by generating virtual architectural designs, which respond to changing environments. Successful developments are encouraged and evolved. Architecture is considered as a form of artificial life, subject, like the natural world, to principles of morphogenesis, genetic coding, replication and selection (Frazer 1995).

Such evolutionary systems consist of a cyclical process whereby whole populations of designs are continuously being manipulated. A generative system uses code-scripts of instructions in order to produce computer models of alternative designs. These designs are used to simulate the development of prototypical forms that are then evaluated on the basis of their performance in a simulated environment. By mutating and manipulating the code scripts, new forms are generated. Very large numbers of evolutionary steps can be processed in a short space of time and the emergent forms are often unexpected.

3.1.1 Typical evolutionary model

Most engineering applications of the evolutionary approach are interested in convergence on an optimal solution in response to clearly defined computable criteria for selection. Typically, such systems predefine an existing design and those parts that are thought to require improvement are parameterised.

Figure 6 summarises the typical evolutionary model. This model identifies two tasks: codifying the parametric model and evolving designs. First, mapping rules that specify how the parameters should be mapped to the parametric model, and the evaluation rules that specify how the model should be evaluated are defined. Second, designs are evolved in response to the environment.

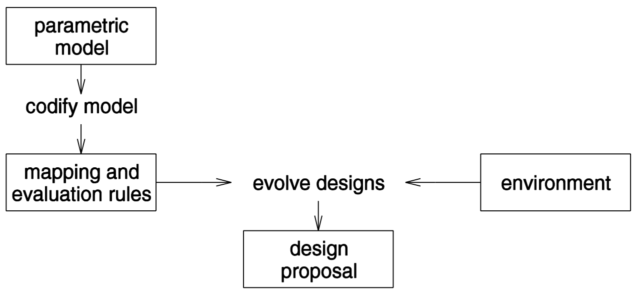

Figure 6: The typical evolutionary model

3.1.2 Typical evolutionary system

Figure 7 summarises the typical evolutionary system. A mapping rules produces the design from an encoded set of parameter values by inserting the values into the parametric model. The evolutionary system evolves these parameter values. We define this as *convergent evolution by natural selection*.

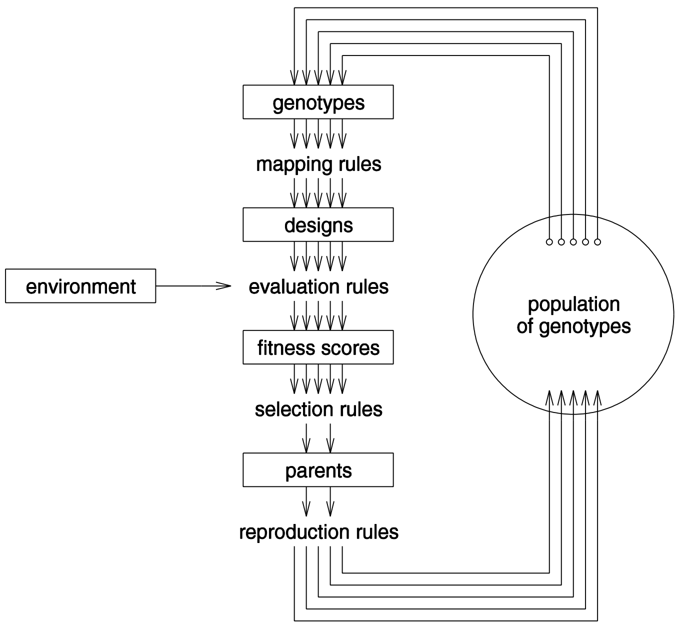

Figure 7: The typical evolutionary system

Although this approach has been very successful, the variability of the design alternatives that are produced is very limited. Since the designs are all based on the same parametric model, the designs will all have the same overall organisation and configuration. As a result, these programs are seriously limited in their ability to evolve new designs.

3.1.3 Convergent and divergent evolution

Convergent evolution by natural selection is not the only possibility. In the Origin of Species, Darwin talks first of the technique of artificial selection as practised by the racehorse or dog breeder. In our model *artificial selection* by the designer or user opens up the opportunity to demonstrate preference. It can be useful as a means of dealing with ill-defined selection criteria, particularly user centred concerns. It also provides the opportunity for the designer to use their experience and intuition to jump to faster results.

Nature also relies on divergence to keep a varied gene pool active to cope with sudden changes such as the increasing success of a predator or change in the environment. So to, our model provides for *divergent evolution* for the generation of alternative ideas. This gives a matrix (Figure 8) of four possible combinations of *natural/artificial selection* and *convergent/divergent evolution* (Dawkins 1986, Simms 1991, Graham 1993, Frazer 1995).

	convergent evolution	divergent evolution
natural selection by analysis	●	●
artificial selection by user	●	●

Figure 8: Matrix of four possible types of evolution.

These evolutionary methods are frequently based on techniques such as the application of genetic algorithms developed by John Holland (Holland 1975). Holland saw the genetic algorithm as a direct analogy with the evolutionary processes of nature. It is useful to remind ourselves that his original book was most appropriately titled “Adaptation in Natural and Artificial Systems ” (Holland 1975). The process of natural selection can generate a wealth of alternative experiments, and the better ones survive. There is no one solution, there is no optimal solution, but there is continuous experiment. Holland did therefore not see the genetic algorithm as a convergent system. Holland illustrates applications in genetics, economics, game-playing, pattern-recognition, control and function optimisation and the central nervous system.

3.2 Generative-evolutionary approach

A generative-evolutionary system replaces the mapping step with a generative step. (See Figure 7 and Figure 10.) This generative step consists of an embedded generative system. Dawkins described such an approach in his book *The Blind Watchmaker* (Dawkins 1986), where he develops a simple demonstration of a generative-evolutionary system that allowed two-dimensional insect-like structures to be evolved by artificial selection.

