





Proposing a Comprehensive Theoretical Training Framework (Concepts, Elements and Design Process) for Computational Design (Algorithmic, Parametric and Generative Design Systems)

Mina Ramyar¹ , Cyrus Bavar²  , Parisa Alimohammadi³ 

1. Department of Architecture, Sav.C., Islamic Azad University, Saveh, Iran. E-mail: mina.ramyar@iau.ac.ir

2. Corresponding Author, Department of Architecture, Sav.C., Islamic Azad University, Saveh, Iran. E-mail: cyrusbavar@iau-saveh.ac.ir

3. Department of Architecture, CT.C., Islamic Azad University, Tehran, Iran. E-mail: par.alimohammadi@iau.ac.ir

Article Info

Article type:

Research Article

Article history:

Received December 04, 2024

Received in revised form June 13, 2025

Accepted July 14, 2025

Published online August 15, 2025

Keywords:

Computational design,
Education,
Algorithmic design,
Parametric design,
Generative design.

ABSTRACT

In recent decades, computer technologies like computational design have made an impact on architectural design. They were first used for automation and form finding, later used for performance-based design and optimization. Computational design lead to the development of algorithmic, parametric, and generative design systems, which are now extensively used in architectural design education. According to previous studies, computational design education mainly focuses on the application of coding and related software, and theoretical knowledge of computational design not proposed and taught in a separate course before its use in the design studio. However, due to the complexities of computational design, an extensive training course is needed to fully understand its capabilities. Therefore, this research proposes a comprehensive theoretical training framework for computational design. To accomplish this objective in the first stage of this research, the current status of its training was examined, and deficiencies in computational design education have been identified through library resources. In the second stage, important concepts for comprehending computational design knowledge were examined, and in the third stage, with the goal of overcoming the deficiencies of the current educational program, a comprehensive theoretical training framework which includes two phases of 1. Learning computational design principles 2. Learning an analysis of computational design principles is proposed. The proposed program includes concepts such as definitions, types, distinctions, components and process of computational design. The findings of this study could serve as a framework for curriculum development in computational design.

Cite this article: Ramyar, M., Bavar, C., & Alimohammadi, P. (2025). Proposing a Comprehensive Theoretical Training Framework (Concept, Elements and Design Process) for Computational Design (Algorithmic, Parametric and Generative Design Systems). *International Journal of Applied Arts Studies*, 10(2), 49-96.



© The Author(s).

Publisher: Islamic Azad University, Yazd Branch.

Introduction

Computational design (CD) has gained popularity in architectural design and education in recent decades (Caetano, Santos, and Leitão, 2020; Ostrowska-Wawryniuk, Strzała, and Słyk, 2022). However, there are two problems with training CD: 1. Learning to program (Austin and Qattan, 2016). 2. extensive CD knowledge (Caetano et al., 2020). To solve the first problem, it was proposed to offer programming courses separately (Austin and Qattan, 2016). The second issue is extensive CD knowledge (Caetano et al., 2020), which some students struggle to apply during the design process (Agkathidis, 2015). However, while studying and assessing existing CD research, it became apparent separate comprehensive course CD knowledge course had not been proposed prior to its implementation in the design studio. For example, students in research (Abdelmohsen, 2013) should acquire knowledge of CD. However, training is required. Some research such as (Fischer, 2002) have exclusively focused on programming. Additionally in some research, only some aspect of CD knowledge is considered. For example (Bianconi and Filippucci, 2018; Lakhanpuria and Naik, 2023), focused on generative and parametric design, although it is obvious that students need to be familiar with algorithmic design before applying these methods. Algorithmic design is the fundamental system underlying other CD systems (parametric and generative) (El-Khaldi, 2007). Lack of CD understanding resulted in limited use of this technology because CD applications in architectural design are various, including automation, form finding (Caetano et al., 2020), performance-based design, and optimization (Alfaris, 2009). To solve this problem, comprehensive theoretical training framework is being proposed in this research that include topics such as: 1. The concept of CD and digital design (DD) and their distinction 2. CD systems (Algorithmic, Generative, Parametric) 3. Differentiation of CD systems 4. Elements and concepts that shape CD systems 5. The concept of system 6. similarity of system concept and CD 7. Prescriptive models and CD Process. The first step is to understand CD concepts and differentiate between DD and CD (Caetano et al., 2020). The Next step is to learn about CD systems and how they differ (El-Khaldi, 2007). Systems consist of units and institutions that work together to achieve a common goal (Schmidt and Taylor, 1970). Systems include concepts such as hierarchy, relationships and rules (Alfaris, 2009). CD systems also takes these factors into account (El-Khaldi, 2007). Additionally according to MIT research, the performance-based CD design process consists of decomposition, formulation, synthesis, analysis, evaluation and optimization (Alfaris, 2009). By learning CD knowledge, its application in architectural design becomes more targeted and conscious. In fact, there are prerequisites in the field of CD that should be provided (Fasoulaki, 2008), to achieve better results in this field. Therefore, in the next section, CD training status and important CD concepts in the theoretical literature is examined.

