MULTI-STAGE EVOLUTIONARY DESIGN

A METHOD AND SYSTEM FOR GENERATING AND ADAPTIVELY EVOLVING BUILDING DESIGNS

David Ryan Cook
May 2013

Submitted towards the fulfillment of the requirements for the Doctor of Architecture Degree

School of Architecture
University of Hawai‘i

MULTI-STAGE EVOLUTIONARY DESIGN

A METHOD AND SYSTEM FOR GENERATING AND ADAPTIVELY EVOLVING BUILDING DESIGNS

David Ryan Cook
May 2013

We certify that we have read this Doctorate Project and that, in our opinion, it is satisfactory in scope and quality in fulfillment as a Doctorate Project for the degree of Doctor of Architecture in the School of Architecture, University of Hawai'i at Mānoa.

Doctorate Project Committee

David Rockwood, Chairperson

Lance Walters

David Garmire

Kostas Terzidis

Abstract

More than three decades of interdisciplinary research have laid the foundations for a new breed of computational design tool that has the potential transform the practice of architecture. Described as evolutionary design systems (EDSs), the new tools are based on the evolutionary process in nature, and can not only analyze and evaluate, but also automatically generate and explore whole populations of alternative design proposals. A number of EDSs have been developed, implemented and tested with varying degrees of success, but there are still obstacles standing in the way of their practical application in architecture. Over the past decade, the focus of research has shifted to a large extent away from tackling these obstacles. Instead, recent research has focused on developing systems that employ more generic representations in order to generate a wider variety of forms, and more complex generative processes that pursue a closer analogy to developmental processes in nature in order to generate intricate forms from small amounts of input information. In opposition to this trend, and in support of the goal of extending the functionality of EDSs in order that they may be quickly made available as tools for architects to use in mainstream practice, this thesis takes the following steps:

- It argues that pursuing more generic generative representations or more complex generative processes beyond a certain point becomes purely academic, because highly generic or complex systems either produce forms that cannot be understood as building designs or else they lack evolvability. Consequently, such systems appear, at the moment, to be incapable of affecting the practice of architecture, at least in the short term. If, on the other hand, efforts are refocused on developing EDSs that employ generative systems constrained by architectural concepts, that path could quickly lead to the development of powerful tools that offer architects a range of potential benefits, including enhanced productivity and the ability to evolve designs for buildings that out-perform buildings designed in a conventional manner in key areas such as energy consumption.

- Drawing upon the literature as well as new research on current mainstream design methods, it analyzes challenges to the practical application of EDSs in architecture.

- Informed by this analysis, it develops the concept of a multi-stage EDS (MSEDS), an associated design methodology, and a system architecture. The MSEDS is a novel variety of EDS in which different parts of a design are generated and evolved separately in a series of stages according to an organizational scheme developed earlier in the design process.

- Within the framework of the MSEDS design methodology and system architecture, it develops an MDEDS design for adaptively evolving solutions to particular category of design problem - the skyscraper.

- Based on the MSEDS design, it develops a fully functioning MSEDS implementation for adaptively evolving skyscraper designs.

- Finally, it evaluates the performance of the MSEDS design and implementation by comparing the simulated performance of a design evolved by the MSEDS and the designs of three internationally renowned architects/firms. This verifies that the MSEDS design and implementation work as intended and further helps to validate the overall proposed multi-stage evolutionary design approach.

Contents

PART 1

INTRODUCTION

1.1.1 Overview

Conventional computer-aided design (CAD) tools boost productivity by automating tasks in the design process that would otherwise be carried out by hand. They also enable the designer to explore more of the design space through their ability to handle complex geometry, they enhance design communication with rotatable 3-dimensional models, physics-based photorealistic renderings and fly-though animations, and they even enable the designer to evaluate design ideas through a variety of performance simulation and analysis functions. However, conventional CAD tools do not play an active role in cognitive design; they leave the imagining of new designs up to the human designer.

Generative design systems, on the other hand, support the cognitive design process by generating form using rule-based procedures prescribed by the designer. They enable the designer to explore unimagined regions of the design space, supplementing his own creativity and freeing him from 'design fixation' and the limits of conventional wisdom.[1] However, simple generative design systems, that is, generative design systems that consist only of an algorithm for generating form, are restricted to exploring the design space one design proposal at a time, and they lack any means of evaluating alternative design proposals in order to guide the exploratory process.

Evolutionary design systems (EDSs) offer a way of overcoming the drawbacks of simple generative design systems. Loosely based on the neo-Darwinian model of evolution through natural selection, EDSs generate and evaluate not only one or several design proposals, but entire populations of alternative design proposals, and some are able to gradually hone in on better designs by replacing old populations of designs with new populations genetically bred from the fittest parent designs. EDSs have benefitted from a large amount of interest and research.[2] [3] [4]

The purpose of creating such systems is not only to enhance productivity by further automating the design process. Just as conventional CAD tools provide a number of other benefits over

[1] P.J. Bentley, "An Introduction to Evolutionary Design by Computers," in *Evolutionary Design by Computers*, ed. P. Bentley. (San Francisco: Morgan Kaufman Publishers, 1999), 1-73.
[2] J.H. Frazer, *An Evolutionary Architecture* (London: AA Publications, 1995)
[3] P.J. Bentley, ed., *Evolutionary Design by Computers* (San Francisco: Morgan Kaufmann Publishers, 1999)
[4] P.J. Bentley and D.W. Corne, eds., *Creative Evolutionary Systems* (London: Academic Press, 2002)

manual design processes, EDSs offer a number of other potential benefits over conventional cognitive design approaches. These include encouraging experimentation and innovation in the design process, facilitating the consideration of a wide range of design alternatives, flexibly accommodating changes, allowing the client more control over the design process, and enabling the evolution of designs for buildings that out-perform buildings designed in a conventional manner in key areas such as energy consumption.[5]

1.1.2 Evolution in Nature

In nature, evolution is the process by which all living organisms change and adapt to their environment over generations. The evolution of a population of organisms is driven by the continuous cycle of life and death of individual organisms in the population. This cycle can be broken down into 3 steps: *reproduction*, *development*, and *natural selection*.

- In reproduction, new organisms are conceived. Each new organism inherits from its parents a unique set of genes that encodes information about various biological traits - the *genotype*. In the case of sexual reproduction, new child genotypes are created by reshuffling the genes of the parents in a process called crossover, and by altering existing genes to create new genes through random accidental copying errors called mutations.
- In development, each new organism grows from a seed or egg into a fully developed organism - the *phenotype*. This takes place gradually as a result of processes of cell growth, differentiation and morphogenesis, which are triggered by genes in the genotype under suitable environmental conditions.
- In natural selection, the organism must successfully compete with others for limited resources in the environment, and the longer it survives, the more likely it is to reproduce. Some organisms possess traits or *adaptations* that help them to survive longer and/or to reproduce more. These organisms are more likely to pass on their genes each generation, causing those genes and adaptations to become more common in the population as a whole over time.

[5] Patrick H.J. Janssen, "A Design Method and Computational Architecture for Generating and Evolving Building Designs" (PhD diss., Hong Kong Polytechnic University, 2004)

Not all biological organisms have *evolvability* - the capacity to acquire useful adaptations through genetic changes in reproduction. Populations of organisms that lack evolvability may still be capable generating genetic diversity, but the developmental processes that produce the phenotype are unable to produce useful adaptations. As a result, such organisms do not become better adapted to their environment through natural selection.

1.1.3 Evolutionary Algorithms

Universal Darwinism is the theory that evolution by natural selection is *substrate-independent,* meaning that processes analogous to evolution in nature can emerge in systems outside the realm of biology.[6] This thinking has led to the successful application of evolutionary models to explain phenomena in a wide range of domains, including economics, psychology, medicine, computer science and physics.[7]

Evolutionary algorithms encode some of the underlying principals and mechanisms of evolution in algorithmic form so that they can be implemented as computer programs. When executed by a computer, they result in a computational process that is loosely analogous to evolution in nature. A population of individual entities is created and selectively bred over generations so that the entire population gradually evolves in a desired direction. Each individual entity in the population can represent an alternative solution to some type of problem, such as a parameter of an equation or even a complete building design. Thus, evolutionary algorithms are capable of evolving solutions to a wide variety of problems.[8] [9]

As in natural evolution, the solution being evolved has both a genotype representation and a phenotype representation. The genotype representation is an encoded version of the solution and the phenotype representation is the decoded genotype representation. The computational process can be broken down into an initialization step of randomly generating a starting

[6] R. Dawkins, "Universal Darwinism," in *Evolution from Molecules to Men*, ed. D.S. Bendall. (Cambridge: University Press, 1983)
[7] Wikipedia contributors, "Universal Darwinism," *Wikipedia, The Free Encyclopedia* http://en.wikipedia.org/w/index.php?title=Universal_Darwinism&oldid=540271232 (accessed 3 Mar. 2013)
[8] M. Mitchell, *An Introduction to Genetic Algorithms* (Cambridge: MIT Press, 1999), 15-16.
[9] D. Beasley, "Possible Applications of Evolutionary Computation," in *Evolutionary Computation 1: Basic Algorithms and Operators*, ed. T. Bäck et al. (Philadelphia: Institute of Physics Publishing, 2000)

population of genotype representations, and an evolutionary cycle comprising four steps: *reproduction*, *development, evaluation,* and *selection*.

- In reproduction, a new population of genotype representations are created by transforming parent genotypes into new child genotypes using *genetic operators* such as crossover and mutation, according to predetermined reproduction rules.
- In development, the new population of genotype representations are transformed into phenotype representations by applying predetermined developmental rules.
- In evaluation, the performance of each phenotype representation is evaluated with respect to one or more objectives. For each objective, an evaluation score is calculated using a mathematical function called the *objective function* and stored together with the solution.
- In selection, the evaluation scores of each alternative solution are summarized in a single figure indicating the merit of the solution - its *fitness*. Here, the mathematical function used to calculate fitness is called the *fitness function*. Solutions that rank higher in fitness are selected for reproduction and those not selected are deleted.

Many aspects of evolutionary algorithms depend upon the details of the problem to which they are applied. The evolutionary algorithm and the problem to be solved together make up an *evolutionary system* with variables, including the types of genotype and phenotype representations, the reproduction and developmental rules, the type of evaluation performed in the evaluation step, the form of the objective function, and the form of the fitness function. In order for the evolutionary algorithm to evolve solutions to the problem, these variables all have to be specified.

Evolvability of evolutionary systems is analogous to evolvability of biological organisms. Evolutionary systems that lack evolvability are able to generate diverse population of solutions, but they do not tend to increase in fitness over generations. As in natural evolution, evolvability is related to the nature of the developmental process used to generate phenotype representations from genotype representations.

A wide range of evolutionary algorithms exist that differ in implementation details and the type of problems they are designed to tackle. The four main types are *genetic algorithms*,[10] [11] *genetic programming*,[12] *evolutionary programming*,[13] and *evolution strategies*.[14] Of these, genetic algorithms are the best known and most commonly used.

1.1.4 Evolutionary Design

In the design domain, evolutionary algorithms are used to evolve populations of alternative designs.[15] [16] [17] In an evolutionary design system, each genotype representation is encoded information that can be used to construct a model of a design, and the corresponding phenotype representation is the fully constructed design model. The reproduction rules can be the typical rules for applying genetic operators used by the evolutionary algorithm, and the developmental rules are the generative or transformational modeling procedures used to produce the design model from the information encoded in the genotype representation. The evaluation step may consist of performing one or more measurements or simulations on the design model, which may incorporate information on the environment in which the design is to be realized, and the objective functions and fitness function are formulated to convert the evaluation scores into a single figure measuring how close the design comes to fulfilling certain design objectives.

There are two main approaches to evolutionary design: *evolutionary design optimization* and *evolutionary design exploration*.[18] Evolutionary design optimization uses evolutionary algorithms to optimize existing designs. In a typical evolutionary design optimization process, parts of an existing design that the designer wants to improve are parameterized, and the system evolves the parameter values. In this case, the developmental rules (modeling procedures) specifying

[10] J.H. Holland, *Adaptation in Natural and Artificial Systems* (Ann Arbor: University of Michigan Press, 1975)
[11] D.G.. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Reading: Addison-Wesley, 1989)
[12] J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (Cambridge: MIT Press, 1992)
[13] D.B. Fogel, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. (IEEE Press, 1995)
[14] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. (New York: Oxford University Press, 1996)
[15] J.H. Frazer, *An Evolutionary Architecture* (London: AA Publications, 1995)
[16] P.J. Bentley, ed., *Evolutionary Design by Computers* (San Francisco: Morgan Kaufmann Publishers, 1999)
[17] P.J. Bentley and D.W. Corne, eds., *Creative Evolutionary Systems* (London: Academic Press, 2002)
[18] Patrick H.J. Janssen, "A Design Method and Computational Architecture for Generating and Evolving Building Designs" (PhD diss., Hong Kong Polytechnic University, 2004)

how to construct the phenotype representation (parametric design) from the genotype representation (parameter values) are linear maps. Evolutionary design optimization systems are typically *convergent,* meaning that the entire population of alternative designs gradually adapt and improve in fitness until they all eventually resemble a single design that has the highest fitness. Numerous examples of evolutionary design optimization systems can be found in the literature.[19] [20] [21]

Evolutionary design exploration, on the other hand, uses evolutionary algorithms to generate and explore new designs, without any ambition to optimize them. In a typical evolutionary design exploration process, information encoded in the genotype representation is used to guide some kind of generative growth process that results in a new design. Often, this involves starting with a collection of components, such as geometric objects, and then, according to the information encoded in the genotype, selecting certain components and arranging them in a certain way to generate a design. In this case, the developmental rules specifying how to construct the phenotype representation from the genotype representation are non-linear maps. Evolutionary design exploration systems are typically *divergent,* meaning that the population of alternative designs does not adapt and improve in fitness but retains a certain level of diversity throughout evolution. Another way to put this is that evolutionary design exploration systems lack evolvability. Numerous examples of evolutionary design exploration systems can be found in the literature.[22] [23]

These two approaches have different limitations with regard to the types of designs they are able to evolve. Evolutionary design optimization is capable of adaptively evolving designs that have some optimized features, but the number and overall configuration of parts in the design are typically determined in advance and do not change during the evolution process. Accordingly, there is nothing really surprising or creative about designs evolved through this approach. The evolved designs can be said to lack *novelty*. Evolutionary design exploration, on the other hand, is capable of evolving a wide variety of different designs, but none of them

[19] D. Dasgupta and Z. Michalewicz, eds., *Evolutionary Algorithms in Engineering Applications*. (Dusseldorf: Springer-Verlag, 1997)
[20] L. Caldas, "An Evolution-based Generative Design System: Using Adaptation to Shape Architectural Form" (Doctoral dissertation, Massachusetts Institute of Technology, 2001)
[21] L. Caldas, "Generation of Energy-efficient Architecture Solutions Applying GENE_ARCH: An Evolution-based Generative Design System," Advanced Engineering Informatics, Vol.22 (2008), http://www.sciencedirect.com.
[22] Frazer, *Evolutionary Architecture*
[23] Bentley, *Evolutionary Design by Computers*

can claim to be optimal or well adapted solutions to the design problem because the system lacks evolvability and the evolutionary process is divergent. The evolved designs can characterized as lacking *optimality*.

These limitations are a consequence of the development process employed to build the phenotype representation (design model) from the information encoded in the genotype representation. In the case of evolutionary design optimization, the development process is nothing but a linear *mapping* process, wherein each gene undergoes a simple linear transformation into a corresponding parameter of a pre-constructed parametric model. This simple linear mapping process is the reason why the evolutionary process is convergent and optimization is possible, however it is also the reason why designs evolved by this approach vary in predictable ways from the pre-constructed parametric model. Meanwhile, in the case of evolutionary design exploration, the development process is a non-linear *generative* process, wherein the genotype specifies a set of rules and procedures that are performed to grow a new design from scratch. Such generative processes can produce novel and sophisticated designs, but they tend to have certain properties that seriously hinder adaptive evolution.

One of these properties is *epistasis*, which means the degree of dependency between multiple genes in a genotype. If a generative system has high epistasis, and they usually do, there may be many genes in the genotype representation whose effect on development of the phenotype representation relies to a large degree on other genes. For example, in a generative system where genes in the genotype representation specify geometric transformations, such as 'scale,' 'rotate', 'mirror,' etc., which are applied in succession to generate a design from some initial object, the transformation specified by each and every gene will have a very different effect on the resulting design, depending on which other transformations precede and succeed it. In such systems, individual genes effectively become elements of a single gene, in the same way that individual binary digits are elements of a single binary number. Changing any individual gene in an attempt to improve just one small part of the design will inevitably change the phenotypic effect of the entire genotype representation to yield an entirely different design. Under these circumstances, there can be no incremental improvement, making convergent evolution to optimal designs impossible.[24]

[24] Peter Bentley, "Aspects of Evolutionary Design by Computers," in Proceedings of the 3rd On-Line World Conference on Soft Computing:WSC3 (1998): 10-11, http://arxiv.org/html/cs/9809049/dss5.html.

Another problematic property of some generative processes is *interdependent elements*, which are elements of a generated design whose proper functioning relies to a large degree on other elements. Interdependent elements of generated designs (phenotype representations) are in some ways analogous to interdependent genes of epistatic genotype representations. If a generative system works with many interdependent elements, as is generally required to evolve sophisticated building designs, variation in the overall configuration of elements is highly restricted by numerous constraints on the relationships between interdependent elements that must be adhered to, in order for the design to work as a whole. For example, in a generative system where genes in the genotype representation specify rules for picking and arranging elements of a high-level design representation, such as floors, walls, doors, and staircases, it is difficult to design rules to prevent misplaced doors and staircases that lead to nowhere, etc. without overly restricting variation. Relaxing the rules a little allows more variation, but this can result in large numbers of *local optima*, which are designs that are better than most in the population but still lack all the adaptations of the best or *globally optimal* design. In complex systems with many interdependent elements, local optima usually include designs that are not even functional. Under these circumstances, it becomes difficult for the system to generate enough functional designs to provide the level of *adaptive genetic diversity* needed for evolution to progress from local to global optima. Evolution may thus commit itself too early to sub-optimal approximations of the best design, some of which are not even functional.[25]

Some researchers have pursued a third approach to evolutionary design that effectively tries to combine optimization with exploration: *integral evolutionary design*.[26] Integral evolutionary design uses evolutionary algorithms to create new designs from scratch *and* optimize them. The main strategy explored thus far is to employ low-level *generic representations* to overcome the problems associated with epistasis and interrelated elements.[27] [28] [29] Generic representations are phenotype representations (design models) based on low-level geometric

[25] Bentley, "Aspects of Evolutionary Design by Computers," 13-14, http://arxiv.org/html/cs/9809049/dss5.html
[26] Bentley, "Aspects of Evolutionary Design by Computers," 9, http://arxiv.org/html/cs/9809049/dss5.html
[27] P.J. Bentley, "Generic Evolutionary Design of Solid Objects using a Genetic Algorithm" (Doctoral dissertation, Division of Computing and Control Systems, Department of Engineering, University of Hudersfield. 1996).
[28] P. Baron et al., "A Voxel-based Representation for the Evolutionary Shape Optimization of a Simplified Beam: A Case-study of a Problem-centered Approach Genetic Operator Design," in 2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2), 1997.
[29] P. Baron et al., "A Voxel-based Representation for Evolutionary Shape Optimization," AI EDAM Special Issue: Evolutionary Design, Vol.13, Nr. 3 (1999).

primitives, such as small cubes or *voxels*, arranged in highly unrestricted ways according to information encoded in the genotype representation. Such representations are 'generic' because they are flexibly capable of representing a wide variety of different forms.

Evolutionary design systems employing generic representations have had some success at evolving new designs for a variety of different tasks and simultaneously optimizing them with respect to certain functional criteria, however they are limited in other ways. The main difficulty with generic representations is that they are too generic for producing designs as complex as buildings. Evolutionary design systems working with generic representations are not constrained in a way that ensures that the forms evolved can be understood as building designs. Though there are a number of different ways of implementing constraints within an evolutionary design system, some of which are discussed in Chapter 2.2, the only way to ensure that the system will evolve forms that resemble building designs and fulfill at least some minimal functional requirements is to partially design the phenotype representation.[30] This might mean designing an architectural kit of parts to be manipulated in a generative process that transforms the genotype representation into the phenotype representation, and/or encoding complex constraints into the rules of such a generative process. However it is done, care should be taken not to overdesign the phenotype representation and block off potentially interesting areas of the design space.

[30] Bentley, "Aspects of Evolutionary Design by Computers," 11-13, http://arxiv.org/html/cs/9809049/dss5.html

1.2.1 Problem Overview

In the domain of architecture, the application of evolutionary design could be highly beneficial. In the past, evolutionary design optimization has been successfully used to fine-tune one or more parts of a design, usually in a later stage of the design process.[31] [32] [33] Evolutionary design exploration of overall building designs early on the design process has proven to be somewhat more difficult, but methods have been successfully developed.[34] However, being able to adaptively evolve overall building designs that converge on optimal performance with regard to predefined functional requirements would result in greater potential benefits than either evolutionary exploration or optimization alone. For this to be possible, an evolutionary design system would have to meet the following three requirements:

- To evolve new building designs and not just parameters of an existing design, a generative process would be needed.
- To ensure that functional building designs are evolved, phenotype representations would have to be partially designed.
- To ensure that the evolved designs can converge on optimal performance, epistasis and interdependent elements would have to be minimized.

A number of generative evolutionary design systems have been developed, implemented and tested with varying degrees of success. They use a variety of generative techniques to create designs, such as *cellular automata*, *L-systems*, *shape grammars*, and *generative algorithms*. These techniques will be discussed in Chapter 2.2. Examples of generative evolutionary design systems include the following:

- Baron et al. have developed systems that employ a matrix of voxels to evolve a variety of forms.[35] [36]

[31] Dasgupta, *Evolutionary Algorithms in Engineering Applications*
[32] Caldas, "Evolution-based Generative Design System"
[33] Caldas, "Generation of Energy-efficient Architecture Solutions"
[34] Janssen, "Design Method and Computational Architecture."
[35] Baron et al., "A Voxel-based Representation" (1997)
[36] Baron et al., "A Voxel-based Representation" (1999)

- Shea has developed a system for generating space frame structures using a shape grammar and optimizing them using simulating annealing.[37] [38] [39]

- Bentley has developed a system for adaptively evolving solid objects using a representation scheme based on non-overlapping solid primitives and a set of evaluation routines. [40] [41]

- Hornby has developed a system using a parametric context-free L-system as a generative process, and turtle graphics that activate voxels in a matrix as a representation scheme, to evolve a variety of table designs based on functional criteria.[42] [43] [44]

- Janssen has developed a system for exploring building designs based on a combination of parametric and combinatorial generative modeling techniques.[45] [46] [47] [48]

Many of these systems are capable of evolving complex forms that possess novelty. A few can even evolve novel three-dimensional forms that resemble building designs. However, none are capable of adaptively evolving novel three-dimensional forms resembling building designs in a convergent process that optimizes design performance.

[37] K. Shea, "Essays of Discrete Structures: Purposeful Design of Grammatical Structures by Directed Stochastic Search" (Doctoral dissertation, Carnegie Mellon University, 1997).
[38] K. Shea. "An Approach to Multiobjective Optimization for Parametric Synthesis," in 13th International Conference on Engineering Design (ICED 01) – Design Methods for Performance and Sustainability, WDK 28 (2001).
[39] K. Shea, "Directed Randomness," in Digital Tectonics, eds. N. Leach et al. (London: Academic Press, 2004).
[40] Bentley, "Generic Evolutionary Design."
[41] Bentley, *Evolutionary Design by Computers*
[42] Gregory S. Hornby, "Functional Scalability through Generative Representations: the Evolution of Table Designs," *Environment and Planning B: Planning and Design*, Vol.31 (2004).
[43] Gregory S. Hornby, "Generative Representations for Computer-Automated Evolutionary Design" (paper presented at 2006 ERCOFTAG Design Optimization: Methods and Applications. Las Palmas, Spain, Apr. 5-7, 2006).
[44] Gregory S. Hornby and Jordan B. Pollack "The Advantages of Generative Encoding for Physical Design" (paper presented at 2001 IEEE Congress on Evolutionary Computation. Seoul, Korea, May 27-30, 2011).
[45] Janssen, "Design Method and Computational Architecture."
[46] Patrick Janssen, "A Generative Evolutionary Design Method," Digital Creativity, Vol.17, Nr.1 (2006)
[47] Patrick Janssen, John Frazer, and Ming-Xi Tang, "A Computational System for Generating and Evolving Building Designs," in Digital Opportunities: Proceedings of the 10th International Conference on Computer-Aided Architectural Design Research in Asia (2005)
[48] Patrick H.T. Janssen, John H. Frazer, and Ming-Xi Tang, "A Framework for Generating and Evolving Building Designs," *International Journal of Architectural Computing*, Vol.3, Nr.4 (2005)

The forms evolved by these systems suffer from at least one of two problems: they are not building designs and/or they are not optimal. Forms of the first kind result from using generative rules and representations that are either too generic or otherwise ill-suited for architecture, while forms of the second kind are produced by divergent evolutionary processes in evolutionary design systems that lack evolvability due to excessively high degrees of epistasis and/or too many interdependent elements.

1.2.2 Problem Statement

The main problem is to find a way to reduce epistasis and interdependent elements enough to allow convergent adaptive evolution of novel designs, without resorting to representations that are too generic to convey important building design information. This problem is referred to as the *integration problem*.

1.3.1 Research Objectives

The overall goal of this research is to contribute to the development of a practical evolutionary design approach that would enable designers to design and implement computational systems to adaptively evolve novel building designs based on performance criteria.

In order to achieve this goal, the main objective is to develop a framework for applying the evolutionary design approach that can overcome the integration problem. This framework is referred to as the *multi-stage evolutionary design framework*. The framework comprises two parts: a system design methodology and a system architecture.

- The design methodology broadly defines a set of procedures for designing *multi-stage evolutionary design systems* (MSEDSs) and using them to evolve design solutions.
- The system architecture specifies the general configuration of software and hardware components for an MSEDS.

A second objective is to design an MSEDS to evolve solutions to a particular class of design problem - the skyscraper. A third objective is to build a prototype implementation of the skyscraper MSEDS. And a fourth objective is to evaluate the performance of the prototype by simulating and comparing the energy performance of an evolved design alongside conventional designs.

1.3.2 Research Methodology

This research fits comfortably into the category of *systems development research* in the applied sciences.[49] The systems development research process consists of five stages:

- The first stage is the construction of a conceptual framework. This involves stating a meaningful research question to pursue, ideally one that is new, creative and important in the field. The researcher should investigate the functionalities and requirements of

[49] J. F. Nunamaker, M. Chen, and T.D.M. Purdin, "Systems Development in Information Systems Research," *Journal of Management Information Systems*, Vol.7, Issue 3, (1991)

the system, understand how it is built, and look to other relevant disciplines for new approaches and ideas.

- The second stage is the development of a system architecture. The developed architecture should be unique, extensible, modular, etc. The researcher should specify the functionalities of the system, and define the structural configuration and interrelationships among system components.
- The third stage is the analysis and design of the system. This involves the design of databases, knowledge bases and processes for carrying out system functions. The researcher should consider alternative approaches and choose one as a blueprint for system implementation.
- The fourth stage is the building of the (prototype) system. In this process, the researcher should gain a deeper understanding of the conceptual framework and any problems or complexities associated with the system design. The implementation process can provide valuable insights into potentially better system designs.
- The fifth stage is the observation and evaluation of the system. This involves testing the performance and usability of the system through case studies and experiments, developing new theories or models by interpreting observations and evaluations, and consolidating all experience gained.

System development research is typically carried out according to a long-term research program that divides the work up into a series of research projects. Each project focuses on one of the above five stages and provides foundations for the next project.

1.3.2 Research Scope

The scope of the present research encompasses all five stages of the system development research process, from constructing a conceptual framework and developing a system architecture, through designing, implementing and evaluating the system. For the first stage, a design methodology is developed that broadly defines a set of procedures for designing multi-stage evolutionary design systems and using them to evolve design solutions. For the second stage, a multi-stage evolutionary design system architecture is developed. For the third stage, a multi-stage evolutionary design system adapted for skyscrapers is designed. For the fourth stage, the skyscraper MSEDS is implemented. For the fifth stage, the performance of the skyscraper MSEDS is tested.

1.4.1 Thesis Organization

This thesis is organized into three parts:

- Part one consists of this introduction.
- Part two reviews current design practices and literature related to this research in three chapters. Chapter 2.1 is describes research on conventional performance-based design approaches, and in particular, performance-based skyscraper design approaches. Chapter 2.2 reviews generative design techniques and some example projects. Chapter 2.3 takes up the topic of evolutionary design, first reviewing its theoretical foundations, then exploring some evolutionary design systems, and lastly taking a look at some evolutionary design experiments described in the design literature.
- Part three documents the main body of research carried out for this thesis in five chapters. Chapter 3.1 introduces the multi-stage evolutionary design framework and explains how it works. Chapter 3.2 describes how to design an MSEDS to evolve skyscraper designs. Chapter 3.3 documents the design and implementation of an MSEDS to evolve solutions to the a particular skyscraper design problem. Chapter 3.4 documents an evaluation of the system's performance through a comparative performance evaluation of a design evolved by the system alongside three conventional designs. Chapter 3.5 reviews the research objectives and outcomes, draws important conclusions and discusses future research activities.

PART 2

LITERATURE REVIEW

2.1.1 Introduction

This chapter discusses some of the methods and procedures employed in mainstream architectural design practice to develop schematic building designs. In particular, it focuses on a process for developing schematic designs for tall buildings based on performance criteria.

2.1.2 Conventional Skyscraper Design Process

The following paragraphs describe a skyscraper design process. It is suggested that the described process can be regarded as typical of how many mainstream architectural design firms approach the design of tall buildings in a large number of cases. This is supported by practice-based research conducted during employment at a large global design firm engaged in the design of skyscrapers for growing Asian metropolises. Skyscrapers are defined here as multi-storey structures having a single independent load-bearing structure consisting of a steel framework, and a minimum total height of 100 m (330 ft). It is of course recognized that there may be other equally valid design processes that differ in some or all aspects from the description given below.

- Before a developer commissions an architect to start designing a new skyscraper, he does his own research and comes up with a fairly detailed list of things he wants in the design. This will usually include detailed programming requirements, such as precise square footages for every program element, as well as more general design objectives, such as natural daylighting, a low carbon footprint, commercial attractiveness, community attractiveness, and an iconic image. Typically handed over to the architect in the form of the Design Brief, this information together with the architect's own creative input provides a starting point for schematic design.

- The conventional way in which the architect begins to process the information in the Design Brief is with massing models. This involves building numerous digital or physical models of the different massing configurations that result from varying the floor-plate dimensions, floor-to-floor height, spacing of refuge floors, and other design parameters not fixed in the Design Brief.

- To consider the resulting models within the urban context of the project, a massing model of the surrounding neighborhood may be constructed and the skyscraper models positioned one-by-one on the site.

- Once all the options are modeled, or rather some representative options (there are actually infinite options), the designer picks one to pursue. The decision is typically based on considerations like whether different program types fit neatly into zones divided by refuge floors, whether a typical floor-plate area can be divided up nicely into structural bays of a convenient size, and whether an aesthetically desirable height and slenderness can be achieved. In some cases the tower height is specified by the client in the Design Brief. Also, the finished floor-to-ceiling clearance may be specified by the client, which more or less determines the floor-to-floor height, because there typically isn't much variability in the total floor depth including structure, HVAC, plumbing, electrical and finish materials. The total floor depth, sometimes called the "sandwich depth," can be estimated based on precedent.

- Next, the position and orientation of the tower on the site are determined and typical floorplans are drawn. This begins with selection of a service core type and placement, and adoption of a structural grid. Once these decisions are made, the core(s) is(are) laid out and the shape of the floor-plate is decided. Here, there are a number of considerations, as shown in the Table.

- Armed with typical floorplans, a new, more-detailed model of the building is constructed and attention turns to the building skin. Thermal and daylighting analyses may inform design of the building envelope, or decisions may be based on aesthetics alone. Shaping of the building in elevation will necessitate changes in the floorplans and may alter the overall massing of the tower. This stage of the design process typically involves a great deal of computer modeling, rendering, and remodeling.

- Finally, the podium and site (ground plane) are designed with careful consideration to automobile traffic, pedestrian circulation, fire safety requirements, and various other functional and aesthetic issues. This completes the schematic design.

This conventional design process for tall buildings has a number of inefficiencies and shortcomings. Starting with the building of massing models, perhaps the most obvious issue is that building a bunch of models (even simple massing models) is time consuming. Second, each new massing configuration requires a new set of calculations and it is easy for error to be introduced in this process. For example, when the floor-plate area is changed, not only does the total number of floors change but the number of floors in each program zone changes according to each program zone's percentage of the total GFA. Thus, program zones may no longer fit nicely between refuge floors without adjusting the spacing of refuge floors, which might change the number of refuge floors required. Even if the spacing of refuge floors is not adjusted, the total number of floors does not necessarily change in proportion to the change in floor-plate area, as additional floors may necessitate additional refuge/MEP floors, and fewer floors may enable removal of a refuge/MEP floor. Clearly, use of a parametric model encoding the basic tower massing relationships would save time and effort and avoid errors. Further, one parametric model with sufficient flexibility built into it could be reused for a wide class of tower design problems.

The conditions for evaluating the massing models in the conventional design process are not ideal either. One problem is that each model, whether digital or physical, must be manually placed on the site and the previous model removed in order to see them all in context. Not only is this inconvenient but the time delay between seeing the placed models hinders their comparison. Another drawback is that there is no continuous spectrum of variation with conventional models. The designer must choose from among the available options when a better solution may be hidden somewhere in between the chosen parameters, or else go back and expend additional time and effort to create more massing models. These drawbacks are additional reasons to employ a parametric model.

There is another important shortcoming at this early stage of the conventional design process that is not overcome simply by employing a parametric model. The choice of massing, like orientation, can have important consequences for building performance that are difficult to gauge without computation as a tool. For example, it is difficult to determine with the naked eye whether views from a particular model are open or blocked, whether adding height to a zone or width to a facade would open up new view corridors and if so how much height or width to add. Likewise, it is impossible to assess by critical observation alone the precise effects of massing variations on the amount of solar radiation incident on different parts of the building facade.

Computer simulation can produce measurements but it is still only one part of the solution. To gauge how various changes in massing effect the measurements requires performing numerous simulations on the different massing models, which is time consuming and still won't yield the optimal configuration. Employing a parametric model in the simulations won't do the trick either because manual manipulation is still needed to vary the model and performing each simulation, and the designer has no reliable way to know how to find the optimal configuration within a reasonably short number of iterations. The solution is to employ an evolutionary solver or some other form of optimizer to evolve/optimize the parametric model in response to feedback from simulations.

2.1.3 Conventional Performance-Based Design

Performance-based design has been described as "an emerging approach to architecture in which building performance is a guiding design principal."[50] Here, performance is broadly defined - "its meaning spans multiple realms, from spatial, social and cultural to purely technical (structural, thermal, acoustical, etc.)."[51] In performance-based design, computer simulation is used in the conceptual stages of design to inform early decisions that will eventually have the greatest impact on building performance.[52] They are also used later in the design process to optimize particular elements of a design that need improvement. [53]

An example of a design developed using a performance-based approach is the glazed roof that spans the British Museum Great Court in London, UK, by Foster + Partners, shown in Figure 2.1.1. The curved shape of the roof and the pattern of steel members constituting the structure of the roof were arrived at using a flexible modeling approach that enabled numerous

[50] Branko Kolarevic and Ali Malkawi, eds., *Performative Architecture: Beyond Instrumentality* (London: Routledge,2005), 3.
[51] Ibid.
[52] Ibid.
[53] Jane Burry and Mark Burry, *The New Mathematics of Architecture* (London; Thames and Hudson, 2010), 116-155.

different options to be generated quickly and tested against different performance criteria, such as solar gain, acoustics, and structural efficiency.[54]

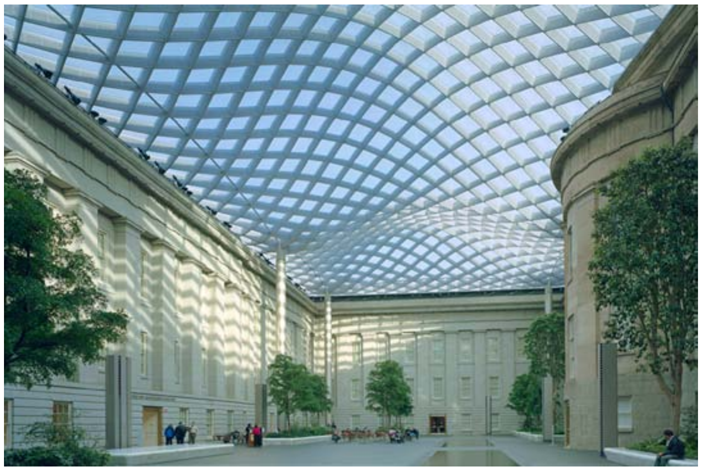


Figure 2.1. 1 British Museum Great Court, London, UK - roof designed by Foster + Partners

Another example is the design for the Al Raha Development in Abu Dhabi, UAE, also by Foster + Partners, shown in Figure 2.1.2. The designers constructed a mutable parametric model of the building form that could morph in response to feedback from simulations of its performance when exposed to the desert sun and wind.[55]

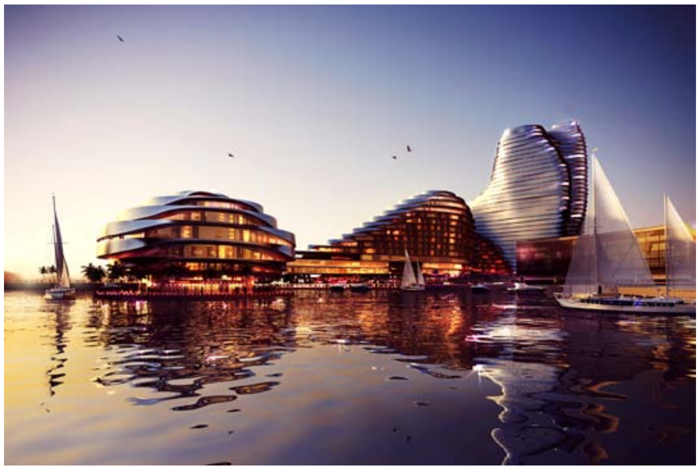


Figure 2.1. 2 Al Raha Development, Abu Dhabi, UAE - Design by Foster + Partners

[54] Jane Burry and Mark Burry, *The New Mathematics of Architecture* (London; Thames and Hudson, 2010), 122-125.
[55] Burry and Burry, *New Mathematics*, 148-151.

2.2.1 Introduction

In contemporary architectural design, digital tools and media are used not only for visualization and representation, but for the derivation and transformation of architectural form. Transcending the well-established practice of mechanically constructing models and drawings by directly manipulating geometry displayed on a computer screen, this emerging practice employs rule-based procedures that are interpreted by software to computationally generate and transform geometry. Generative design is "an approach to design that seeks to challenge the hegemony of top-down processes of form-making, and replace it with a bottom-up logic of form-finding."[56]

Conventional computer-aided design (CAD) tools function like sophisticated electronic drawing instruments enabling designers to construct digital models and drawings. The designer has no real perception of the computational processes governing the behavior of the digital environment in response to his/her manipulations. Generative design systems and techniques, on the other hand, enable the designer to work with the underlying form-generating processes, thus activating a whole new field of possible interactions between the human designer and the design tool.

Generative design techniques give the designer explicit control over design parameters, their logical relationships, and the form-generating process itself. This can help to unpack dependencies, externalize logic, and clarify distinctions between qualities and quantities. They can also free the designer from the limitations in the inbuilt functionality of CAD software, support the cognitive design process by enabling the designer to explore unimagined regions of the design space, and supplement the designer's own creativity, freeing him/her from 'design fixation' and the limits of conventional wisdom.[57]

This chapter outlines four different categories of techniques for computationally generating three-dimensional form. Any of these generative techniques can be used alone or in combination with other techniques to develop a generative system that can be used within the developmental step of an evolutionary design system in order to generate alternative designs.