This approach requires that the generative rules be described in a genetic code. The code is then mutated and developed into a series of design models in response to a simulated environment. The models are then evaluated in the simulate environment and the code of successful models is selected. The selected code is then used to reiterate the cycle until a particular stage in the development a model is selected for prototyping in the real world.

The generative-evolutionary approach does not require the embedded generative system to incorporate the idea of concept seeding. Instead, the generative rules that are developed tend to be very general, and are not intended to reflect particular architectural concepts. However, all such rules contain biases and constraints, and as a result the forms produced may nevertheless all share certain characteristics.

3.2.1 The generative-evolutionary model

Figure 9 summarises the generative-evolutionary model. The model identifies two tasks: codifying the generative concepts and evolving designs. First, generation and evaluation rules are defined. The generation rules generate designs from encoded code-scripts and the evaluation rules evaluate the design that are generated. Second, designs are evolved in response to the design environment. This second task requires a generative-evolutionary system.

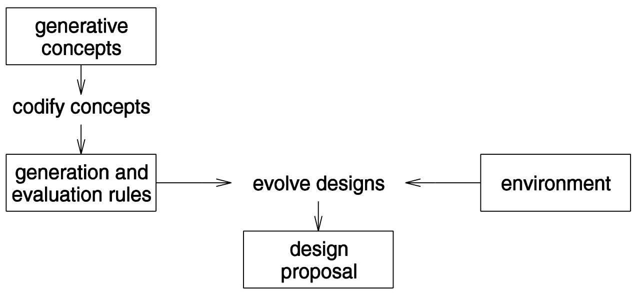

Figure 9: The generative-evolutionary model.

3.2.2 The generative-evolutionary system

Figure 10 summarises the system required by the generative-evolutionary model. Initially, a population of code scripts is created. The evolutionary process then consists of four steps. First, designs are generated from the genetic code scripts through some form of epigenetic development in an environment. Second, designs are evaluated by simulating and analysing their performance within their environment. Third, the most successful are selected. Fourth, the selected code scripts are transformed by genetic operators such as crossover and mutation. These four steps then repeat. At some point the process may be stopped.

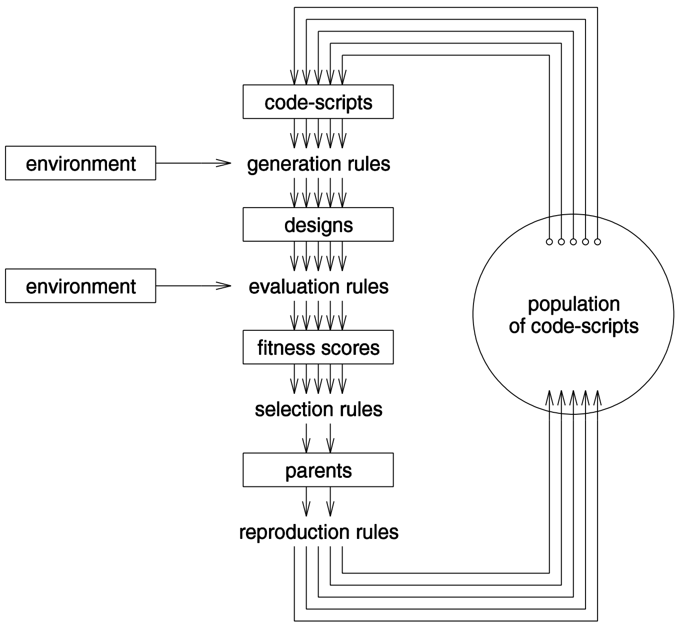

Figure 10: Generative-evolutionary system.

3.2.3 The role of the environment

An important aspect of this approach is that the generative system generates designs in response to an environment. The environment consists of the design context and the design criteria. The generative rules may require information about the context and criteria. This allows the developmental process that ‘grows’ a design to be an adaptive process. This adaptive process determines successive structural modifications in response to the environment and measures the performance of different structures in the environment. The adaptive process produces a sequence of structures from a set of operators. Information obtained from the environment influences the selections 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.

3.3 1995: The Interactivator

On the 25th January 1995, a generative-evolutionary experiment was launched to involve global participation in the evolution of a virtual environment. The experiment was at the centre of the exhibition entitled *An Evolutionary Architecture* being the work of Frazer, his wife and their students at the Architectural Association and the School of Design and Communication at the University of Ulster. It proposed the evolutionary model of nature as the generating process for architectural form.

The model proposed a generative-evolutionary system that was accessible via the Internet with the intention of encouraging wide participation thus creating biodiversity in the genetic design pool on which the model is dependent. Janssen was involved in developing a special demonstration version, known as The Interactivator. Although the theoretical system has been simplified, all the key elements are represented. Participation in the evolution of the model could be achieved either in the exhibition or in virtual form on the Internet.

The system is based on the sequential evolution of a family of cellular structures in an environment. Each cellular structure begins development from a single cell inheriting genetic information from its ancestors. Each cell in the cellular structure contains the same chromosomes, which make up the genetic code. The cells divide and multiply based on the genetic code script and the environment with each new cell containing the same genetic information.

3.3.1 Chromosome encoding

The data structure used to represent the cellular growth is based on a universal state space or iso-spatial data structure where each cell in the world has a maximum of 12 equidistant neighbours and can exist in one of 4096 states, the state of a cell being determined by the number of spatial arrangements of its neighbours.

The local environment of a cell in the world can thus be coded in a 12 bit binary string. For example, the string type (110110000110) would spatially represent a cell with six neighbours in particular positions.