Theoretical Literature

CD Training Status

Over the past two decades, CD has been used in architecture to solve a variety of design problems (Caetano et al., 2020). Therefore, the curriculum should be adapted to the current situation (Shtepani and Yunitsyna, 2023). Using CD requires extensive theoretical knowledge (Caetano et al., 2020) and programming skills (Shtepani and Yunitsyna, 2023) that many students lack. To solve first problem, (Austin and Qattan, 2016) proposes separate programming courses. Analyzing previous research shows that there is no separate comprehensive theoretical training framework before its use in design process (Vrouwe et al., 2020; Agirbas, 2022)). The training program is based on the research plans of professors (Oxman, 2008). However Students should have detailed theoretical knowledge, such as algorithmic thinking, before applying it. (Abdelmohsen, 2013)) aimed to integrate generative design and digital construction into architectural design education. The students have personally dealt with generative design. But CD knowledge should be taught fully, and professors play an important role (Agkathidis, 2015). Additionally Parametric design and CNC production have been applied in educational research (Karzer and Matcha, 2009). (Gürbüz, Çağdaş, and Alaçam, 2010) used fractals to create design solutions in the early stages of design education. Furthermore (Guidera, 2011) conducted research on parametric generative design education. Other studies taught generative design approaches such as shape grammar through collaborative design (Knight, 2012). Also architectural spaces were reconfigured using generative design and digital construction by students (Abdelmohsen, 2013). Another research in education created a generative model using an ecological approach (Yavuz and Çelik, 2014). In addition, generative design and physical testing have led to a new design process in the design studio (Huang and Xu, 2015). Agkathidis, (2015) examined the impact of generative design on architectural design education and (Bianconi and Filippucci, 2018) examined education in generative design and how design thinking can be transformed through the use of these systems. During landscape design education, a database for generative design and landscape design concepts were introduced (Vaz and Celani). Other studies have included mathematical and algorithmic in early design education (Ostrowska-Wawryniuk, Strzała, and Słyk, 2022). In another research (Abdelmohsen et al., 2017) discussed combination of generative design and intuition can be beneficial in design education. Also recent research used problem-solving based learning based on parametric design thinking in an architectural studio in India (Lakhanpuria and Naik, 2023). Another article evaluated 3D printing and parametric modeling tools by architecture students (Shtepani and Yunitsyna, 2023). Additionally, (Nazidizaji and Safari, 2013) developed algorithmic approaches and reverse engineering for architectural analysis. A significant trend involves integrating algorithmic and parametric thinking (Peteinarelis and Yiannoudes, 2018; Vazquez, 2024). Also, pedagogical

innovations include proposing new teaching methods, such as using incomplete instructions (Vazquez, 2024), fostering interactive learning environments and developing collaboration skills (Vrouwe et al., 2020; Agirbas, 2022).