[56] Neil Leach, "Digital Morphogenesis." *Architectural Design* 79, no.1 (2009): 34.
[57] Bentley

The four categories are *parametric techniques*, *combinatorial techniques*, *replacement techniques*, and *agent-based techniques*.

2.2.2 Parametric Techniques

Parametric techniques involve varying constraints associated with a predefined model or modeling procedure to generate a variety of forms. When applied to a model, certain dimensions are typically defined as variable parameters, and these are varied to transform the model into a range of different forms. When applied to a modeling procedure, certain constraints on operations in the procedure are defined as variable parameters, and these are varied to transform the end result of the procedure. Parametric techniques are sometimes found in the literature under different headings, such as associative modeling, variational design, constraint-based design, etc.

2.2.3 Combinatorial Techniques

Combinatorial techniques involve selecting elements from a predefined set and combining the selected elements in various arrangements to generate a variety of forms. Techniques in this category may differ from one another in terms of the predefined set of elements they work with, and the way selected elements are combined. The predefined set of elements may include a number of different object types, such as solid geometric primitives, which can be selected and combined to generate a wide range of different forms. Alternatively, there may be only one type of element, such as a tiny three-dimensional cube or voxel (volumetric pixel), which can be combined in large numbers to generate almost any form. The procedure for combining elements may involve selecting certain operations that position and combine the elements in space from a predefined set of such operations. Alternatively, the procedure may make use of a predefined organizational template into which different elements are inserted to generate forms.

2.2.4 Replacement Techniques

Replacement techniques involve repeatedly applying a set of rules that manipulate parts of a predefined initial form to generate a new form in a series of steps. The initial form is described as a *seed*, and the set of rules are referred to as *transition rules.* At each step, the transition rules are applied to the *current form* to produce a new form*,* and after a certain number of steps a *final form* is produced.

The transition rules are essentially *if-then* rules. They specify that *if* the current form has a certain part or configuration, *then* that part or configuration must be replaced by another to produce a new form. The *if* part of the rule is called the antecedent, and the *then* part is called the consequent. The first step in applying a transition rule is to examine the current form for instances of the part or configuration specified in the antecedent. If one or more matches are found, the transition rule replaces each of them with the part or configuration specified in the consequent. When this happens, the transition rule is said to be *fired*. If, on the other hand, no matches are found, then the rule is not fired and simply does nothing.

Many replacement techniques employ not only *iteration* but also *recursion*. In recursive replacement techniques, each iteration involves reapplying the same transition rules to the result of their application in the previous iteration. In order for a rule to continue firing after more than one iteration, the part or configuration specified in its consequent must contain at least one copy of the part or configuration specified in its antecedent. Recursive replacement techniques employing fairly simple transition rules can be used to build up highly complex, *self-similar* forms, after only a few iterations. Although many generative systems employing recursive replacement techniques are deterministic,[58] their rapid, exponential growth is difficult to control and the final forms they produce are notoriously unpredictable.

L-Systems

Lindenmayer-Systems or L-Systems are a kind of parallel replacement system that begin with alpha-numeric string data as a seed and employ recursive transition rules to grow. A simple example of an L-systems begin with a seed *A* and transition rules *A -> B* and *B - > AB*. The transition rules are applied at each step to yield the following alternating pattern:

Step 0: A
Step 1: B
Step 2: AB

[58] A replacement-based generative system is deterministic as long as its transition rules are deterministic. Some systems employ stochastic transition rules and therefore are not deterministic.

Step 3: BAB
Step 4: ABBAB
Step 5: BABABBAB
Step 6: ABBABBABABBAB
Step 7: BABABBABABBABBABABBAB

The string values in the final form can be mapped to a variety of geometric operations to produce form. One common strategy is to use LOGO-style 'turtle graphics,' a system of commands including *move forward, move backward, turn right, turn left,* etc. that are used to move a cursor know as the 'turtle' through space displayed on the computer screen. The path traced out by the turtle provides a basis for generating geometry through various modeling operations such as piping and voxel activation.

L-systems can be used to construct iterated function systems (IFSs) and fractals. In an IFS, a contractive function is executed recursively, repeatedly performing the same contractive transformation on its output from the previous iteration, thereby generating an infinite sequence of self-similar forms.[59] A fractal, whose self-similar appearance is well known, is the union of these resulting forms.[60] When used as form-generative mechanisms, the recursive nature of L-systems and IFSs results in formal complexity marked by self-similarity at a range of scales, after relatively few recursions, making them potentially useful for generative design in architecture.

Supermatter by Supermanoeuvre + Matter Design, shown in Figure 2.2.1, is an example of the use of L-systems to generate form. "Supermatter explores the algorithm as a geno-typical morphology, where similarity across the collective is instilled through the instructions of assembly embedded within the algorithm as it operates on a discrete set of geometric aggregates. The input of geometry enables a speciation of each resultant object as the rules of growth and assembly are applied to the specific geometric constraints and potentials of connection particular to each aggregate primitive. Through changing either one or all of the primitives, or the generative rules of the Lindenmayer (L-System) combinatorial algorithm itself, differentiation across the population can be instantiated."[61] As revealed in this slightly cryptic

[59] Wikipedia contributors, "Iterated function system," *Wikipedia, The Free Encyclopedia* http://en.wikipedia.org/w/index.php?title=Iterated_function_system&oldid=460647709 (accessed 5 Dec. 2011)
[60] Ibid.
[61] Supermanoeuvre + Matter Design, "Supermatter," *Contemporary Digital Architecture: Design and Techniques* (Links International Ceg, 2010), 212-217.

description by the architect, Supermatter is based on L-systems, devised by theoretical biologist and botanist Aristid Lindenmayer in 1968 for modeling processes of plant development.[62] More specifically, Supermatter is an experiment in which strings generated by an L-system are translated into geometric transformations performed on geometric primitives, resulting in the generation of an aggregate self-similar form.

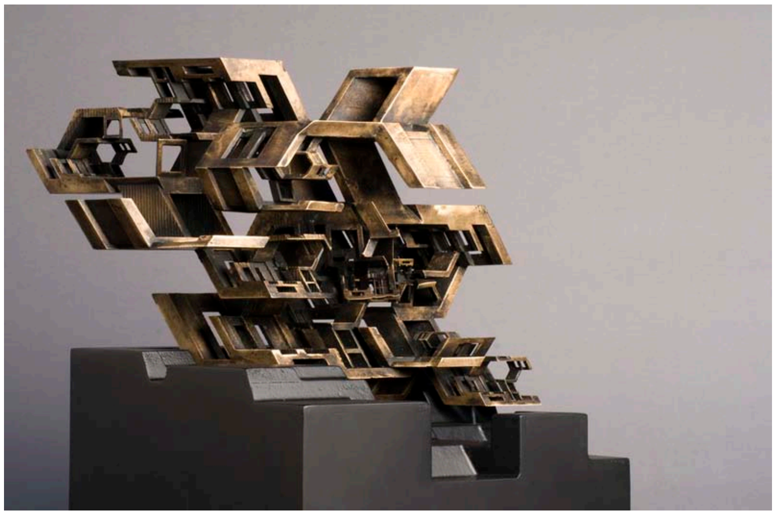


Figure 2.2. 1 Supermatter by Supermanoeuvre + Matter Design

<u>Cellular Automata</u>

A cellular automaton (CA) is a system made up of a finite matrix of cells (e.g., a two-dimensional grid), each cell having a finite number of states (e.g., two states: "on" and "off"), a neighborhood in the matrix defined relative to itself (e.g., all cells within a 2-cell distance of itself), an initial state (e.g., "on") which serves as a seed, and a transition rule for updating the state, expressed in terms of its own current state and the states of the other cells in its neighborhood (e.g., when 5 or more cells in the neighborhood are "on," change state from "on" to "off" or vice versa).[63] CAs are typically implemented on a computer, and are fascinating examples of how simple programs operating according to simple rules can lead to complex and unpredictable behavior with no further human input after they are set up.[64] Such behavior

[62] Aristid Lindenmayer, "Mathematical Models for Cellular Interaction in Development: Parts I and II," *Journal of Theoretical Biology, 18* (1968), 280-315
[63] Wikipedia contributors, "Cellular automaton," *Wikipedia, The Free Encyclopedia* http://en.wikipedia.org/w/index.php?title=Cellular_automaton&oldid=464924768 (accessed 5 Dec. 2011)
[64] Stephen Wolfram, *A New Kind of Science,* Wolfram Media (Champaign, IL: 2002).

points to the latent potential of algorithmic abstraction as a vehicle for architectural exploration within a domain of abstract reality that is independent of experience and "extends beyond the limits of perception."[65]

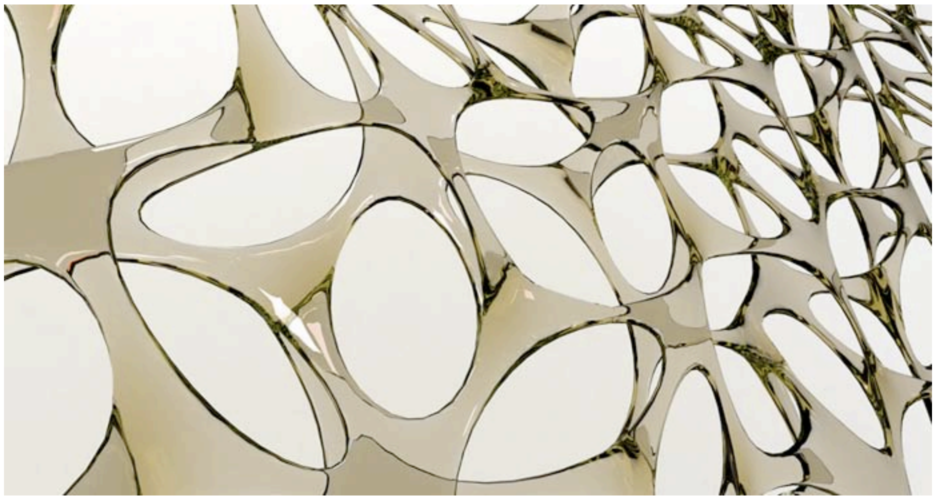


Figure 2.2. 2 Morphogenetic Lattice by Supermanoeuvre + Kokkugia

Morphogenetic Lattice, by Supermanoeuvre + Kokkugia, shown in Figure 2.2.2, is an experiment in the use of systems "designed to generate ornamental distortions within geometry through the internal logic of cellular automata. A technique where a population of self-similar elements in space continually change their state based on the states of their neighbors in a feedback loop, giving rise to emergent patterns."[66] The architects of Morphogenetic Lattice developed a series of encoded procedures to construct a three-layer system in which the lattice geometry (top layer) is dependent on a network of springs (middle layer) whose behavior is calculated using a physics solver in response to changes in the state of a hex-grid of cellular automata (bottom layer). The network of springs (middle layer) simply serves to smoothly translate changes trickling through the cellular automaton (bottom layer) into smooth elastic transformations throughout the lattice geometry (top layer), enabling Morphogenetic Lattice to appear alive.

---

[65] Kostas Terzidis, *Expressive Form: A Conceptual Approach to Computational Design,* Spoon Press (London and New York: 2003), 73
[66] Supermanoeuvre + Kokkugia, "Morphogenetic Lattice," *Contemporary Digital Architecture: Design and Techniques* (Links International Ceg, 2010), 208-211.

2.2.5 Agent-Based Techniques

Agent-based techniques involve defining virtual autonomous agents that move around and interact with one another to collaboratively generate form in space. These techniques offer an alternative model to systems of centrally orchestrated form growth and development. They are often inspired by the natural behavior of certain animals, for example, the smarm behavior of certain flocks of birds and schools of fish, and the insect colonies built by ants and bees.

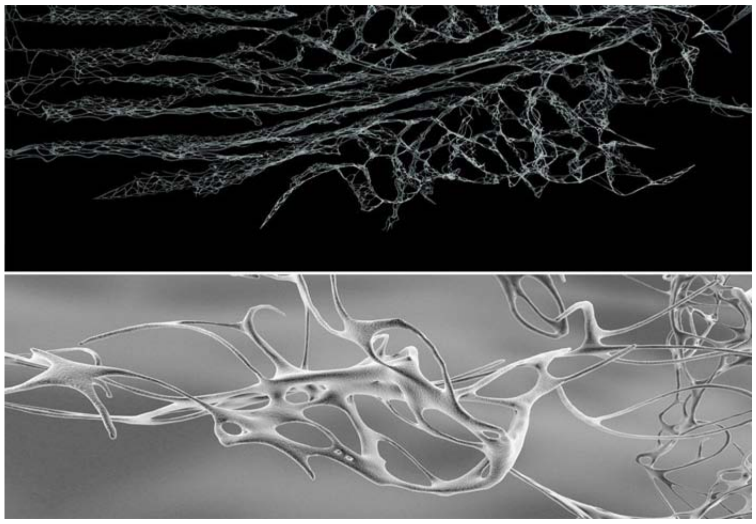


Figure 2.2. 3 'Swarm Matter' by Kokkugia (top image-feild; bottom image-close-up)

Swarm Matter by Kokkugia, shown in Figure 2.2.3, is an experiment in the use of agent-based techniques to dynamically generate form. "Swarm Matter is an ongoing research project exploring the generation of ornamental geometries though the agent based formation of emergent hierarchies and non-linear patterns."[67] This experiment is essentially concerned with the forms that dynamically emerge in a system of components (agents) that each move independently according to predetermined rules without any central coordination. More particularly, Swarm Matter is based on a computer simulation of swarm behavior, the collective

[67] Kokkugia, "Swarm Matter," *Contemporary Digital Architecture: Design and Techniques,* Jacobo Krauel, ed. (Links International Ceg, 2010), 126-129.

motion of a large number of self-propelled entities.[68] In nature, swarm behavior is "a collective behavior exhibited by animals of similar size which aggregate together, perhaps milling about the same spot or perhaps moving *en masse* or migrating in some direction."[69] It is witnessed, for example, as the swarming of locusts, the flocking of birds, the herding of sheep, the schooling of fish, and the blooming of algae.[70] Swarm behavior can be modeled by programming similarly sized agents, representing particles or some other objects, to follow three simple rules: (1) move in the same direction as your neighbors, (2) remain close to your neighbors, and (3) avoid collisions with your neighbors.[71]

Thomas and Torben Fischer disclose a very different approach to digital form-finding employing an agent-based model (ABM).[72] Instead of the neo-Darwinian evolutionary principals of variation and selection, their approach concentrates more on the genetically instructed developmental pathway from genotypes to phenotypes. To investigate and simulate the developmental growth of geometric form, they have created a voxel-based form-modeling software system in which individual voxels (volumetric pixels) can be programmed to behave (change state) according to predetermined rules in relation to neighboring voxels and time. The voxels' state transformations according to preprogrammed rules works like a 3-dimmensional cellular automaton. However, unlike conventional cellular automata, these voxels can all follow different rules (genetic instructions), and, by employing parallel processing, their state updates need not be synchronized. This approach and the software tool created (*Zellkalkül*) is an attempt to model the "cellular incorporation of morphologic and behavioral blueprints."[73] , [74]

---

[68] O'Loan, Evans, "Alternating Steady State in One-Dimensional Flocking," *Journal of Physics A: Mathematical and General 32 (8)*, (1998).
[69] Wikipedia contributors, "Swarm behavior," *Wikipedia, The Free Encyclopedia* http://en.wikipedia.org/w/index.php?title=Swarm_behaviour&oldid=460044985 (accessed 5 Dec. 2011)
[70] Ibid.
[71] Ibid.
[72] Thomas Fischer and Torben Fischer, "Toolmaking for Digital Morphogenesis," *International Journal of Design Computing, asd Vol. 6*, University of Sydney (2003), 1-23
[73] Ibid, 5
[73] See "The Problem of the Blueprint" in Frazer, *An Evolutionary Architecture*, Architectural Association, (London: 1995)

2.3.1 Introduction

2.3.1.1 Overview

Evolutionary Design is an interdisciplinary field of research that merges the boundaries of evolutionary biology, computer science, and design, as shown in Figure 2.3.1. The central objective of evolutionary design research is the development of methods and computational systems for generating and adaptively evolving designs, based on the Neo-Darwinian model of evolution through natural selection.

Frazer was among the first to advocate the use of evolutionary design techniques in the domain of architecture. He envisioned a computational design system in which "*architectural concepts are expressed as generative rules so that their evolution and development can be accelerated and tested by the use of computer models. Concepts are described in a genetic language which produces a code script of instructions for form-generation. Computer models are used to simulate the development of prototypical forms which are then evaluated on the basis of their performance in a simulated environment. Very large numbers of evolutionary steps can be generated in a short space of time and the emergent forms are often unexpected.*"[75]

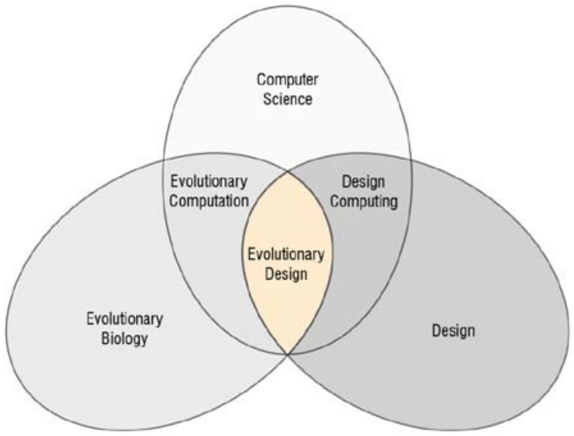


Figure 2.3. 1 - Evolutionary design is rooted in computer science, evolutionary biology, and design

[75] J. H. Frazer, *An Evolutionary Architecture*, (London: AA Publications, 1995)

Evolutionary design systems are capable of generating and evaluating not only one or several design proposals, but entire populations of alternative design proposals, and gradually honing in on better designs by replacing old populations of designs with new populations genetically bred from the fittest parent designs. Evolutionary design systems have benefitted from a large amount of interest and research.[76] [77] [78]

The purpose of creating such systems is not only to enhance productivity by further automating the design process. Evolutionary design systems have the potential to encourage experimentation and innovation in the design process, to facilitate the exploration of a wide range of design alternatives, to flexibly accommodate design changes, to allow the client more control over the design process, and to produce designs for buildings that out-perform buildings designed in a conventional manner in key areas such as energy consumption.[79]

The central issue in the design of evolutionary design systems is the development of generative systems capable of producing adequate representations of architectural concepts in response to manipulations of genetic variables. Another important issue is the formulation of fitness criteria from various conflicting and ill-defined design criteria. A further challenge is the issue of how the simulated interactions between generated forms and the environment are linked with the processes of form generation and selection.[80]

2.3.1.2 Evolution in Nature

In nature, evolution is the process by which all living organisms change and adapt to their environment over generations. The evolution of a population of organisms is driven by the continuous cycle of life and death of individual organisms in the population. This cycle can be broken down into 3 steps: *reproduction*, *development*, and *natural selection*.

- In reproduction, new organisms are conceived. Each new organism inherits from its parents a unique set of genes that encodes information about various biological traits - the *genotype*. In the case of sexual reproduction, new child genotypes are created by

[76] Ibid.
[77] Bentley, *Evolutionary Design by Computers*.
[78] Bentley and Corne, *Creative Evolutionary Systems*.
[79] Janssen, “A Design Method and Computational Architecture.”
[80] Kolarevic, "Digital Morphogenesis."

reshuffling the genes of the parents in a process called crossover, and by altering existing genes to create new genes through random accidental copying errors called mutations.

- In development, each new organism grows from a seed or egg into a fully developed organism - the *phenotype*. This takes place gradually as a result of processes of cell growth, differentiation and morphogenesis, which are triggered by genes in the genotype under suitable environmental conditions.
- In natural selection, the organism must successfully compete with others for limited resources in the environment, and the longer it survives, the more likely it is to reproduce. Some organisms possess traits or *adaptations* that help them to survive longer and/or to reproduce more. These organisms are more likely to pass on their genes each generation, causing those genes and adaptations to become more common in the population as a whole over time.

Not all biological organisms have evolvability - the capacity to acquire useful adaptations through genetic changes in reproduction. Populations of organisms that lack evolvability may still be capable generating genetic diversity, but the developmental processes that produce the phenotype are unable to produce useful adaptations. As a result, such organisms do not become better adapted to their environment through natural selection.

Kirschner describes three characteristics of biological systems that are conducive to evolvability, which he defines as "efficiency of generating novelty in evolution." [81] The first is an ability to "maximize variation in the amount of mutation." The more variation there is in a population, there greater are the chances of useful traits being selected and passed on to new generations. The second is an ability to "suppress the fallacy of variation produced." He goes on to explain, "Only nonlethal variations contribute to evolution. In an architectural sense, if you were making structures at random, structures that would fall down, structures that nobody would be interested in - these would be lethal structures. Thus, you'd like to have a way of biasing the output so at least you would be looking at things that would hold up." The third characteristic is "the provision of useful variation, even for conditions not previously

[81] Mark Kirschner, "Variations in Evolutionary Biology," *Research & Design: The Architecture of Variation* (London: Thames and Hudson, 2009), 26-33

encountered." In addition to maximizing overall variation, and suppressing fatal variation, evolvability requires the ability to promote useful variation.

2.3.1.3 Evolutionary Algorithms

Universal Darwinism is the theory that evolution by natural selection is *substrate-independent,* meaning that processes analogous to evolution in nature can emerge in systems outside the realm of biology.[82] This thinking has led to the successful application of evolutionary models to explain phenomena in a wide range of domains, including economics, psychology, medicine, computer science and physics.[83]

Evolutionary algorithms encode some of the underlying principals and mechanisms of evolution in algorithmic form so that they can be implemented as computer programs. When executed by a computer, they result in a computational process that is loosely analogous to evolution in nature. A population of individual entities is created and selectively bred over generations so that the entire population gradually evolves in a desired direction. Each individual entity in the population can represent an alternative solution to some type of problem, such as a parameter of an equation or even a complete building design. Thus, evolutionary algorithms are capable of evolving solutions to a wide variety of problems.[84] [85]

Figure 2.3.2 illustrates the cyclical process of computational evolution by evolutionary algorithms. As in natural evolution, the solution being evolved has both a genotype representation and a phenotype representation. The genotype representation is an encoded version of the solution and the phenotype representation is the decoded genotype representation. The computational process can be broken down into an initialization step of randomly generating a starting population of genotype representations, and an evolutionary cycle comprising four steps: *reproduction*, *development, evaluation,* and *selection*.

- In reproduction, a new population of genotype representations are created by transforming parent genotypes into new child genotypes using *genetic operators* such as crossover and mutation, according to predetermined reproduction rules.

[82] Dawkins, *Universal Darwinism.*
[83] Wikipedia contributors, "Universal Darwinism."
[84] Mitchell, *An Introduction to Genetic Algorithms*, 15-16.
[85] Beasley, "Possible Applications of Evolutionary Computation."

- In development, the new population of genotype representations are transformed into phenotype representations by applying predetermined developmental rules.
- In evaluation, the performance of each phenotype representation is evaluated with respect to one or more objectives. For each objective, an evaluation score is calculated using a mathematical function called the *objective function* and stored together with the solution.
- In selection, the evaluation scores of each alternative solution are summarized in a single figure indicating the merit of the solution - its *fitness*. Here, the mathematical function used to calculate fitness is called the *fitness function*. Solutions that rank higher in fitness are selected for reproduction and those not selected are deleted.

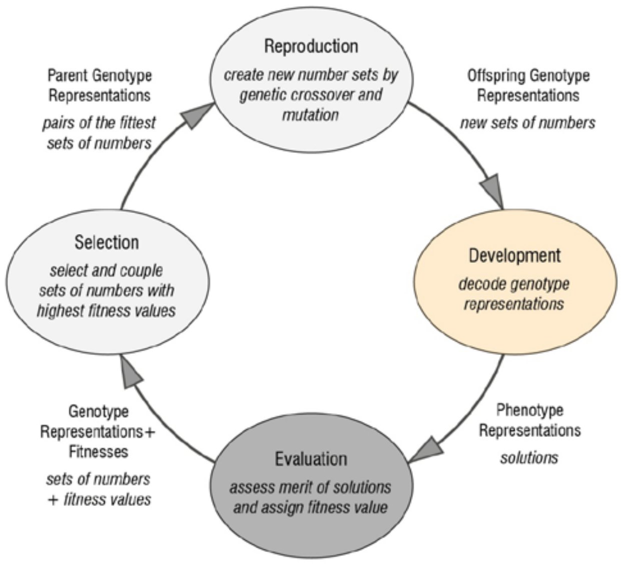

Reproduction
create new number sets by genetic crossover and mutation
Offspring Genotype Representations
new sets of numbers
Development
decode genotype representations
Phenotype Representations
solutions
Evaluation
assess merit of solutions and assign fitness value
Genotype Representations + Fitnesses
sets of numbers + fitness values
Selection
select and couple sets of numbers with highest fitness values
Parent Genotype Representations
pairs of the fittest sets of numbers

Figure 2.3. 2 - The Cycle of Computational Evolution

Many aspects of evolutionary algorithms depend upon the details of the problem to which they are applied. The evolutionary algorithm and the problem to be solved together make up an *evolutionary system* with variables, including the types of genotype and phenotype representations, the reproduction and developmental rules, the type of evaluation performed in the evaluation step, the form of the objective function, and the form of the fitness function. In order for the evolutionary algorithm to evolve solutions to the problem, these variables all have to be specified.

Evolvability of evolutionary systems is analogous to evolvability of biological organisms. Evolutionary systems that lack evolvability are able to generate diverse population of solutions, but they do not tend to increase in fitness over generations. As in natural evolution, evolvability is related to the nature of the developmental process used to generate phenotype representations from genotype representations.

A wide range of evolutionary algorithms exist that differ in implementation details and the type of problems they are designed to tackle. The four main types are *genetic algorithms*,[86] [87] *genetic programming*,[88] *evolutionary programming*,[89] and *evolution strategies*.[90] Of these, genetic algorithms are the best known and most commonly used.

2.3.1.4 Evolutionary Design

In the design domain, evolutionary algorithms are used to evolve populations of alternative designs.[91] [92] [93] Figure 2.3.3 illustrates the cycle of computational evolutionary design. Each genotype representation is encoded information that can be used to construct a model of a design, and the corresponding phenotype representation is the fully constructed design model. The reproduction rules can be the typical rules for applying genetic operators used by the evolutionary algorithm, and the developmental rules are the generative or transformational modeling procedures used to produce the design model from the information encoded in the genotype representation. The evaluation step may consist of performing one or more measurements or simulations on the design model, which may incorporate information on the environment in which the design is to be realized, and the objective functions and fitness function are formulated to convert the evaluation scores into a single figure measuring how close the design comes to fulfilling certain design objectives.

[86] Holland, *Adaptation in Natural and Artificial Systems*
[87] Goldberg, *Genetic Algorithms*
[88] Koza, *Genetic Programming*
[89] Fogel, "Evolutionary Computation"
[90] Bäck, *Evolutionary Algorithms*
[91] Frazer, *An Evolutionary Architecture*
[92] Bentley, *Evolutionary Design by Computers*
[93] Bentley and Corne, *Creative Evolutionary Systems*

The ability of evolutionary algorithms to evolve solutions to a wide variety of problems stems from the ability to represent solutions to a wide variety of problems as encoded genotypes. In the evolutionary cycle, the processes of selection and reproduction operate only on genotype representations, and thus are largely independent of the nature of the problem. In an evolutionary design system, these steps can be performed by a system component called an *evolutionary solver,* and the development and evaluation steps can be performed by a system component called a *design developer-evaluator*, as indicated in Figure 2.3.3.

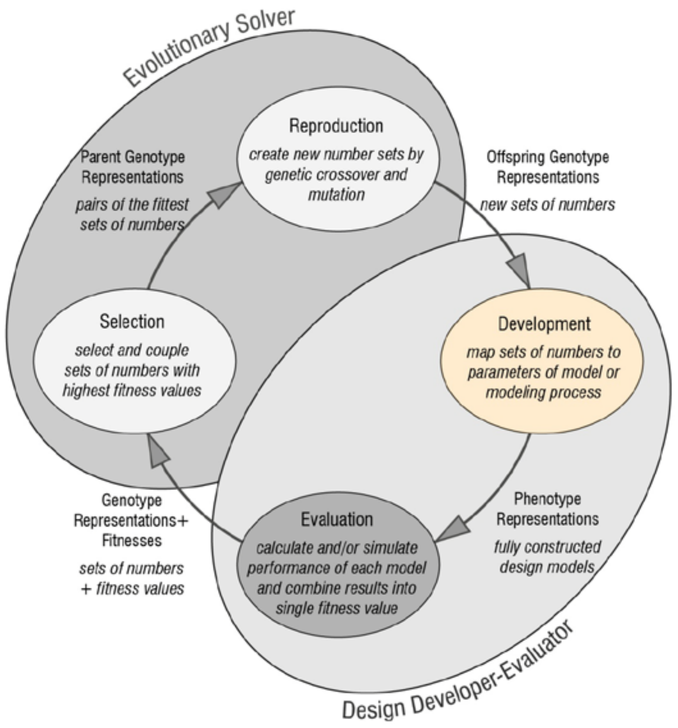

Evolutionary Solver
Reproduction
create new number sets by genetic crossover and mutation
Parent Genotype Representations
pairs of the fittest sets of numbers
Offspring Genotype Representations
new sets of numbers
Selection
select and couple sets of numbers with highest fitness values
Development
map sets of numbers to parameters of model or modeling process
Genotype Representations+ Fitnesses
sets of numbers + fitness values
Evaluation
calculate and/or simulate performance of each model and combine results into single fitness value
Phenotype Representations
fully constructed design models
Design Developer-Evaluator

Figure 2.3. 3 - The Evolutionary Design Cycle

Figure 2.3.4 is a system diagram illustrating the basic architecture of an evolutionary design system, and Figure 2.3.5 is a flowchart illustrating the process of evolving designs using the evolutionary design system shown in Figure 2.3.4. Referring to Figure 2.3.4, the system includes a user interface 100, an evolutionary solver 200, and a design developer-evaluator 300. The evolutionary solver 200 includes a selection system 210 and a reproduction system 220. The design developer-evaluator 300 includes a development system 310 and an evaluation system 320.

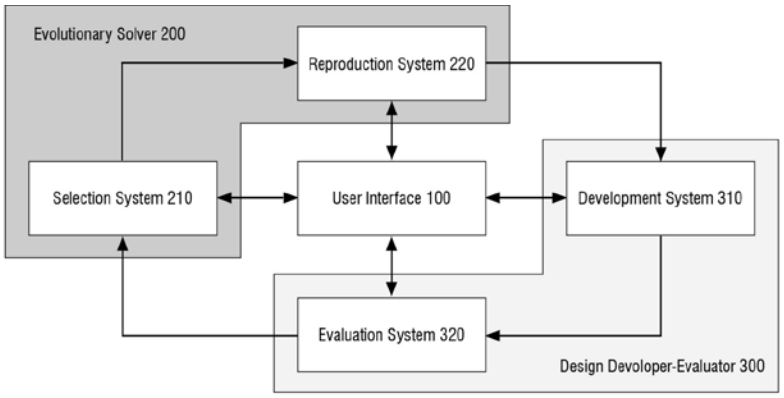


Figure 2.3. 4 - Basic Architecture of An Evolutionary Design System

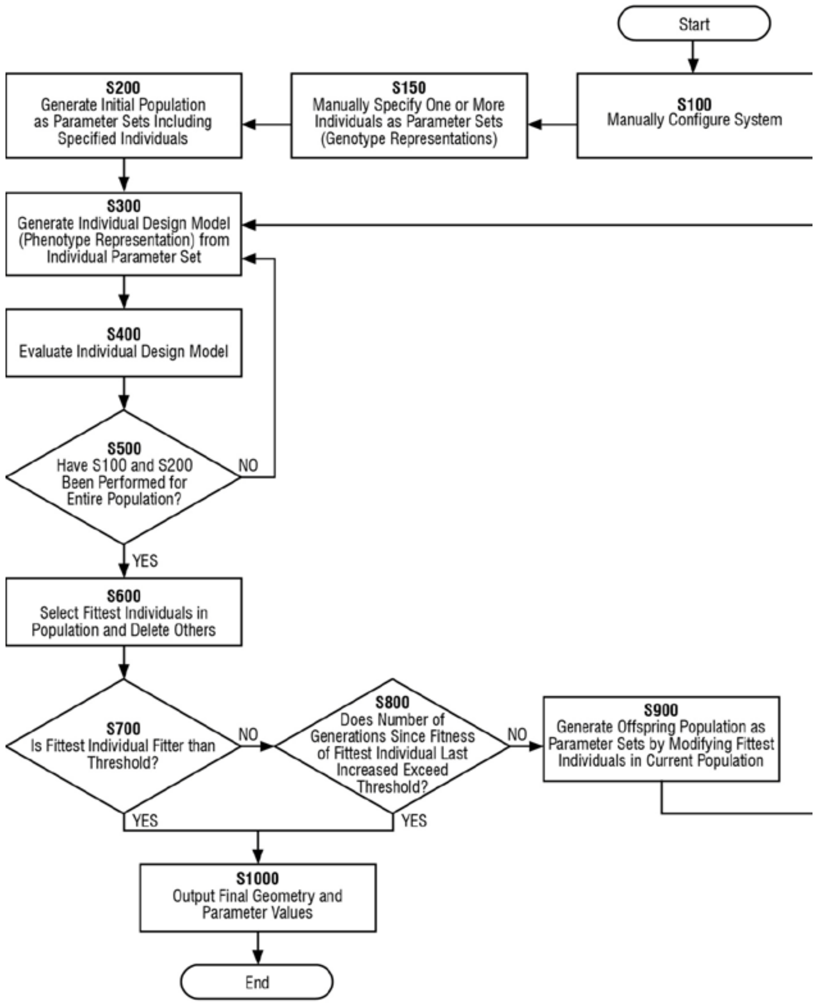

Start
S100 Manually Configure System
S150 Manually Specify One or More Individuals as Parameter Sets (Genotype Representations)
S200 Generate Initial Population as Parameter Sets Including Specified Individuals
S300 Generate Individual Design Model (Phenotype Representation) from Individual Parameter Set
S400 Evaluate Individual Design Model
S500 Have S100 and S200 Been Performed for Entire Population?
NO
YES
S600 Select Fittest Individuals in Population and Delete Others
S700 Is Fittest Individual Fitter than Threshold?
NO
S800 Does Number of Generations Since Fitness of Fittest Individual Last Increased Exceed Threshold?
NO
S900 Generate Offspring Population as Parameter Sets by Modifying Fittest Individuals in Current Population
YES
YES
S1000 Output Final Geometry and Parameter Values
End

Figure 2.3. 5 - Evolutionary Design Process

2.3.2 Evolutionary Design Systems

2.3.2.1 Overview

In Chapter 1.1, three types of evolutionary design were identified: evolutionary design optimization, evolutionary design exploration, and integral evolutionary design. This section reviews a number of evolutionary design systems in the literature, identifies each them as one of these three types, and discusses their strengths and weaknesses.

2.3.2.2 GENE_ARCH

Caldas has developed an evolutionary design optimization system, called GENE_ARCH, for helping architects to design energy-efficient and sustainable architectural solutions.[94] GENE_ARCH can operate in two modes depending on design requirements. When applied to problems where the overall building geometry is fixed, it generates populations of alternative solutions from a parametric model, and in cases where the building geometry is allowed to vary, it employs shape grammars. The system employs a Pareto genetic algorithm (GA) as a search engine, and uses a sophisticated building energy simulation application DOE-2.1E to evaluate alternative design solutions. Figure 2.3.6 illustrates GENE_ARCH's main components, and Figure 2.3.7 illustrates three-dimensional architectural solutions generated by GENE_ARCH.

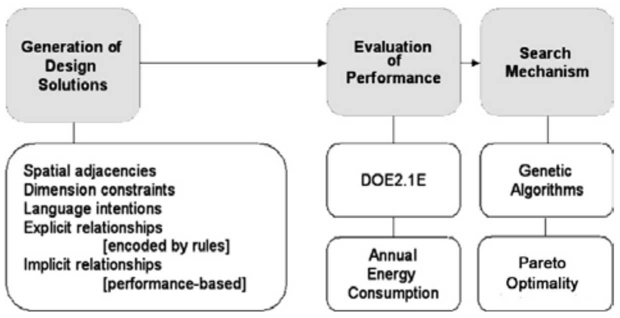


Figure 2.3. 6 GENE_ARCH's Main Components

[94] Caldas, "Generation of Energy-efficient Architecture Solutions," p59-70.

A strength of this system lies in its ability to evolve designs on the detailed level of their material systems, when the overall building shape and spatial layout are fixed. Caldas writes, "One of the distinctive aspects of GENE_ARCH is that it generates complete building designs, both in terms of geometry, spatial layout and room characteristics, as in terms of construction materials, internal finishes, types and characteristics of window and glazing systems, and even mechanical and electrical installations."[95] The main weakness of GENE_ARCH is its inability to explore different building shapes and layouts are not specifically encoded. Even using shape grammars, variation in the generated geometry is highly restricted. Caldas writes, "The on-going experiments with shape grammars suggest that the method may be too limited to provide the necessary handles on complex three-dimensional problems, and to allow the emergence of unexpected design characteristics, thus suggesting the need for other paradigms."

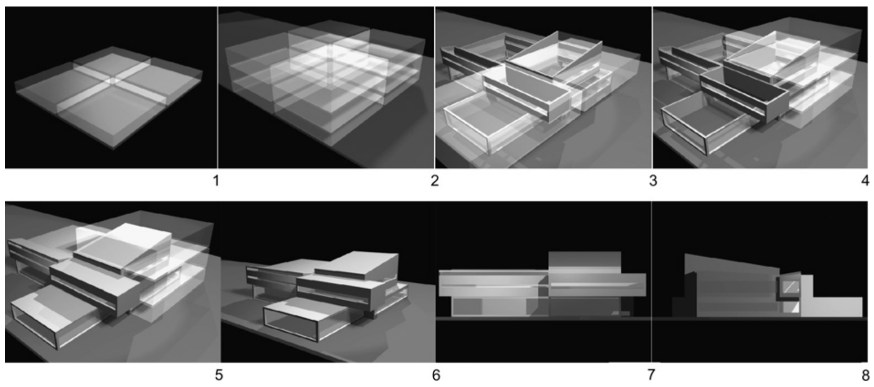


Figure 2.3. 7 3-D Building Designs Generated by GENE_ARCH

2.3.2.3 eifForm

Kristina Shea has developed a design optimization system, called eifForm, that employs a combination of a structural shape grammar and parametric techniques to generate and transform structural design geometry and topology.[96] It executes a recursive cycle of form and structure generation, structural performance evaluation, and parametric modification stages. It

[95] Ibid.
[96] Kristina Shea, et al., "Towards Integrated Performance-based Generative Design Tools," W. Dokonal, ed., *Digital Design, ECAADE 2003*, (Graz Austria: 2003), 253-264

can accommodate various loading conditions and structural and material constraints. In the generative stage, parametric structural shapes are added, removed or modified by a non-deterministic, non-monotonic search algorithm based on simulated annealing, rather than an evolutionary algorithm.[97] In the evaluation stage, structural analysis and stochastic optimization[98] are carried out, outputting a new set of optimally-directed parameters. In the modification stage, the new parameters are input by the human designer, completing the first iteration of the three-stage iterative process. Figure 2.3.8 shows an installation structure generated and optimized by eifForm.