Chromosomes control the growth and development of the cellular structure. A typical chromosome consists of the following 5 parts:

(	patrick@velcro.demon.co.uk	(1011010*11**)	(000011011010)	1	192.5	)
	origin of chromosome	condition	action	flag	strength	

The five parts are decoded as follows:

- *Origin of chromosome*: The Internet address site from where the chromosome is received.
- *Condition*: The local environment of a cell (* being a don't care situation)
- *Action*: The state of the cell in the next generation.
- *Flag*: Whether a chromosome is dominant or recessive.
- *Strength*: Fitness of the chromosome with respect to the environment.

3.3.2 Developmental processes

The developmental process of each member of the family consists of three parts – cellular growth, materialization and the genetic search landscape and is based on Goldberg's simple classifier system. A genetic algorithm is used to ensure that future generations of the system learn from the previous ones as well as provide for biodiversity during the evolutionary process. The three developmental processes each consist of cyclical processes.

- *Cellular growth*: Chromosomes are generated by either being sent in by a remote user, an active site or as a function of selection, crossover and mutation within cellular activity and are maintained in a main chromosomal pool. The physical environment determines which part of the main chromosomal pool becomes dominant. The local environment of each cell then determines which part of the genetic code is switched on. The cell then multiplies and divides in accordance with that genetic code (Figure 11 and Figure 12).
- *Materialization*: As cellular division takes place, unstable cells are generated. In the next generation this leftover material creates a space of exclusion within the cellular space. This space of exclusion interacts with the physical environment to create a materialization. Boundary layers are identified in the unstable cells as part of their state information and an optimised surface is generated to skin the structure. This material continues to exist throughout the evolution of the model and will initially affect the cellular growth of future generations (Figure 13 and Figure 14).
- *Genetic search landscape*: The selection criteria are not defined but are an emergent property of the evolutionary process, and is based on the relationship between the chromosomes, cellular structure and the environment over time. A genetic search landscape is generated for each member graphically representing the evolving selection criteria. Form, or the logic of form, emerges as a result of travelling through this search space (Figure 15).

Once chromosomal stability has been achieved, the parent cellular activity is terminated. The final cellular structure, the materialization and the genetic search space are posted out. A daughter cellular activity is then initiated from a single cell. The fittest chromosome from the parent generation are bred using selection, crossover and mutation and combined with the new list of dominant chromosomes from the main chromosome pool to form a new chromosome set for daughter generation.

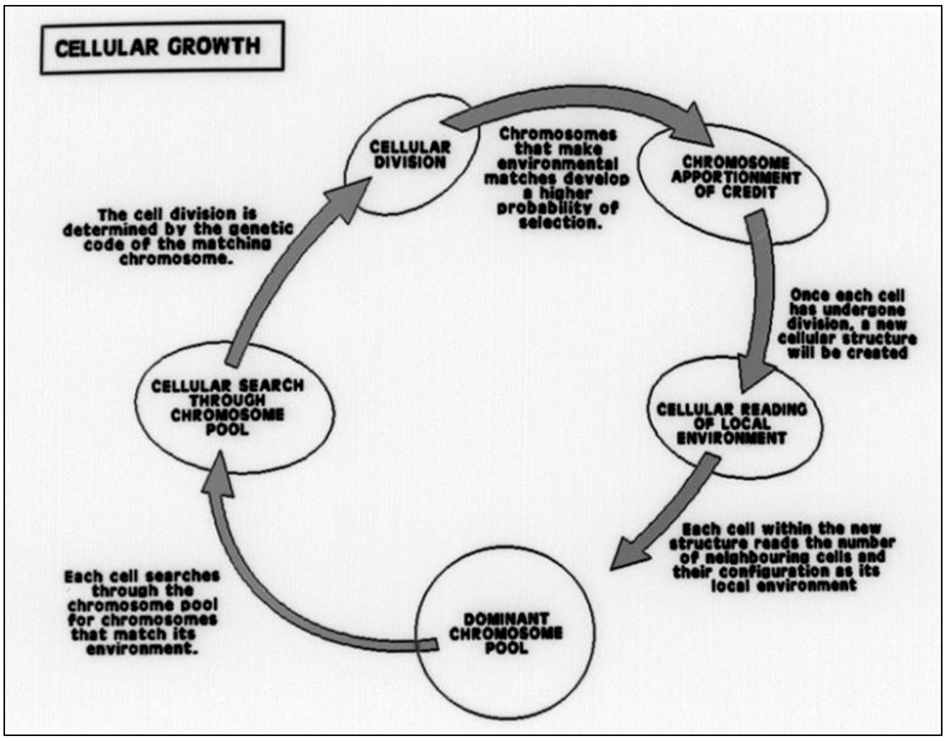
CELLULAR GROWTH
CELLULAR DIVISION
Chromosomes that make environmental matches develop a higher probability of selection.
CHROMOSOME APPORTIONMENT OF CREDIT
Once each cell has undergone division, a new cellular structure will be created
CELLULAR READING OF LOCAL ENVIRONMENT
Each cell within the new structure reads the number of neighbouring cells and their configuration as its local environment
DOMINANT CHROMOSOME POOL
Each cell searches through the chromosome pool for chromosomes that match its environment.
CELLULAR SEARCH THROUGH CHROMOSOME POOL
The cell division is determined by the genetic code of the matching chromosome.

Figure 11: Cellular growth process.