Furthermore, a review of internal references revealed that they also did not present a comprehensive educational program covering CD concepts and processes. Instead, they primarily focus on digital design education description, computer-aided design (CAD), and the general application of computer technology in education. For example, Poursistany et al. (2016) analyzed the impact of digital education on architectural creativity. Additionally (Asefi and Imani, 2017) investigates the impact of digital software on enhancing creativity in design education. Mahmoudi and Naghizadeh, (2010) addresses the transformation of architectural education due to the introduction of Information Technology (IT) as a design tool for idea representation, speed, flexibility, and 3D visualization, which manual tools lack. Additionally (Ahmadi Tabatabaie and Moosav, 2024) focuses on identifying the appropriate time and method for teaching software to enhance students' creativity. Their findings strongly recommend that software training should commence after students acquire a strong foundation in design and hand drawing. Furthermore (Eynifar and Hosseini, 2014) suggests that digital technologies in architectural design education be viewed as "media" rather than merely tools, as they serve as mediators and shape ideas. Therefore, it is apparent that several studies have incorporated CD systems in their curriculum but separate comprehensive CD knowledge training program was not proposed. The integration of technology into architectural design education precedes the development of its theoretical framework (Schumacher and Krish, 2010). CD training lacks complete training program (Fischer & Herr, 2001) and is not fully covered in the architectural design curriculum (Gürer, Alaçam, & Çağdaş, 2012). Understanding algorithmic thinking is crucial in CD education (Ozkar, 2017). Architectural education should provide future architects with algorithmic thinking skills and thinking (Ostrowska-Wawryniuk, Strzała, and Słyk, 2022). Architectural education must respond to these changes (Soliman, Taha, and El Sayad, 2019) and students should learn fundamental CD concepts. The next section will cover the fundamental CD concepts that students need study in a distinct course in order to meet these changing demands on architecture education.

Computational design and Digital design

Digital design (DD) and computational design (CD) have been driven by the advances in computer technology over the last decades. While DD requires computer tools, CD can be performed with or without a computer (Caetano et al., 2020). Architects used computing and algorithms to break down complicated design problems and solve them more effectively (M Rocker, 2006). Recent advances have resulted in CD replacing CAD (computer-aided design) in architecture (Kalay, 2004). This methodology drastically changes the standard design method by introducing innovative methodologies (Gurcan Bahadir and Tong, 2025). CD requires extensive

design knowledge and enables automation, form finding, optimization and performance based design (Caetano et al., 2020). CD concepts has developed algorithmic, parametric and generative design methods in architecture (Michelle and Gemilang, 2022). These methods have gained popularity in optimization, simulation (Oxman, 2017). Their applicability went beyond design automation and form finding (Mitchell & Terzidis, 2004). CD methods follow a system structure (El-Khaldi, 2007). A system is a collection of units working toward a coherent goal (Schmidt and Taylor, 1970; Alfari, 2009). System includes CD-related ideas such as hierarchy (El-Khaldi, 2007), relationship (Gu, Yu, and Behbahani, 2021) and rule (Doe, 2018). As technology advances, CD is becoming an increasingly important component in architectural design (Fatai, 2024) and education (Indraprastha, 2018). CD systems are more important than digital technologies for promoting CD thinking in architectural education (Adem and Çağdaş, 2020).