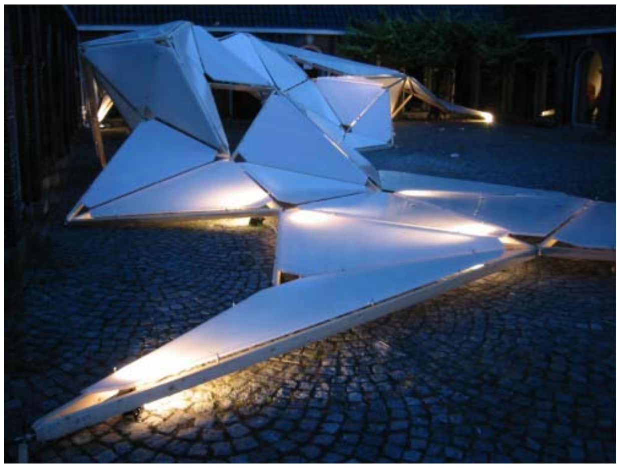


Figure 2.3. 8 Installation Generated and Optimized by eifForm

2.3.2.4 Schema-Based Approach

Janssen has developed an evolutionary design exploration system capable of generating a variety of designs based on a predetermined *design schema*, using a combination of parametric and combinatorial generative modeling techniques.[99] The design schema is a

[97] Wikipedia contributors, "Simulated annealing," *Wikipedia, The Free Encyclopedia* http://en.wikipedia.org/w/index.php?title=Simulated_annealing&oldid=460918505 (accessed 5 Dec. 2011)
[98] Wikipedia contributors, "Stochastic optimization," *Wikipedia, The Free Encyclopedia* http://en.wikipedia.org/w/index.php?title=Stochastic_optimization&oldid=456468729 (accessed 5 Dec. 2011)
[99] Janssen, "A Design Method and Computational Architecture."

conceptualization that "captures the essential and identifiable character of a family of designs."[100] The design schema is encoded in a form that can be used by the evolutionary design system to enable generation of designs that differ in overall organization and configuration but that preclude non-functional or 'chaotic' forms. Essentially, the design schema prescribes a set of constraints used to construct systems to generate designs with controlled variability.

The development system gradually modifies a three-dimensional matrix of cuboid cells, whose dimensions are prescribed by the design schema, using a set of modeling operations that push, pull and incline the faces of cuboids, merge and remove cuboids, and insert architectural objects like columns and staircases. This process is illustrated in Figure 2.3.9. The system is successful in achieving controlled variation, as demonstrated by the sample of generated building designs shown in Figure C, however the types of allowed variation are considerably restricted. In particular, the scale of designs that can be generated is limited to low-rise buildings, and the geometry that can be generated is limited to rectilinear geometry.

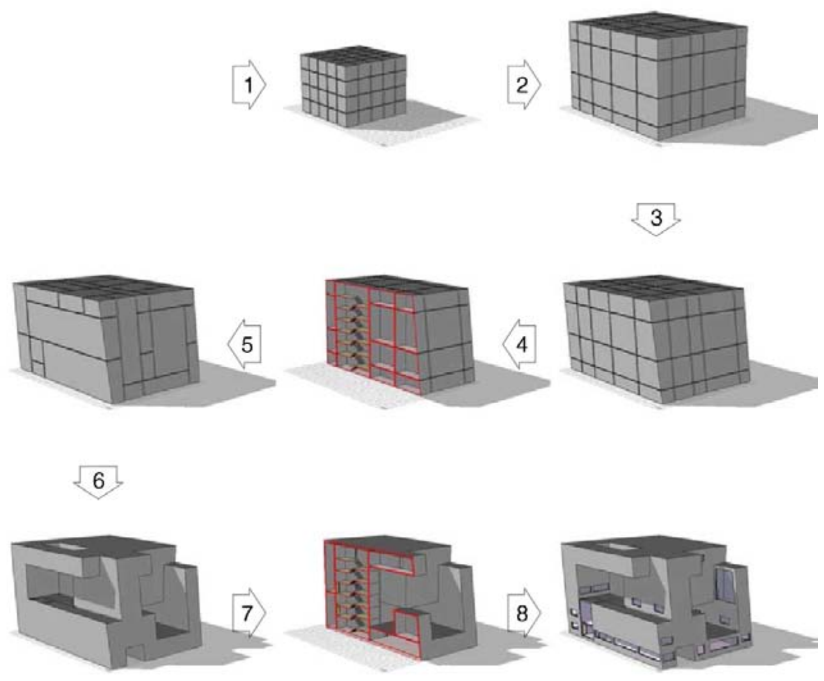


Figure 2.3. 9 Eight-Step Design Generation Process of Jansen's Schema-Based System

[100] Ibid, 20

2.3.2.5 P0L-Systems

Hornby has developed an integral evolutionary design system for evolving 3-D static structures, like tables.[101][102][103] The system combines parametric context-free Lindenmayer systems (P0L-systems) with a LOGO-style turtle to construct objects out of voxels. A string of commands is generated by the P0L-system generates a string of commands that move the turtle around inside an initially empty three-dimensional grid of voxels. Every empty voxel the turtle moves through gets filled in to gradually build up a structure. An evolutionary algorithm is used to evolve populations of P0L-systems by varying conditions, arguments, symbols, and characters of the production rules. The system can be used to evolve table designs containing thousands of voxels by defining a fitness function in terms of table height, surface structure, stability and number of excess voxels used.

As shown in Figure 2.3.10, the P0L-system is able to produce surprising table designs that exhibit complex regularities, and is reported to be faster and produce designs with higher fitness than a comparative system using a non-generative encoding. However, to achieve this performance (for tables) requires the use of no less than twenty complex conditional production rules with two parameters each.

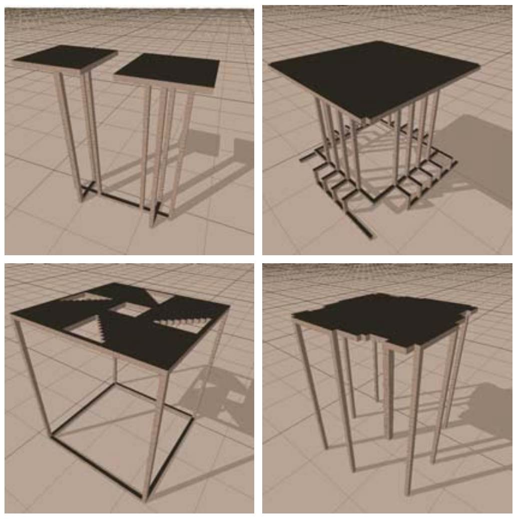


Figure 2.3. 10 Table Designs Evolved Using P0L-System

[101] Hornby, "Functional Scalability," 569-587.
[102] Hornby, "Generative Representations."
[103]Hornby and Pollack, "Advantages of Generative Encoding."

2.3.2.6 GADES

Bentley has developed an integral evolutionary design system, called GADES (Genetic Algorithm Designer), that can adaptively evolve many different types of design.[104] The system generates forms using "clipped stretched cuboids," which are cubes with variable length, width, height, and variable three-dimensional location, that may be clipped by a plane of variable orientation. Each clipped stretched cuboid is thus defined by nine parameters. Forms are generated by combining a number of non-overlapping cuboids based on instructions encoded in hierarchically structured genotype representations. To enable reproduction of offspring from parents with different sized genotype representations, the system employs a modified crossover operator. The evolutionary algorithm is a modified genetic algorithm (GA) using a mapping stage between genotype and phenotype representations, preferential selection of parents, and a life-span operator. Individual designs are evaluated using software modules selected from a library, according to the particular design task.

One of the main features of GADES is its flexibility, in terms of the types of designs it is able to evolve. Bentley claims to have been inspired by nature, which, he explains by quoting Dawkins, uses the same genetic machinery to generate everything from bacteria to blue whales.[105] The flexibility of GADES is due, in large part, to the generic nature of its clipped stretched cuboid-based representations. Bentley has demonstrated how a wide variety of different forms, from the coffee tables shown in Figure 2.3.11 to the race car shown in Figure 2.3.12, can be created by assembling a limited number of these primitives. The cost of this flexibility, however, is that the forms evolved are blocky and exhibit very little detail.

Another important feature of GADES is that it is an integral evolutionary design system, capable of both generating novel forms and optimizing them. This capability too is largely due to the generic nature of the phenotype representations employed. Each clipped stretched cuboid in a design is generated by a linear mapping process of nine parameters, encoded in the genotype representation. Accordingly, epistasis is low and any small change in some particular part of the genotype representation will cause a corresponding small change in the phenotype

[104] Bentley, *Evolutionary Design by Computers*, 405-423.
[105] Dawkins, *Universal Darwinism*

representation. This minimizes discontinuities in the evolutionary search space, allowing the GA to easily find paths from poor designs to better ones.[106]

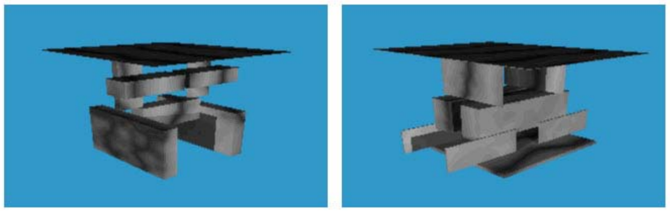


Figure 2.3. 11 Table Designs Evolved by GADES

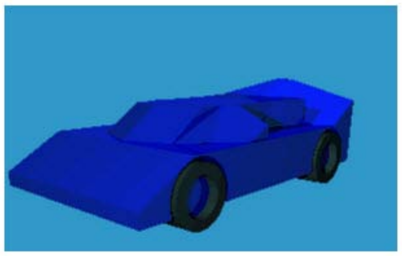


Figure 2.3. 12 Race Car Evolved by GADES

2.3.2.7 HEAD

Bukhari, Frazer, and Drogemuller have proposed an integral evolutionary design system, called the hierarchical evolutionary algorithmic design (HEAD) system, that is divided into three levels "to respond to the hierarchical decomposition of the design problem into sub-problems."[107] The HEAD system includes a room-level algorithm, a building-level algorithm, and an optimization-level algorithm in the configuration shown in Figure 2.3.13.

[106] Bentley, "Generic Evolutionary Design."
[107] Fakhri Bukhari, John H. Frazer, and Robin Drogemuller, "Evolutionary Algorithms for Sustainable Building Design" (paper presented at the 2nd International Conference on Sustainable Architecture and Urban Development, Amman, Jordan, July 12-14, 2010).

The room-level algorithm evolves designs for each space in the building based on unspecified design criteria, standards, codes, etc., and outputs not just one but a number of the most successful variants. The building level algorithm evolves arrangements of the space designs output by the room-level algorithm based on adjacency criteria and outputs a number of the most successful configurations to the optimization-level algorithm. The optimization-level algorithm optimizes the successfully evolved configurations based on thermal, daylighting, acoustical and cost performance.

Only the system architecture shown in Figure 2.3.13 is disclosed. There is no information on the type of generative processes employed in the development systems. Neither is there any data on the performance of the system because, at the time of publication, it had not been implemented.

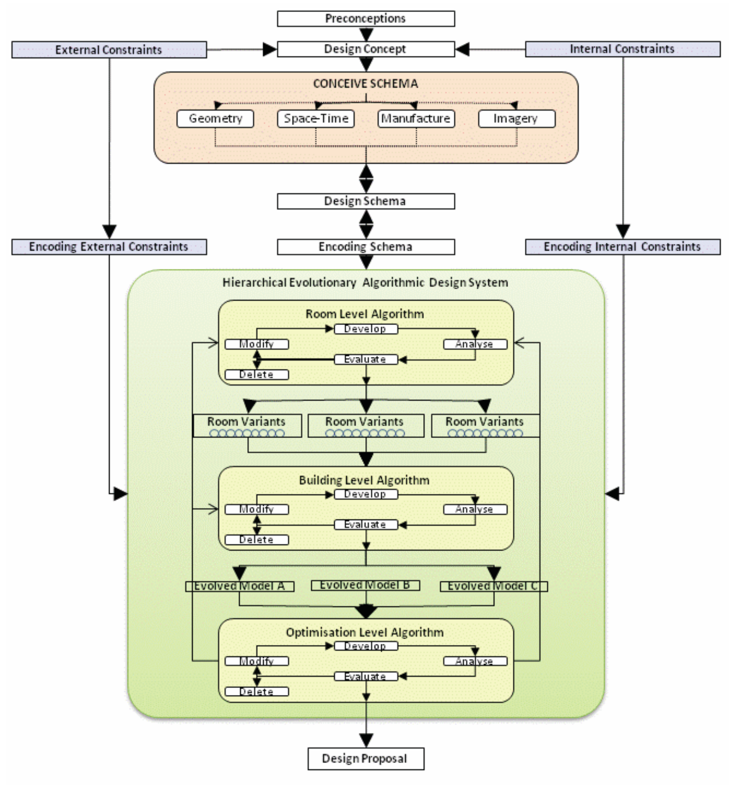


Figure 2.3. 13 HEAD Architecture

While the concept of decomposing the design problem into further levels of sub-problems is quite promising, the HEAD system seems to have some major drawbacks. First, the partial design solutions evolved by the room-level algorithm restrict the variability of room configuration solutions that can be evolved by the building-level algorithm, which in turn restrict the variability of the overall building designs evolved by the optimization-level algorithm. However, this particular hierarchy of constraints may not reflect the priorities of the design problem, in which case they could block off important areas of the design space that should be explored for useful adaptations. For example, in a particular design problem, the overall form of the building or the layout of spaces may take precedence over the design of individual rooms. In this case, the algorithm for evolving the overall building form or layout of spaces should sit at the top of the hierarchy for maximum variability, rather than lower down where the range of allowed variation is constrained by previously evolved design elements.

Further, room, building and overall optimization may not be the most effective choice of sub-problems. Depending on the particular design problem, an alternative deconstruction may be more in-line with building type characteristics and design priorities. For example, a skyscraper design problem might best be broken down into the design of overall massing, then floor-plates, and finally building envelope. And the design of a museum or gallery might best be approached by making circulation the top priority, followed by detailed plans, the building enclosure, and fenestration. The particular breakdown should follow from the building typology and design priorities of the particular problem.

2.3.3 Experimental Design Projects

In the design literature, good examples of applied evolutionary design are still few and far between. There are a number of experiments disclosed in publications of research conducted over the past several years by the students and directors of the Emergent Technologies and Design program at the Architectural Association in London. [108] [109] [110] [111] Most of these

[108] Michael Hensel, Achim Menges and Michael Weinstock, eds., *Emergence: Morphogenetic Design Strategies (Architectural Design)* (New Jersey: Wiley, 2004)
[109] Michael Hensel, Achim Menges and Michael Weinstock, eds., *Techniques and Technologies in Morphogenetic Design (Architectural Design)* (New Jersey: Wiley, 2006)
[110] Michael Hensel and Achim Menges, eds., *Versatility and Vicissitude: Performance in Morpho-Ecological Design* (Architectural Design) (New Jersey: Wiley, 2008)

experiments (hereinafter referred to as the "AA experiments") are characterized by modeling some system or element of the architecture based on the morphology of a chosen plant, animal or other naturally-occurring construct, to provide a starting point and framework for subsequent evolution or optimization. The plant, animal or whatever is chosen based on its performance in relation to predetermined architectural performance requirements. Its morphology is analyzed, and, in an attempt to emulate its performance capacities, certain properties of its morphology are incorporated into a computer model. The model is then evolved or optimized in response to feedback from performance simulations.

A typical example of an AA experiment is the M.Sc. Dissertation of Ioannis Douridas (2006), which focused on the design of a new envelope for the Piraeus Tower in Athens, Greece, a climatically deficient 1970s mid-rise office building with a glass curtain wall.[112] Owing to a lack of insulation, the building suffers from overheating in the summer and insufficient heating in the winter. Research into a cactus revealed that the cactus' hydrostatic and ribbed body reduce thermal gain by self-shading and utilizing airflow. Based on this analysis, a base element was developed from which to algorithmically build up an articulated surface for the building. In this process, a hill-climbing algorithm informed by feedback from performance simulations was used in a process of multi-objective optimization. Figure 2.3.14 is a flowchart of the overall iterative design development process.

The AA experiments demonstrate the need to establish a starting point and constraints for the evolutionary design approach. While the kind of biomimicry[113] employed by Douridas and many others is one strategy for doing this, it may considerably restrict the evolutionary search space, blocking off areas of the design space that may contain useful adaptations. Moreover, the completely different scales and natures of plants and buildings brings into question the appropriateness of these kinds of analogies.

[111] Michael Hensel et al., *Emergent Technologies and Design: Towards a Biological Paradigm for Architecture* (London: Routledge, 2010)
[112] Ibid, 65-73
[113] Wikipedia contributors, "Biomimicry," *Wikipedia, The Free Encyclopedia* http://en.wikipedia.org/w/index.php?title=Biomimicry&oldid=465552583 (accessed 5 Dec. 2011)

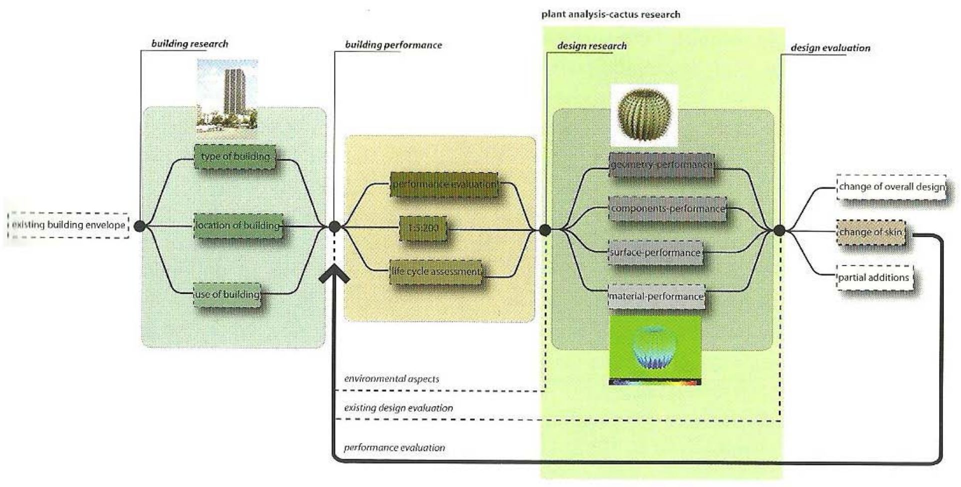


Figure 2.3. 14 Flowchart of iterative design development process of typical AA experiment

Another example of experimental performance-based evolutionary design is shown in Figures 2.3.15 and 2.3.16. Strange Attractor, by MESNE, embodies a different approach to the initial setup of the conditions for form discovery.[114] The architects begin with a strong formal concept, the warped figure-8 of a Lorentz attractor, and employ a bi-directional evolutionary structural optimization (BESO) algorithm to find optimally-directed structural designs. In a fully automated process, "the form of the pavilion emerges through the actions of a guided feedback loop." The architects comment, "What is most interesting about this process of form discovery is the negotiation between the positioning of structural 'boundary conditions' and the design constraints of concept, planning strategy and site."

Strange Attractor illustrates how performance-based evolutionary design can proceed from a formal concept. BESO operates by iteratively removing under-utilized volume elements from a 3-dimentional matrix of elements representing the user-defined maximum building envelope, under user-defined loading conditions and constraints.[115] Though form-finding is driven by only

[114] MESNE, "Strange Attractor," *Contemporary Digital Architecture: Design and Techniques,* Jacobo Krauel, ed. (Links International Ceg, 2010), 200-203.
[115] O.M. Querin , G.P. Steven and Y.M. Xie, "Evolutionary structural optimisation (ESO) using a bidirectional algorithm," *Engineering Computations*, Vol. 15 No. 8 (MCB University Press : 1998), 1031-1048.

one type of performance, structural efficiency, issues of program and site are addressed in the synthesis of the design concept. And though it is a strong formal concept, it is also quite flexible, having no explicit constraints on interior partitioning or fenestration. Thus, it successfully provides a balance of specificity and generality that is conducive to useful variations, which is important in the algorithmic form-evolution process.

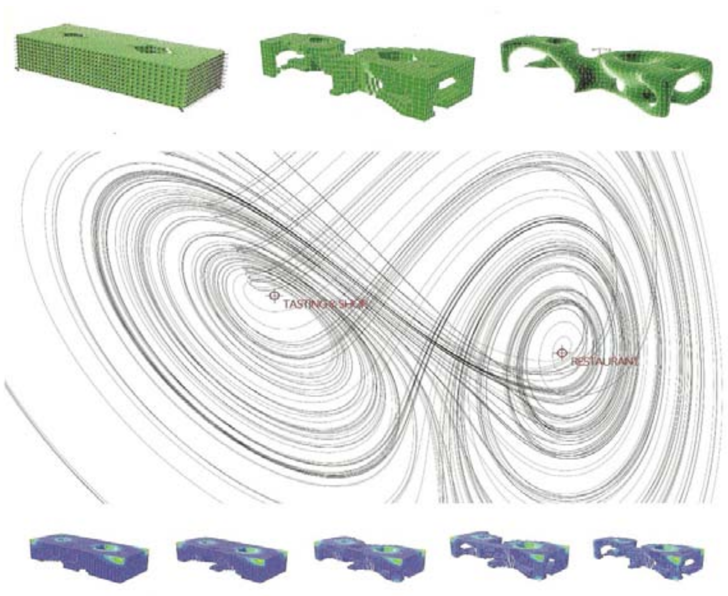


Figure 2.3. 15 Form-finding for 'Strange Attractor' using BESO

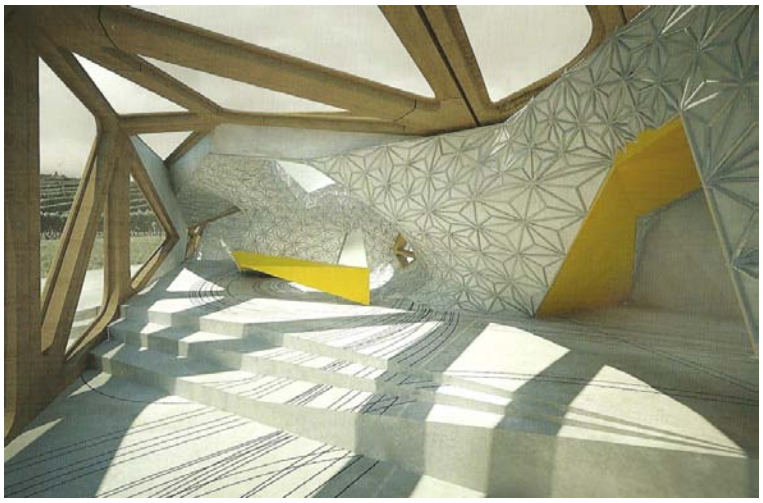


Figure 2.3. 16 Interior Computer Rendering of 'Strange Attractor'

PART 3

RESEARCH DOCUMENTATION

3.1.1 Introduction

Some researchers have suggested a possible alternative strategy for achieving integral evolutionary design that calls for the combination of multiple evolutionary design systems to overcome the barriers to adaptive evolution faced by generative evolutionary design systems, without resorting to low-level generic rules and representations.[116] [117] The proposed combined systems might be called *compound evolutionary design systems*.

Compound evolutionary design systems are composed of a hierarchy of sub-systems, which are themselves evolutionary design systems, working together in coordination on the same design problem but at different resolutions. For example, a compound evolutionary design system might include one sub-system dedicated to evolving individual rooms, another sub-system dedicated to evolving layouts, and yet another system whose purpose is to evolve building envelopes containing stacked layouts. By feeding the outputs of lower-level sub-systems to the next higher-level sub-systems, the overall system functions like a *computational pipeline*. The individual sub-systems of the pipeline may operate one at a time in succession, or concurrently by synchronizing their outputs. There are very few examples of compound evolutionary design systems in the literature, and it is likely that even fewer have been implemented.

The main problem with compound evolutionary design systems is that they are complex, both conceptually and computationally. This is particularly so of compound systems whose sub-systems operate concurrently. Such compound systems tend to have phenotype representations with many interdependent elements and, consequently, large numbers of local optima. If care is not taken in the design of the rules and representations, evolution may commit itself too early to sub-optimal approximations of the best design, some of which are not even functional. While this problem may be relatively minor in the case of simple design problems that do not require solutions with interdependent elements, it can seriously hinder the evolutionary process when applied to more complex architectural design problems.

[116] Bukhari et al., "Evolutionary Algorithms."
[117] Ian Parmee, "Exploring the Design Potential of Evolutionary Search, Exploration and Optimisation," in *Evolutionary Design by Computers*, ed. Peter J. Bentley. (San Francisco: Morgan Kaufmann Publishers, Inc.).

However, when the different sub-systems operate one by one in succession, interdependent elements in different sub-systems are treated independently. This helps to avoid problems that stem from interdependent elements, but it also changes the way the system is capable of optimizing designs. It abandons the idea of a single search for an optimal overall configuration of all elements in favor of a series of searches for optimal configurations of sub-systems of elements. Each sub-system evolves an optimal configuration of some part of the overall design, which, once fixed, imposes constraints on the next sub-system, and so on, until a complete design has been evolved. To emphasize that they evolve complete designs in multiple stages with evolving constraints, rather than in one process with many interdependent elements, compound evolutionary design systems employing a hierarchy of sub-systems operating one-by-one in succession are herein referred to as *multi-stage evolutionary design systems*.

3.1.2 Multi-Stage Evolutionary Design Systems

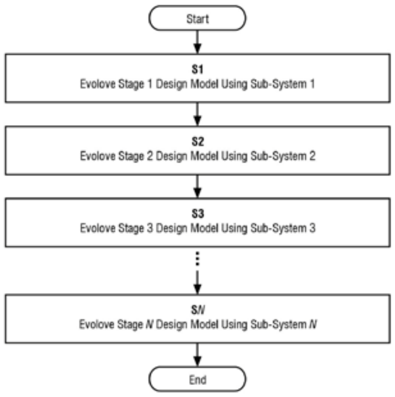

Start
S1
Evolove Stage 1 Design Model Using Sub-System 1
S2
Evolove Stage 2 Design Model Using Sub-System 2
S3
Evolove Stage 3 Design Model Using Sub-System 3
SN
Evolove Stage N Design Model Using Sub-System N
End

Figure 3.1. 1 Multi-Stage Evolutionary Design Process

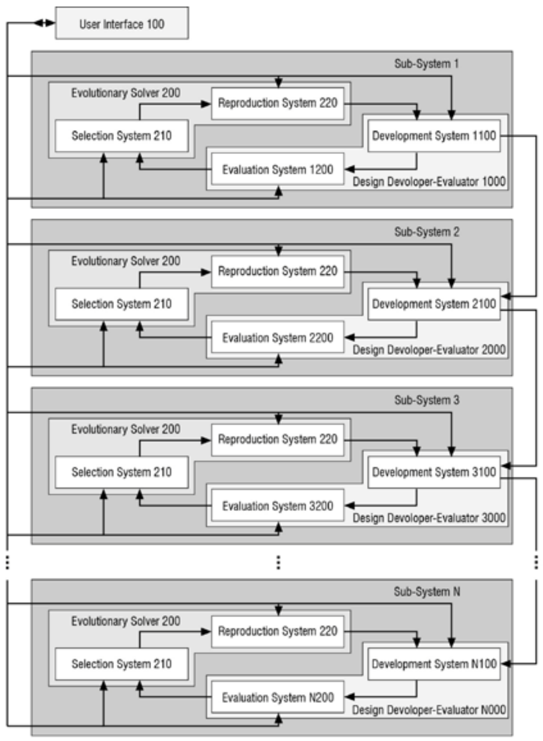


Figure 3.1. 2 General Architecture of N-Stage MSEDS

Figure 3.1.1 is a flowchart illustrating a multi-stage evolutionary design process comprising *N* steps, and Figure 3.1.1 is a schematic diagram illustrating the general architecture of a multi-stage evolutionary design system (MSEDS) comprising *N* sub-systems. While there is no particular limit to the number *N*, in most cases it will be smaller than 10. In general, the number *N* will depend on the size, complexity and level of resolution of the building design to be evolved by the MSEDS, as well as the number of different performance criteria used to evaluate designs in the evolutionary process. This is discussed in more detail below.

Referring to Figures 3.1.1 and 3.1.2, the multi-stage evolutionary design process includes step S1 of evolving a stage 1 design model using sub-system 1, step S2 of evolving a stage 2 design model using sub-system 2, step S3 of evolving a stage 3 design model using sub-system 3, and so on up to step S*N* of evolving a stage *N* design model using sub-system *N.* Here, since sub-systems 1 to *N* are used one by one in succession to evolve different parts of an overall building design, the parts evolved by sub-systems 1 to *N* are referred to as stage 1 to *N* design models. Taken together, they constitute the overall building design evolved by the MSEDS.

Referring to Figure 3.1.2, each of sub-systems 1 to *N* has the same general EDS architecture shown in Figure 2.3.4, except for one difference. Each of sub-systems 1 to *N* is connected to a single user interface 100, which is not considered to be included in any sub-system but rather is a basic component of the overall MSEDS. Each of sub-systems 1 to *N* includes the same evolutionary solver 200, which includes the selection system 210 and the reproduction system 220, but a different design developer-evaluator. Sub-systems 1 to *N* respectively include design developer-evaluators 1000 to *N*000, which respectively include development systems 1100 to *N*100 and evaluation systems 1200 to *N*200. This configuration enables each of sub-systems 1 to *N* to evolve a different part of the overall design.

Sub-systems 1 to *N* are connected in series from sub-system 1 to sub-system *N* to form a computational pipeline. More specifically, development systems 1100 to *N*100 are connected in series from development system 1100 to development system *N*100, so that the design evolved by sub-system 1 is output from development system 1100 to development system 1200, where it acts as a constraint on the evolutionary process in sub-system 2, the design evolved by sub-system 2 is output from development system 1200 to development system

1300, where it acts as a constraint on the evolutionary process in sub-system 3, and so on down the line all the way to sub-system *N*.

3.1.3 System Design

The task of developing an MSEDS to evolve building designs is twofold. It entails:

- breaking down the overall design problem into a set of smaller component problems, and then
- developing individual sub-systems to evolve solutions to the component problems, which, together, constitute a solution to the overall design problem.

These two steps, which in some ways mirror the initial steps of a conventional cognitive design process, will be referred to as: *problem deconstruction* and *sub-system design.*

3.1.3.1 Problem Deconstruction

Problem deconstruction involves strategically breaking down the overall design problem into smaller interrelated design problems to be solved by separate sub-systems within the multi-stage evolutionary design system. This can be approached in the following three steps.

- The first step toward deconstructing the overall design problem is to carefully rank all of its constraints and objects in order of importance. This is to provide a basis for appropriately deconstructing the design problem into smaller parts. The prioritized constraints will also inform the design of a generative process to be employed in each part later in the development of the system.
- The second step is to actually deconstruct the problem. In this process, fundamental architectural concepts are leveraged to impose a basic structure on the evolutionary search space. For example, the preliminary design of a high-rise office building could be broken down into three steps that call for the design of: a generic massing model that works with the program and other constraints, a set of floorplans to populate the massing model and give it shape, and an enclosure system to enclose the model.

- The third step is to hierarchically order the parts of the deconstructed design problem based on the priority of the constraints they introduce. This is to ensure that the evolutionary search space is not constrained in a way that blocks off more important areas where better designs are likely to be found in favor of expanding less important areas. Returning to the above example of a high-rise office building, suppose a design objective specifies, for aesthetics or some other reason, that the building have some predetermined three-dimensional shape, say, the shape of an egg, and further suppose that objective is considered to be more important than any potentially conflicting constraints on the floorplans. Then the task of evolving the egg-shaped enclosure around the massing model probably ought to precede the task of evolving floorplans, to ensure that no compromise in form need be made to accommodate floorplans evolved in advance.

3.1.3.2 Sub-System Design

The term 'sub-system' refers to each of the separate evolutionary design systems needed to evolve solutions to a corresponding part of the decomposed design problem. The sub-systems must be integrated in such a way that the solutions they evolve can be combined to constitute a solution to the overall design problem. Design of the evolutionary algorithms used in the sub-systems is beyond the scope of the present research. Instead, the commercially available Galapagos evolutionary solver is used to implement each of the sub-systems. The task of designing the sub-systems can be broken down into two parts: *development system design* and *evaluation system design.*

<u>Development System</u>

Development system design involves designing a generative process to create phenotype representations from genotype representations for each sub-system of the multi-stage evolutionary design system corresponding to each part of the deconstructed design problem. This step further constrains the evolutionary search space to ensure that elements of building designs are evolved. Considerable care should be taken not to constrain the generative processes in ways that conflict with the priorities assigned in the problem deconstruction step, as this could block off regions of the evolutionary search space where more desirable designs might have been found only to open up regions with relatively less potential.

Returning again to the above example of a high-rise office building, the process used to generate floorplans should first off be constrained to produce plans that fit within the egg-shaped envelope. This could be accomplished by intersecting a horizontal plane at the height of each floor with the surface of the envelope to produce a set of curves yielding the outer perimeters of the floor-plates. A number of other constraints have to be encoded into the generative process, and there is often the potential for conflict. For example, the designer may wish to consider the possibility of an atrium of some kind. In encoding the potential for an atrium into the generative process, he must pay careful attention to the problem constraints. Suppose one design requirement specifies a minimum average daylight factor, while another of higher priority specifies a minimum floor-plate efficiency (area of rentable space divided by total area of floor-plate). In this case, the atrium should be encoded into the generative process in such a way that its size is limited by the minimum floor-plate efficiency.

Evaluation System

Evaluation system design involves designing a system for evaluating generated designs based on functional criteria that reflect relevant performance objectives. Evaluation results are input to a fitness function formulated to output a single-figure appraisal of the quality of the design. If designs are to be evaluated based on multiple conflicting criteria, then multi-objective optimization techniques will have to be employed.

AOF

One way to handle conflicting performance criteria in a multi-objective optimization scenario is to simply add them together. Weight coefficients can be used to reflect their relative importance. The resulting expression is called an *aggregate objective function (AOF).* to illustrate how weights are calculated, consider the following pair of objectives:

$$\text{average daylight factor } (DF_{AVE}) = 5$$

$$\text{area-averaged solar heat gain coefficient } (SHGC_{AVE}) = 0$$

Adding these equations together yields:

$$|\,DF_{AVE} - 5\,| + SHGC_{AVE} = 0$$

Inserting weight coefficients A and B to express the relative importance of the two objectives yields:

$$f \equiv f(DF_{AVE}, SHGC_{AVE}) = A(\,|\,DF_{AVE} - 5\,|\,) + B(\,SHGC_{AVE})$$

Values for the weight coefficients A and B can be determined by specifying appropriate boundary conditions for f, which may be derived from various constraints and/or preferences. For example, if the International Energy Conservation Code (IECC) requires an area-averaged solar heat gain coefficient of 0.4 or less, and the program calls for a daylight factor of at least 2.0, the ratio A/B can be calculated as follows:

$$f\ (5, 0.4) = f(2.0, 0)$$

$$A(\,|\,5 - 5\,|\,) + B(\,0.4) = A(\,|2.0 - 5\,|\,) + B(0)$$

$$0.2B = A$$

Since only the ratio A/B is meaningful, not their individual values, we can conveniently let A = 1 and B = 1/0.2 = 5 to obtain the following AOF:

$$AOF \equiv f(DF_{AVE}, SHGC_{AVE}) = |\,DF_{AVE} - 5\,| + 5(\,SHGC_{AVE})$$

In the general case of multi-objective optimization with n mutually conflicting performance objectives, the AOF can be defined:

$$AOF \equiv f(x_1, x_2, x_3, \dots x_n) = \sum_{i=1}^{n} w_i |x_i - x_i^{opt}|$$

where x_i represents the actual result of the i^{th} performance simulation, x_i^{opt} represents the optimal result of the i^{th} performance simulation, and w_i represents the i^{th} weight coefficient. As in the above example, the weight coefficients may be determined by choosing a set of equally fit (non-optimal) simulation results $(x_1^{unf}, x_2^{unf}, x_3^{unf}, \dots x_n^{unf})$, and writing:

$$w_1|x_1^{unf} - x_1^{opt}| = w_2|x_2^{unf} - x_2^{opt}| = w_3|x_3^{unf} - x_3^{opt}| = \dots = w_n|x_n^{unf} - x_n^{opt}|$$

As the weight coefficients are calculated from boundary conditions set by the designer, this method provides a way to encode design priorities into the evaluation process for multiple evaluation criteria.

3.2.1 Introduction

Most building design problems are sufficiently complex that when design solutions are explored using a single evolutionary design system employing a sufficiently constrained generative process, epistasis and interdependent elements severely hinder adaptive evolution. The underlying strategy of multi-stage evolutionary design is to deconstruct the complex building design problem into smaller, interrelated component design problems that can each be effectively tackled by a different convergent evolutionary design system without epistasis and interdependent elements becoming barriers to adaptive evolution. This approach does limit the evolutionary search space, but it does so in a way that is consistent with the prioritized objectives and constraints of the design problem.

This chapter presents research focused on designing multi-stage evolutionary design systems to adaptively evolve design solutions to a particular category of design problem - the skyscraper.

3.2.2 Design Problem

As a category of design problem, the skyscraper is defined in this research to include all multi-story structures that have a single independent load-bearing structure consisting of a steel framework, and a minimum total height of 100 m (330 ft).

3.2.3 Skyscraper MSEDS Design

According to the multi-stage evolutionary design approach, a given design problem is tackled by first designing a system that addresses a whole class of design problems to which the given problem belongs. The multi-stage evolutionary design system is like a developmental meta-model consisting of generative rules and procedures that can be instantiated as a model of any design in the evolutionary search space by varying its genetic parameters. In these terms, each sub-system is a specific part of the meta-model that can be instantiated as a corresponding part of any design model in the evolutionary search space by variation of its genetic parameters. Designing the system is, in effect, equivalent to designing the evolutionary search space.

3.2.3.1 Problem Deconstruction

Constraint Prioritization

The first step in designing the system involves looking at the general architectural constraints of the particular class of design problem. The following is a list of typical constraints on the skyscraper design problem, grouped according to their source:

General

- multiple storeys
- single independent vertical structure capable of resisting vertical and lateral loads.
- one or more vertical service cores capable of providing required vertical transportation (elevators) and space for routing HVAC risers, plumbing, electrical and data transmission systems.
- flexible enclosure system capable of providing thermal barrier, admitting daylight, and permitting views
- maximum slenderness ratio of 1:20 (beyond this limit there are no precedents)

Codes and Regulations

- means of egress on every floor (details specified by local building code)
- fireproof refuge areas located at minimum intervals (specified by local building code / fire code)
- maximum usable floor area per fire zone
- maximum height specified by local zoning regulations
- maximum floor area ratio (FAR) specified by local zoning regulations
- podium setbacks from lot boundary specified by local zoning regulations
- tower setbacks from lot boundary specified by local zoning regulations
- maximum site coverage (plinth area) specified by local zoning regulations
- minimum green area coverage of ground plane specified by local zoning regulations and design brief
- emergency vehicle access routes and maneuvering zones with access to all elevations and satisfying minimum area requirement specified in fire safety regulations

Design Brief

- gross floor area (GFA)
- breakdown of GFA for program
- target building height
- minimum floor-to-ceiling clearance
- minimum floor-plate efficiency
- maximize floor-plate efficiency
- maximize flexibility of space
- maximize views
- minimize energy consumption
- maximize natural daylight
- maximize indoor comfort
- minimize wind loads
- minimize construction costs
- aesthetic/cultural/stylistic preferences
- preferences related to structural grid

The first group of constraints above relates to the basic functionality and structural soundness of the skyscraper, and the second group relates to legal requirements specified in relevant codes and regulations . Both groups are necessary constraints and therefore may be hard-coded into the generative rules and procedures of the system. Here, constraints related to specific codes and regulations may be encoded as variables so that the system can be easily modified for application to different site conditions or in different localities, and can easily cope with changes in regulations. However, for any given skyscraper design problem, these constraints will be fixed according to the particular site conditions (lot size and boundary, etc.) and the applicable codes and regulations.