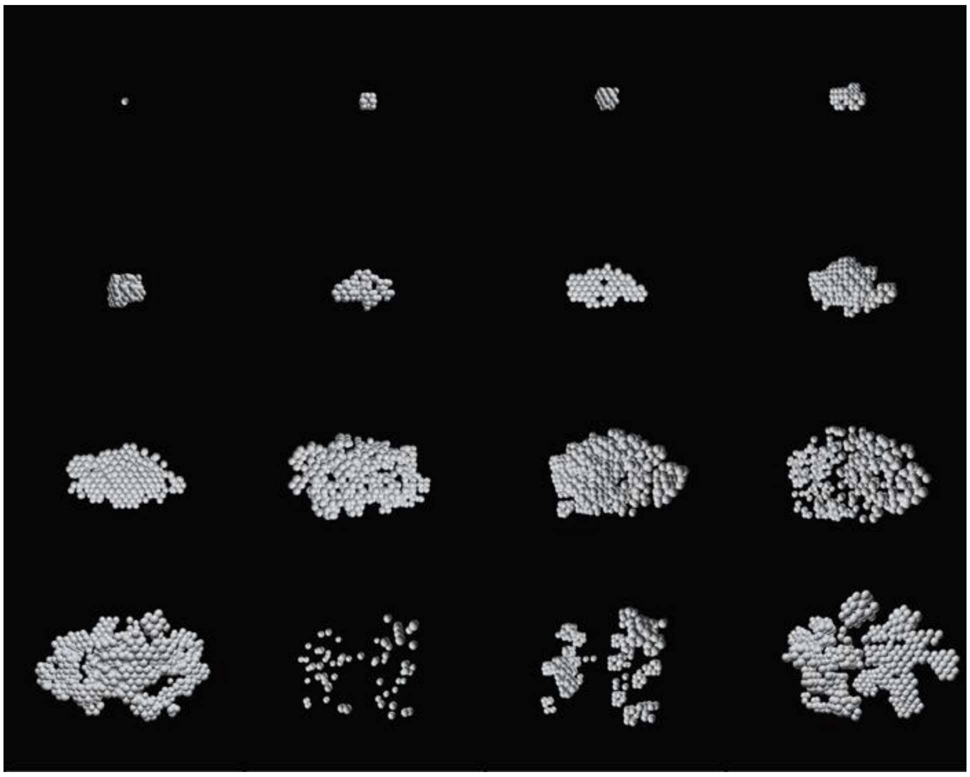

Figure 12: A sequence of cellular growths.

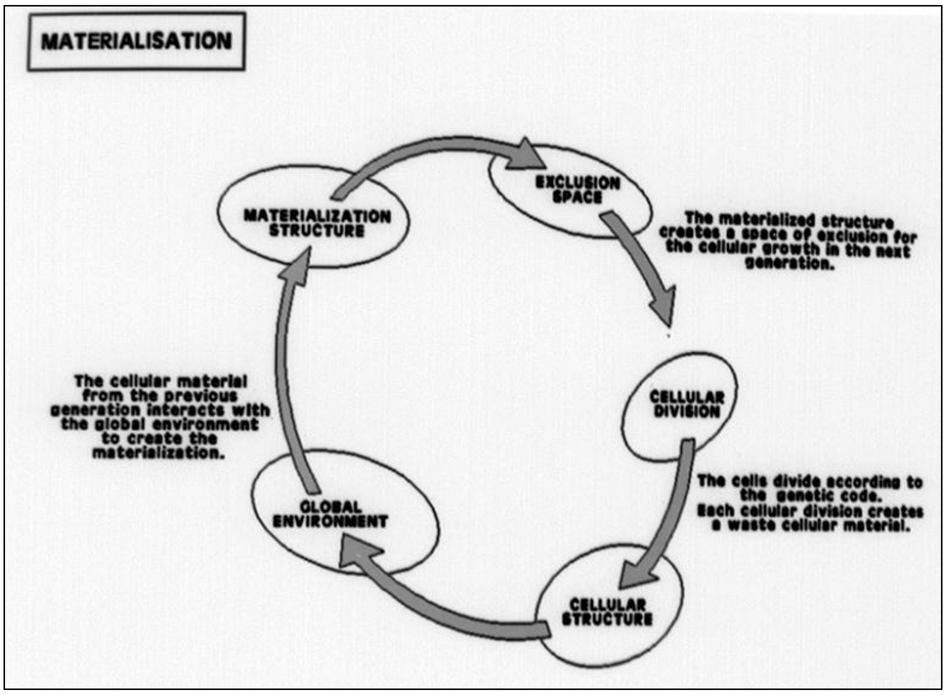
MATERIALISATION
MATERIALIZATION STRUCTURE
EXCLUSION SPACE
The materialized structure creates a space of exclusion for the cellular growth in the next generation.
CELLULAR DIVISION
The cells divide according to the genetic code. Each cellular division creates a waste cellular material.
CELLULAR STRUCTURE
GLOBAL ENVIRONMENT
The cellular material from the previous generation interacts with the global environment to create the materialization.

Figure 13: Materialisation process.