a. Algorithmic design system

Algorithmic design (AD) systems serve as a basis for the development of other CD systems (El-Khaldi, 2007). Online Cambridge dictionary defined the word algorithmic as “connected with or using algorithms.” AD become more and more popular because of its versatility and ability to establish work environments free from constraints (Castelo-Branco, 2020). Terzidis proposed AD (Terzidis, 2004), a process that uses algorithms (Sammer, Leitão, and Caetano, 2019). AD Thinking provides a step-by-step guide to achieve design goals and it supports designers in analyzing the context and understanding connections (El-Khaldi, 2007). AD is used in 3D building printing (Guerguis et al., 2017), residential project design (Chen, 2020) and building facades (Caetano, Garcia, Pereira, and Leitão, 2020) and envelope design Figure 1, (El-Khaldi, 2007). Algorithms can find the nth member of an infinite set (Leeds, 1977). It has the potential to produce a novel method of idea generation that is beyond human perception (Terzidis, 2006). Algorithms can be executed in parallel, sequentially (Figure 2,3) or randomly (El-Khaldi, 2007). AD uses algorithms to create design models (Michelle and Gemilang, 2022), the relationship between the algorithm and the design is evident in algorithmic designs such as Morpheus Hotel Figure 4, (Caetano et al., 2020). Algorithms manipulate numbers, alphabets, geometric elements and fixed/variable units (Caetano and Leitão, 2021). Functions connect algorithms to units using equations including operators and architectural operators include activities like movement and rotation (El-Khaldi, 2007). Furthermore (Moussavi, 2009) explores the influence of function on form. In algorithmic design, inheritance refers to a directional relationship in which the child inherits the characteristics of its parent (El-Khaldi, 2007). A rule-based algorithm can be described as follows: If the condition... is true, start the function (De Souza and Ferreira, 2002). Decomposition (dividing a task into subtasks) is a key concept in algorithms. (Fried et al., 2018).

b. Parametric design system

Parametric design (PD) is one of the most commonly used CD methods that allows the creation of a parametric model by specifying dimensions and geometry (Wahbeh, 2017). Online Cambridge dictionary, defined the word parametric as “relating to the parameters of something.” Morty introduced parametric design in 1971 as the study of dimensional relationships through the use of parameters (Moretti, 1971). Its powers were further enhanced by the advent of parametric animation in the late 1990s to manipulate forms dynamically by adjusting dimensions, constraints, and connections (Mark, 2008). Greg Lane's work based on transformations is well-known examples. Catia creates models in Figure 5, is regulated by two main parameters: thickness and height (El-Khalidi, 2007). PD is defined by its ability to create multiple solutions through rule-based algorithms, allowing for dynamic adjustments (Gu, Yu, and Behbahani, 2021; Jabi et al., 2017; Eastman, 2011). It enhances creativity by enabling designers to visualize and manipulate complex relationships within their designs (Campbell and Shea, 2014). PD has been used in green building design (Zhang, 2020) and energy efficient design (Touloupaki and Theodosiou, 2017). PD has been described in different ways (Caetano et al., 2020) as an optimization technique that identifies solutions within constraints (Eggert, 2005) and as a design style (Schumacher, 2008). Any system capable of connecting pieces is parametric whereas object properties are established through connections and inheritance (El-Khalidi, 2007). It is a subset of both algorithmic and code-based design (Elghandour et al., 2016). PD can accommodate any unit and relies on relationships. Designers can use inheritance to create families of objects, with changes in the first generation affecting the second generation (El-Khalidi, 2007). When parameters are used in algorithmic and generative design, they can be parametric (Caetano et al., 2020). Parametric design can shift the focus from form to function. For example in Figure 6, PD have been used to discover shape and achieve goals such as user visual comfort, energy optimization and solar protection (Tabadkani et al., 2019). This method can help choose the best solutions from a variety of design options (Khamis et al., 2022).

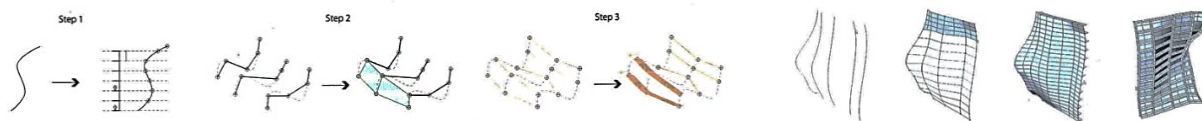


Figure 1. An example of an algorithmic envelope design (El-Khalidi, 2007).

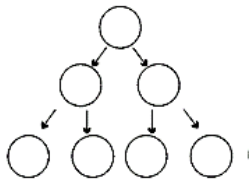


Figure 2. Parallel execution of algorithm (El-Khaldi, 2007).