The third group of constraints and objectives relates to design requirements and objectives expressed by the client or suggested by the designer. For simplicity, the source of this group will be referred to as the Design Brief, although, as mentioned, some or all objectives in the group may originate from other sources. Not all constraints in the group are immutable and not all are of equal necessity and importance. Rather, they can be selected for consideration or omitted, and they can be arranged along a scale from highest priority to lowest. For any given design problem, the choices of which of these constraints to implement and what level of priority to assign to each of them, are subjective decisions made by the client and/or architect. Some of these decisions may affect the way constraints are hard-coded into the system, which could have an impact on the applicability of the system to other skyscraper design problems with conflicting priorities.

Problem Deconstruction

Design constraints identified above as 'general' or stemming from 'codes and regulations' are necessary constraints, though they won't always specify the same limits for every skyscraper design problem. Together they impose some initial limits on the basic form or *massing* the skyscraper is allowed to have. For example, they directly specify setbacks, a site coverage limit, and a height limit, and, combined with the GFA and floor-to-ceiling clearance requirements from the Design Brief, they specify the general massing of the skyscraper as a function of four or five different design parameters. The precise form of the parametric *massing equations* that result from combining these constraints will be described below. Regarding the present task of deconstructing the design problem, the realization that necessary design constraints can be combined to parametrically describe the general massing of a design solution suggests a way forward that doesn't require any additional assumptions. That is, the skyscraper design problem can be initially broken down into the relatively simple problem of designing its general massing and the remaining problem of designing its specific form and details, without constraining the design space in any unnecessary ways.

The remaining problem of designing the specific form and details of the skyscraper is still quite complex and needs to be further deconstructed. Aside from general massing, some important aspects of the design that need to be addressed early on are:

- the shapes and dimensions of typical and atypical floor-plates
- the type and placement of the service core(s)
- typical floorplans showing partitioning and circulation
- the overall form of the building envelope

At a low level of resolution, these four aspects wholly constitute the remaining part of the design problem. Thus, to further deconstruct the problem, they must be divided into at least two groups. Although they are all interdependent to some degree, the first three aspects are more closely related to one another than to the fourth. The first three all deal with horizontal organization, and they directly address functional space requirements at the human scale, which is a high priority. The fourth aspect, on the other hand, is not directly relevant to functional space requirements at the human scale, but rather to issues such as aesthetic/stylistic preference and wind loading (aerodynamics). Since these issues are of a very different nature than functional space requirements, it is reasonable to expect one of the two sets of issues to dominate the other in priority for any given design problem. Therefore, is it also reasonable to expect that severing the connection between these two components of the design problem will not improperly restrict the design space. Splitting off the fourth aspect listed above from the remaining design problem still leaves a fairly complex problem consisting of the first three highly interdependent aspects. However, it is essentially a 2-dimensional space planning problem and, if carefully encoded, such problems can typically be solved in a convergent evolutionary process.

There is another aspect of the overall design problem that corresponds to a higher level of design resolution and is an important factor affecting a number of performance objectives - the detailed structure and properties of the building envelope or *building skin*. This includes the materials used in the curtain wall facade, the precise pattern of those materials, their thermal and optical properties, as well as the detailed configuration of any shading system employed. Because of its high degree of relevance to various performance goals, it is important to consider design of the building skin early on in the design process. As it corresponds to a higher level of resolution of aspects of the design problem that are predetermined, presumably, in response to higher priority objectives, adding design of the building skin as a separate part of the overall design problem to be tackled by the multi-stage evolutionary design system will not improperly restrict the design space.

The process of deconstructing the overall design problem thus results in identification of the following four sub-problems:

- design of general massing
- schematic design of typical and atypical floor-plates, including type and placement of service core(s), plan partitioning and circulation
- design of overall form of building envelope

- detailed design of building skin

This is not the only possible way to deconstruct the design problem. Care should be taken to ensure that the sub-problems are framed in a way that reflects the particular priorities of the design problem.

Also, the ordering of sub-problems shown above is not the only possible ordering. For example, suppose the client or designer has an aesthetic or stylistic preference for an overall building form that embodies certain sculptural qualities, which is nearly always true. Should such a preference rank higher in priority than the goal of optimizing the functionality of typical floorplans, then the order of the second and third sub-problems could be switched. In this case, the overall form of the skyscraper would largely predetermine the shape(s) of the floor-plates, rather than the other way around.

This flexibility in defining and ordering sub-problems translates into a considerable degree of flexibility in designing the multi-stage evolutionary design system. Reframing and/or reordering sub-problems calls for modifying the system in ways that can redefine the evolutionary search space. By exploiting this flexibility, modified systems can be implemented to explore different design alternatives that represent optimal solutions for different sets of design constraints and priorities. This adds another level of controlled variability to designs that can be produced.

<u>System Architecture</u>

Figure 3.2.1 is a flowchart illustrating the process of evolving a skyscraper design in four stages corresponding to the above four sub-problems. Figure 3.2.2 is a schematic diagram illustrating the general architecture of a multi-stage evolutionary design system (MSEDS) comprising 4 sub-systems designed to evolve a skyscraper design by executing steps S1-S4 of Figure 3.2.1. Figure 3.2.3 is an image illustrating examples of design models evolved in each of steps S1-S4 of Figure 3.2.1.

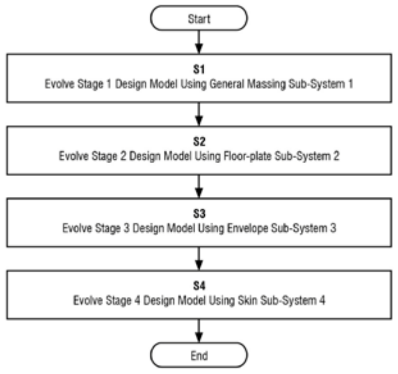

Start
S1
Evolve Stage 1 Design Model Using General Massing Sub-System 1
S2
Evolve Stage 2 Design Model Using Floor-plate Sub-System 2
S3
Evolve Stage 3 Design Model Using Envelope Sub-System 3
S4
Evolve Stage 4 Design Model Using Skin Sub-System 4
End

Figure 3.2. 1 Skyscraper Multi-Stage Evolutionary Design Process

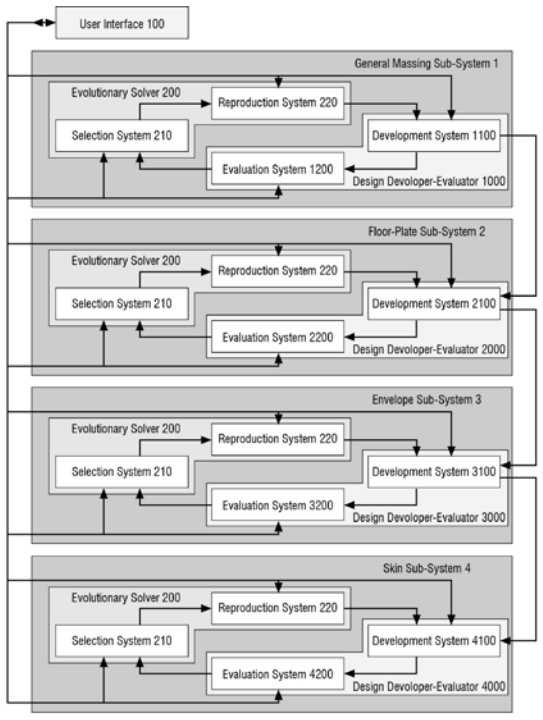


Figure 3.2. 2 Skyscraper MSEDS Architecture

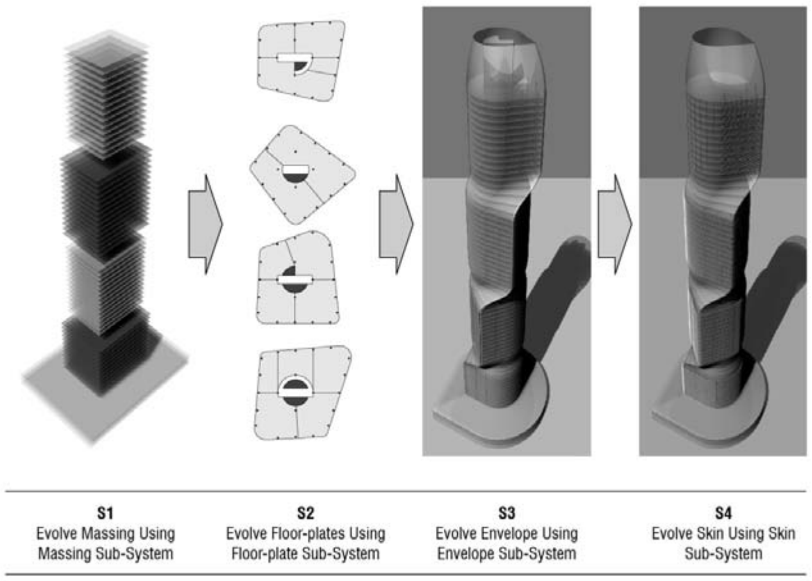

S1
Evolve Massing Using Massing Sub-System
S2
Evolve Floor-plates Using Floor-plate Sub-System
S3
Evolve Envelope Using Envelope Sub-System
S4
Evolve Skin Using Skin Sub-System

Figure 3.2. 3 Examples of Models Produced at Different Stages of the Skyscraper MSED Process

Referring to Figures 3.2.1, 3.2.2, and 3.2.3, in S1, a stage 1 design model representing the general massing of the skyscraper using rectangular planes vertically arrayed and grouped into service zones is evolved using general massing sub-system 1. In S2, a stage 2 design model including at least one floor-plate design for each service zone is evolved using floor-plate sub-system 2. In S3, a stage 3 design model, in which the floor-plates evolved in S2 are vertically arrayed according to the general massing evolved in S1, the arrayed floor-plates are wrapped with a surface representing the building envelope, and service core and structural members are extruded, is evolved using envelope sub-system 3. Finally, in S4, a stage 4 design model including the stage 3 design model evolved in S3 as well as further details such as shading devices and building information representing material construction is evolved using skin sub-system 4.

3.2.3.2 General Massing Sub-System

<u>Development System</u>

As described above, general constraints, constraints specified in relevant codes and regulations, and GFA, when taken together allow the general massing of the skyscraper to be specified in terms of several different design parameters. The main parameters are listed in the table below:

Quantity	Variable
No. floors between refuge/MEP floors	R
Floor-to-floor height	h
Target building height	H
Total no. of floors (including podium)	N
No. of refuge/MEP floors	n_r
No. of tenant floors	n_t
Gross floor area	GFA
Area of typical floor-plate	A

Here, 'refuge/MEP floors' are floors reserved for refuge areas and mechanical rooms, and 'No. floors between refuge/MEP floors' refers to the interval between refuge floors which is limited by the relevant codes and regulations. The other parameters are self explanatory. With these parameter definitions, the parametric massing equations are:

$$N = \frac{H}{h}$$

$$n_r = \frac{N}{R}$$

$$n_t = N - n_r = N\left(1 - \frac{1}{R}\right)$$

$$A = \frac{GFA}{n_t} = \frac{GFA}{\left(N - \frac{N}{R}\right)}$$

These equations can be used to define a parametric massing model of a skyscraper. By setting the GFA to the value required in the Design Brief, and setting the floor-to-floor height based on the floor-to-ceiling clearance required in the Design Brief, the other parameters can be varied within applicable constraints to explore different massing options.

Evaluation System

Objectives for optimizing the massing of the skyscraper are the same as in a conventional manual design process. Some typical objectives are:

- achieve target building height
- minimize number of refuge floors required
- neatly fit program zones between refuge floors
- maximize typical floor-plate area within limit of one fire zone per floor
- typical floor-plate areas that can be neatly divided up into convenient structural bays

The parametric massing model can be evaluated with respect to these criteria without having to simulate or calculate anything, simply by visualizing the model. Thus, while evolutionary algorithms could be used to carry out the optimization, it may be a simple enough task to perform manually. Optimization of the general massing system yields typical floor-plate areas for each program zone.

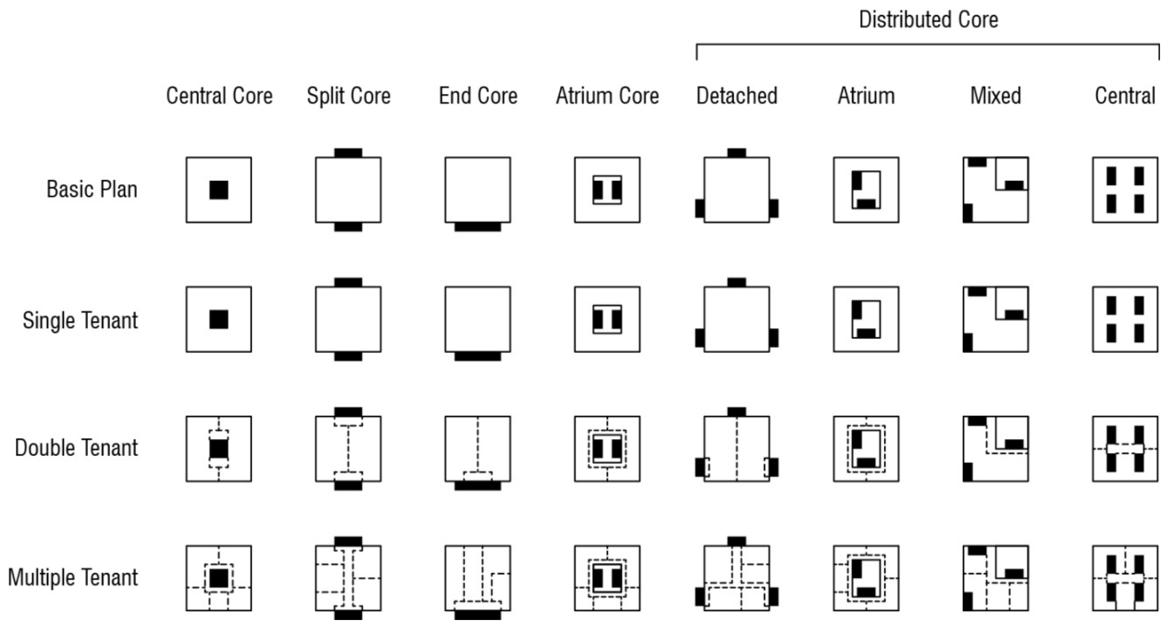
3.2.3.3 Floor-plate Sub-System

Development System

In designing an evolutionary design system to tackle the problem of floor-plate design, the following issues have to be considered:

- service core type and placement
- overall floor-plate shape and dimensions
- structural layout
- partitioning and circulation

Of these, service core type and placement may be the most fundamental and consequential consideration.



3.2. 1 - Service Core Type and Placement Schemes

In skyscrapers, a single centrally located service core is almost universal. The main reason for this is that a central core maximizes the amount of unobstructed, rentable area around the daylit perimeter of the building. However, to allow for the possible discovery of designs that defy this conventional wisdom, it may be desirable to allow the service core to vary in type and placement. Figure 3.2.3 is a chart summarizing some of the different options for core type and placement, along with some of their advantages and disadvantages.

The first task then becomes designing and implementing a generative process that is capable of generating a range of core type and placement options, like those shown in Figure 3.2.3, in a general way that can be applied to different floor-plate shapes, and without generating non-functional configurations. Such a generative process is described in Chapter 3.4.

The next task is to create a generative process for generating a wide variety of floor-plate shapes, to which the core type and placement generative process can be linked. Here, floor-plate shapes are preferably generated with as much variability as possible within certain practical limits. For example, the generated shapes may preferably range from primitive circles and rectangles to complex, irregular curvilinear and rectilinear shapes, so long as the total area for each typical floor-plate stays close to the value determined by optimizing the massing model. It is also preferable to exclude chaotic shapes and shapes with very fine features. A few examples of applicable generative processes are described in Chapter 3.4.

The next task is to devise a process for generating a structural grid and structural members such as columns based on core type, placement and floor-plate shape. The generative process should be capable fitting the structural grid to the floor-plate in structurally logical manner to enable evaluation of the general structural scheme, however it is not necessary for the generative process to be capable of developing detailed structural designs. Such a generative process is disclosed in Chapter 3.4

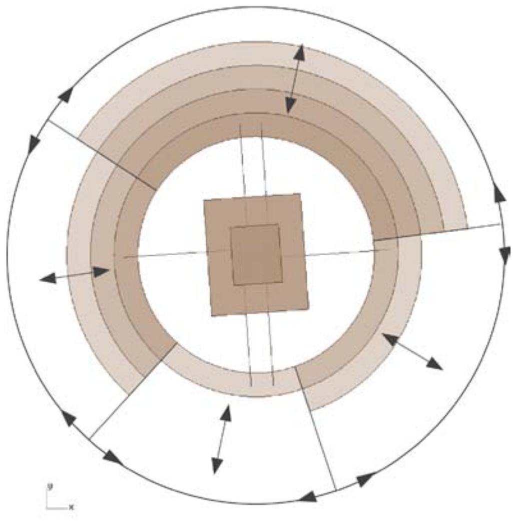
The final task is to encode circulation and variable partitioning as a function of the number of separate rental spaces and the area each specified in the Design Brief. Again, circulation and variable partitioning should be encoded in a general way that can be applied to different floor-plate shapes, and only functional configurations should be generated. Such an encoding is described in Chapter 3.4.

<u>Evaluation System</u>

The next step in designing the floor-plate sub-system 2 is to design an evaluation system for evaluating generated floor-plate designs. The criteria for evaluation should reflect performance objectives specified in the Design Brief. Evaluation results are input to a fitness function formulated to output a single-figure appraisal of the quality of the design. If designs are to be evaluated based on multiple conflicting criteria, then an aggregate objective function (AOF) will have to be constructed as a fitness function. The following are a selection of evaluation techniques that could be used alone or in combination.

View Value

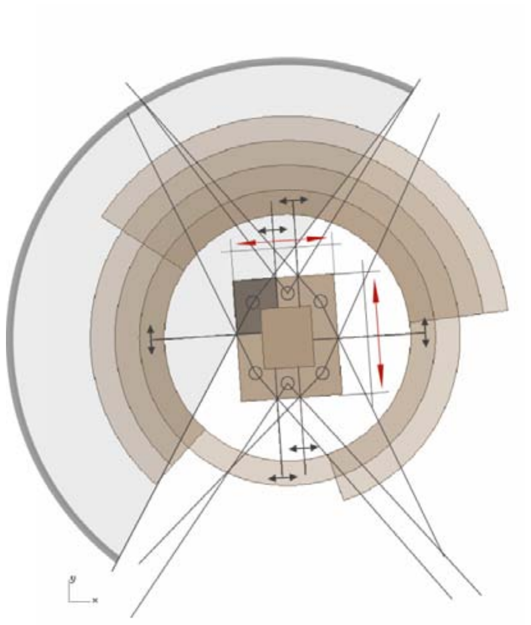
During the course of this research, a method was devised for evaluating floor-plate designs based on a quantitative measure of the quality of views opened up to tenants by the design, referred to as a *view value*. The view value depends not only on the floor-plate and layout of rental spaces, but also on the particular context in which the design is to be constructed and utilized.



3.2. 2 - Parametric View Ring

To evaluate floor-plate designs based on view value, a parametric *view ring* is constructed as shown in Figure 3.2.4. The view ring is centered on the site and divided up into sections corresponding to different views. It can be divided into any number of segments and the angles subtended by the segments can be freely adjusted to match the site conditions of the design problem. Then, the views are ranked from best to worst and the thicknesses of the corresponding sections of the view ring are adjusted to reflect the rankings, with better views getting thicker sections.

Once the view ring is set up, evaluations can be performed during the evolutionary process as shown in Figure 3.2.5. Rays are projected from the geometric center of each rental space defined by a given floor-plate design, in the direction of both corners of the same space at the building facade. If the line of sight from the center of a rental space to either of its corners at the building facade is obstructed by a partition or by the building core, then a ray is projected from the center of the space toward the edge of the obstruction instead of toward the corner. For each rental space, the angular range between the two rays projected from its center is defined as the view range, and the area of the view ring enclosed by the view range is defined as the view value. Defined in this way, the view range of a rental space reflects its facade frontage and depth, and the view value of a rental space reflects its view range and the quality of views within its view range, as represented using the view ring.



3.2. 3 - View Evaluation Process

If all tenants on a floor are to be treated equally, then summing the individual view values calculated for each rental space in a given floor-plate design yields the total view value for that design. If, on the other hand, the intention is to create 'premium' rental spaces for higher paying tenants as well as 'standard' tenant spaces on the same floor, and it is more important that the premium rental spaces have high view values, then weight coefficients may be applied to the individual view values before summing to obtain the total view value of the design. In either case, the larger its total view value, the better the design takes advantage of the views naturally available at the site. In a simple case where this is the only criteria for judging the fitness of floor-plate designs, then total view value may be equated with fitness in the evolutionary process.

Floor-plate Efficiency

Another common design objective is to maximize floor-plate efficiency, defined as the percentage of gross floor area (GFA) that is net rentable area (NRA). Floor-plate efficiency can be approximated by subtracting the total area reserved for the service core(s) and circulation from the total floor-plate area, dividing the result by the total floor-plate area, and multiplying by 100%. In a simple case where this is the only criteria for judging the fitness of floor-plate designs, then the floor-plate efficiency may be equated with fitness in the evolutionary process.

Facade insolation

Another important design objective is to minimize energy consumption, and one of the largest causes of energy consumption in many buildings is the need for mechanical cooling. In any climate, solar radiation incident on the building facade usually makes up a large part of the total cooling load. Accordingly, in order to minimize energy consumption, it is important to try to reduce exposure of the building envelope to solar radiation. This has a direct bearing on the overall 3-dimensional form of the building envelope and therefore, the shapes and orientations of individual floor-plates.

The evaluation of facade insolation is somewhat more complicated than the evaluation of view value or floor-plate efficiency. It requires the use of solar simulation to calculate the angle of incidence of solar radiation on the building envelope over the course of a day and throughout a year. One strategy for deriving an approximate model of the building envelope to use in the facade insolation simulation is to simply extrude the floor-plate vertically. The extrusion may begin from a height corresponding to the lower end of the stack of floor-plates and end at the top of the stack, according to the general massing model.

The facade insolation simulation will typically produce several sets of results, including facade insolation in Watts per square meter averaged over the year, and averaged over each season of the year. Depending on the local climate at the site, it may be desirable to minimize facade insolation year-round, or it only be desirable during hot seasons. Whichever the case may be, the appropriate facade insolation data is selected, and if there are no other evaluation criteria, the negative of the data is equated with fitness. Here, it is necessary to take the negative of the facade insolation results because fitness increases as facade insolation decreases.

Daylight

Numerous studies have shown that the level of natural daylight inside buildings is linked to the health and performance of building occupants. Natural daylighting can also reduce the need for artificial lighting, thereby reducing energy consumption. Accordingly, in designing for occupant health, productivity, and low energy consumption, it is desirable for natural daylight levels to fall within an optimal range in all occupied spaces.

Daylight levels inside skyscrapers vary across each floor according to proximity to the building envelope, the geometry and optical properties of the building envelope, internal partitioning, and solar geometry. Thus, like facade insolation, daylight analysis requires the use of solar simulation and an appropriately constructed analysis model. A model for daylight analysis can consist of surfaces representing two adjacent floor-plates, all internal partitions and the building envelope between those floor-plates, and data representing the optical properties of each surface in the model.

It is difficult to accurately gauge daylight levels without considering the detailed form and properties of the building skin, which will typically be evolved after the building floor-plates. Nevertheless, daylight analysis using a simplified envelope having fixed optical properties and no shading devices can be performed to isolate the daylight performance contribution of the floor-plate design. Here, floor-plate designs may be evaluated based on lowest daylight level, range of variation in daylight level, and/or average daylight level, measured in the simulation. It is typically preferable to maximize lowest daylight level, minimize range of variation, and aim for some target average daylight level. While excessive daylight can be mitigated with shading devices, etc., inadequate penetration of daylight into deep or partitioned-off areas of the floor-plate can only be rectified by changing the floor-plate design.

3.2.3.4 Envelope Sub-System

Development System

The third component of the skyscraper multi-stage evolutionary design system illustrated in Figure 3.2.2 is the envelope sub-system 3. The building envelope is the outer wall (typically a curtain wall) that wraps around the floor-plates and encloses the interior space of the skyscraper. As such, its form is largely determined by the combination of general massing and floor-plate designs evolved using the general massing sub-system 1 and the floor-plate sub-system 2. However, neither of those sub-systems directly develop the form of the skyscraper in elevation. Thus, the envelope sub-system 3 is included as a means of adding a limited degree of variability to the overall form of the skyscraper viewed in elevation, within the constraints imposed by the evolved general massing and floor-plate designs.[118]

The envelope can be treated as a tubular surface wrapped around a stack of floor-plates evolved using the floor-plate sub-system 2 and stacked according the general massing scheme evolved using the general massing sub-system 1. Since its horizontal cross-section at any elevation corresponds to the outer perimeter of the floor-plates arrayed at that elevation, the envelope can be generated using well-known geometric modeling operations, such as lofting, sweeping, or blending, by taking the outer perimeters of floor-plates as profile curves. Accordingly, one strategy for modifying the envelope is to modify its profile curves. This involves selecting floor-plate outlines to use as profile curves, and selectively modifying some of the profile curves before creating the envelope surface. A specific implementation of this strategy is described in Chapter 3.4.

[118] As mentioned in Chapter 3.2, in alternative design problem deconstruction scenarios, the overall form of the skyscraper viewed in elevation may take priority over other design objectives more closely related to floor-plate design. In such a case, the envelope sub-system 3 would precede the floor-plate sub-system 2 in the hierarchy of sub-systems constituting the skyscraper MSEDS and, accordingly, would not be constrained by floor-plate designs but only by general massing.

Evaluation System

Even small modifications to the form of the building envelope can significantly impact building performance in a number of areas such as aerodynamics and self-shading.[119] The aerodynamics of skyscrapers is structurally very important. Aerodynamic forms may require less structure due to lower wind loads, and properly modulated forms may be less susceptible to wind-driven oscillation. Shaping of the building envelope might also aim to capture winds as part of a natural ventilation strategy, or to channel winds through wind turbines to produce electricity.

Whatever the particular design objective(s), evaluation of aerodynamic performance generally requires computational fluid dynamics (CFD) simulation. For external CFD analysis, site-specific wind data and a simple model of the building envelope surface are all that is required, whereas for CFD analysis of internal wind-driven airflow, a model of the envelope fenestration and internal partitions are additionally required. In some cases, it may be desirable to evaluate buoyancy-driven airflow inside a skyscraper or inside the air cavity of a double-skin facade (DSF). This will require a more sophisticated model and CFD analysis.

Meanwhile, inclination of the surface of the building envelope with respect to the vertical can produce self-shading effects, thereby lowering facade insolation and energy consumption associated with cooling. Two potential evaluation strategies are outlined below.

Wind Load

The wind pressure on the envelope can be simulated using wind-driven CFD and a simple surface model of the building envelope. If the envelope model is encoded to have variable profile curves and/or variable openings through which air can flow, these properties may be varied by manipulating corresponding genetic parameters to produce a population of alternative envelope designs. The maximum wind pressure on the envelope under design conditions may be simulated using CFD and used to evaluate the alternative designs. Here, the fitness should be defined to increase as the simulated wind pressure decreases.

[119] Other considerations include reflective glare, solar lensing, etc.

Facade insolation

Reduced facade insolation as a result of self-shading provides other potential criteria for evaluating different envelope designs. In the simplest case, a solar simulation is performed on simple surface models of envelope design variants, produced using a development system employing variable profile curves. Depending on whether it is desirable to minimize facade insolation year-round or only during hot seasons, the appropriate facade insolation data is selected, and fitness is defined to increase as facade insolation decreases.

3.2.3.4 Building Skin Sub-System

Development System

The final component of the skyscraper multi-stage evolutionary design system illustrated in Figure 3.2.2 is the skin sub-system 4. Here, 'skin' refers to the detailed material construction of the building envelope, including shading devices. While the envelope sub-system 3 is used to evolve the overall form of the building envelope, the skin sub-system 4 is used to evolve the material properties and any other performance-related features of the building envelope, such as shading devices or double skin systems.

Environmental inputs such as solar loads are not uniform across the entire surface of the building envelope. Therefore, building skin designs that yield the best performance are not likely to have the same characteristics all over. In order to enable different regions of the building skin to have different properties, it is necessary to sub-divide the envelope into regions. Here, the method of sub-division should allow the regions to vary in size and location. One such method is described in Chapter 3.4.

Material properties can generally be designated by appropriately setting parameter values and require no generative process. Physical objects like louvers, fins, double skin facades (DSFs), and secondary cladding systems, on the other hand, must be generated for each region of the sub-divided building envelope. It is important to encode sufficient variability into the generative processes used to generate these systems. In the case of louvers, for example, louver width, displacement from the surface of the building envelope, and spacing interval should all be controlled variable genetic parameters. One example of such a generative process is described in Chapter 3.4.

Evaluation System

The material systems of the building skin, together with the geometry of the building envelope, have a considerable impact on thermal performance, daylight penetration, and access to views. They are also a major factor affecting the overall cost performance and aesthetics of the skyscraper design. Any one or combination of these performance areas could be selected as criteria for evaluating building skin designs. A few examples are outlined below.

Facade insolation

The degree to which shading devices shade the building facade from direct solar radiation is a key factor affecting thermal performance. Facade insolation of a facade clad in shading devices can be measured by performing a solar simulation on a model of the facade including the shading devices. Here, the length of time required to perform the simulation typically increases dramatically as the geometry of the model increases in number of parts and complexity, and the evolutionary process entails repeatedly performing the simulation for each new design in the population, each generation. Therefore, to prevent the evolutionary process carried out by the skin sub-system 4 from taking too long, it is preferable to include only as much of the building skin as is necessary to obtain reliable simulation results and no more. This might mean extracting a horizontal slice of the skin one or two floor-heights thick to represent a larger region in the analysis.

Daylight

Certain material systems and shading devices can significantly restrict penetration of daylight through the building envelope. In some cases this may be desirable, but it is often an unwanted consequence of using such systems and devices to reduce solar heat gain. For each different skin design, daylight levels inside the building can be measured by performing a daylight simulation on a model consisting of surfaces representing two adjacent floor-plates, all internal partitions and the building envelope between those floor-plates, all external and internal shading devices, and data representing the optical properties of each surface in the model.

Cost Performance

Every aspect of a building design has an associated cost performance, however it is often difficult to measure cost performance until a later stage of the design process when the design is quite highly resolved. If, however, the construction costs of different material systems under consideration for the building skin are known in advance, then rough energy performance simulation results could be coupled with cost data to perform a net present value (NPV) analysis for each different skin design. By setting the term of the NPV analysis, the discount rate, the cost of energy, and the inflation rate, the calculated net present value of each design could be defined as proportional to fitness.

3.3.1 Introduction

This Chapter describes the design and implementation of a multi-stage evolutionary skyscraper design system (skyscraper MSEDS) based on the methodology and general architecture described in Chapter 3.3. A specific skyscraper design problem is taken as a reference for problem deconstruction and other specific system design tasks: The Rolansberg Hanking International Schematic Design Competition held in August and September, 2012 (hereinafter, referred to as the "Hanking Competition"). The end result is a skycraper MSEDS design and implementation tailored to adaptively evolve designs that respond to the requirements and objectives of the Hanking competition.

3.3.2 Design Problem

Figure 3.3. 1 Hanking Competition Site and Land Use Map

Figure 3.3. 2 View of Shenzhen at Night Showing Location for Skyscraper

The Hanking competition called for the design of a 330m-tall office skyscraper boasting a combined GFA of 110,000m^2 in the city of Shenzhen in Nanshan, China. The client was Rolansberg Hanking, the second largest land developer based in Shenzhen, and the top of the Hanking Center was to serve as their headquarters.

Figure 3.3.1 is a map showing the site for Hanking Center and land use in the neighborhood, and Figure 3.3.2 is a view of Shenzhen at night, showing the location of Hanking Center. The site was 11,000m^2, roughly rectangular with the broad side oriented North-South. The surrounding neighborhood consisted mainly of low-rise residential, with a few high-rise office buildings under 150m tall in the vicinity. The site bordered on a major thoroughfare with the best views in either direction down the thoroughfare and to the south over a university campus.

3.3.3 Skyscraper MSEDS Design

3.3.3.1 Problem Deconstruction

Constraint Prioritization

The first step in designing the system involves studying the problem constraints and objectives and arranging them in order of priority. General constraints on the skyscraper class of design problem are discussed in Chapter 3.3. The following is a prioritized list of constraints specific to the Hanking Competition:

Codes and Regulations (necessary)

- land coverage = 11,000m^2
- land use - commercial office building
- maximum height specified by zoning regulations = 330m + 20m cheat zone = 350m
- maximum floor area ratio (FAR) specified by local zoning regulations = 10
- podium setbacks from lot boundary specified by local zoning regulations = 10m (North-South), 8m (East-West)
- tower setbacks from lot boundary specified by local zoning regulations
- maximum site coverage (plinth area) specified by local zoning regulations = 63%
- minimum green area coverage of ground plane specified by local zoning regulations and design brief = 15%

- means of egress on every floor (details specified by local building code)
- fireproof refuge areas located at minimum intervals of fourteen floors
- maximum usable floor area per fire zone = 2000m^2
- emergency vehicle access routes and maneuvering zones with access to all elevations and satisfying minimum area requirement specified in fire safety regulations

<u>Design Brief (requirements)</u>

- gross floor area (GFA) = 110,169m^2
- breakdown of GFA for program:
- commercial office area = 62,719m^2
  - innovative industries = 4,400m^2
  - business apartments = 33,050m^2
  - commercial = 10,000m^2
  - commercial part of basement - 4,000 - 5,000m^2
- target building height - not specified
- minimum floor-to-ceiling clearance = 3m
- minimum floor-plate efficiency - not specified

<u>Design Brief (performance objectives)</u>

- maximize floor-plate efficiency
- maximize flexibility of space
- minimize energy consumption
- maximize views
- maximize natural daylight
- maximize indoor comfort
- iconic landmark
- minimize wind loads
- minimize construction costs

Problem Deconstruction

The above prioritized list of constraints confirm that constraints related to codes, regulations and requirements in the Design Brief together specify the general massing of the skyscraper as a function of four or five different design parameters. Also, they indicate that the client places greater importance on efficiency, functionality and sustainability than on achieving some particular aesthetic, which suggests that the overall building form should follow, in large part, from the design of the floor-plates and the overall massing. These observations confirm the appropriateness of deconstructing the problem into the following four parts, as described in Chapter 3.3:

- design of general massing
- schematic design of typical and atypical floor-plates, including type and placement of service core(s), plan partitioning and circulation
- design of overall form of building envelope
- detailed design of building skin

System Architecture

This problem deconstruction yields the general system architecture illustrated in Figures 3.3.1 and 3.3.2. Figure 3.3.3 is a schematic diagram illustrating the skyscraper MSEDS general architecture of Figure 3.3.1 in more detail. Referring to Figure 3.3.3, the design developer-evaluator of each of sub-systems 1 through 4 includes X number of development modules (shown in blue) in the development system, and Y number of performance evaluation modules (shown in blue) in the evaluation system. The task of designing a particular skyscraper MSEDS based on this architecture entails specifying the number, configuration, functions and interrelationships of these modules. The main objectives and considerations relevant to this task are discussed in Chapter 3.3. The remainder of this chapter describes the sub-systems of a specific MSEDS design, aimed at evolving design solutions for the Hanking Competition, and its implementation.

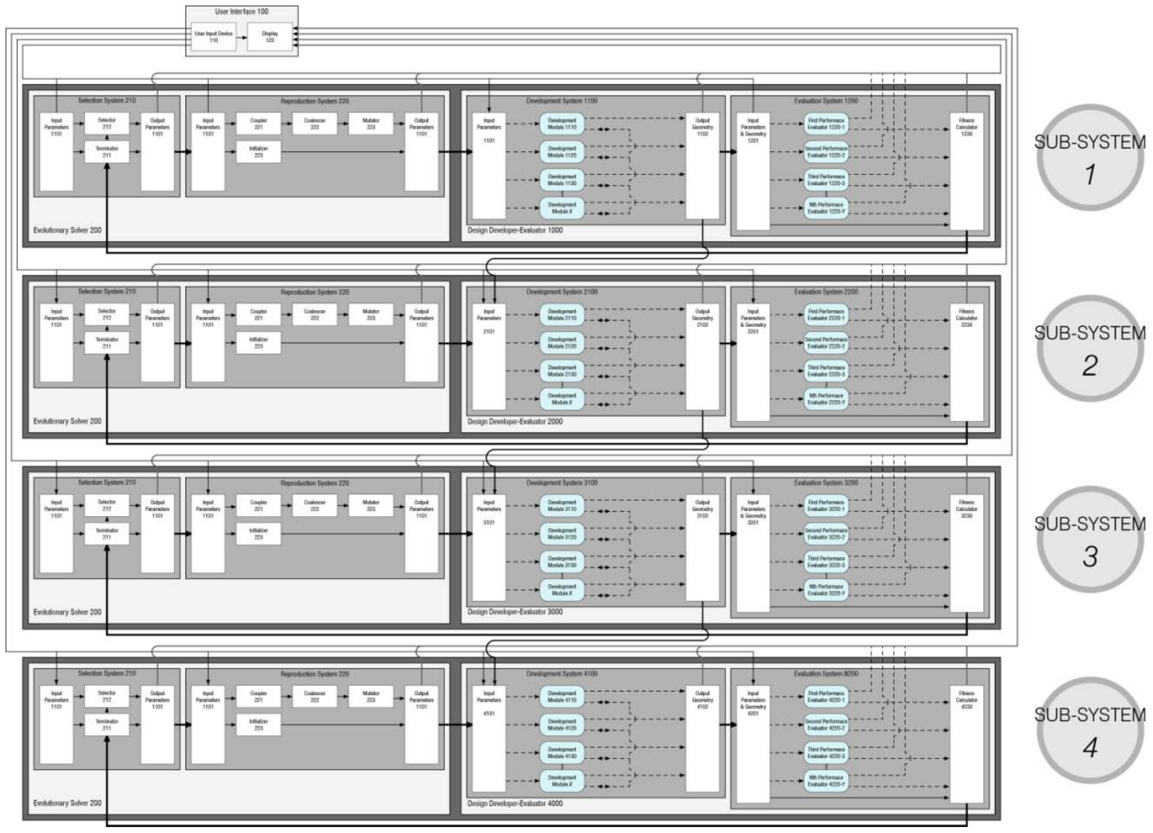
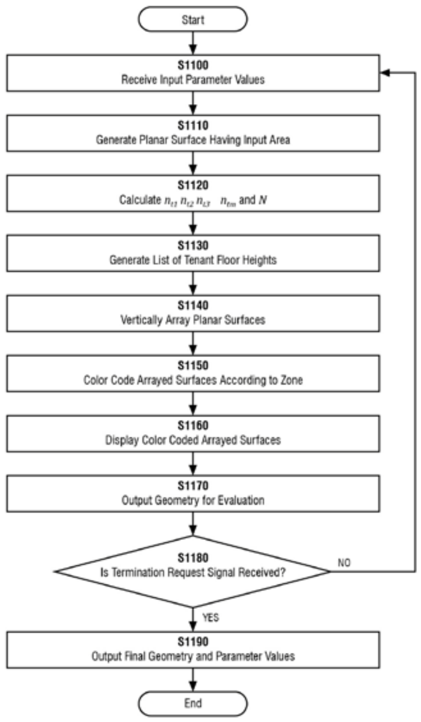


Figure 3.3 1 Skyscraper MSED Architecture

3.3.3.2 General Massing Sub-System

Development System

Figure 3.3.4 is a flowchart illustrating the process of developing general massing models using the development system 1100 of the design developer-evaluator 1000 of the general massing sub-system 1 shown in Figure 3.3.3. Figure 3.3.5 is a schematic diagram of the general massing sub-system 1 shown in Figure 3.3.3, illustrating the general design of the development system 1100 of the design developer-evaluator 1000.