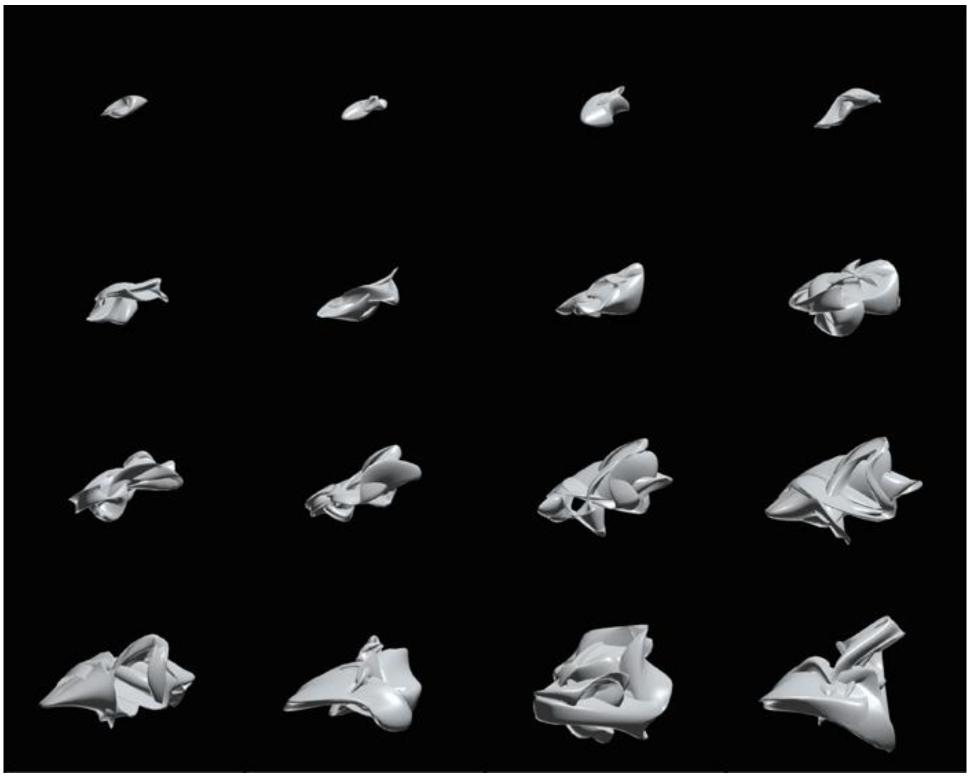

Figure 14: A sequence of materialisations.

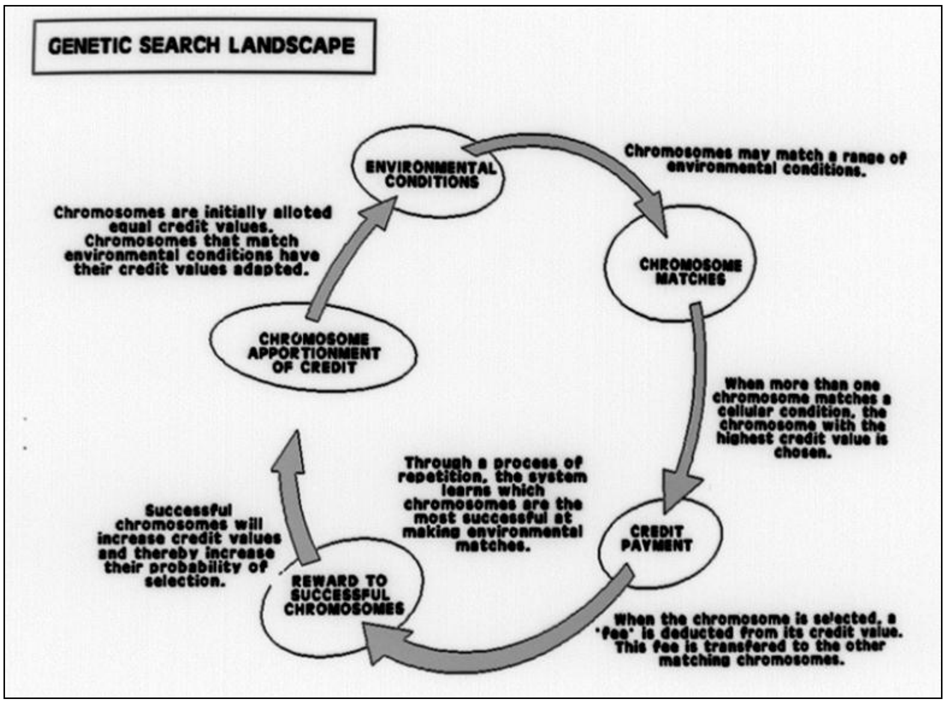
GENETIC SEARCH LANDSCAPE
ENVIRONMENTAL CONDITIONS
Chromosomes may match a range of environmental conditions.
CHROMOSOME MATCHES
When more than one chromosome matches a cellular condition, the chromosome with the highest credit value is chosen.
CREDIT PAYMENT
When the chromosome is selected, a 'fee' is deducted from its credit value. This fee is transfered to the other matching chromosomes.
REWARD TO SUCCESSFUL CHROMOSOMES
Successful chromosomes will increase credit values and thereby increase their probability of selection.
Through a process of repetition, the system learns which chromosomes are the most successful at making environmental matches.
CHROMOSOME APPORTIONMENT OF CREDIT
Chromosomes are initially alloted equal credit values. Chromosomes that match environmental conditions have their credit values adapted.

Figure 15: Genetic search process.

4 COMBINED MODEL

Combining the concept seeding model with the generative-evolutionary model result in a new type of model that integrates the advantages of both previous models. Such an approach was first proposed by Frazer in 1992 (Frazer 1990), and was further developed in the book *An Evolutionary Architecture* (Frazer 1995).

The concept seeding model allows designs to be generated that embody particular architectural concepts. However, the model does not use an evolutionary system, and as a result it is up to the designer to discover the generative modifications that produce the most suitable design. The generative-evolutionary model allows designs to be evolved that are adapted to their environment in complex ways. However, the designs do not embody any architectural concepts and therefore tend to be highly abstract. The combined model synthesises these two previous models, thereby allowing designs to be evolved that - as well as being adapted to their environment – also embody particular architectural concepts.

4.1 The combined approach

The combined model evolves the generative modifications that, in the case of the concept seeding model, the designer was required to make manually. These modifications introduced small changes to either the concept seed or the rules that transformed that seed. Representing these modifications as code-scripts allows an evolutionary system to evolve the modifications. The generative modifications encoded within the code-scripts are used to generate designs, these designs are then evaluated, and based on such evaluations new populations of modifications are created. This approach requires the range of valid generative modifications to be rigorously defined. Such a rigorous definition is required in order to allow rules to be developed that can autonomically create new generative modifications.

Such a combined model is applicable to a wide range of architectural concepts and geometries, all conceived as generative systems susceptible to development and evolution, all possessing the quality characterised by Viollet-le-Duc as ‘style’: ‘the manifestation of an ideal established upon a principle’.