Start
S1100
Receive Input Parameter Values
S1110
Generate Planar Surface Having Input Area
S1120
Calculate n_{t1} n_{t2} n_{t3} n_{tm} and N
S1130
Generate List of Tenant Floor Heights
S1140
Vertically Array Planar Surfaces
S1150
Color Code Arrayed Surfaces According to Zone
S1160
Display Color Coded Arrayed Surfaces
S1170
Output Geometry for Evaluation
S1180
Is Termination Request Signal Received?
NO
YES
S1190
Output Final Geometry and Parameter Values
End

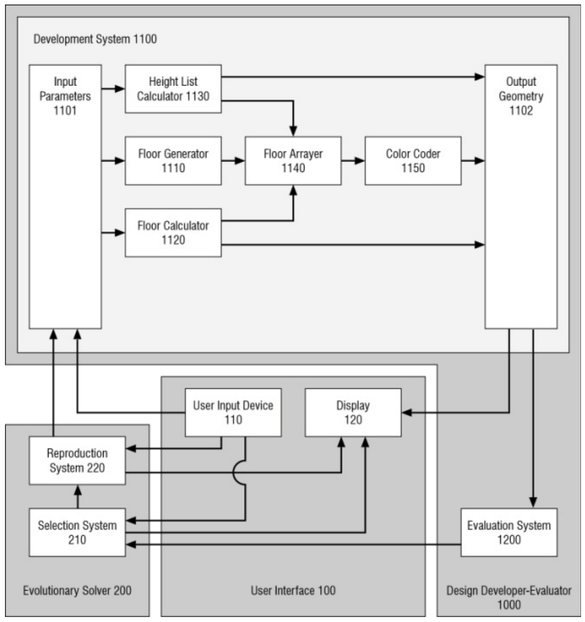
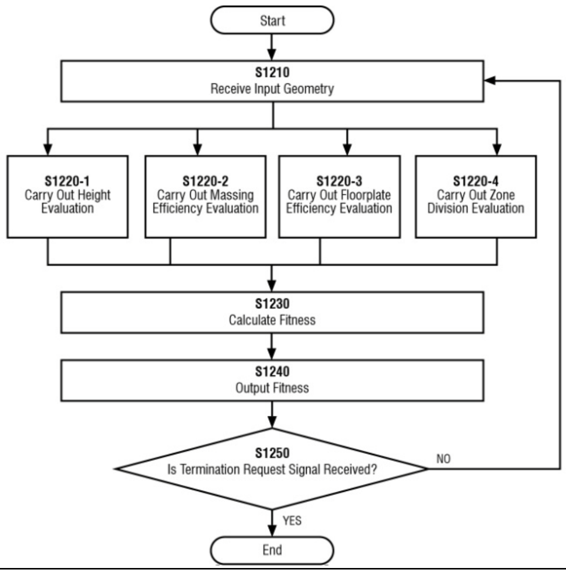


Figure 3.3 2 Massing Model Development Process

Figure 3.3 3 Massing Sub-System 1 Showing Design of Development System 1100

Referring to Figures 3.3.4 and 3.3.5, in S1100, values of input parameters 1101 are received from the user input device 110 of the user interface 100 and/or the reproduction system 220 of the evolutionary solver 200. In S1110, a floor generator 1110 generates a planar surface of area specified by input parameter values for each program zone. In S1120, a floor calculator 1120 calculates the total number of floors *N* and the number of tenant floors n_t in each service zone from input parameter values using the parametric massing equations described in Chapter 3.2. In S1130, a height list calculator 1130 calculates a list of elevations of all tenant floors from input parameter values. In S1140, a floor arrayer 1140 vertically arrays a number of each of the planar surfaces generated by the floor generator 1110 corresponding to the number of tenant floors n_t calculated by the floor calculator 1120 at the at the heights calculated by the height list calculator 1130. In S1150, a color coder 1150 color codes the planar surfaces vertically arrayed by the floor arrayer 1140 according to program zone. In S1160, the massing model including the vertically arrayed planar surfaces is displayed on the display 120 of the user interface 100. In S1170, the outputs of the floor calculator 1120, the height list calculator 1130 and the color coder 1150 are output to the evaluation system 1200. In S1180, the selection system 210 of the evolutionary solver 200 determines whether or not a termination request signal is received from the user input device 110 or the terminator 211 (refer to Figure 3.3.3). If the termination request signal is received, in S1190, the outputs of the floor calculator 1120, the height list calculator 1130 and the color coder 1150 are output to the development system 2100 of the floor-plate sub-system 2. If it is determined that the termination request signal is not received, the process returns to S1100 and new input parameter values are received.

Evaluation System


Start
S1210
Receive Input Geometry
S1220-1
Carry Out Height Evaluation
S1220-2
Carry Out Massing Efficiency Evaluation
S1220-3
Carry Out Floorplate Efficiency Evaluation
S1220-4
Carry Out Zone Division Evaluation
S1230
Calculate Fitness
S1240
Output Fitness
S1250
Is Termination Request Signal Received?
NO
YES
End

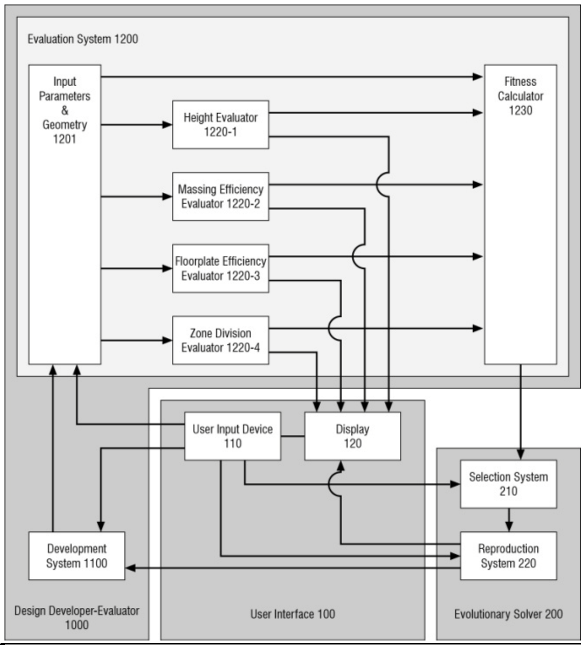


Figure 3.3 4 Massing Model Evaluation Process

Figure 3.3 5 General Massing Sub-System 1 Showing Design of Evaluation System 1200

Figure 3.3.6 is a flowchart illustrating the process of evaluating general massing models using the evaluation system 1200 of the design developer-evaluator 1000 of the general massing sub-system 1 shown in Figure 3.3.3. Figure 3.3.7 is a schematic diagram of the general massing sub-system 1 shown in Figure 3.3.3, illustrating the general design of the evaluation system 1200 of the design developer-evaluator 1000.

Referring to Figures 3.3.6 and 3.3.7, in S1210, geometry and parameter values output from the development system 1100 are received as input parameter and geometry 1201. In S1220-1, a height evaluator evaluates the overall height of the received geometry and outputs the result to the display 120 of the user interface 100 and to a fitness calculator 1230. In S1220-2, a massing efficiency evaluator 1220-2 evaluates the efficiency of the received geometry and outputs the result to the display 120 of the user interface 100 and to the fitness calculator 1230. In S1220-3, a floor-plate efficiency evaluator 1220-3 evaluates the efficiency of the received geometry and outputs the result to the display 120 of the user interface 100 and to the fitness calculator 1230. In S1220-4, a zone division evaluator 1220-4 evaluates how closely divisions between program zones coincide with refuge/MEP floors that divide service zones and.outputs the result to the display 120 of the user interface 100 and to the fitness calculator 1230. In S1230, the fitness calculator 1230 calculates the fitness of the received geometry by adding the outputs of the height evaluator 1220-1, the massing efficiency evaluator 1220-2, the floor-plate efficiency evaluator 1220-3, and the zone division evaluator 1220-4. In S1240, the fitness calculator outputs the calculated fitness value to the selection system 210 of the evolutionary solver 200. In S1250, the selection system 210 of the evolutionary solver 200 determines whether or not a termination request signal is received from the user input device 110 or the terminator 211 (refer to Figure 3.3.3). If the termination request signal is received, the evaluation process ends, and if the termination request signal is not received, the evaluation process is repeated from S1210 for new input geometry and parameter values.

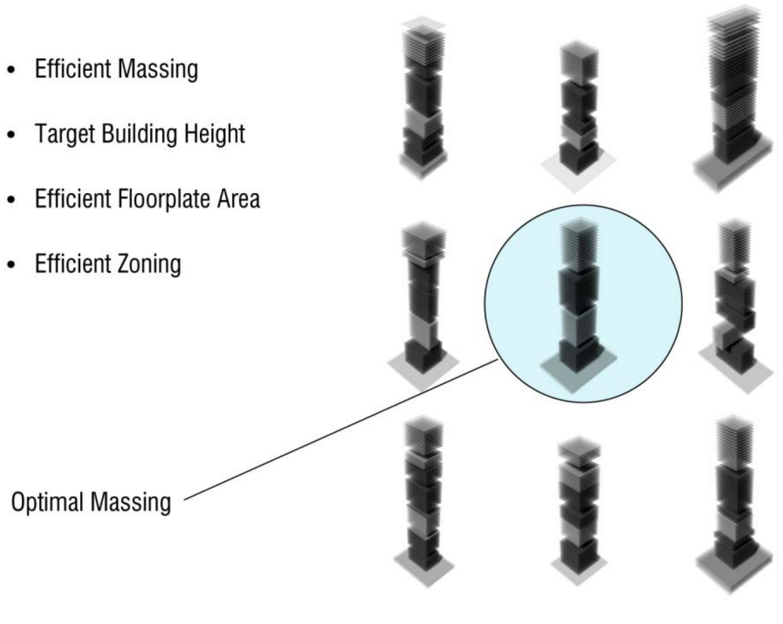


Figure 3.3 6 Selection of Optimal Massing Model Generated by Development System 1100

Figure 3.3.8 illustrates different massing models generated by the development system 1100, including the optimal massing model evaluated to be most fit in terms of massing efficiency (total number of floors / number of refuge zones), proximity to target height, ability to divide floor-plates into convenient structural bays, and how closely divisions between program zones coincide with refuge/MEP floors that divide service zones.

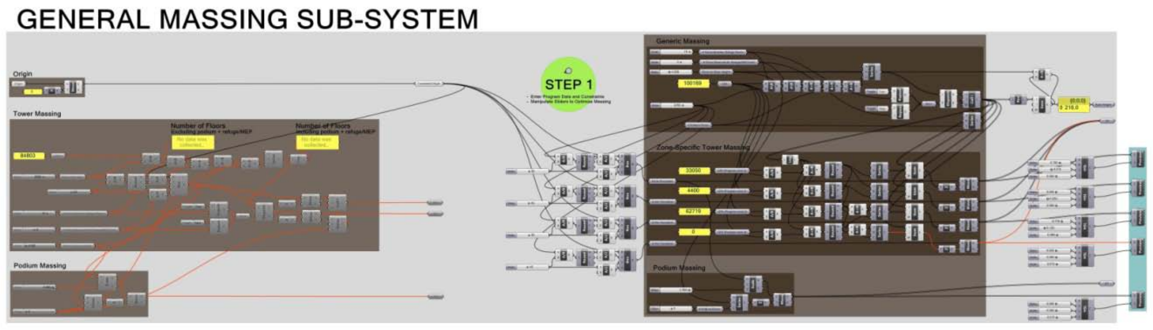


Figure 3.3 7 Grasshopper Implementation of General Massing Sub-System 1

Figure 3.3.9 illustrates an implementation of the general massing sub-system 1 in the Grasshopper graphical scripting language.

3.3.3.3 Floor-plate Sub-System

Development System

Figure 3.3.10 is a flowchart illustrating the process of developing floor-plate designs using the development system 2100 of the design developer-evaluator 2000 of the floor-plate sub-system 2 shown in Figure 3.3.3. Figure 3.3.11 is a schematic diagram of the floor-plate sub-system 2 shown in Figure 3.3.3, illustrating the general design of the development system 2100 of the design developer-evaluator 2000.

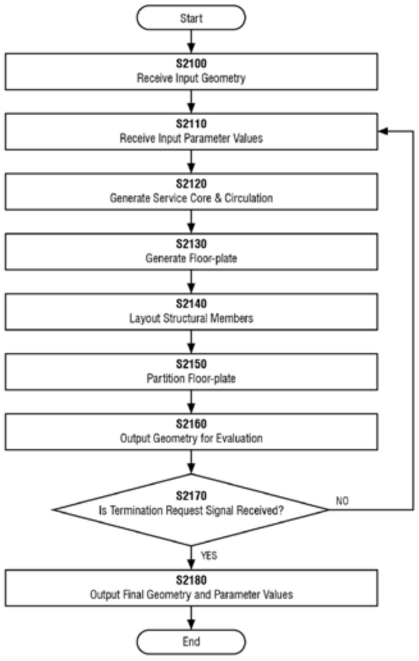

Start
S2100
Receive Input Geometry
S2110
Receive Input Parameter Values
S2120
Generate Service Core & Circulation
S2130
Generate Floor-plate
S2140
Layout Structural Members
S2150
Partition Floor-plate
S2160
Output Geometry for Evaluation
S2170
Is Termination Request Signal Received?
NO
YES
S2180
Output Final Geometry and Parameter Values
End

Figure 3.3 8 Floor-plate Design Development Process

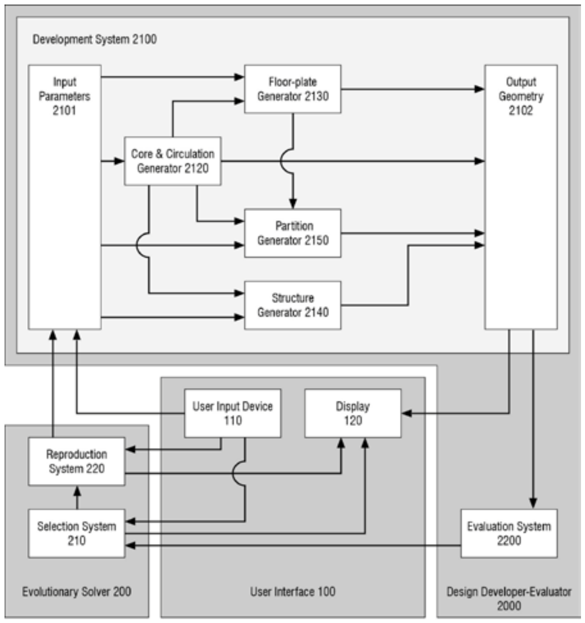
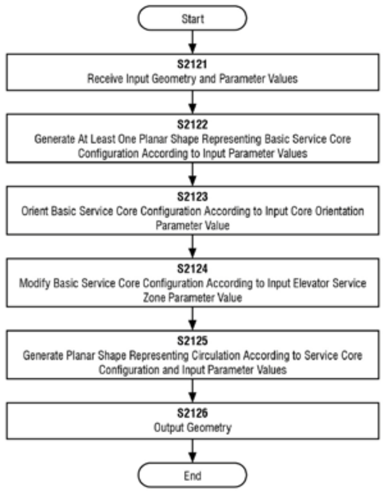


Figure 3.3 9 Floor-plate Sub-System 2 Showing Design of Development System 2100

Referring to Figures 3.3.10 and 3.3.11, in S2100, final geometry and parameter values output from the general massing sub-system 1 are received as input parameters 2101. In S2110, additional input parameter values are received from the user input device 110 of the user interface 100 and/or the reproduction system 220 of the evolutionary solver 200. In S2120, a core & circulation generator 2120 generates geometry representing a service core and circulation for a floor-plate design, based on input parameter values. In S2130, a floor-plate generator 2130 generates a planar surface representing a floor-plate, based on input parameter values and the core and circulation geometry generated by the core & circulation generator 2120. In 2140, a structure generator generates a structural grid and lays out geometry representing structural members such as columns, based on the floor-plate geometry generated by the floor-plate generator 2130 and the service core and circulation geometry generated by the service core & circulation generator 2120. In S2150, a partition generator 2150 partitions the floor-plate into rental spaces based on input parameter values and the core and circulation geometry generated by the core & circulation generator 2120. In S2160, geometry generated by the floor-plate generator 2130, the structure generator 2140, and the partition generator 2150 is output as output geometry 2102 to the display 120 of the user interface 100, and to the evaluation system 2200. In S2170, the selection system 210 of the evolutionary solver 200 determines whether or not a termination request signal is received from the user input device 110 or the terminator 211 (refer to Figure 3.3.3). If the termination request signal is received, in S2180, the output geometry 2102 is output to the development system 3100 of the envelope sub-system 3. Meanwhile, if the termination request signal is not received, the process is repeated from S2110 for new input parameter values.


Start
S2121
Receive Input Geometry and Parameter Values
S2122
Generate At Least One Planar Shape Representing Basic Service Core Configuration According to Input Parameter Values
S2123
Orient Basic Service Core Configuration According to Input Core Orientation Parameter Value
S2124
Modify Basic Service Core Configuration According to Input Elevator Service Zone Parameter Value
S2125
Generate Planar Shape Representing Circulation According to Service Core Configuration and Input Parameter Values
S2126
Output Geometry
End

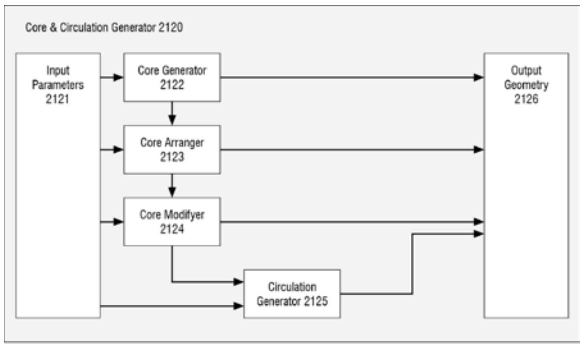


Figure 3.3 10 Core & Circulation Generation Process

Figure 3.3 11 Core & Circulation Generator 2120

Figure 3.3.12 is a flowchart illustrating the process of generating geometry representing the service core and circulation for a floor-plate design using the core & circulation generator 2120 shown in Figure 3.3.11. Figure 3.3.13 is a schematic diagram of the core & circulation generator 2120 shown in Figure 3.3.11.

Referring to Figures 3.3.12 and 3.3.13, in S2121, input geometry and parameter values are received as input parameters 2121. In S2122, a core generator 2122 generates at least one planar shape representing a basic service core configuration based on input parameter values. In S2123, a core arranger 2123 orients the service core geometry generated by the core generator 2122 based on an input core orientation parameter value. In S2124, a core modifier 2124 modifies the basic service core geometry generated by the core generator 2122 and oriented by the core arranger 2123 based on an input elevator service zone parameter value. In S2125, a circulation generator 2125 generates a planar shape representing circulation between the service core and rental spaces based on the service core configuration generated by the core generator 2122, oriented by the core arranger 2123, and modified by the core modifier 2124, and input parameter values. In S2126, the geometry output by the core modifier 2124 and the circulation generator 2125 is output to the floor-plate generator 2130, the structure generator 2140, and the partition generator 2150, and also as output geometry 2102, all shown in Figure 3.3.11.

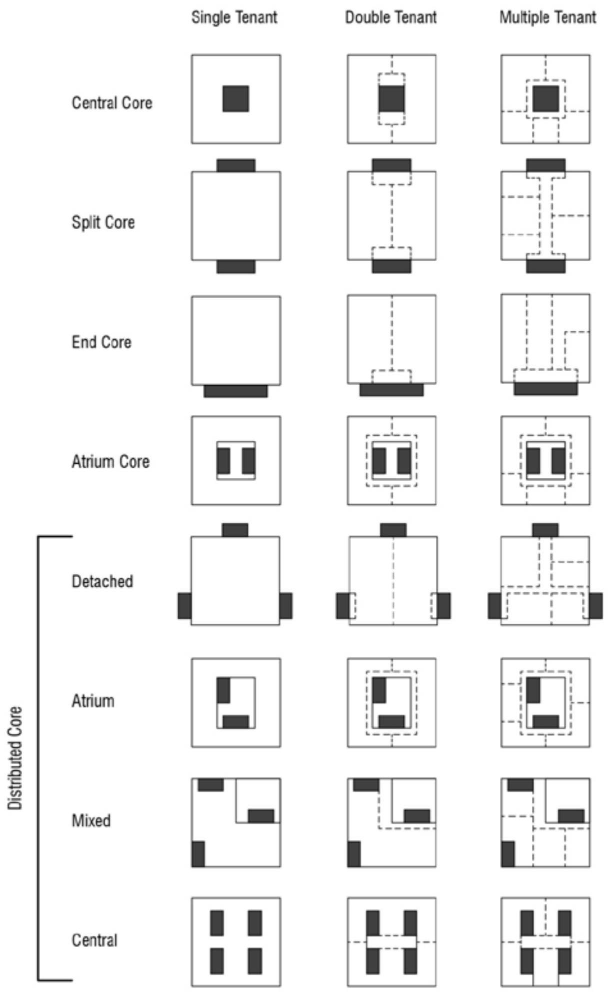


Figure 3.3 12 Core & Circulation Configurations

Figure 3.3.14 illustrates different service core and circulation layouts that may be generated by the core & circulation generator 2120 as a function of core type and number of separate rental units on the floor-plate.

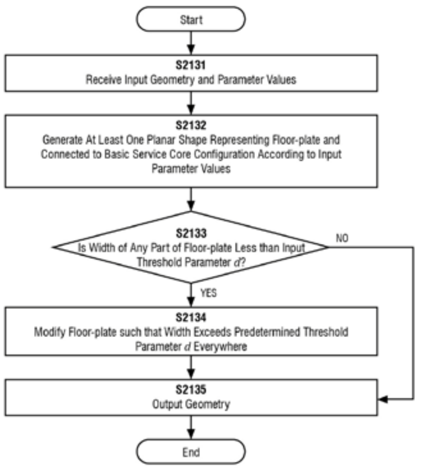

Start
S2131
Receive Input Geometry and Parameter Values
S2132
Generate At Least One Planar Shape Representing Floor-plate and Connected to Basic Service Core Configuration According to Input Parameter Values
S2133
Is Width of Any Part of Floor-plate Less than Input Threshold Parameter d?
NO
YES
S2134
Modify Floor-plate such that Width Exceeds Predetermined Threshold Parameter d Everywhere
S2135
Output Geometry
End

Figure 3.3 13 Floor-plate Generation Process

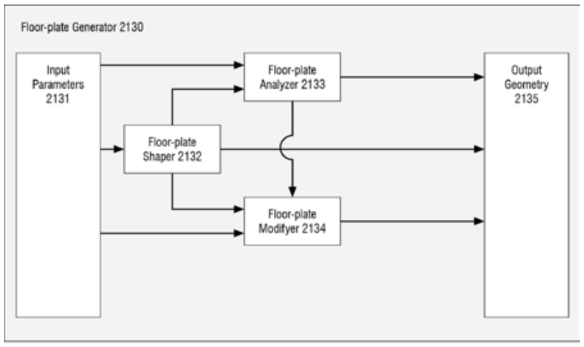

Floor-plate Generator 2130
Input Parameters 2131
Floor-plate Shaper 2132
Floor-plate Analyzer 2133
Floor-plate Modifyer 2134
Output Geometry 2135

Figure 3.3 14 Floor-plate Generator 2130

Figure 3.3.15 is a flowchart illustrating the process of generating geometry representing the floor-plate itself for a floor-plate design using the floor-plate generator 2130 shown in Figure 3.3.11. Figure 3.3.16 is a schematic diagram of the floor-plate generator 2130 shown in Figure 3.3.11.

Referring to Figures 3.3.15 and 3.3.16, in S2131, input geometry and parameter values are received as input parameters 2131. In S2132, a floor-plate shaper 2132 generates at least one planar shape representing the entire floor-plate and connected to the service core configuration output by the core & circulation generator 2120 shown in Figure 3.3.11 based on input parameter values. In S2133, a floor-plate analyzer 2133 determines whether the width of any part of the floor-plate generated by the floor-plate shaper 2132 is less than an input threshold value *d*. If the width of some part of the floor-plate is less than *d*, in S2134, the floor-plate is modified such that its width exceeds *d* everywhere. Then, in S2135, the planar shape representing the floor-plate is output to the structure generator 2140, the partition generator 2150, and as output geometry 2102, shown in Figure 3.3.11. If, on the other hand, the width is not less than *d* anywhere on the floor-plate, then S2134 of modifying the floor-plate is skipped and S2135 of outputting the floor-plate is carried out directly.

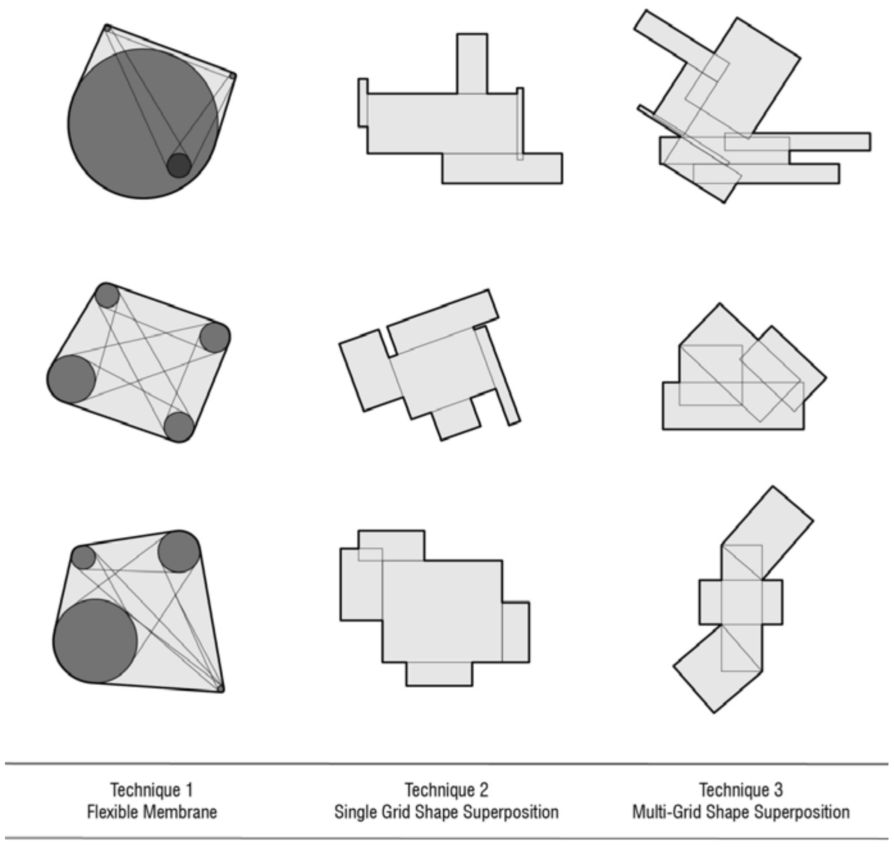

Technique 1
Flexible Membrane
Technique 2
Single Grid Shape Superposition
Technique 3
Multi-Grid Shape Superposition

Figure 3.3 15 Floor-plate Generation Techniques

Figure 3.3.17 illustrates three different techniques that may be employed by the floor-plate shaper 2132 to generate planar shapes representing floor-plates. Technique 1 uses a variable number of circles with variable radii and variable center coordinates to define the outer perimeter of a floor-plate. Technique 2 employs a fixed number of connected rectangles of variable dimensions to produce a floor-plate shape. Technique 3 superimposes a variable number of rectangles of variable dimensions and orientation to form a floor-plate.

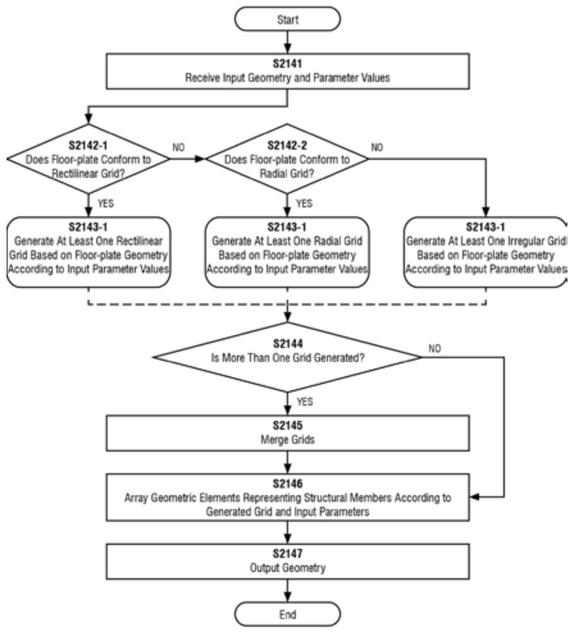

Start
S2141
Receive Input Geometry and Parameter Values
S2142-1
Does Floor-plate Conform to Rectilinear Grid?
NO
S2142-2
Does Floor-plate Conform to Radial Grid?
NO
YES
YES
S2143-1
Generate At Least One Rectilinear Grid Based on Floor-plate Geometry According to Input Parameter Values
S2143-1
Generate At Least One Radial Grid Based on Floor-plate Geometry According to Input Parameter Values
S2143-1
Generate At Least One Irregular Grid Based on Floor-plate Geometry According to Input Parameter Values
S2144
Is More Than One Grid Generated?
NO
YES
S2145
Merge Grids
S2146
Array Geometric Elements Representing Structural Members According to Generated Grid and Input Parameters
S2147
Output Geometry
End

Figure 3.3 16 Structure Generation Process

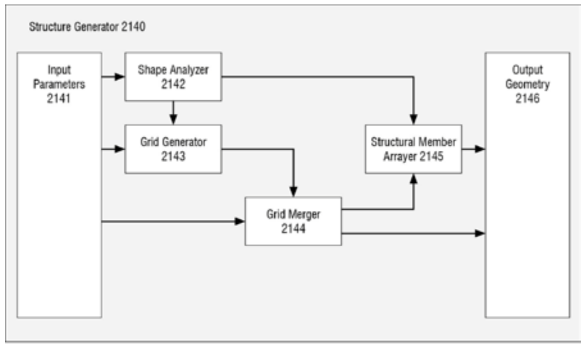


Figure 3.3 17 Structure Generator 2140

Figure 3.3.18 is a flowchart illustrating the process of generating a structural grid and geometry representing structural members for a floor-plate design using the structure generator 2140 shown in Figure 3.3.11. Figure 3.3.19 is a schematic diagram of the structure generator 2140 shown in Figure 3.3.11.

Referring to Figures 3.3.18 and 3.3.19, in S2141, input geometry and parameter values are received as input parameters 2141. In S2142-1, a shape analyzer 2142 determines whether the floor-plate conforms to a rectangular grid. If it does, in S2143-1, a grid generator 2143 generates at least one rectangular grid based on the floor-plate geometry and input parameter values. If the floor-plate does not conform to a rectangular grid, in S2142-2, the shape analyzer 2142 determines whether the floor-plate conforms to a radial grid. If it does, in S2143-2, the grid generator 2143 generates at least one radial grid based on the floor-plate geometry and input parameter values. If the floor-plate does not conform to a radial grid either, the in S2143-3, the grid generator 2143 generates at least one irregular grid based on the floor-plate geometry and input parameter values. After S2143-1, S2143-2, or S2143-3, in S2144, a grid merger 2145 determines whether more than one grid has been generated. If so, in S2145, the grid merger 2144 merges the generated grids according to a predetermined method. Then, in S2146, a structural member arrayer 2145 generates and arrays geometric elements representing structural members such as columns, based on the grid and input parameter values. If the grid merger 2144 determines that only one grid has been generated, then S2145 of grid merging is skipped and S2146 of arraying structural members is carried out directly. Finally, in S2147, geometry generated by the grid generator 2143, the grid merger 2144, and the structural member arrayer 2145 is output as output geometry 2146.

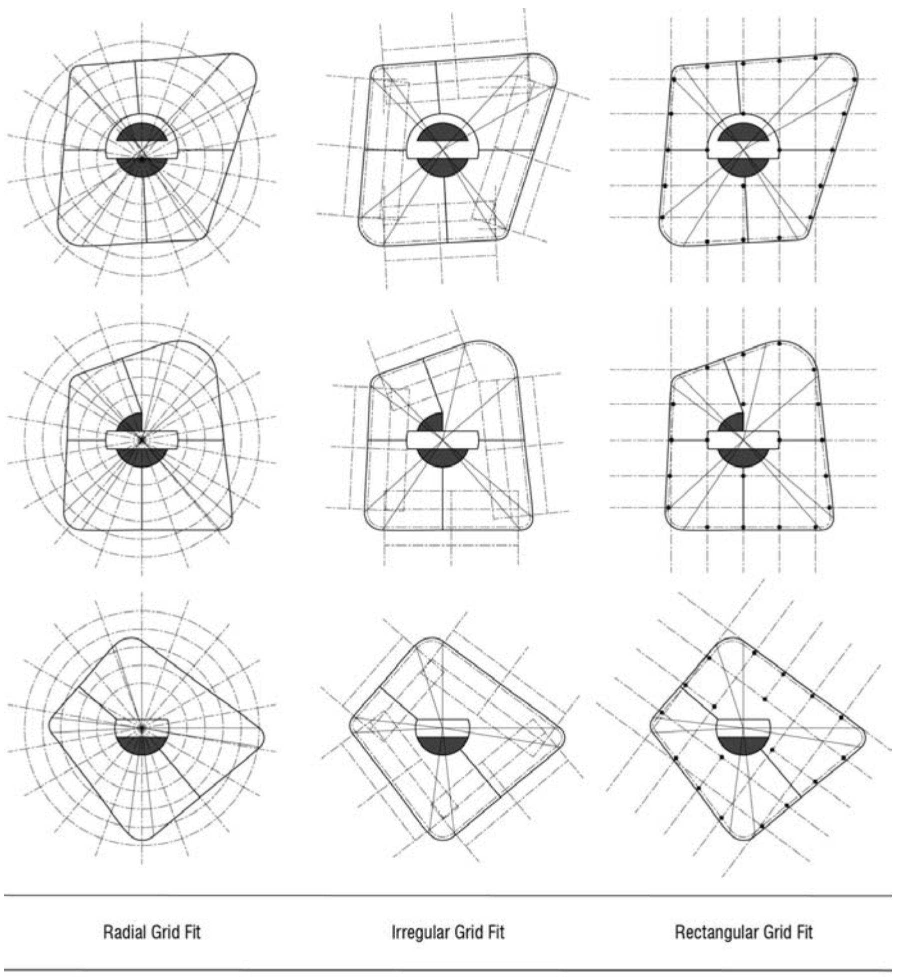


Figure 3.3 18 Structural Grid Fitting Options

Figure 3.3.20 illustrates rectangular, radial, and irregular grids fit to three different floor-plate shapes. In all three cases, the floor-plate conforms to a rectangular grid, so only the rectangular grid is generated, and columns are placed at locations on the grid around the perimeter of the floor-plate and surrounding the service core.

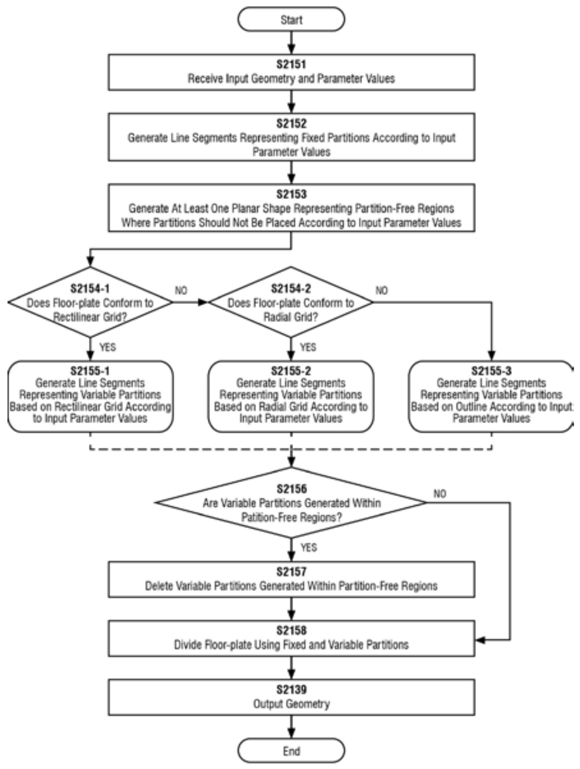

Start
S2151
Receive Input Geometry and Parameter Values
S2152
Generate Line Segments Representing Fixed Partitions According to Input Parameter Values
S2153
Generate At Least One Planar Shape Representing Partition-Free Regions Where Partitions Should Not Be Placed According to Input Parameter Values
S2154-1
Does Floor-plate Conform to Rectilinear Grid?
NO
S2154-2
Does Floor-plate Conform to Radial Grid?
NO
YES
YES
S2155-1
Generate Line Segments Representing Variable Partitions Based on Rectilinear Grid According to Input Parameter Values
S2155-2
Generate Line Segments Representing Variable Partitions Based on Radial Grid According to Input Parameter Values
S2155-3
Generate Line Segments Representing Variable Partitions Based on Outline According to Input Parameter Values
S2156
Are Variable Partitions Generated Within Patition-Free Regions?
NO
YES
S2157
Delete Variable Partitions Generated Within Partition-Free Regions
S2158
Divide Floor-plate Using Fixed and Variable Partitions
S2139
Output Geometry
End

Figure 3.3 19 Partition Generation Process

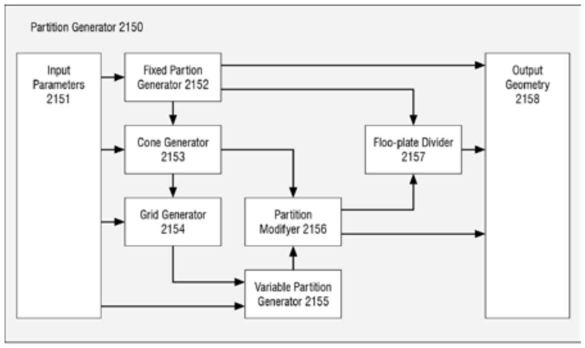


Figure 3.3 20 Partition Generator 2150

Figure 3.3.21 is a flowchart illustrating the process of partitioning a floor-plate design into a predetermined number of rental spaces using the partition generator 2150 shown in Figure 3.3.11. Figure 3.3.22 is a schematic diagram of the partition generator 2150 shown in Figure 3.3.11.

Referring to Figures 3.3.21 and 3.3.22, in S2151, input geometry and parameter values are received as input parameters 2151. In S2152, a fixed partition generator 2152 generates line segments representing fixed partitions according to input parameter values. Here, 'fixed' means that the placement of the partitions is fully determined by the core configuration and does not vary with respect to the same core configuration during the evolutionary process. In S2153, a cone generator 2153 generates at least one planar shape representing a 'partition-free' region where partitions should not be placed according to input parameter values. The partition-free regions may be shaped like cones. In S2154-1, a grid analyzer 2154 determines whether the floor-plate conforms to a rectangular grid. If so, in S2154-1, a variable partition generator 2155 generates line segments representing variable partitions based on the rectangular grid and input parameter values. If the floor-plate does not conform to a rectangular grid, in S2154-2, the grid analyzer 2154 determines whether the floor-plate conforms to a radial grid. If so, in S2154-2, the variable partition generator 2155 generates line segments representing variable partitions based on the radial grid and input parameter values. If the floor-plate does not conform to a radial grid either, then the variable partition generator 2155 generates line segments representing variable partitions based on the perimeter of the floor-plate and input parameter values. Once variable partitions have been generated in one of S2155-1, S2155-2 and S2155-3, a partition modifier 2156 determines whether any variable partitions are generated within the partition-free region generated by the cone generator 2153. If one or more variable partitions do enter the partition-free region, then in S2156, the partition modifier 2156 deletes those partitions. Next, in S2158, a floor-plate divider 2147 divides the planar surface representing the floor-plate along the line segments representing the fixed and remaining variable partitions to create separate planar surfaces corresponding to separate rental spaces. If no variable partitions pass through the partition-free region, then S2156 is skipped and S2158 is performed directly. Finally, in S2139, geometry output from the fixed partition generator 2152, the partition modifier 2156, and the floor-plate divider 2157 is output as output geometry 2158 to the display 120 of the user interface 100 and to the evaluation system 2200.

Figure 3.3.23 illustrates general floor-plate partitioning schemes as a function of elevator service zone and number of rental spaces on the floor, for a central service core layout. Fixed partitions are aligned with the hallway through the core (vertically on the page), all other partitions are variable partitions, and partition-free regions shown as grey cones. As shown, when the elevator service zone increases, a portion of the floor-plate allocated to the service core is regained as tenant space, reflecting the termination of elevator shafts not needed to service higher zones due to the use of destination control. Note also how the circulation between the core and the rental spaces changes with service zone and number of rental spaces to allow every tenant access. In this example, the partition-free regions are designed to minimize the need for circulation connecting the elevator lobby with the outside of the core, and to prevent awkward partitioning of regained core space.

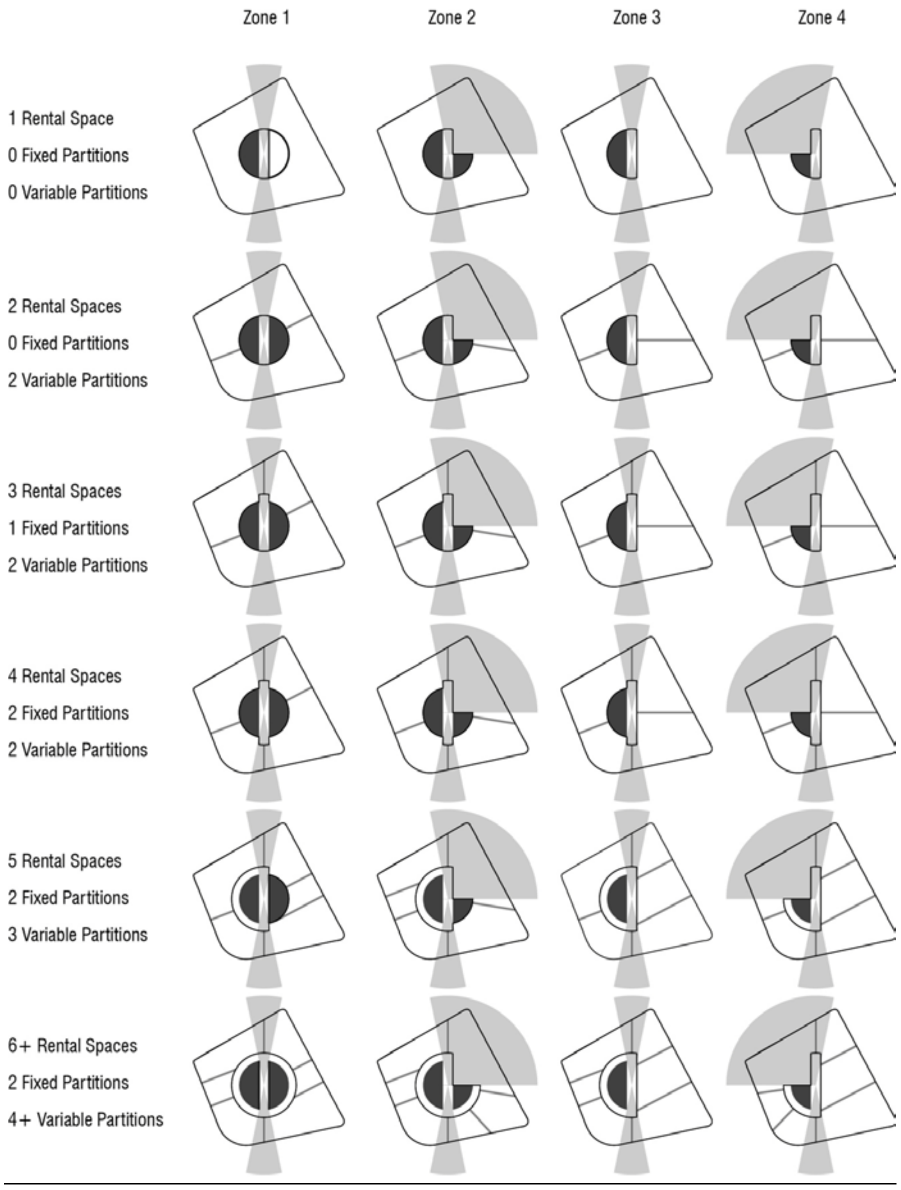


Figure 3.3 21 Floor-plate Partitioning Schemes

Evaluation System

Figure 3.3.24 is a flowchart illustrating the process of evaluating floor-plate designs using the evaluation system 2200 of the design developer-evaluator 2000 of the floor-plate sub-system 2 shown in Figure 3.3.3. Figure 3.3.25 is a schematic diagram of the floor-plate sub-system 2 shown in Figure 3.3.3, illustrating the general design of the evaluation system 2200.

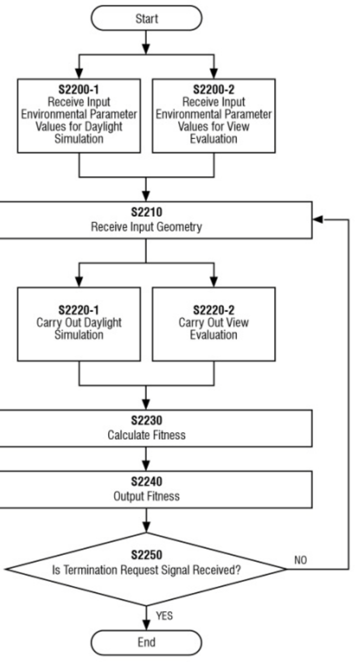

Start
S2200-1
Receive Input Environmental Parameter Values for Daylight Simulation
S2200-2
Receive Input Environmental Parameter Values for View Evaluation
S2210
Receive Input Geometry
S2220-1
Carry Out Daylight Simulation
S2220-2
Carry Out View Evaluation
S2230
Calculate Fitness
S2240
Output Fitness
S2250
Is Termination Request Signal Received?
NO
YES
End

Figure 3.3 22 Floor-plate Design Evaluation Process

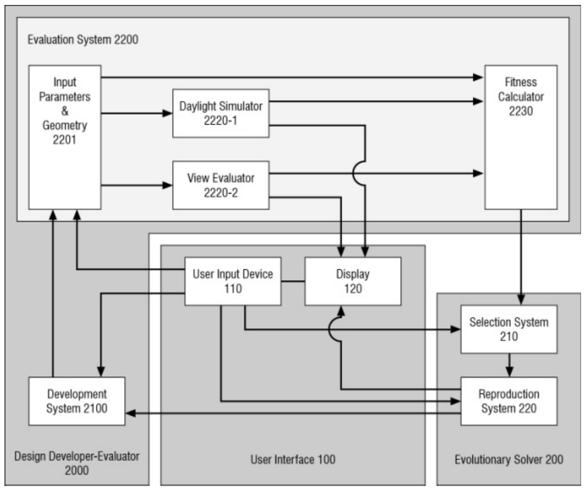


Figure 3.3 23 Floor-plate Sub-System 2 Showing General Design of Evaluation System 2200

Referring to Figures 3.3.6 and 3.3.7, in S2200-1, a daylight evaluator 2220-1 receives environmental parameter values input from the user input device 110 of the user interface 100. The parameters include weather data based on geographical location, sky conditions, and material properties of glazing assemblies for the skyscraper. In S2200-2, a view evaluator 2220-2 receives environmental parameter values input from the user input device 110 of the user interface 100. The parameters include angles specifying view corridors from the site, and values ranking the quality of view down each of the view corridors. The parameter values are used to generate a view ring as described in Chapter 3.3. In S2210, geometry and parameter values output from the development system 2100 are additionally received as input parameters and geometry 2201. In S2220-1, the daylight evaluator 2220-1 carries out a daylight simulation on the received geometry based on the received environmental parameter values and outputs the results to the display 120 of the user interface 100 and to a fitness calculator 2230. Here, the geometry used in the daylight simulation may include surfaces representing two adjacent floor-plates, all internal partitions, and a section of the building enclosure. In S2220-2, the view evaluator 2220-2 calculates the view value of each rental space and the overall view value of the floor-plate design as described in Chapter 3.3 and outputs the results to the display 120 of the user interface 100 and to the fitness calculator 2230. Steps S2220-1 and S2220-2 may be performed concurrently or sequentially. In S2230, the fitness calculator 2230 calculates a fitness value based on the outputs of the daylight evaluator 2220-1 and the view evaluator 2220-1. Here, an aggregate objective function (AOF) may be used to adjust the relative priorities of daylight and view performance in calculating fitness, as described in Chapter 3.3. In S2240, the calculated fitness value is output to the selection system 210 of the evolutionary solver 200. In S2250, the selection system 210 determines whether a termination request signal is received from the user input device 110 or the terminator 211 (refer to Figure 3.3.3). If the termination request signal is received, the evaluation process ends, and if the termination request signal is not received, the evaluation process is repeated from S2210 for new input geometry.

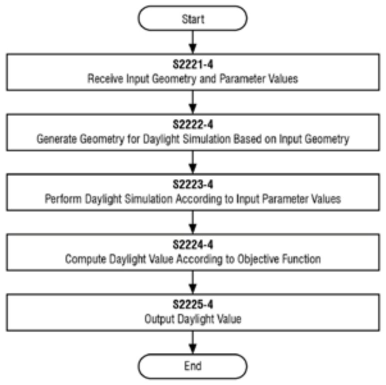

Start
S2221-4
Receive Input Geometry and Parameter Values
S2222-4
Generate Geometry for Daylight Simulation Based on Input Geometry
S2223-4
Perform Daylight Simulation According to Input Parameter Values
S2224-4
Compute Daylight Value According to Objective Function
S2225-4
Output Daylight Value
End

Figure 3.3 24 Daylight Evaluation Process

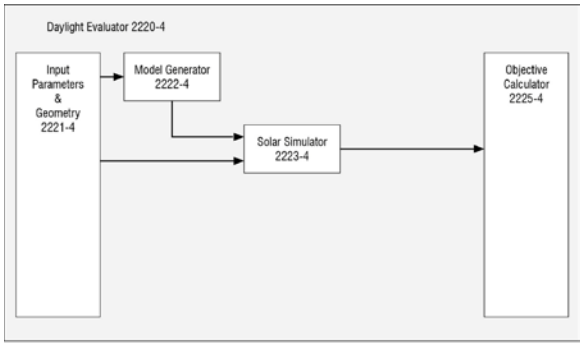

Daylight Evaluator 2220-4
Input Parameters & Geometry 2221-4
Model Generator 2222-4
Solar Simulator 2223-4
Objective Calculator 2225-4

Figure 3.3 25 Daylight Evaluator 2220-1

Figure 3.3.26 is a flowchart illustrating the daylight evaluation process using the daylight evaluator 2220-1 shown in Figure 3.3.25. Figure 3.3.27 is a schematic diagram of the daylight evaluator 2220-1 shown in Figure 3.3.25.

Referring to Figures 3.3.26 and 3.3.27, in S2221-1, input geometry and parameter values are received as input parameters and geometry 2221-1. In S2222-1, a model generator 2222-1 generates the geometry needed to perform a daylight simulation on the floor-plate design based on geometry output by the development system 2100 shown in Figure 3.3.11. As mentioned above, this may include surfaces representing two adjacent floor-plates, all internal partitions, and a section of the building enclosure. The floor-plates may be vertically separated by the floor-to-ceiling clearance distance, internal partitions may be generated by extruding the outputs of the core & circulation generator 2120 and the partition generator 2150 of the development system 2100 shown in Figure 3.3.11, and the building enclosure may be approximated by extruding the outer perimeter of one of the floor-plates. Next, in S2223-1, a daylight simulator 2223-1 performs a daylight simulation on the model generated by the model generator 2222-1 based on input parameter values, including the environmental parameter values received from the user input device 110. In S2224-4, an objective calculator 2225-1 calculates a single figure measure of daylight performance, for example, average daylight factor (DF) for all rental spaces on floor-plate, from the simulation results. In S2225-1, the objective calculator 2225-4 outputs the daylight performance value to the fitness calculator 2230 of the evaluation system 2200 shown in Figure 2.34.25.

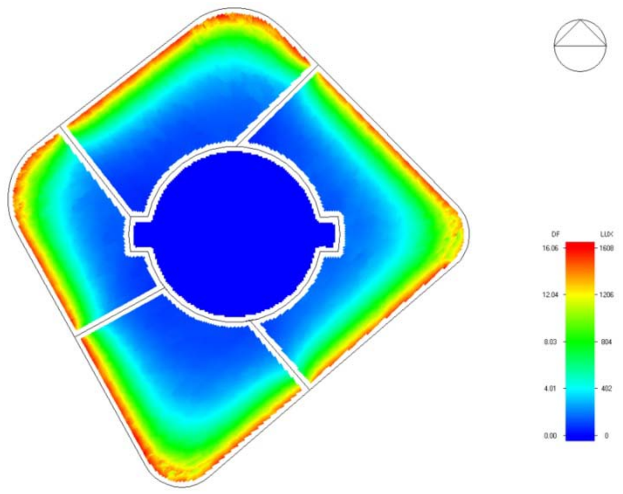


Figure 3.3 26 Visualization of Daylight Simulation Results

Figure 3.3.28 is a visualization of daylight simulation results for a typical floor-plate design. Daylight simulations have to be very course in precision in order to minimize delay in system operation. This is no significant compromise because simulation results are only used to establish the relative ranking of alternative floor-plate designs, not to provide precise and accurate measurements.

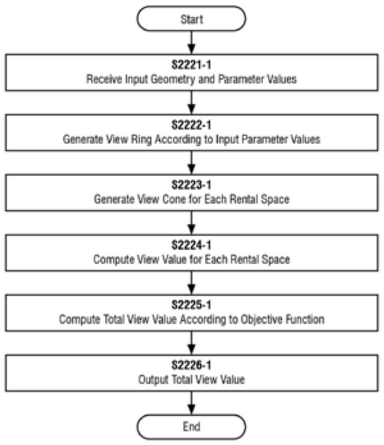

Start
S2221-1
Receive Input Geometry and Parameter Values
S2222-1
Generate View Ring According to Input Parameter Values
S2223-1
Generate View Cone for Each Rental Space
S2224-1
Compute View Value for Each Rental Space
S2225-1
Compute Total View Value According to Objective Function
S2226-1
Output Total View Value
End

Figure 3.3 27 View Performance Evaluation Process

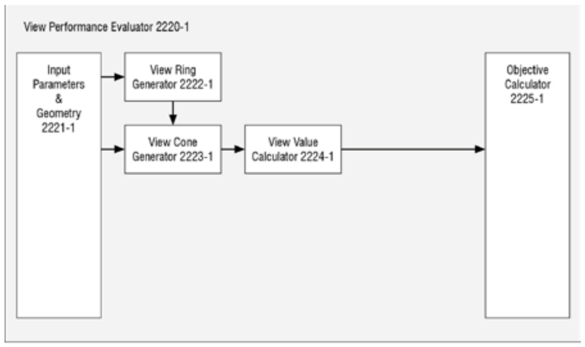

View Performance Evaluator 2220-1
Input Parameters & Geometry 2221-1
View Ring Generator 2222-1
View Cone Generator 2223-1
View Value Calculator 2224-1
Objective Calculator 2225-1

Figure 3.3 28 View Performance Evaluator 2220-2

Figure 3.3.29 is a flowchart illustrating the view evaluation process using the view evaluator 2220-2 shown in Figure 3.3.25. Figure 3.3.30 is a schematic diagram of the view evaluator 2220-2 shown in Figure 3.3.25.

Referring to Figures 3.3.29 and 3.3.30, in S2221-2, input geometry and parameter values are received as input parameters and geometry 2221-2. In S2222-2, a view ring generator 2222-1 generates a view ring quantitatively describing the quality of different view corridors available from the site. In S2223-2, a view cone generator 2223-2 generates view cones representing the total view range of each rental space as a function of facade frontage and geometric center of the space as described in Chapter 3.3. In S2224-2, a view value calculator 2224-2 calculates the area of the portion of the view ring generated by the view ring generator 2222-2 that overlaps with each view cone generated by the view cone generator 2223-2 as the view value for each rental space. In S2225-2, an objective calculator 2225-2 calculates a total view value for the floor-plate design as a weighted sum of the view values of individual rental spaces according to input parameter values. In S2226-2, the objective calculator 2225-2 outputs the total view value to the fitness calculator 2230 of the evaluation system 2200 shown in Figure 2.34.25.

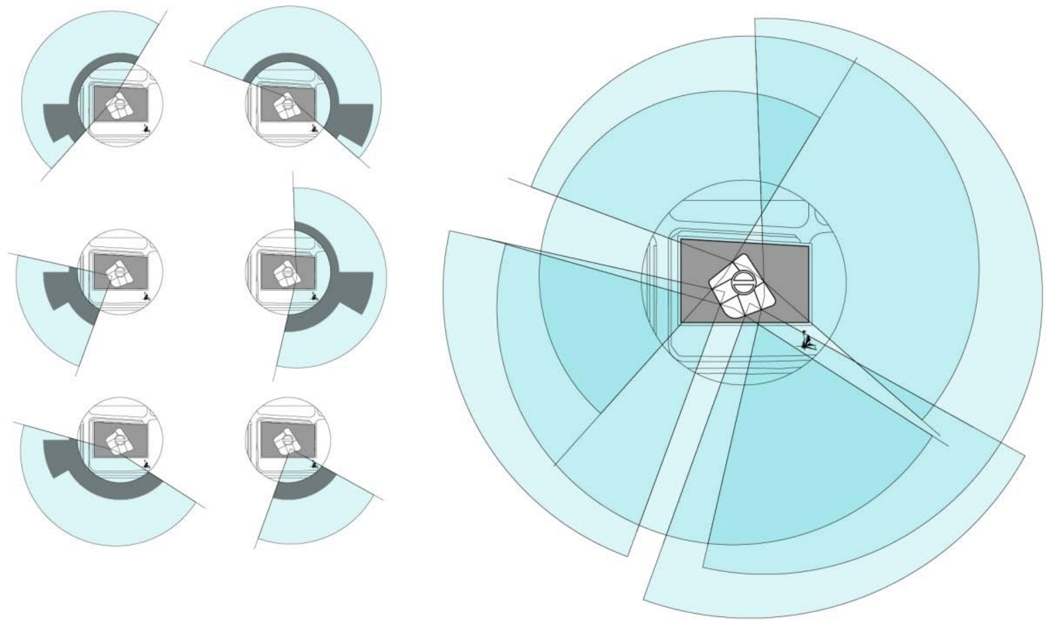


Figure 3.3 29 Visualization of View Evaluation Process

Figure 3.3.31 is a visualization of the view evaluation process for a typical floor-plate design. The six smaller objects on the right show the regions of the view right that overlap with the view cones of each of six rental spaces in the floor-plate design, and the larger object on the right shows the view cones of all six rental spaces together.

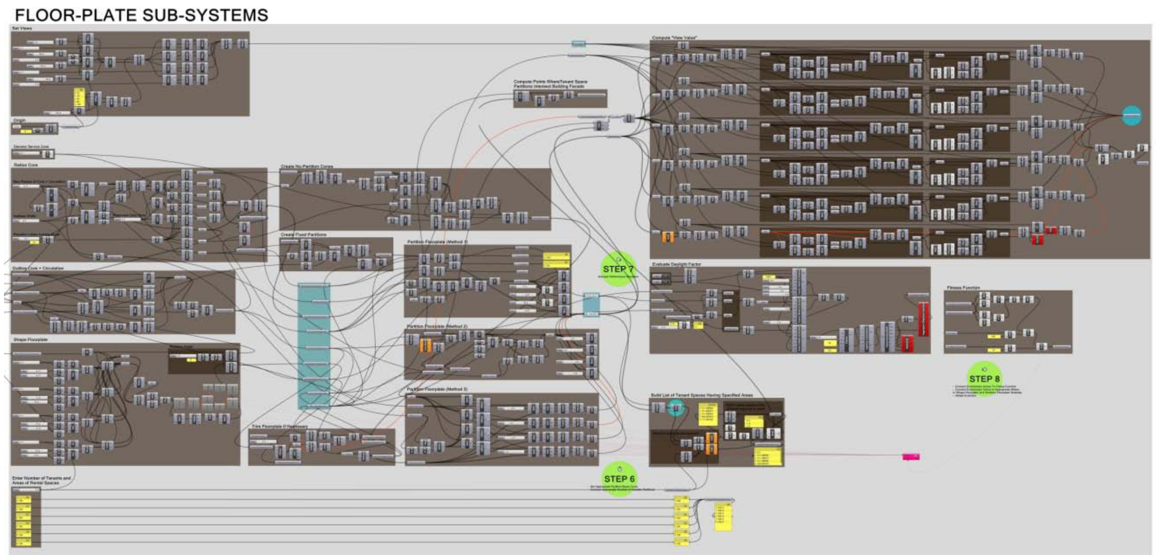
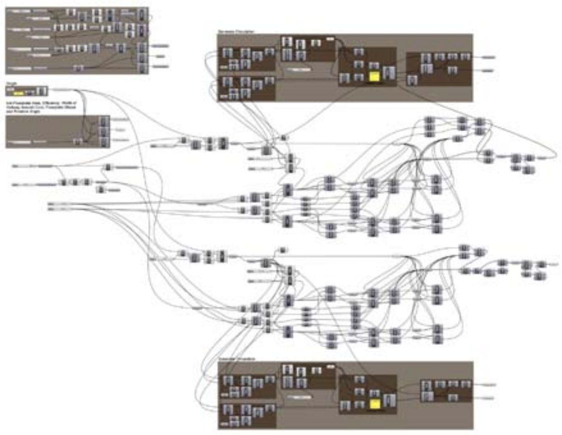


Figure 3.3 30 Grasshopper Implementation of Floor-plate Sub-System 2



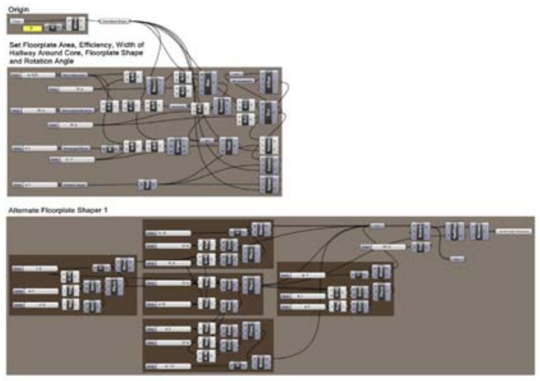

Origin
Set Floorplate Area, Efficiency, Width of Hallway Around Core, Floorplate Shape and Rotation Angle
Alternate Floorplate Shaper 1

Figure 3.3 31 Alternative Grasshopper Implementation of Core & Circulation Generator 2120

Figure 3.3 32 Alternative Grasshopper Implementation of Floor-plate Generator 2130

Figure 3.3.32 illustrates an implementation of the floor-plate sub-system 2 in the Grasshopper graphical scripting language, Figure 3.3.33 illustrates an alternative Grasshopper implementation of the core & circulation generator 2120, and Figure 3.3.34 illustrates an alternative grasshopper implementation of the floor-plate generator 2130.

3.3.3.4 Envelope Sub-System

Development System

Figure 3.3.35 is a flowchart illustrating the process of developing building envelope designs using the development system 3100 of the design developer-evaluator 3000 of the envelope sub-system 3 shown in Figure 3.3.3. Figure 3.3.36 is a schematic diagram of the envelope sub-system 3 shown in Figure 3.3.3, illustrating the general design of the development system 3100 of the design developer-evaluator 3000.

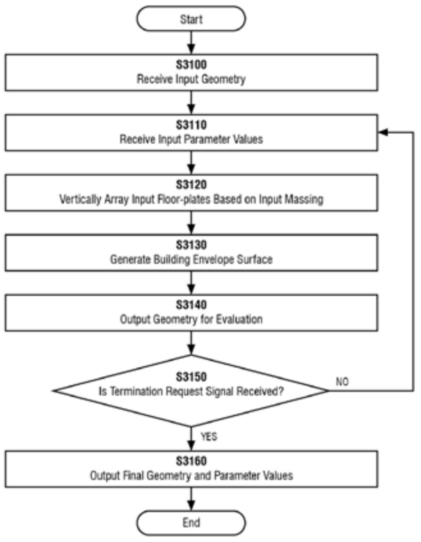

Start
S3100
Receive Input Geometry
S3110
Receive Input Parameter Values
S3120
Vertically Array Input Floor-plates Based on Input Massing
S3130
Generate Building Envelope Surface
S3140
Output Geometry for Evaluation
S3150
Is Termination Request Signal Received?
NO
YES
S3160
Output Final Geometry and Parameter Values
End

Figure 3.3 33 Building Envelope Development Process

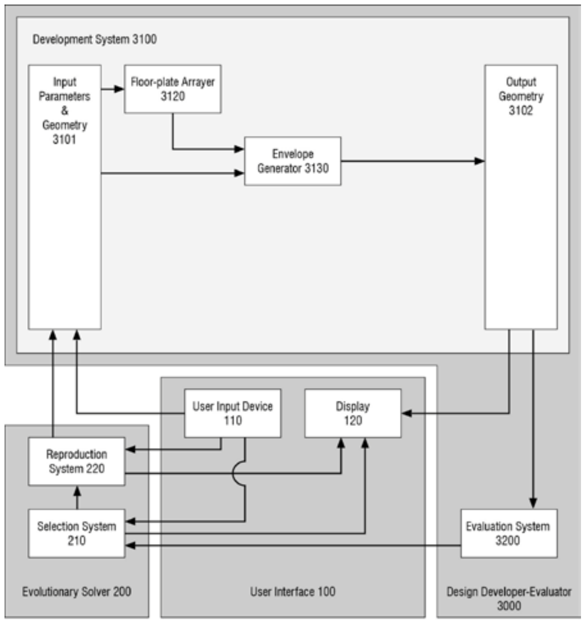
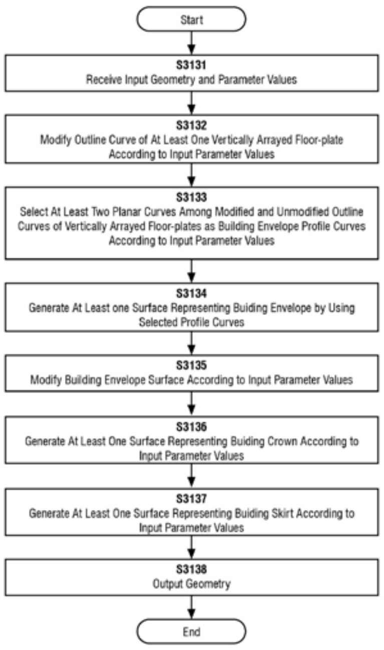


Figure 3.3 34 Envelope Sub-System 3 Showing Design of Development System 3100

Referring to Figures 3.3.35 and 3.3.36, in S3100, final geometry and parameter values output from the general massing sub-system 1 and the floor-plate sub-system 2 are received as input parameters 3101. In S3110, additional input parameter values are received from the user input device 110 of the user interface 100 and/or the reproduction system 220 of the evolutionary solver 200. In S3120, a floor-plate arrayer 3120 vertically arrays the input floor-plates based on the input massing model. In S3130, an envelope generator 3130 generates at least one surface enclosing the floor-plates arrayed by the floor-plate arrayer 3120 and representing the building envelope according to input parameter values. In S3140, the generated building envelope is output as output geometry 3102 to the display 120 of the user interface 100, and to the evaluation system 3200. In S3150, the selection system 210 of the evolutionary solver 200 determines whether or not a termination request signal is received from the user input device 110 or the terminator 211 (refer to Figure 3.3.3). If the termination request signal is received, in S3160, the output geometry 3102 is output to the development system 4100 of the skin sub-system 4. Meanwhile, if the termination request signal is not received, the process is repeated from S3110 for new input parameter values.


Start
S3131
Receive Input Geometry and Parameter Values
S3132
Modify Outline Curve of At Least One Vertically Arrayed Floor-plate According to Input Parameter Values
S3133
Select At Least Two Planar Curves Among Modified and Unmodified Outline Curves of Vertically Arrayed Floor-plates as Building Envelope Profile Curves According to Input Parameter Values
S3134
Generate At Least one Surface Representing Buiding Envelope by Using Selected Profile Curves
S3135
Modify Building Envelope Surface According to Input Parameter Values
S3136
Generate At Least One Surface Representing Buiding Crown According to Input Parameter Values
S3137
Generate At Least One Surface Representing Buiding Skirt According to Input Parameter Values
S3138
Output Geometry
End

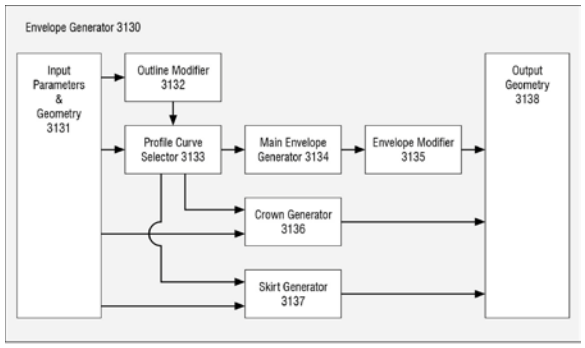


Figure 3.3 35 Building Envelope Generation Process

Figure 3.3 36 Envelope Generator 3130

Figure 3.3.37 is a flowchart illustrating the process of generating at least one surface representing the building envelope using the envelope generator 3130 shown in Figure 3.3.36. Figure 3.3.38 is a schematic diagram of the envelope generator 3130 shown in Figure 3.3.36.

Referring to Figures 3.3.37 and 3.3.38, in S3131, input geometry and parameter values are received as input parameters 3131. In S3132, an outline modifier 3132 modifies the outline curve of at least one of the vertically arrayed floor-plates output by the floor-plate arrayer 3120 shown in Figure 3.3.36. The nature of the modification is up to the discretion of the system designer and may include modest offsets and various other forms of articulation in the plane of the corresponding floor-plate. Here, the modification process should be sufficiently constrained so that the modified outline curve does not significantly cut into the floor-plate or extend too far out from the floor-plate, rendering it incompatible with the evolved floor-plate design. In the present implementation, a modest, variable outward offset is the chosen modification. Such modifications expand the evolutionary search space to include potentially useful adaptations such as tapered, notched and wavy facades that can have self-shading properties and distribute wind loads.

In S3133, a profile curve selector 3133 selects at least two planar curves from among the outline curves modified by the outline modifier 3132 and the remaining unmodified outline curves of the vertically arrayed floor-plates, to use as building envelope profile curves in response to input parameter values. In S3134, a main envelope generator 3134 generates at least one surface enclosing the vertically arrayed floor-plates and representing the main building envelope, based on the profile curves selected by the profile curve selector 3133. Here, the generation process may include some form of curve lofting, sweeping, or extruding. In S3135, an envelope modifier 3135 modifies the at least one surface generated by the main envelope generator 3134 according to input parameter values to produce a final building envelope. The form of modification may include various geometric modeling operations such as trimming, rebuilding, surface blending, etc.

In S3136, a crown generator 3136 generates at least one surface beginning from the uppermost profile curve of the main building envelope, representing the 'crown' of the skyscraper, in response to input parameter values. In S3137, a skirt generator 3137 generates at least one surface between the uppermost surface of the podium and the lowermost profile curve of the main building envelope representing the 'skirt' of the skyscraper. Finally, in S3138, the surface geometry generated/transformed by the envelope modifier 3135, the crown generator 3136, and the skirt generator 3137 is output as output geometry 3138 to the display 120 of the user interface 100 and to the evaluation system 3200.

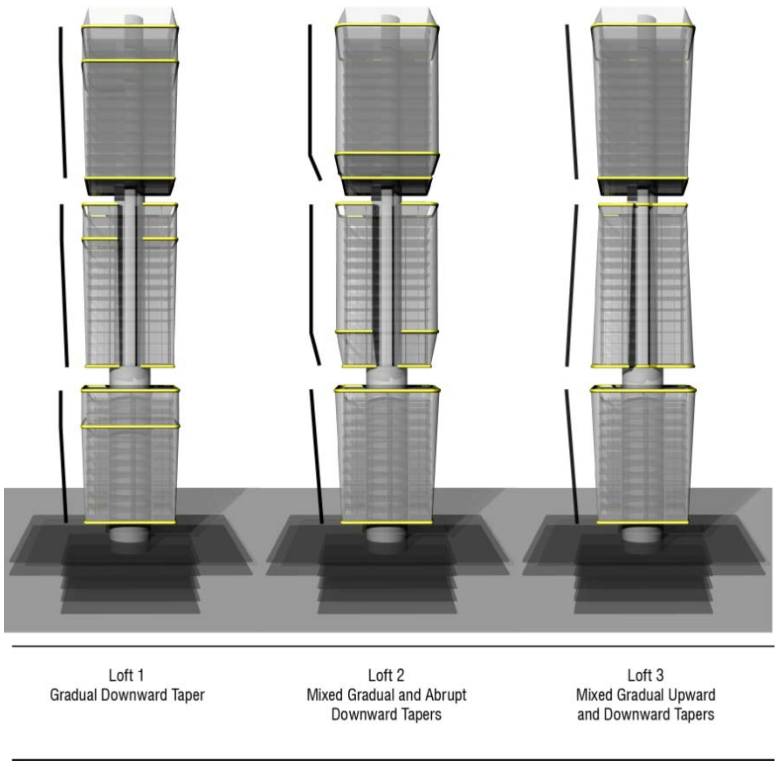

Loft 1
Gradual Downward Taper
Loft 2
Mixed Gradual and Abrupt Downward Tapers
Loft 3
Mixed Gradual Upward and Downward Tapers

Figure 3.3 37 Visualization of Building Envelope Generation Process

Figure 3.3.39 is a visualization of the building envelope generation process, showing three simple examples of main building envelope forms generated from different profile curves (highlighted in yellow). The evolved massing and floor-plate designs in the illustrated example result in a tower with a central service core and three blocks of floor-plates. The first variation on the building envelope has a gradual downward taper, the third mixes more abrupt downward tapers on the top two blocks with a gradual downward taper on the bottom block, and the fourth blends gradual upward and downward tapers to produce a subtle wave or undulation that connects the three blocks and slightly reduces the segmented appearance of the tower.

Evaluation System

Figure 3.3.40 is a flowchart illustrating the process of evaluating envelope designs using the evaluation system 3200 of the design developer-evaluator 3000 of the envelope sub-system 3 shown in Figure 3.3.3. Figure 3.3.41 is a schematic diagram of the envelope sub-system 3 shown in Figure 3.3.3, illustrating the general design of the evaluation system 3200.

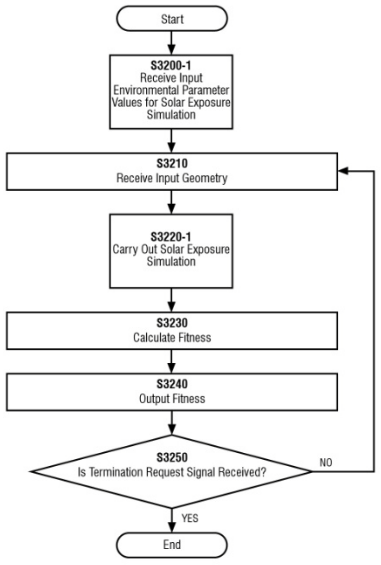

Start
S3200-1
Receive Input Environmental Parameter Values for Solar Exposure Simulation
S3210
Receive Input Geometry
S3220-1
Carry Out Solar Exposure Simulation
S3230
Calculate Fitness
S3240
Output Fitness
S3250
Is Termination Request Signal Received?
NO
YES
End

Figure 3.3 38 Envelope Design Evaluation Process

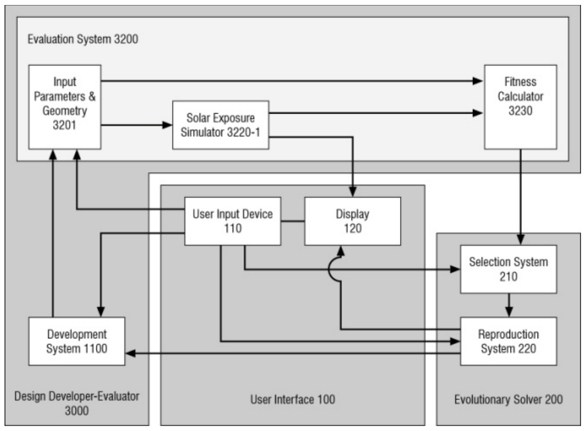


Figure 3.3 39 Envelope Sub-System 3 Showing General Design of Evaluation System 3200

Referring to Figures 3.3.40 and 3.3.41, in S3200-1, a facade insolation evaluator 3220-1 receives environmental parameter values input from the user input device 110 of the user interface 100. The parameters include weather data based on geographical location, sky conditions, and a daily time period during which the tower is occupied and requires climate control. In S3210, the surface geometry and parameter values output from the development system 3100 are additionally received as input parameters and geometry 3201. In S3220-1, the facade insolation evaluator 3220-1 carries out a facade insolation simulation to predict the amount of solar energy that would be directly incident on the glazed portion of the building envelope geometry developed by the development system 3100 during the specified period of occupation, in kWh/m^2, averaged over an entire year. In S3230, the simulation results are output to the display 120 of the user interface 100 and to a fitness calculator 3230. Since there are no other performance simulation modules in the evaluation system 3200, the fitness calculator 3230 equates fitness with the negative of the average facade insolation value output from the facade insolation evaluator 3220-1, so that lower insolation values correspond to less negative, i.e., greater, fitness values. In S3240, the fitness calculator 3230 outputs the calculated fitness value to the selection system 210 of the evolutionary solver 200. In S3250, the selection system 210 determines whether a termination request signal is received from the user input device 110 or the terminator 211 (refer to Figure 3.3.3). If the termination request signal is received, the evaluation process ends, and if the termination request signal is not received, the evaluation process is repeated from S3210 for new input geometry.

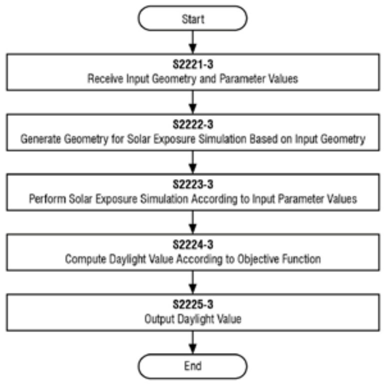

Start
S2221-3
Receive Input Geometry and Parameter Values
S2222-3
Generate Geometry for Solar Exposure Simulation Based on Input Geometry
S2223-3
Perform Solar Exposure Simulation According to Input Parameter Values
S2224-3
Compute Daylight Value According to Objective Function
S2225-3
Output Daylight Value
End

Figure 3.3 40 Facade Insolation Evaluation Process

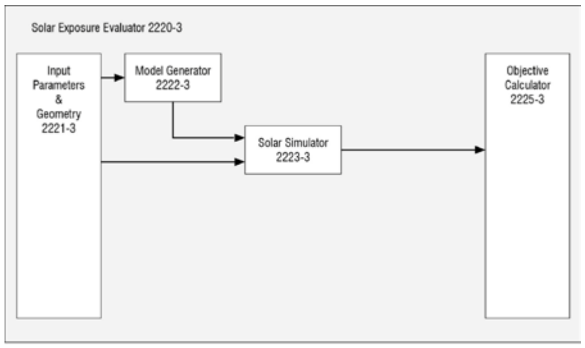

Solar Exposure Evaluator 2220-3
Input Parameters & Geometry 2221-3
Model Generator 2222-3
Solar Simulator 2223-3
Objective Calculator 2225-3

Figure 3.3 41 Facade Insolation Evaluator 3220-1

Figure 3.3.42 is a flowchart illustrating the facade insolation evaluation process using the facade insolation evaluator 3220-1 shown in Figure 3.3.42. Figure 3.3.43 is a schematic diagram of the facade insolation evaluator 3220-1 shown in Figure 3.3.41.