4.1.1 The combined model

Figure 16 summarises the combined model. The model defines three tasks: codifying generative concepts, codifying architectural concepts and evolving designs. First, as with the previous model, generation and evaluation rules are defined. However, in this case, the generation rules must generate designs from the seed rather than from the code-script. Second, the concept seed is defined that codifies a set of architectural concepts. Third, the design alternatives are evolved in response to the design environment. This last task requires a generative-evolutionary system that includes concept seeding.

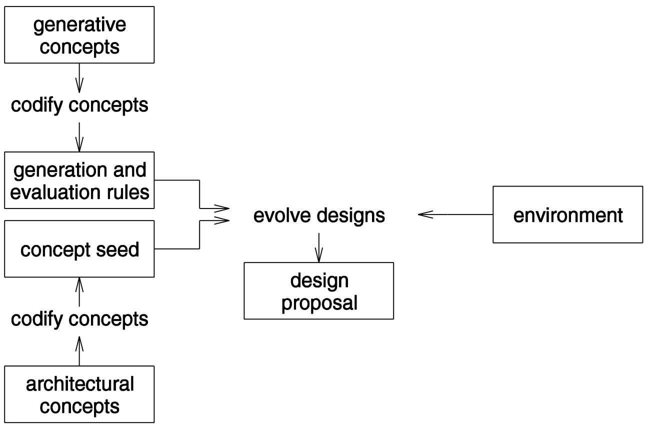

Figure 16: The combined model.

4.1.2 The combined system

Figure 17 describes the system required by the combined model. The system is similar to the generative-evolutionary system previously described. However, in this case, the embedded generative system generates designs from concept seeds. The generated designs will therefore embody the set of architectural concepts codified by the seed. The code-scripts encode the generative modifications. These modifications will make small changes either to the concept seed itself or to the generative rules that transform the seed. These generative modifications result in different designs being produced and evaluated. The generative modifications that result in designs with the highest fitness scores are then selected. The genetic operators are then used to create a new population of generative modifications, which will be used to generate a new population of designs, and so forth.

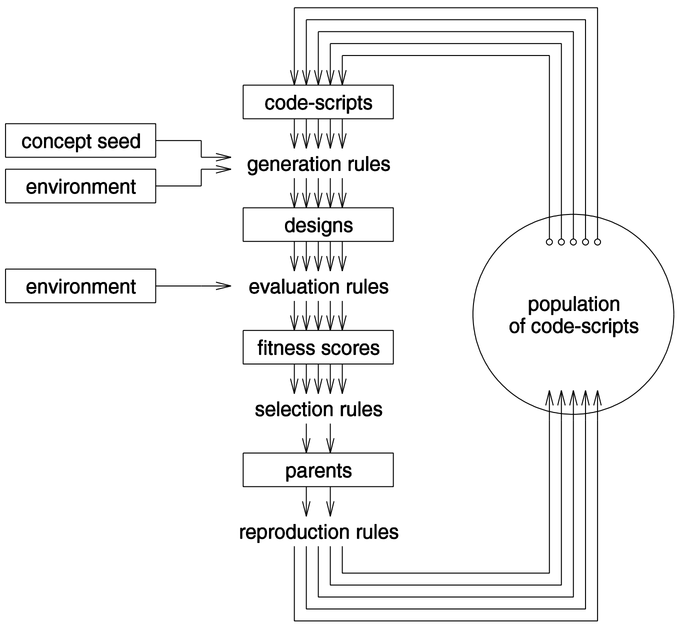

Figure 17: The combined system.

Within the area of generative-evolutionary design system, the concept seeding idea constitutes an important departure from the traditional research direction. Most researchers developing divergent evolutionary systems see any constraints and biases as negative side effects that must be minimised (although they can never be completely eliminated). The concept seeding approach, on the other hand, amplifies the constraints and biases, but at the same time ensures that they reflect the design ideas developed by the design team. The constraints and biases are therefore seen a positive rather than a negative phenomena.

4.2 2003: Janssen's model and system

Various aspects of the combined model have been demonstrated in isolation. For example a wide variety of generative systems using concept seeding have been developed. However, a comprehensive demonstration of the combined model has not yet been achieved. Janssen is currently developing such a demonstration. In addition, Janssen has further developed and refined Frazer's combined model (Janssen 2003a, 2003b).

4.2.1 Janssen's combined model

Broadly speaking, Janssen's model follows the pattern developed in Frazer's combined model. However, some fundamental modifications have been introduced. These modifications focus on two areas: the nature of the codified concepts and the structure of the overall model.

First, the idea of capturing and codifying the architectural concept has been refined in a number of ways. The aim of developing an architectural concept is to be able to use this concept in order to create a variety of designs. However, Janssen has defined 'variety' more precise definition by using a distinction made in evolutionary biology between diversity and disparity (Jaanusson 1981, Runnegar 1987, Gould 2000). Diversity refers designs that differ in the proportions and dimensions of their parts, but that share the same overall organisation and configuration of parts. Disparity refers to designs that have a fundamentally different organisation and configuration of parts.

The architectural concept must contain enough flexibility and adaptability to allow disparate designs to be created. The architectural concept should therefore not predefine the overall organisation and configuration of the designs, but should instead focus on defining the parts of the design and their interactions and overlaps. These parts, interactions, and overlaps might be thought of as defining the character of the designs without specifying their overall form. Janssen therefore refers to the set of architectural concepts to as the *character schema*.

The codifying of the character schema also differs from the Frazer model. In the Frazer model, the architectural concepts are codified in the form of a concept seed, but it is not clear to what extent this architectural concept affects the encoding of the other rules such as the developmental and mapping rules. In the new model, the codifying of the character schema is seen to affect all the evolutionary rules and data structures, including the developmental and mapping rules. The set of evolutionary rules and data structures are collectively referred to as the *evolution schema*. This evolution schema imbues the generative-evolutionary design system with biases and constraints that reflect the ideas developed by design team.