Referring to Figures 3.3.42 and 3.3.43, in S3221-1, input geometry and parameter values are received as input parameters and geometry 3221-1. In S3222-1, a model generator 3222-1 generates surface geometry representing the glazed portion of the building envelope developed by the development system 3100 an converts the geometry to mesh form if not already in mesh form. In S3223-1, a solar simulator 3223-1 carries out the facade insolation simulation based on the input parameter values described above. In the Grasshopper implementation disclosed herein, the solar simulator 3223-1 is implemented using Autodesk's Ecotect software application together with [uto]'s Gecko plug-in for Grasshopper, which allows remote control of the Ecotect simulation kernel from the Grasshopper user interface. In S3224-1, an objective calculator 3224-1 calculates the desired quantity - direct insolation during the specified period of occupation, in kWh/m^2, averaged over an entire year - from the simulation results output by the solar simulator 3223-1. Here, the decision to take the yearly average was made based on the climate of Shenzhen, the site of the Hanking competition. It was found that cooling during the daytime is typically required year-round in Shenzhen. Therefore, to help minimize cooling loads, it is desirable to minimize direct insolation on the glazed building facade year-round. In S3225-1, the objective calculator 3224-1 outputs the calculated insolation value to the fitness calculator 3230 shown in Figure 3.3.41.

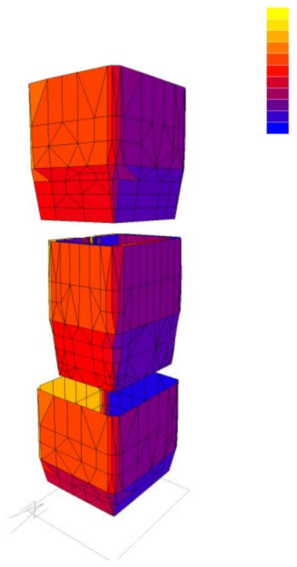


Figure 3.3 42 Visualization of Facade Insolation Simulation Results

Figure 3.3.44 is an image of facade insolation simulation results generated by the facade insolation evaluator 3220-1 and output to the display 120 of the user interface 100. As with the daylight simulations carried out in the evaluation system 2200 of the floor-plate sub-system 2 described above, it is important for the facade insolation simulation be very course in precision in order to minimize delay in system operation. This is no significant compromise because simulation results are only used to establish the relative ranking of alternative envelope designs, not to provide precise and accurate measurements.

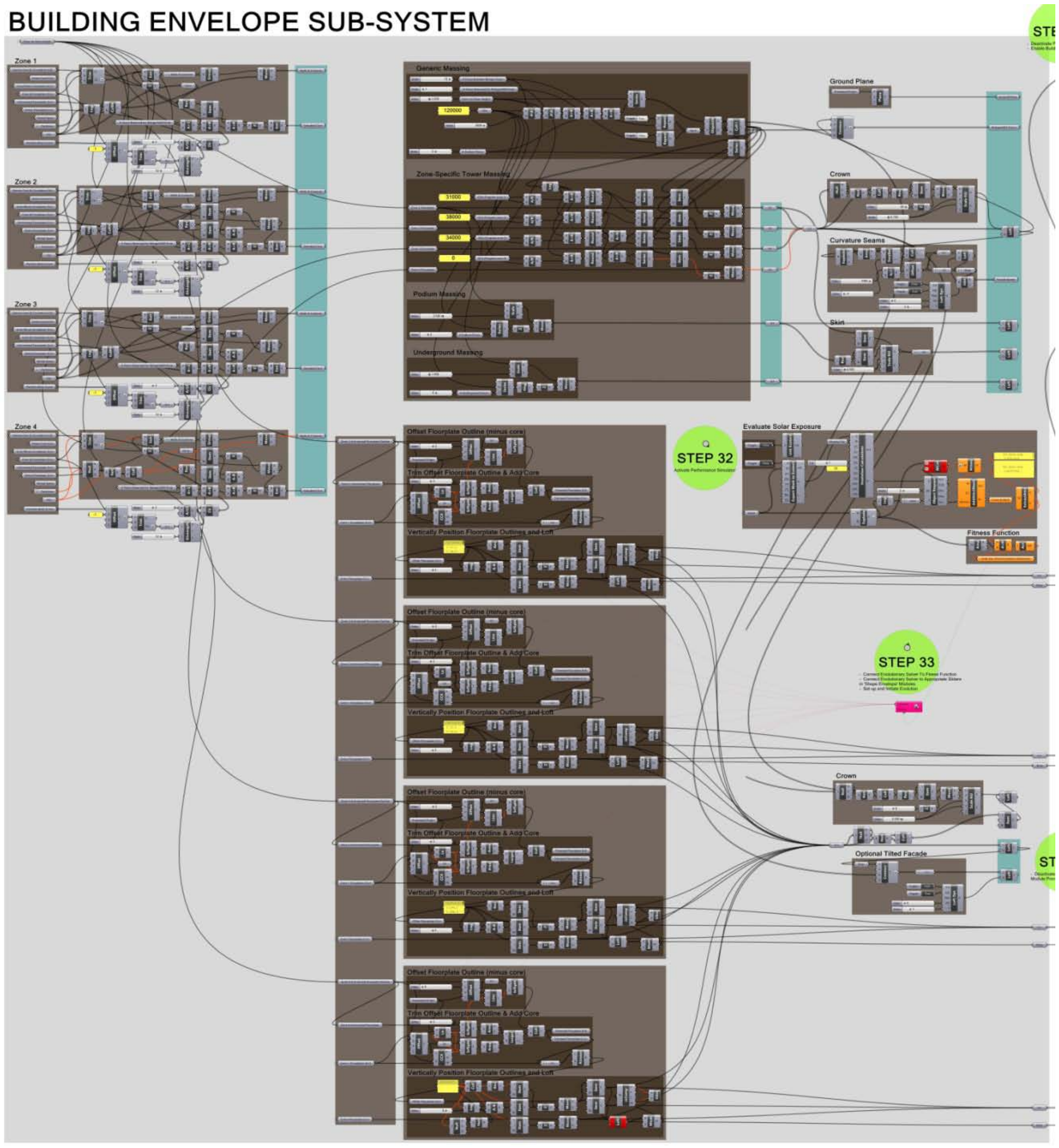


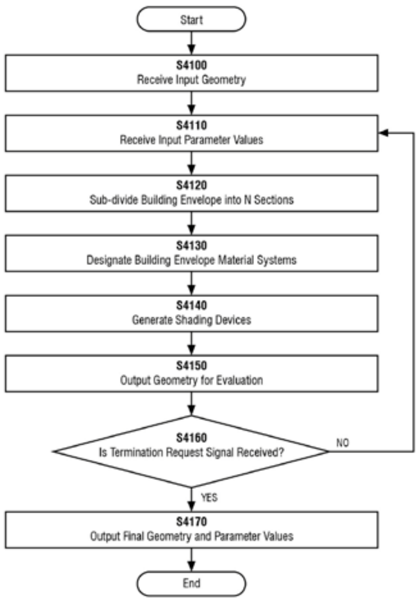
Figure 3.3 43 Grasshopper Implementation of Building Envelope Sub-System 3

Figure 3.3.45 illustrates an implementation of the envelope sub-system 2 in the Grasshopper graphical scripting language.

3.3.3.5 Skin Sub-System

Development System

Figure 3.3.46 is a flowchart illustrating the process of developing building envelope designs using the development system 4100 of the design developer-evaluator 4000 of the skin sub-system 4 shown in Figure 3.3.3. Figure 3.3.47 is a schematic diagram of the skin sub-system 4 shown in Figure 3.3.3, illustrating the general design of the development system 4100 of the design developer-evaluator 4000.


Start
S4100
Receive Input Geometry
S4110
Receive Input Parameter Values
S4120
Sub-divide Building Envelope into N Sections
S4130
Designate Building Envelope Material Systems
S4140
Generate Shading Devices
S4150
Output Geometry for Evaluation
S4160
Is Termination Request Signal Received?
NO
YES
S4170
Output Final Geometry and Parameter Values
End

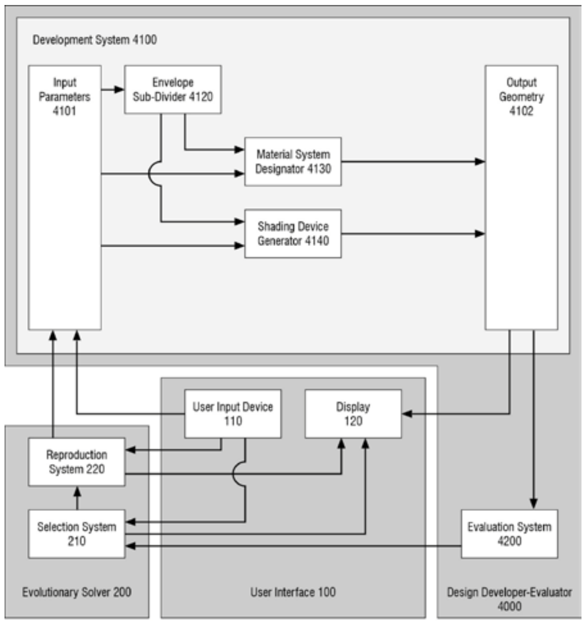


Figure 3.3 44 Building Skin Development Process

Figure 3.3 45 Building Skin Sub-System 4 Showing General Design of Development System 4100

Referring to Figures 3.3.46 and 3.3.47, in S4100, final geometry and parameter values output from the general massing sub-system 1, the floor-plate sub-system 2 and the envelope sub-system 3 are received as input parameters 4101. In S4110, additional input parameter values are received from the user input device 110 of the user interface 100 and/or the reproduction system 220 of the evolutionary solver 200. In S4120, an envelope sub-divider 4120 sub-divides the building envelope geometry received from the development system 3100 of the envelope sub-system 3 into *N* sections based on input parameter values. In S4130, a material system designator 4130 assigns predetermined material systems to each of the *N* envelope sections output from the envelope sub-divider 4120. In S4140, a shading device generator 4140 generates shading devices of variable type, dimensions and configuration, separately for each of the *N* envelope sections. In S4150, the *N* envelope sections with material systems designated by the material system designator 4140 and the shading devices generated by the shading device generator 4140 are output as output geometry 4102 to the display 120 of the user interface 100 and to the evaluation system 4200.

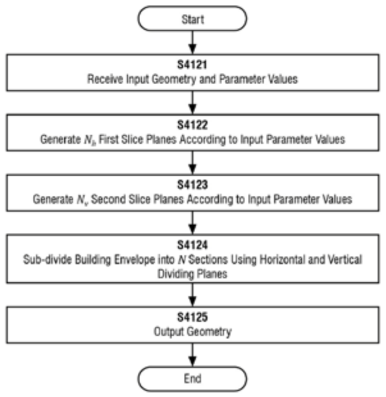

Start
S4121
Receive Input Geometry and Parameter Values
S4122
Generate N_h First Slice Planes According to Input Parameter Values
S4123
Generate N_v Second Slice Planes According to Input Parameter Values
S4124
Sub-divide Building Envelope into *N* Sections Using Horizontal and Vertical Dividing Planes
S4125
Output Geometry
End

Figure 3.3 46 Building Envelope Sub-Division Process

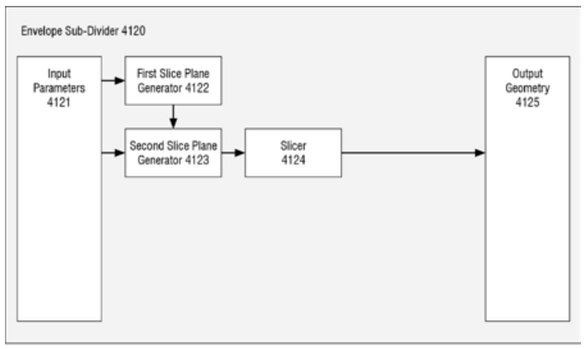

Envelope Sub-Divider 4120
Input Parameters 4121
First Slice Plane Generator 4122
Second Slice Plane Generator 4123
Slicer 4124
Output Geometry 4125

Figure 3.3 47 Envelope Sub-Divider 4120

Figure 3.3.48 is a flowchart illustrating the process of sub-dividing the building envelope into *N* sections using the envelope sub-divider 4120 shown in Figure 3.3.47. Figure 3.3.49 is a schematic diagram of the envelope sub-divider 4120 shown in Figure 3.3.47.

Referring to Figures 3.3.47 and 3.3.48, in S4121, input geometry and parameter values are received as input parameters 4121. In S4142, a first slice plane generator 4122 generates N_h horizontally oriented planar surfaces according to input parameter values. In S4142, a second slice plane generator 4123 generates N_v vertically oriented planar surfaces according to input parameter values. In S4124, a slicer 4124 sub-divides the building envelope into *N* sections using the horizontal N_h horizontally oriented planar surfaces and the N_v vertically oriented planar surfaces as slicing planes. In S4125, the resulting geometry consisting of *N* sections of the building envelope is output as output geometry 4125.

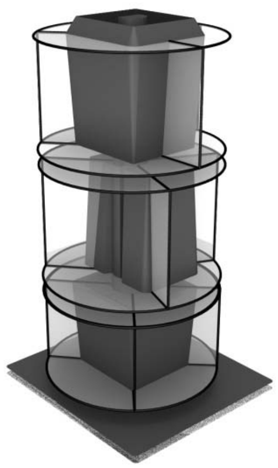
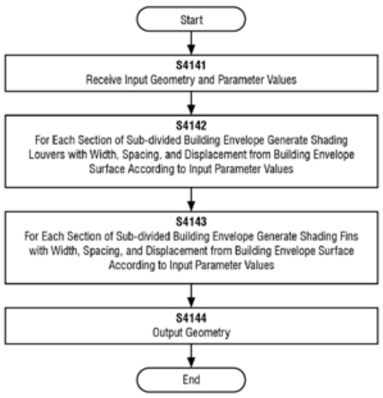


Figure 3.3 48 Visualization of Sub-Division of Building Envelope Using Slice Planes

Figure 3.3.50 is a visualization of how the building envelope is divided into *N* sections using a variable number of horizontal and vertical slice planes of variable height and rotation angle, respectively, according to input parameter values.


Start
S4141
Receive Input Geometry and Parameter Values
S4142
For Each Section of Sub-divided Building Envelope Generate Shading Louvers with Width, Spacing, and Displacement from Building Envelope Surface According to Input Parameter Values
S4143
For Each Section of Sub-divided Building Envelope Generate Shading Fins with Width, Spacing, and Displacement from Building Envelope Surface According to Input Parameter Values
S4144
Output Geometry
End

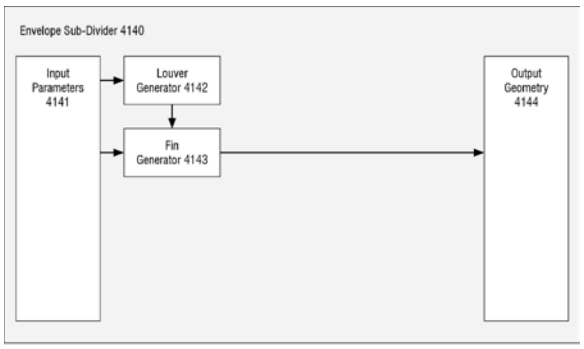


Figure 3.3 49 Shading Device Generation Process

Figure 3.3 50 Shading Device Generator 4140

Figure 3.3.51 is a flowchart illustrating the process of shading devices for each of the *N* sections of the subdivided building envelope using the shading device generator 4140 shown in Figure 3.3.47. Figure 3.3.52 is a schematic diagram of the shading device generator 4140 shown in Figure 3.3.47.

Referring to Figures 3.3.51 and 3.3.52, in S4141, input geometry and parameter values are received as input parameters 4141. In S4142, a louver generator 4142 generates horizontal shading louvers with width, spacing, and displacement from the building envelope surface specified by input parameter values, for each of the *N* sections of the sub-divided building envelope. In S4143, a fin generator 4142 generates vertical shading fins with width, spacing, and displacement from the building envelope surface specified by input parameter values, for each of the *N* sections of the sub-divided building envelope. In S4144, the building envelope geometry together with the generated shading louvers and fins are output as output geometry 4144 to the display 120 of the user interface 100 and to the evaluation system 4200.

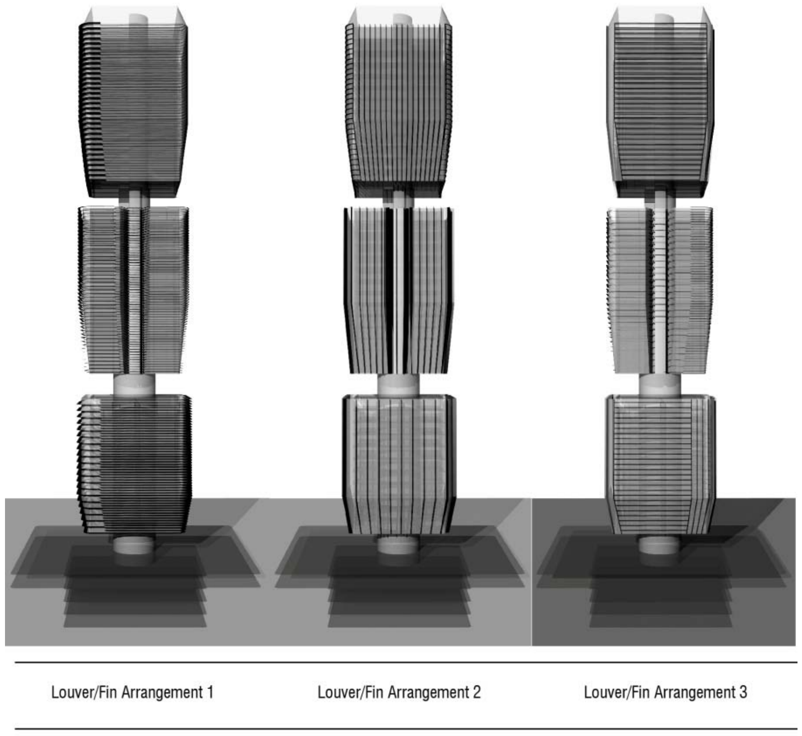


Figure 3.3 51 Visualization of Different Louver/Fin Arrangements

Figure 3.3.53 is a visualization of three different shading louver/fin arrangements generated by the shading device generator 4140 described above.

Evaluation System

Figure 3.3.54 is a flowchart illustrating the process of evaluating building skin designs using the evaluation system 4200 of the design developer-evaluator 4000 of the skin sub-system 4 shown in Figure 3.3.3. Figure 3.3.55 is a schematic diagram of the skin sub-system 4 shown in Figure 3.3.3, illustrating the general design of the evaluation system 4200.

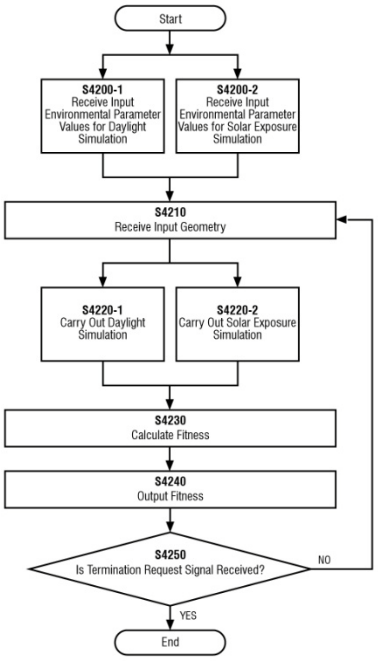

Start
S4200-1
Receive Input Environmental Parameter Values for Daylight Simulation
S4200-2
Receive Input Environmental Parameter Values for Solar Exposure Simulation
S4210
Receive Input Geometry
S4220-1
Carry Out Daylight Simulation
S4220-2
Carry Out Solar Exposure Simulation
S4230
Calculate Fitness
S4240
Output Fitness
S4250
Is Termination Request Signal Received?
NO
YES
End

Figure 3.3 52 Skin Design Evaluation Process

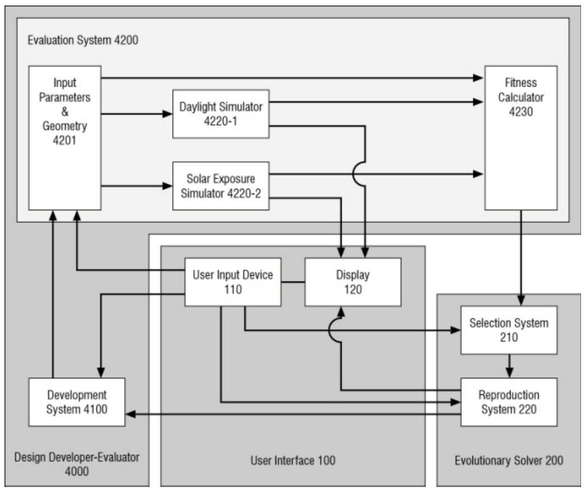


Figure 3.3 53 Skin Sub-System 4 Showing General Design of Evaluation System 4200

Referring to Figures 3.3.54 and 3.3.55, in S4200-1, a daylight evaluator 4220-1 receives environmental parameter values input from the user input device 110 of the user interface 100. The parameters include weather data based on geographical location, sky conditions, and material properties of glazing assemblies for the skyscraper. In S4200-2, a facade insolation evaluator 4220-2 receives environmental parameter values input from the user input device 110 of the user interface 100. These parameters include weather data based on geographical location, sky conditions, and a daily time period during which the tower is occupied and requires climate control. In S4210, the surface geometry and parameter values output from the development system 4100 are additionally received as input parameters and geometry 4201.

In S4220-1, the daylight evaluator 4220-1 carries out a daylight evaluation on the input geometry and outputs the results to the display 120 of the user interface 100 and to a fitness calculator 4230. In S4220-2, the facade insolation evaluator 4220-2 carries out a facade insolation simulation and outputs the results to the display 120 of the user interface 100 and to the fitness calculator 4230. In S4230, the fitness calculator 4230 calculates a fitness value for the skin design based on the simulation results received from the daylight evaluator 4220-1 and the facade insolation evaluator 4220-2. Here, maximizing daylight and minimizing solar exposure are conflicting objectives, and thus multi-objective optimization techniques are required. In the present design, the fitness calculator simply uses an aggregate objective function (AOE) of the form described in Chapter 3.1. In S4240, the fitness calculator 4230 outputs the calculated fitness value to the selection system 210 of the evolutionary solver 200. In S4250, the selection system 210 determines whether a termination request signal is received from the user input device 110 of the user interface 100 or from its internal terminator 211 (refer to Figure 3.3.3). If the termination request signal is received, the process ends and the final output geometry and parameters of the development systems of sub-systems 1 through 4 constitute a fully evolved skyscraper design. If the termination request signal is not received, then the process repeats from S4210 for new input geometry supplied by the development system 4100.

The daylight evaluator 4220-1 has the same general design and functionality as the daylight evaluator 2220-1 described above with reference to Figures 3.3.26, 3.3.27 and 3.3.28, and the facade insolation evaluator 4220-2 has the same general design and functionality as the facade insolation evaluator 3220-1 described above with reference to Figures 3.3.42, 3.3.43, and 3.3.44.

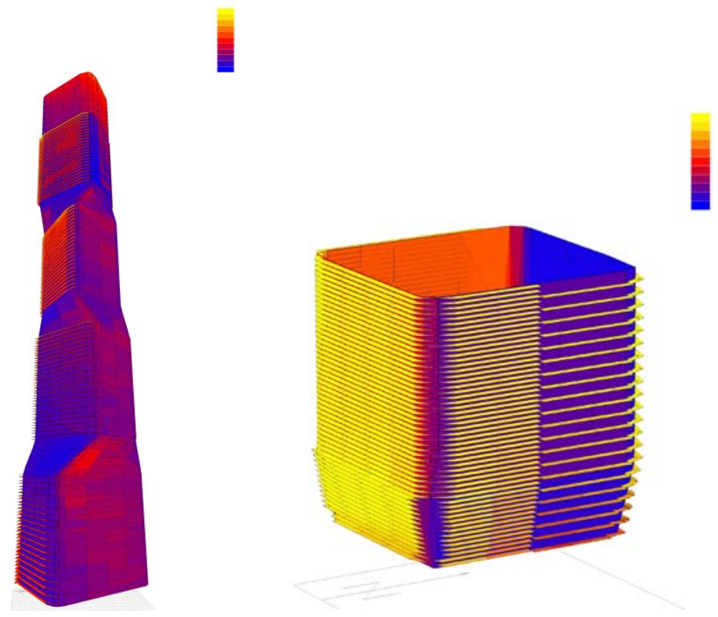


Figure 3.3 54 Visualization of Facade Insolation Simulation Results

Figure 3.3.56 is a visualization of facade insolation simulation results for a building skin model including a dense array of shading louvers.

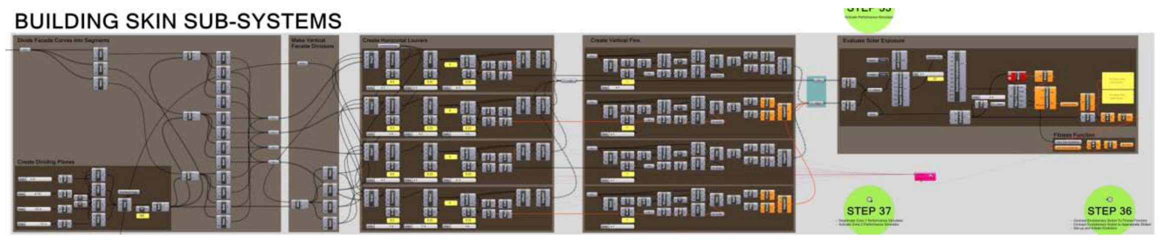


Figure 3.3 55 Grasshopper Implementation of Skin Sub-System 4

Figure 3.3.57 illustrates an implementation of the skin sub-system 4 in the Grasshopper graphical scripting language.

3.3.3.6 Overall System

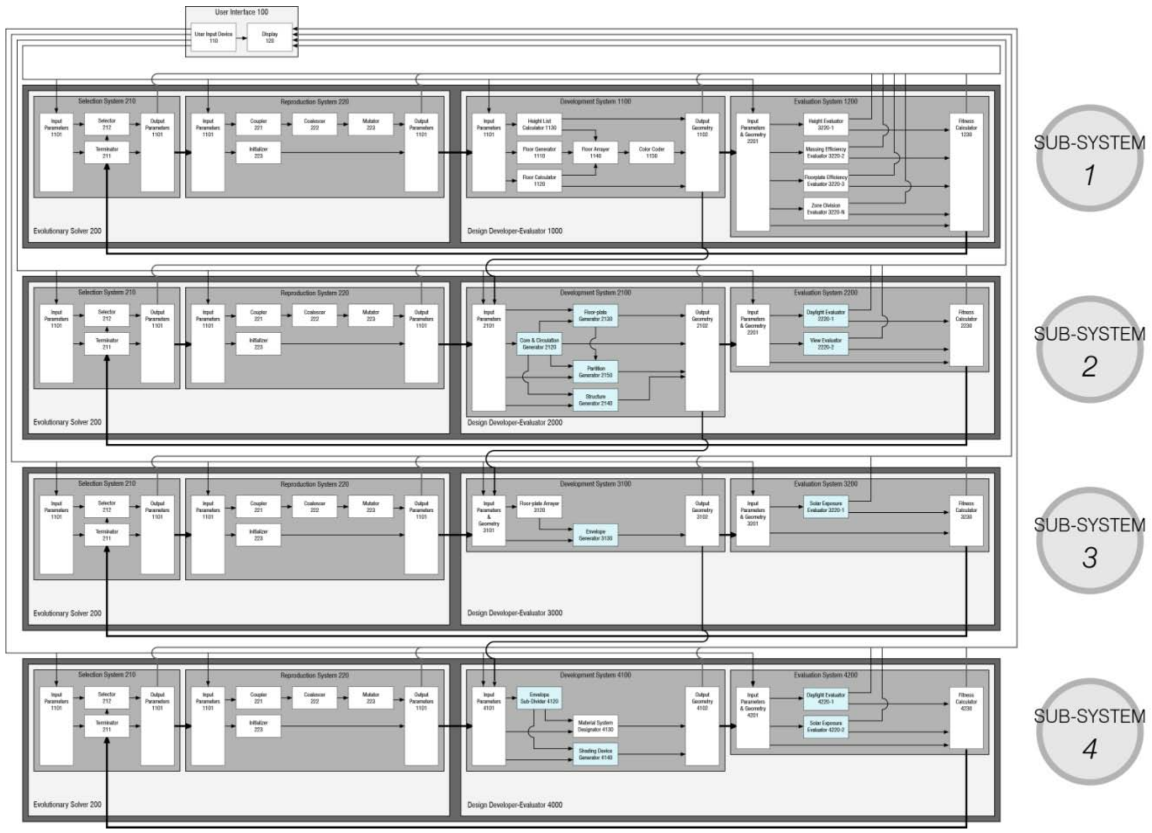

User Interface 100
Evolutionary Solver 200
Design Developer-Evaluator 1000
Design Developer-Evaluator 2000
Design Developer-Evaluator 3000
Design Developer-Evaluator 4000
SUB-SYSTEM 1
SUB-SYSTEM 2
SUB-SYSTEM 3
SUB-SYSTEM 4

Figure 3.3 56 Skyscraper MSEDS Customized for Hanking Competition

Figure 3.3.58 is a schematic diagram illustrating the overall design of a skyscraper MSEDS based on the skyscraper MSEDS architecture shown in Figure 3.3.3 and customized to evolve designs for the Hanking competition. The structures and functions of the components of each sub-system have been described in the preceding sections.

Both the skyscraper MSEDS shown in Figure 3.3.58 and the general architecture shown in Figure 3.3.3 are expansible, modular, and can be reconfigured to address the demands of different skyscraper design problems. Modifications can be made on two levels: the module level and the sub-system level.

- Module-level modifications involve altering or replacing one or more of the component modules of the development and evaluation systems colored blue in Figure 3.3.3 (architecture) and Figure 3.3.58 (working design). Such modifications change the way the modified module operates but do not affect how it interoperates with other modules and functions overall within the system. Examples such as alternative core and circulation generators and alternative floor-plate generators have already been given in the preceding description.

- Sub-system-level modifications are major modifications to the system design shown in Figure 3.3.58, that alter the way modules work together and therefore alter the general design of the MSEDS on the development system or evaluation system level. Such modification include the addition and removal of modules. For example, the envelope sub-divider 4120 might be eliminated from the development system 4100 of the skin sub-system 4, so that the material system designator 4130 and the shading device generator 4140 operate the same way on the entire building envelope.

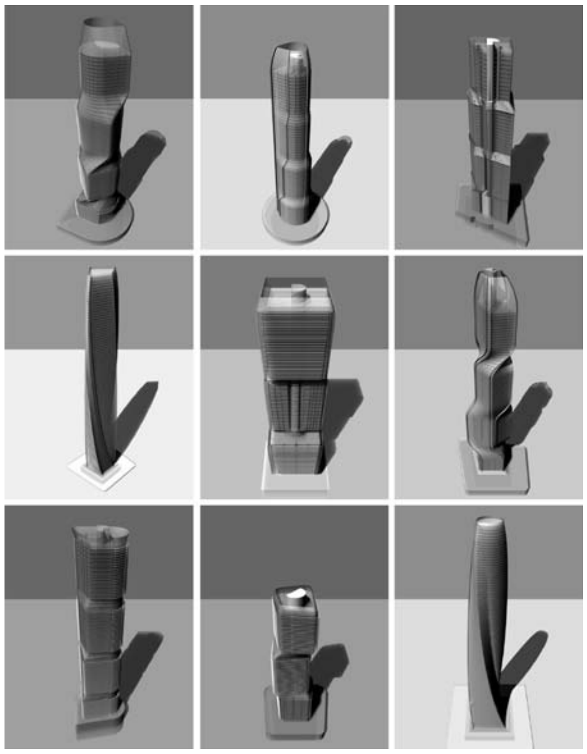


Figure 3.3 57 Skyscraper Designs Developed (Not Evolved) by Skyscraper MSEDS

Figure 3.3.59 illustrates a selection of skyscraper designs generated using the skyscraper MSEDS described herein. These designs demonstrate a high degree of variability in the forms than can be produced.

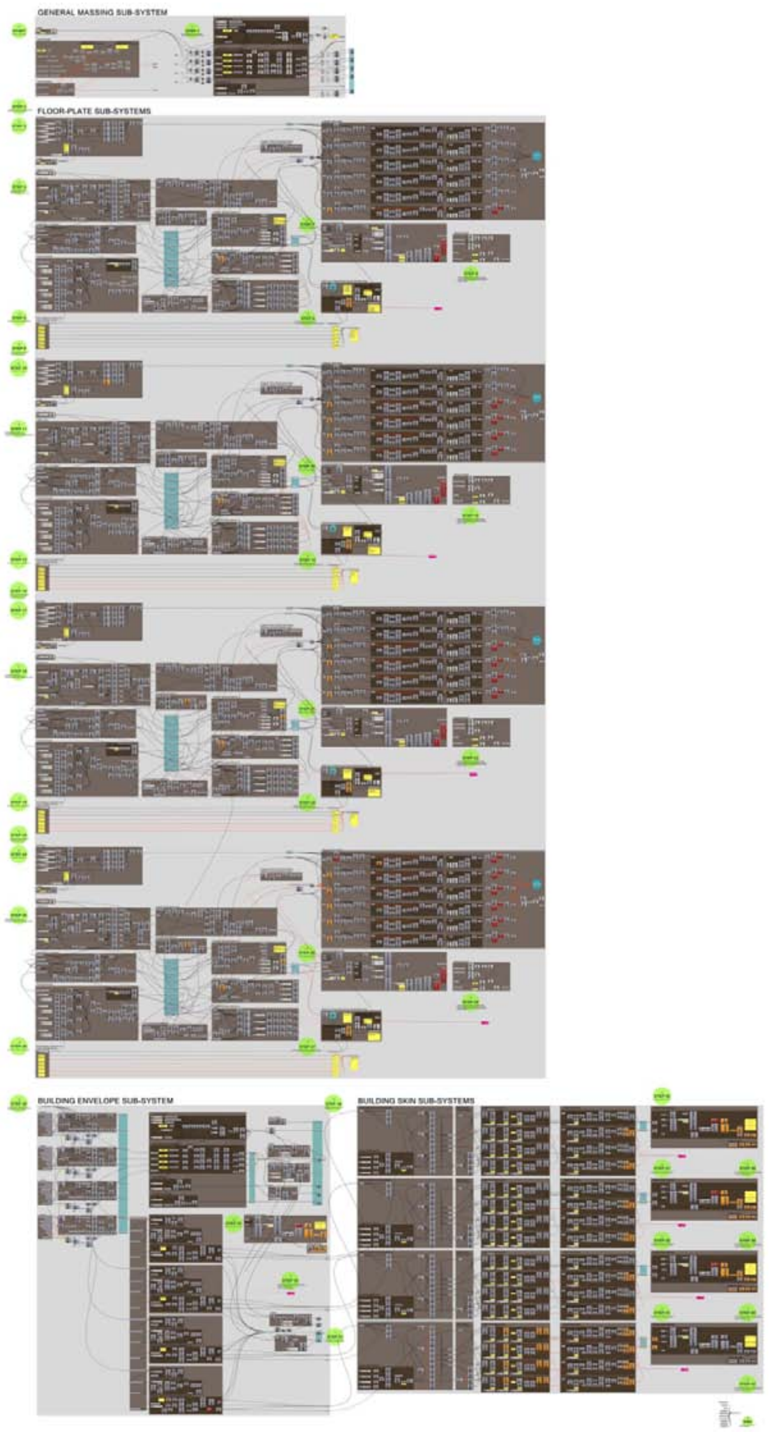

GENERAL MASSING SUB-SYSTEM
FLOOR-PLATE SUB-SYSTEMS
BUILDING ENVELOPE SUB-SYSTEM
BUILDING SKIN SUB-SYSTEMS

Figure 3.3 58 Grasshopper Implementation of Skyscraper MSEDS Customized for Hanking Competition

Figure 3.3.57 illustrates an implementation of the entire skyscraper MSEDS described in the preceding sections in the Grasshopper graphical scripting language.

3.4.1 Introduction

The final objective of this research is to test the skyscraper MSEDS design and implementation described in Chapter 3.3 by using it to evolve a skyscraper design that responds to the requirements and objectives of the Hanking competition (as it has been designed to do) and then evaluating the performance of the evolved design alongside other designs entered in the competition. Not only would this provide evidence about how well the skyscraper MSEDS design and implementation work to fulfill their intended purpose, but it could also help to validate the multi-stage evolutionary design approach itself.

3.4.2 Evaluation Setup

3.4.2.1 Evolved Design

A skyscraper design was evolved using the skyscraper MSED implementation based on the system design described in Chapter 3.3. Although the evolutionary process executed by each sub-system was convergent, repeated test runs of the same system with the same initial setup converged on a slightly different design each time, indicating that the evolved designs were all sub-optimal. Nevertheless, their substantial similarities suggested that they were probably only slightly sub-optimal, so a final run of each sub-system was carried out and the resulting design was saved for evaluation.

3.4.2.2 Competition Designs

The Hanking competition had six participants: internationally renowned architects/firms Thom Mayne (Morphosis) -*1st place Winner*, TFP Farrells -*2nd place*, Gensler - *3rd place*, Urbanis, Adrian Smith, and Leo a Daly. However, limits on time and resources dictated that the list of designs to be taken for comparison be restricted to three. Therefore, the top three designs by Morphosis, TFP Farrells and Gensler were selected and modeled based on documentation submitted during the competition and made available to all participants after judging. However, in the case of the Gensler design, the author was already in possession of a detailed and accurate design model so no reconstruction was required.

3.4.2.3 Evaluation Criteria

As described in Chapter 3.3, the Hanking Competition called for the design of a 110,000m^2, 330m-tall, flexible office tower in the city of Shenzhen, China. Among the design objectives expressed in the Design Brief issued to the competition participants, minimizing energy consumption, maximizing views and maximizing natural daylight are ranked as top priorities. Accordingly, the skyscraper MSEDS was designed to adaptively evolve designs that not only meet the program requirements and obey the relevant regulations, but that also excel in these three performance categories. It is assumed that the designs submitted by the competition participants also aimed to perform well in these areas.[120]

While daylight and views can be easily evaluated using the same techniques employed in the skyscraper MSEDS, it is somewhat more difficult to evaluate energy consumption because the designs are only resolved to the schematic level and there is insufficient information available about them. As the local climate in Shenzhen typically requires daytime cooling year-round, it may be inferred that mechanical cooling will account for the largest portion of overall building energy consumption, and in large, glazed office buildings, one of the largest contributors to daytime cooling loads is facade insolation. Accordingly, for the purposes of this comparative evaluation, facade insolation was taken as an indicator of energy consumption.

3.4.2.4 Evaluation Procedures

The Hanking competition program requirements called for 33,050m^2 of business apartments in a lower zone of the tower, with six apartments per floor, and 62,719m^2 of commercial office space to be distributed between a middle zone with 3-4 rental spaces per floor, and an upper zone with 1-2 rental spaces per floor. Thus, for the sake of a thorough comparison, three typical floors were chosen from each design for evaluation: one in the lower zone with 6 rental spaces, one in the middle zone with either 3 or 4 rental spaces, and one in the high zone with two rental spaces. The following evaluation procedures were carried out:[121]

[120] The author was a member of the Gensler design team and can testify that this was indeed the case for at least one participant.
[121] This work was performed in part for ARCH 690 Design Optimization through Building Simulation, Instructor: Manfred Zapka, phD, PE, LEED AP

- The selected floors were modeled using DesignBuilder software and daylight simulations were carried out using Radiance, the industry standard daylight simulation program.
- Next, the view performance of the selected floors was evaluated using the evaluation routine described in Chapter 3.3 and employed in the skyscraper MSEDS.
- Finally, models of the entire building models were constructed using Rhinocerous3D and then imported into Autodesk Ecotect for solar exposure simulation to determine a daily facade insolation value for each, averaged over a year.

3.4.2.5 Controls

Throughout the evaluation process, the following controls were maintained:

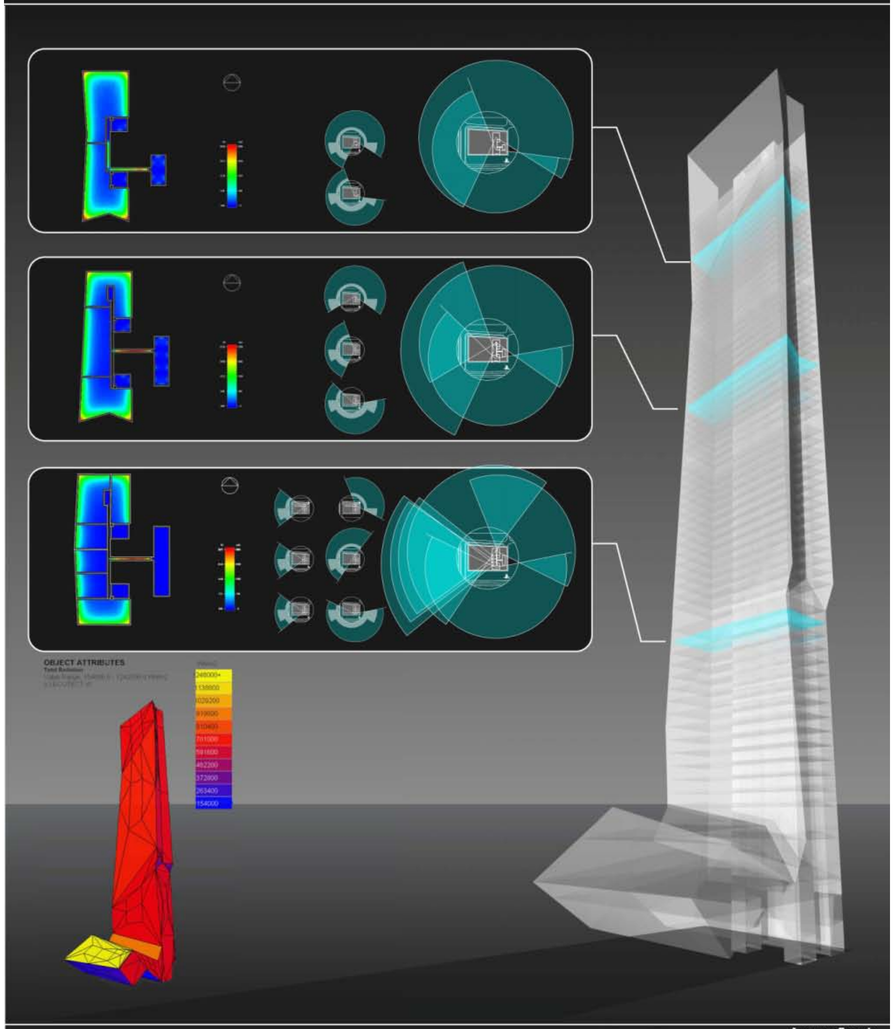
- all designs were modeled to the same level of detail;
- any design parameters not defined in one or more designs were assigned the same default value in all designs; and
- the same design-independent simulation parameters, such as weather data, etc., were applied in all simulations.

3.4.3 Evaluation Results

Figures 3.4.1 through 3.4.4 are images showing the evaluation results alongside a model of each of the four skyscraper designs. At the top of each figure is the name of the designer (the evolved design is indicated as "MSEDS") and a table displaying that design's rank in each of the three performance categories and its rank overall. The overall ranking was calculated by assigning equal weight to all three performance categories.

Design by ***Morphosis***

Performance Category	Rank
View	1
Daylight	1
Facade Insolation	4
Overall	2

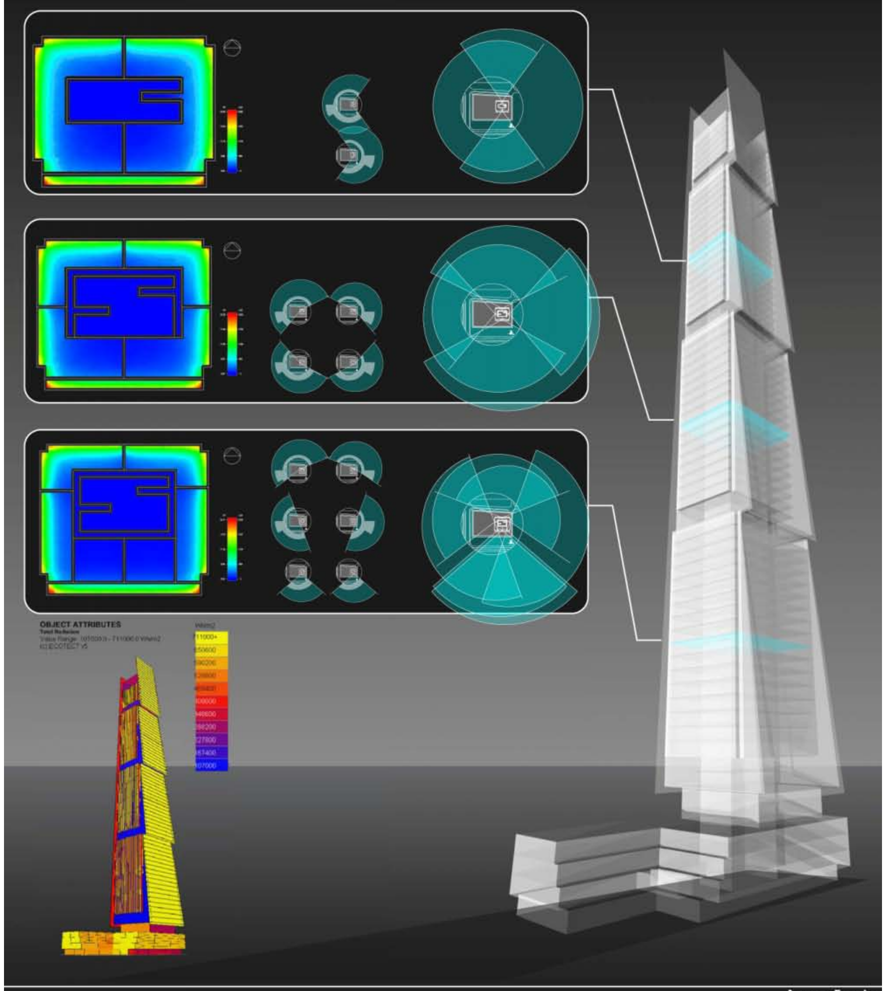


Floor	Rental Space	Area [m2]	View Value	Total View Value	Average View Value	Average DF [%]	UR [%] (Min/Max)	Overall Average DF	Overall UR	Average Facade Insolation [kWh/m2]
A	1	200	7052	70556	11759	7.52	8	5.24 %	4 %	697
	2	300	13319			6.76	5			
	3	160	8585			5.5	3			
	4	160	8587			2.97	2			
	5	160	9169			2.98	2			
	6	300	23844			3.09	2			
B	1	420	18399	55764	18588	6.5	3	4.70 %	3 %	
	2	450	12746			6.96	3			
	3	450	24624			3.71	4			
C	1	615	21966	44761	22380	6.29	4	5.22 %	4 %	
	2	615	22795			6.14	5			

Figure 3.4. 1 Morphosis Design Performance Evaluation Summary

Design by ***TFP Farrells***

Performance Category	Rank
View	3
Daylight	3
Facade Insolation	3
Overall	3

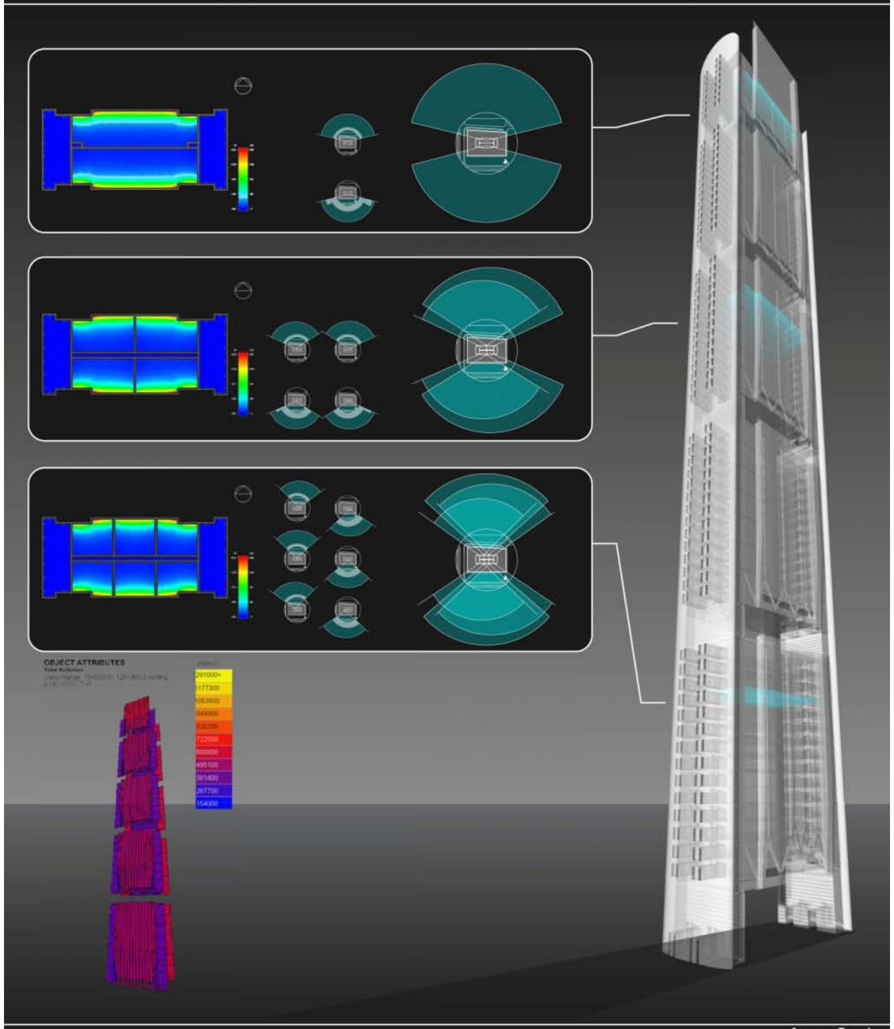


Floor	Rental Space	Area [m2]	View Value	Total View Value	Average View Value	Average DF [%]	UR [%] (Min/Max)	Overall Average DF	Overall UR	Average Facade Insolation [kWh/m2]
A	1	250	13923	67370	11228	6.78	8	4.46 %	7 %	654
	2	250	10571			5.07	6			
	3	250	12420			0.59	6			
	4	250	15795			5.25	6			
	5	200	7323			0.59	6			
	6	200	7338			6.92	7			
B	1	260	13633	62929	15732	3.42	2	5.07 %	5 %	
	2	260	12953			7.06	9			
	3	290	17845			3.51	2			
	4	290	18498			7.09	8			
C	1	510	10794	29443	14722	6.29	0	4.88 %	1 %	
	2	510	18649			6.14	2			

Figure 3.4. 2 TFP Farrells Design Performance Evaluation Summary

Design by **Gensler**

Performance Category	Rank
View	4
Daylight	4
Facade Insolation	2
Overall	4

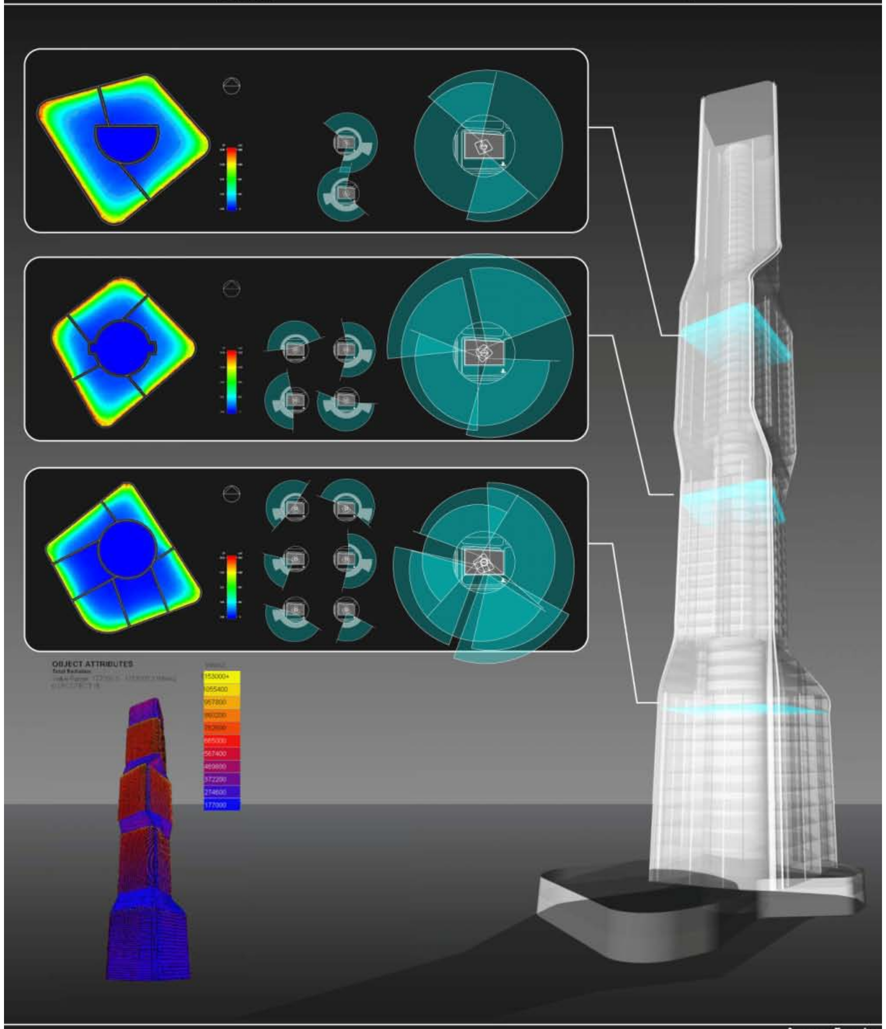


Floor	Rental Space	Area [m2]	View Value	Total View Value	Average View Value	Average DF [%]	UR [%] (Min/Max)	Overall Average DF	Overall UR	Average Facade Insolation [kWh/m2]
A	1	220	8090	34385	5730	3.35	3	3.27 %	3 %	483
	2	220	3518			3.36	2			
	3	220	7782			3.3	3			
	4	220	3387			3.22	2			
	5	220	3518			3.17	3			
	6	220	8090			3.23	2			
B	1	320	10896	30101	7525	3.47	3	3.40 %	3 %	
	2	330	4343			3.45	3			
	3	330	4343			3.33	3			
	4	320	10519			3.35	3			
C	1	670	5520	21531	10765	3.13	2	3.33 %	3 %	
	2	720	16011			3.55	4			

Figure 3.4. 3 Gensler Design Performance Evaluation Summary

MSED

Performance Category	Rank
View	2
Daylight	2
Facade Insolation	1
Overall	1



Floor	Rental Space	Area [m2]	View Value	Total View Value	Average View Value	Average DF [%]	UR [%] (Min/Max)	Overall Average DF	Overall UR	Average Facade Insolation [kWh/m2]
A	1	330	5682	70347	11725	6.63	0.06	4.33 %	3 %	443
	2	330	16180			5.31	0.05			
	3	330	10888			2.34	0.01			
	4	330	14679			4.32	0.03			
	5	330	8848			2.41	0.01			
	6	330	14066			5.2	0.05			
B	1	375	17277	57736	14434	4.91	0.04	5.28 %	4 %	
	2	375	5755			5.42	0.04			
	3	375	21966			5.3	0.05			
	4	375	12737			5.52	0.05			
C	1	750	18070	37139	18570	4.77	3	5.02 %	4 %	
	2	750	19068			5.26	4			

Figure 3.4. 4 MSEDS-Evolved Design Performance Evaluation Summary

The results shown in Figures 3.4.1 through 3.4.2 indicate that if the three evaluation criteria - views, daylighting, and facade insolation - are assigned equal priority, then the design evolved using the MSEDS outperforms all three designs from the competition overall. It overtakes the 2nd and 3rd place designs in every category, but rates slightly below the competition-winning design by Morphosis in the categories of view and daylighting performance. The MSED still comes out on top overall because the curiously oriented Morphosis design, with no shading devices to protect its broad Western facade, ranks last in the category of facade insolation.

3.5.1 Review

The overall goal of this research is to contribute to the development of a practical evolutionary design approach that would enable designers to design and implement computational systems to adaptively evolve novel building designs based on performance criteria. The main problem is to find a way to reduce epistasis and interdependent elements enough to allow convergent adaptive evolution of novel designs, without resorting to representations that are too generic to convey important building design information.

The central proposition of this thesis is that the integration problem can be overcome by strategically deconstructing the design problem into a set of simpler component problems with solutions that have fewer interdependent elements and that can be generated from fewer parameters using low-compression generative encodings that have low epistasis. According to this approach, it is proposed that epistasis and interdependent elements can be reduced enough to facilitate convergent adaptive evolution, and solutions to the component problems can be recombined to produce a novel overall solution with a high degree of variability that makes it difficult to anticipate even with foreknowledge of its component parts.

To implement this strategy, the multi-stage evolutionary design framework was developed. It comprises a system design methodology that broadly defines a set of procedures for designing multi-stage evolutionary design systems (MSEDSs) and using them to evolve design solutions, and a system architecture that specifies the general configuration of software and hardware components for the MSEDS. According to the framework, a design problem is deconstructed into smaller sub-problems based on a prioritized list of constraints and performance objectives, and a hierarchy of evolutionary design sub-systems is constructed to evolve solutions to each of the sub-problems. Sub-problems that relate to high-priority constraints and objectives are tackled first, and the evolved solutions constrain subsequent sub-problems in a manner consistent with design priorities. This allows the evolutionary search space to take in areas of the design space where the most important adaptations are likely to exist.

To sum up, breaking down the overall problem into smaller sub-problems reduces epistasis and interdependent elements to allow a convergent adaptive evolutionary process of partial design solutions, which can then be combined to produce a novel and optimal overall design solution. And hierarchically ordering the sub-systems efficiently structures the evolutionary search space to support the system's potential to develop useful adaptations that can improve performance. If the multi-stage evolutionary design framework is successful in overcoming the integration problem to allow convergent adaptive evolution of novel designs without blocking off important areas of the design space, then this research is successful in achieving its overall goal.

3.5.2 Outcomes

3.5.2.1 System Design

To test the multi-stages evolutionary design framework, it was applied to design an MSEDS for evolving solutions to a particular class of design problem - the skyscraper. The results show that a flexible, modular and expansible MSEDS design can be constructed by carefully deconstructing the design problem into four sub-problems: general massing, design of the floor-plates, design of the overall envelope form, and design of the more detailed properties of the building skin, including material properties and shading devices. This particular break-down and sequence of sub-problems reflects a particular set of performance objectives, described in Chapter 3.2 and 3.3. If the performance objectives of a particular skyscraper design problem are considerably different, then the hierarchical ordering of the sub-problems can be changed and the system reconfigured accordingly. Adding, subtracting or reordering sub-systems are system design-level modifications whose availability renders the framework flexible to address skyscraper design problems with a wide range of different design priorities.

Though not explored in detail in this thesis, it is suggested that the multi-stage evolutionary design framework could be applied to design MSEDSs for evolving solutions to a variety of other architectural design problems, besides skyscrapers, with equal success. This would involve alternative problem deconstructions that reflect the functions and characteristics of different building typologies. For example, a museum or gallery design problem might be deconstructed into an initial sub-problem of designing a circulation scheme to meet predetermined objectives that relate to the choreography of the patron's experience. This could be followed by the problem of laying out a set of floorplans constrained by the circulation scheme and the program requirements. A third sub-problem may be the design of the overall building form to accommodate the evolved plans and embody certain preferred cultural or stylistic qualities. To complete the design, a fourth sub-system may be dedicated the problem of designing the fenestration with careful attention to the manipulation of natural light within different spaces.

For a commercial shopping center, advertising effect and image are high-priority design objectives. Centrally located retail spaces with high visibility to shoppers bring in substantially higher rental fees. A convenient and engaging layout and a comfortable, well-lit, well-ventilated environment are also important objectives. According to the multi-stage evolutionary design framework, the first sub-problem should address these issues as directly as possible. It might come down to the three-dimensional form of the exterior walls that divide the main circulation space from the retail spaces, as they provide the main surfaces for advertising, they are the main obstacles to the visibility of storefronts, and they strongly affect lighting and airflow. Noting that sharp corners and ninety-degree angles restrict vision more than gradual curves and wide angles, one strategy might be to design a main wall sub-system with the encoded potential to develop smoothly curving walls. This could lead to forms with a range of interesting adaptations, from which complete plans could be developed, and finally some kind of roof structure.

These examples, though they are only 'thought experiments,' support the hypothesis that the multi-stage evolutionary design framework is flexible enough to tackle a wide range of architectural design problems having diverse performance objectives without blocking off important areas of the design space. This flexibility stems from the freedom to reframe the overall problem as any set of sub-problems that allows maximum variability of those parts of the design that have the greatest bearing on design performance in identified key areas.

The framework also allows particular system designs to be tailored to match more subtle variations in performance objectives, by modification on the sub-system level. Such modifications alter the functionalities and interrelationships of the modules constituting the development and evaluation systems within a sub-system. Such modifications affect the overall scheme of developing and evaluating design solutions to a given sub-problem, changing the interdependencies among elements of the solutions. For example, in the development system of the floor-plate subsystem of the MSEDS design described in Chapter

3.3, the service core and circulation layout are generated first, and the floor-plate and partitions and generated afterwards based on the location of the service core. Through a sub-system-level modification, the floor-plate could be generated first, and the core, circulation and partitions placed afterwards according to the layout of the floor-plate. Such a design could, for example, allow the core to always be centered at the geometric center of the floor-plate, or located at a given interval from the edge of the floor-plate.

The framework also allows particular generative and evaluative process carried out by modules in the development and evaluation systems to be altered via module-level modifications. Module-level modifications change the way the modified module operates but do not affect how it interoperates with other modules and functions overall within the system. Examples such as alternative core and circulation generators and alternative floor-plate generators are described in Chapter 3.3. Both sub-system-level modifications and module-level modifications can be used to alter constraints and access new fields of design variation.

3.5.2.2 System Implementation

Use of the multi-stage evolutionary design framework to design the skyscraper MSEDS shows that the framework enables the design of flexible, modular and expansible systems that can efficiently structure the evolutionary search space to cover the most important regions of the design space. However, to test whether the framework can overcome the integration problem to allow convergent adaptive evolution of novel designs requires construction and testing of a prototype system implementation, so one was built based on the skyscraper MSEDS design.

The first real test of the system's performance related to the variability of designs it can generate. The selection of generated designs shown in Figure 3.3.59 demonstrates a considerable degree of design variability. This speaks to the potential of the system and, by extension, the framework to produce novelty. Given the variation illustrated in Figure 3.3.59, it is hard to predict what form of design the skyscraper MSED will produce.

The second performance test was to try evolving a design and then observe whether the evolutionary process converges around a single design. Throughout repeated trials, the evolutionary process executed by each sub-system was observed to converge. However, in the case of the floor-plate sub-system, even with the same initial setup, the population converged on a slightly different design each time, indicating that the evolved floor-plate designs were all sub-optimal. But, based on their substantial similarities, they were probably only slightly sub-optimal. It is likely that the fitness landscape of the floor-plate sub-system had a number of closely grouped local optima, and the evolving population was converging on those. In future work, this kind of problem can be corrected by modification of the implementation details, or module-level or sub-system-level modification of the system design.

Whether the MSEDS was finding global optima or local optima, the evolutionary process was convergent, demonstrating that the MSEDS and, by extension, the multi-stage evolutionary design framework had overcome the remaining part of the integration problem. Thus, it is concluded that this research successfully achieved its primary objective.

3.5.2.3 Performance Evaluation

To more directly test the capability of the skyscraper MSEDS design and implementation to evolve high-performance designs, and to help validate the multi-stage evolutionary design framework, a skyscraper design was evolved for the Hanking competition and compared with the designs of the top three competition entries by simulating the daylighting, view and facade insolation performance of each design. The detailed set up and results of this comparative evaluation are described in Chapter 3.4. To recap, the evolved design was found to outperform all three conventional designs in overall performance, despite placing second behind Thom Mayne's design in the daylighting and view performance categories.

This not only supports the hypothesis that the multi-stage evolutionary design approach is capable of producing designs for buildings that outperform designs developed using conventional performance-based approaches, but it also demonstrates one of the most serious potential limitations of evolutionary design - restricted variability. The evaluation results indicate that the Morphosis design excels in view and daylight performance because of its narrow, rectangular floor-plates and detached service core connected to the main floor-plate via a narrow bridge corridor. The potential to generate such a detached core configuration, apparently a very useful adaptation, was not encoded into the MSEDS.

In order to evolve designs with useful adaptations that improve performance, the generative processes employed in development must allow sufficient variation in the generated designs.

On the other hand, if variability is too unrestricted, an over-abundance of non-functional forms could be generated, which would seriously hinder evolution. An important system design objective, then, is to find ways to constrain variability so as to suppress generation of non-functional forms and configurations without blocking off areas of the design space that may contain useful adaptations. To this end, the system designer must draw upon a broad body of architectural knowledge, as well as his/her personal insights, experiences and sensibilities.

3.5.3 Human-Computer Synergy

In evolutionary design, the role of the designer is not to define a single design solution, but rather to define the boundaries of the design solution space, and to specify the criteria for evaluating the merit of individual designs in the space. The task of the evolutionary design system is then to sweep through the solution space in search of the best designs. This partnership between human and computer requires that the human designer put his/her knowledge, insight, experience and sensibilities to work on a higher plane of abstraction than in conventional design processes. Instead of focusing on the lines and proportions of a particular design, guided by personal aesthetic preferences and intuition, the designer creatively devises ways to produce and control variation based on knowledge, experience and skill, and encodes that essential architectural intelligence and creativity into a computational system that can explore and evaluate thousands of options.

This approach plays to the strengths of both the human designer and the computer assistant, maximizing the synergy of human cognition and computation. Because of its ability to take samples from all corners of the evolutionary search space and gradually hone in on the best designs, there is the unique potential to produce optimal designs that are virtually impossible to discover using conventional methods. Once a design is evolved, the human designer is free to further develop and refine the design, or to make adjustments to the system and evolve an alternative design. This process helps to free designers from 'design fixation' and encourages experimentation and innovation.

3.5.4 Future Work

For a designer with coding experience, it requires a substantial amount of time and effort to construct any kind of evolutionary design system from scratch. For a designer with no coding experience, the task is all but impossible. However, if architectural intelligence of the kind discussed above could be gathered and encoded to produce a library of modules and sub-systems that could be assembled to build MSEDSs within a user-friendly application environment, then a designer with no coding experience could select and combine modules and sub-systems from the library to create and customize new systems to address new design problems. Among all evolutionary design systems, this potential is a unique feature of the MSEDS framework developed from this research.

Future research will focus on developing this potential by gathering and encoding design intelligence, and adopting a new platform for implementation to provide a more user-friendly interface. The hope is that this line of research will eventually make evolutionary design tools available for architects to use in mainstream practice.

Bibliography

Aranda, Benjamin, and Chris Lasch. *Pamphlet Architecture 27: Tooling*. New York: Princeton Architectural Press, 2005.

Arora, J.S., and R.T. Marler. "Survey of multi-objective optimization methods for engineering." *Structural Multidisciplinary Optimization*, Vol.26, Nr.6 (2004), 369-395.

Bäck, T. *Evolutionary Algorithms in Theory and Practice*. New York: Oxford Univerity Press, 1996.

Baron, P., R. Fisher, F. Mill, A. Sherlock, and A. Tuson. "A Voxel-based Representation for the Evolutionary Shape Optimization of a Simplified Beam: A Case-study of a Problem-centered Approach Genetic Operator Design." In 2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2), 1997.

Baron, P., R. Fisher, A. Tuson, and F. Mill. "A Voxel-based Representation for Evolutionary Shape Optimization". *AI EDAM Special Issue: Evolutionary Design*, Vol.13, Nr. 3 (1999).

Beasley, D. "Possible Applications of Evolutionary Computation." In *Evolutionary Computation 1: Basic Algorithms and Operators*, edited by T. Bäck, D.B. Fogel, and T. Michalewicz, 4-19. Philadelphia: Institute of Physics Publishing, 2000.

Bentley, Peter. "Aspects of Evolutionary Design by Computers." In Proceedings of the 3rd On-Line World Conference on Soft Computing:WSC3 (1998): 1-17. http://arxiv.org/html/cs/9809049/dss5.html.

Bentley, P. "An Introduction to Evolutionary Design by Computers." In *Evolutionary Design by Computers*, edited by P. Bentley, 1-73. San Francisco: Morgan Kaufman Publishers, 1999.

Bentley, P.J. "Generic Evolutionary Design of Solid Objects using a Genetic Algorithm." Doctoral dissertation, Division of Computing and Control Systems, Department of Engineering, University of Hudersfield. 1996.

Bentley, P.J., ed., *Evolutionary Design by Computers*. San Francisco: Morgan Kaufmann Publishers, 1999.

Bentley P.J., and D.W. Corne., eds. *Creative Evolutionary Systems*. London: Academic Press, 2002.

Bukhari, Fakhri, John H. Frazer, and Robin Drogemuller. “Evolutionary Algorithms for Sustainable Building Design.” Paper presented at the 2nd International Conference on Sustainable Architecture and Urban Development, Amman, Jordan, July 12-14, 2010.

Burry, Jane, and Mark Burry. *The New Mathematics of Architecture*. London: Thames and Hudson, 2010.

Caldas, L. “An Evolution-based Generative Design System: Using Adaptation to Shape Architectural Form.” Doctoral dissertation, Massachusetts Institute of Technology, 2001.

Caldas, Luisa, “Generation of Energy-efficient Architecture Solutions Applying GENE_ARCH: An Evolution-based Generative Design System.” *Advanced Engineering Informatics*, Vol.22 (2008), 59-70. http://www.sciencedirect.com.

Corser, Robert, ed. *Fabricating Architecture: Selected Readings in Digital Design and Manufacturing*. New York: Princeton Architectural Press, 2010.

Dasgupta, D., and Z. Michalewicz, eds. *Evolutionary Algorithms in Engineering Applications*. Dusseldorf: Springer-Verlag, 1997.

Dawkins, R. “Universal Darwinism.” In *Evolution from Molecules to Men*, edited by D.S. Bendall, 403-425. Cambridge: University Press, 1983.

De Vries, Peter. “Morphology and Structural Behavior of the Hyperbolic Lattice.” *Structural Morphology Colloquium* (2000), 374-381.

Di Cristina, Giuseppa, ed. *Architecture and Science (Architectural Design)*. Sussex: Wiley-Academy, 2001.

Evans, M.R., and O.J. O'Loan. "Alternating steady state in one-dimensional flocking." *Journal of Physics A: Mathematical and General*, Vol.32, Nr.8 (1999), L99-L105.

Fischer, Thomas, and Torben Fischer. "Toolmaking for Generative design." *International Journal of Design Computing*, asd Vol.6 (2003), 1-23.

Fogel, D.B. *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. IEEE Press, 1995.

Fox, Michael, and Miles Kemp. *Interactive Architecture*. New York: Princeton Architectural Press, 2009.

Frazer, John H. *An Evolutionary Architecture*. London: Architectural Association, 1995.

Garcia, Mark, ed. *The Patterns of Architecture (Architectural Design)*. New Jersey: Wiley, 2010.

Goldberg, D. G.. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading: Addison-Wesley, 1989.

Grondzik, Walter T., Alison G. Kwok, John S. Reynolds, and Benjamin Stein. *Mechanical and Electrical Equipment for Buildings*. 11th ed. New Jersey: John Wiley and Sons, 2010.

Gurdal, Zafer, and Raphael T. Haftka. *Elements of Structural Optimization*. Norwel: Kluwer Academic Publishers, 1992.

Hensel, Michael, Achim Menges, and Michael Weinstock. *Emergent Technologies and Design: Towards a Biological Paradigm for Architecture*. London: Routledge, 2010.

Hensel, Michael, Achim Menges, and Michael Weinstock, eds. *Emergence: Morphogenetic Design Strategies (Architectural Design)*. New Jersey: Wiley, 2004.

Hensel, Michael, Achim Menges, and Michael Weinstock, eds. *Techniques and Technologies in Morphogenetic Design (Architectural Design)*. New Jersey: Wiley, 2006.

Hensel, Michael, and Achim Menges, eds. *Versatility and Vicissitude: Performance in Morpho-Ecological Design (Architectural Design)*. New Jersey: Wiley, 2008.

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

Hornby, Gregory S. "Functional Scalability through Generative Representations: the Evolution of Table Designs." *Environment and Planning B: Planning and Design*, Vol.31 (2004): 569-587.

Hornby, Gregory S. “Generative Representations for Computer-Automated Evolutionary Design.” Paper presented at 2006 ERCOFTAG Design Optimization: Methods and Applications. Las Palmas, Spain, Apr. 5-7, 2006.

Hornby, Gregory S., and Jordan B. Pollack. “The Advantages of Generative Encoding for Physical Design.” Paper presented at 2001 IEEE Congress on Evolutionary Computation. Seoul, Korea, May 27-30, 2011.

Janssen, Patrick H.J. “A Design Method and Computational Architecture for Generating and Evolving Building Designs.” PhD diss., Hong Kong Polytechnic University, 2004.

Janssen, Patrick. “A Generative Evolutionary Design Method.” *Digital Creativity*, Vol.17, Nr.1 (2006): 49-63.

Janssen, Patrick, John Frazer, and Ming-Xi Tang. “A Computational System for Generating and Evolving Building Designs.” In Digital Opportunities: Proceedings of the 10th International Conference on Computer-Aided Architectural Design Research in Asia (2005): 463-474.

Janssen, Patrick H.T., John H. Frazer, and Ming-Xi Tang. “A Framework for Generating and Evolving Building Designs.” *International Journal of Architectural Computing*, Vol.3, Nr.4 (2005): 449-470.

Kolarevic, Branko. *Architecture in the Digital Age: Design and Manufacturing*. UK: Taylor and Francis, 2005.

Kolarevic, Branko, and Ali Malkawi, eds. *Performance-based architecture: Beyond Instrumentality*. London: Routledge, 2005.

Kolarevic, Branko, and Kevin Klinger, eds. *Manufacturing Material Effects: Rethinking Design and Making in Architecture*. London: Routledge, 2008.

Kolarevic, Branko. "Generative design and Computational Architectures." Presented at SIGraDi 2000, Rio de Janiero, 2000.

Kotnik, Toni. “Digital Architectural Design as Exploration of Computable Functions.” *International Journal of Architectural Computing*, Vol.8, Nr.1 (2010), 1-16.

Kottas, Dmitris. *Contemporary Digital Architecture: Design and Techniques*. Links International, Ceg, 2010.

Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: MIT Press, 1992.

Kubo, Michael, and Farshid Moussavi, eds. *The Function of Ornament*. Barcelona: Actar and Harvard Graduate School of Design, 2006.

Lagios, Kera, Jeff Niemasz, and Christophe F. Reinhart. “Animated Building Performance Simulation (APBS) – Linking Rhinoceros/Grasshopper with Radiance/Dayism.” Paper presented at Simbuild 2010, New York City, New York, August 2010.

Leach, Neil. “Generative design.” *Architectural Design*, Vol.79, Nr.1 (2009), 32-37.

Leach, Neil, David Turnbull, and Chris Williams, eds. *Digital Tectonics*. Chichester: Wiley-Academy, 2004.

Leach, Neil. “Generative design.” Lecture, University of East London, 2008.

Lindenmayer, Aristid. "Mathematical Models for Cellular Interaction in Development: Parts I and II." *Journal of Theoretical Biology* 18 (1968): 280-315.

Markov, A.A., and N.M. Nagorny. *The Theory of Algorithms*. Moscow: Academy of the Sciences USSR, 1954.

McWilliams, Chandler, and Casey Reas. *Form+Code in Design, Art and Architecture (Design Briefs)*. New York: Princeton Architectural Press, 2010.

Meredith, Michael. *From Control to Design: Parametric/Algorithmic Architecture*. Edited by Aranda-lasch and Mutsuro Sasaki. Barcelona: Actar, 2008.

Mitchell, M. *An Introduction to Genetic Algorithms*. Cambridge: MIT Press, 1999.

Mitchell, William J. *Logic of Architecture: Design, Computation, and Cognition*. Cambridge: The MIT Press, 1990.

Moussavi, Farshid. *The Function of Form*. Edited by Daniel Lopez and Garrick Ambrose, Ben Fortunato, Ryan R Ludwig, and Ahmadreza Schriker. Barcelona: Actar and Harvard Graduate School of Design, 2009.

Oxman, Rivka. "Performance-based Design: Current Practices and Research Issues." *International Journal of Architectural Computing*, Vol.6, Nr.1 (2008), 1-17.

Parmee, Ian. "Exploring the Design Potential of Evolutionary Search, Exploration and Optimisation." In *Evolutionary Design by Computers*, edited by Peter J. Bentley, 119-143. San Francisco: Morgan Kaufmann Publishers, Inc., 1999.

Querin, O.M., G.P. Stevens and Y.M. Xie. "Evolutionary structural optimisation (ESO) using a bidirectional algorithm." *Engineering Computations*, Vol.15, Nr.8 (1998), 1031-1048.

Rizos, Ioannis. "Next Generation Energy Simulation Tools: Coupling 3D sketching with Energy Simulation Tools." MSc Thesis, University of Strathclyde, 2007.

Roudavski, Stanislav. "Towards Morphogenesis in Architecture." *International Journal of Architectural Computing*, Vol.7, Nr.3 (2009), 345-374.

Shea, Kristina, R. Aish, and M. Gourtovaia. "Towards Integrated Performance-based Generative Design Tools." Edited by W. Dokonal, *Digital Design*. Presented at ECAADE 2003, Graz Austria, 2003, 253-264.

Shea, K. "Essays of Discrete Structures: Purposeful Design of Grammatical Structures by Directed Stochastic Search." Doctoral dissertation, Carnegie Mellon University, 1997.

Shea, K. "An Approach to Multiobjective Optimization for Parametric Synthesis." In 13th International Conference on Engineering Design (ICED 01) – Design Methods for Performance and Sustainability, WDK 28 (2001), 203-210.

Shea, K. "Directed Randomness." In *Digital Tectonics*, edited by N. Leach, D. Turnbull, and C. Williams, 10-23. London: Academic Press, 2004.

Silver, Mike, ed. *Programming Cultures: Architecture, Art and Science in the Age of Software Development (Architectural Design).* Academy Press, 2006.

Spuybroek, Lars, ed. *Research & Design: the Architecture of Variation*. London: Thames and Hudson, 2009.

Stiny, George. *Shape: Talking about Seeing and Doing*. Cambridge: MIT Press, 2006.

Teknomo, Kardi, Yasushi Takeyama, and Hajime Inamura. “Determination of Pedestrian Flow Performance Based on Video Tracking and Microscopic Simulations.” *Proceedings of Infrastructure Planning Conference*, Vol.23, Nr.1 (2000), 639-642.

Terzidis, Kostas. *Algorithmic Architecture*. London: Architectural Press, 2006.

Terzidis, Kostas. *Expressive Form: A Conceptual Approach to Computational Design*. London: Spoon Press, 2003.

Weinstock, Michael. *The Architecture of Emergence: the Evolution of Form in Nature and Civilisation*. New Jersey: Wiley, 2010.

Wolfram, Stephen*. A New Kind of Science*. Champaign: Wolfram Media, 2002.

Woodbury, Robert. *Elements of Parametric Design*. London: Routledge, 2010.