The second area that Frazer’s combined model has been modified is its overall structure. The two tasks of Frazer’s combined model – codifying concepts and evolving design – are still present in a modified form, sandwiched between two new tasks. At the beginning, Janssen has added a task that focuses on the development of the character schema; at the end, he has added a task that focuses on the development of a detailed design proposal.

These four tasks have been grouped into pairs to create two phases: the schema development phase and the design development phase. (See Figure 18.) This split reflects two different levels of environment. On the one hand, the schema development phase creates a schema that is specific to a general category of environment, referred to as the *niche environment* for the schema. 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 design must satisfy and the context within which the design will exist.

Splitting the design procedure into two phases in this way emphasises the fact that these two phases can be executed in very different ways. The first phase develops, and codifies the design character. This character will reflect the beliefs and preferences of the design team – called the *design stance* – and will be developed in response to the niche environment. This niche environment can be defined before any specific design environment has actually been found. As a result, this phase creates a generic design entity – the evolution schema – that can be reused many times within different projects.

The second phase evolves and details a design proposal for a specific design project. The proposal will be adapted to a design environment for the project, which will include aspects such as the site, spatial requirements, performance targets, and a budget. This phase therefore creates a design entity that cannot be reused.

Figure 18 summarises the Janssen model.

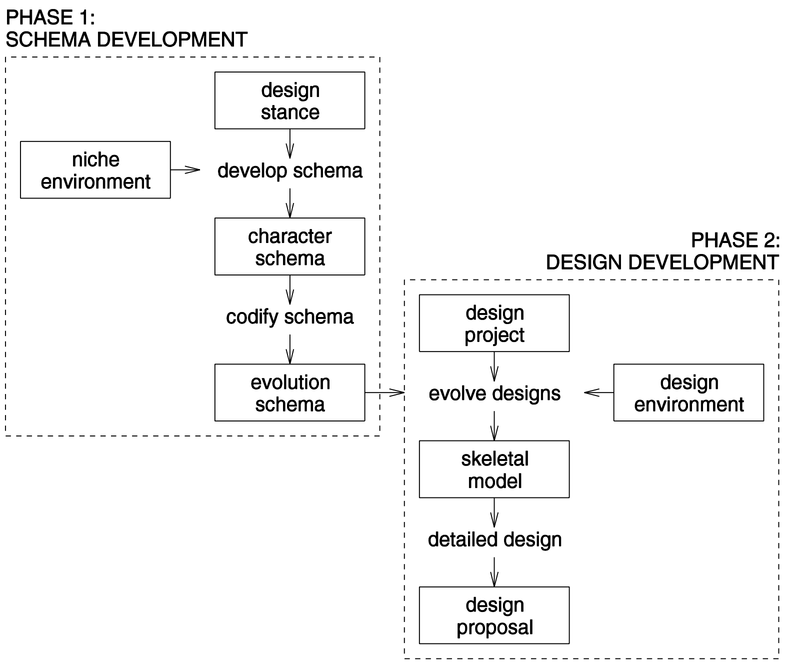

Figure 18: Janssen's model.

4.2.2 Janssen's combined system

In addition to the overall model, Janssen has also developed a detailed hardware and software architecture for implementing a generative-evolutionary system. In order to overcome certain limitations, this architecture departs from the typical architecture of such systems in three key ways.

First, the software architecture describes an evolutionary process that progresses in an asynchronous rather than a synchronous manner. Typical evolutionary systems create an evolutionary process where each individual in the population progresses through the various evolutionary steps in tandem with one another. With the asynchronous approach, individuals progress through the evolutionary steps irrespective of what other individuals in the population are doing.

Second, the software architecture proposes a core generative-evolutionary system that is highly generic and that can therefore be reused to evolve a wide variety of designs. This generic core does not actually predefine any of the evolutionary rules and data structures required in order to create the evolutionary process. In order to use the system, the design team first customises the system by specifying the rules and data structures as a separate set of data files that the system can then access. Together these rules and data structures constitute the evolution schema, which is the codified version of the character schema. They imbue the system with the biases and constraints that will ensure that the system will generate and evolve designs that embody a particular type of character.

Third, two additional evolutionary steps have been introduced: a validation step and a prediction step. The validation step allows the complex skeletal model produced by the generation step to be checked for inconsistencies and errors. The prediction step uses existing third-party simulation and analysis software in order to predict how the design would perform if built. Previously this step was part of the evaluation step. However, decoupling the process of making predictions from the process of tallying these predictions is necessary if the predictions are to be performed in an asynchronous manner.

Figure 19 summarises the software configuration of the generative-evolutionary system proposed by Janssen. Each evolutionary step downloads certain data from the population, applies the rules specified in the evolution schema, and then uploads the newly created data.

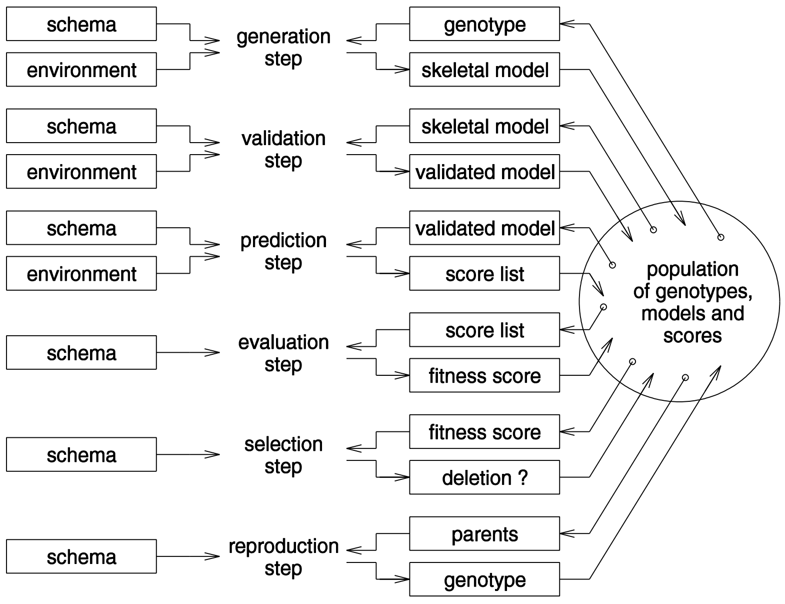

Figure 19: Janssen's generative-evolutionary software configuration.

The hardware architecture identifies a set of software components and proposes a networked configuration for these components. These components make extensive use of a number existing software technologies and systems.

Figure 20 summarises the hardware configuration proposed by Janssen. The architecture decomposes the system into a server program that maintains a population of designs within a database and a set of client programs that communicate with the server. The evolutionary steps are each performed by a separate client. Other clients include an initialisation client that allows the population to be initialised, and a visualization client that allows users to view a particular design in the population.

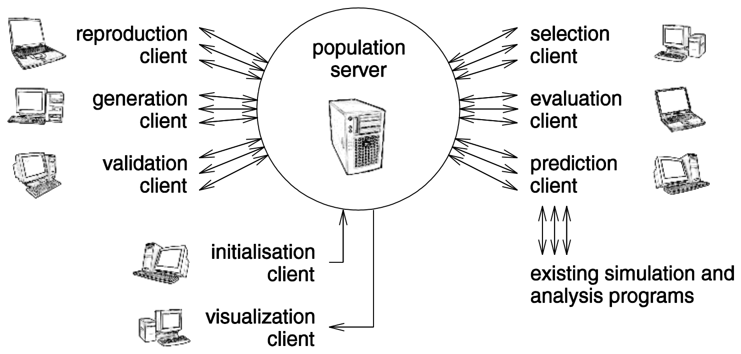

Figure 20: Janssen's generative-evolutionary hardware configuration.

The asynchronous evolutionary process means that any of the clients can be duplicated many times. For example, the prediction client running a particular simulation program can be duplicated, thereby allowing designs to be simulated in parallel. Furthermore, these clients can run on different operating systems, thereby facilitating the integration of third-party CAAD simulation and analysis programs.

5 CONCLUSION

The approach so far described implies some changes in architects' working methods. The generic approach adopted by many designers has to be made explicit, rigorous, and stated in terms which enable a concept to be expressed in genetic code. Ideally, the computer could deduce this information from normal work methods without any conscious changes being necessary. Architects have to be very clear about the criteria for evaluating an idea, and prepared to accept the concept of client- and user-participation in the process. The design responsibility changes to one of overall concept and embedded detail, but not individual manifestation. Overall the roles of the architect is enhanced rather than diminished as it becomes possible to seed far more generations of new designs than could be individually supervised, and to achieve a level of sophistication and complexity far beyond the economics of normal office practice.

REFERENCES

Dawkins, R. 1986. *The Blind Watchmaker*, Longman 1986.

Frazer, J. H. 1974. Reptiles, *Architectural Design*, April, p. 231-9.

Frazer, J. H. and 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, 1979), pp. 425 – 34.

Frazer, J. H. 1982. Use of Three-Dimensional Computer Input Devices to Encourage Public Participation in Design, *Proceedings of the conference on Computer Aided Design 82*, Butterworth Scientific, pp. 143 – 151.

Frazer, J. H. 1990, A Genetic Approach to Design – Towards an Intelligent Teacup, in *The Many Faces of Design*, Nottingham.

Frazer, J. H. 1992. Datastructures for Rule-Based and Genetic Design, in *Visual Computing – Integrating Computer Graphics with Computer Vision*, Springer Verlag.

Frazer, J. 1995. *An Evolutionary Architecture*, London: Architectural Association Publications.

Frazer, J.H. and Frazer, J.M.:1996, The Evolutionary Model of Design, *in* A. Asanowicz and A. Jakimowicz (eds.) *Approaches to Computer Aided Architectural Composition*, Technical University of Bialystok, pp. 105-117.

Frazer, J. 2002. Creative design and the generative evolutionary paradigm. In *Creative Evolutionary Systems*, eds. P. Bentley and D. Corne, 253-257. San Francisco: Morgan. Kaufmann Publishers.

Gould, S. J.: 2000, *Wonderful Life: The burgess shale and the nature of history*, Vintage, London.

Graham, P.C., Frazer, J. H. and Hull, M. E. C. 1993. The Application of Genetic Algorithms to Design Problems with Ill-defined or Conflicting Criteria, *Conference on Values and (In)Variants*, Amsterdam (Systematica, vol 10, 1995, pp 61 – 76).

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

Jaanusson, V. 1981. Functional thresholds in evolutionary progress, Lethaia 14:251-260.

Janssen, P., Frazer, J. H. and Tang, M. X. 2003a. Evolution Aided Architectural Design: An Internet based evolutionary design system, *Proceedings of the 9th EuropIA International Conference*, (to appear).

Janssen, P., Frazer, J. H. and Tang, M. X. 2003b. Evolutionary Design Exploration Systems, *Proceedings of the International Conference on Construction Information Technology (INCITE 2004)*, (to appear).

Runnegar, B. 1987. Rates and modes of evolution in the Mollusca, in K.S.W. Campbell and M.F. Day (eds.), Rates of evolution, pp. 39-60, Allen and Unwin, London.

Simms, K. 1991. Artificial Evolution for Computer Graphics, *Computer Graphics,* Vol. 25, No. 4, July, pp. 319 – 28.