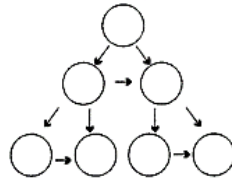


Figure 3. Sequential execution of algorithm (El-Khaldi, 2007).



Figure 4. Algorithmic design of Morpheus hotel (Source: Archdaily).

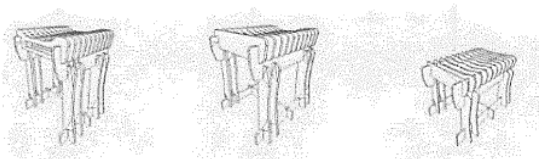


Figure 5. An example of a parametric design (El-Khaldi, 2007).



Figure 6. An example of a parametric facade design (Tabadkani et al., 2019).

c. Generative design system

Generative design (GD) systems have long been used and Durand applied it to architecture in 1803, developing new ways to create plans by assembling structural elements (Fasoulaki, 2008). Online Cambridge Dictionary defined generative as “able to produce or create something.” These systems use parallel, sequential and random algorithms (El-Khaldi, 2007). GD is an algorithmic or rule-based technique that creates a variety of possible design solutions (Ashour and Gogo, 2024). This approach executes programmed instructions until the necessary conditions are met, and simple algorithms produce sophisticated results (Humppi, 2015). Cellular automata, L-system and shape grammar are three examples of generative design systems (El-Khaldi, 2007) ; (Fasoulaki, 2008; Abdelmohsen, 2013; Toussi, 2020). Fractals have also been considered as a GD system (El-Khaldi, 2007; Fasoulaki, 2008). There is a relationship between algorithm and design output in AD, but not in GD, sophisticated creations based on simple algorithms (Caetano et al., 2020). L-systems model plant growth (Prusinkiewicz et al., 2018), cellular automata model reproduction (El-Khaldi, 2007), fractals model self-similarity (Lorenz, 2011), and shape grammars mimic the human ability to visually observe and calculate (Stiny, 2022).

d. Generative design system (cellular automata)

Cellular automata (CA) is a GD technique that simulates reproduction (Caetano et al., 2020). John von Neumann's abstract model served as the original inspiration for cellular automata (El-Khaldi, 2007). CA facilitate the generation of spatial layouts by considering user-defined parameters such as geometry and adjacency requirements (Ng, Chen, and Sathikh, 2024). It can model ecological dynamics, allowing for the integration of environmental factors into architectural design (Liu and Herr, 2023). The network of interconnected cells adjusts their state according to its neighbors and local regulations (Patt, 2015). Its applications in architecture can range from facades and interior elements (Herr and Ford, 2015) to the design of urban districts (de Oliveira and Celani, 2019). CA consist of replacement rules, cells (can contain geometric descriptions, colors, numbers, and other data) and initial states and inheritance is not possible with cellular automata because information is not passed on across generations (El-Khaldi, 2007). Chris Langton's diagram illustrates the behavioral transition of CA from fixed rules (generate cells in a fixed section) to random behavior (generates them in random mode) Figure 7, (Flake, 2000). Cellular automata can generate intricate patterns in architectural design (Herr & Ford, 2015). Wolfram, (2002) has studied one-dimensional cellular automata. He found the eight fundamental combinations of primitive cellular automata. Two states (black or white) yield eight (23) combinations Figure 8, and according to the initial combinations, there are 256 potential states (28). The Figure 9, shows CA with rules, an initial state and replaced states. John Fraser used them to form shapes (Januszkiewicz and Paszkowska-Kaczmarek, 2023). The Figure 10, shows application of cellular automata in envelope design (El-Khaldi, 2007). They are crucial to the development of CD thinking in design studios (Adem and Çağdaş, 2020). The integration of CA in academic settings promotes innovative design thinking, preparing future architects to leverage these tools in real-world applications (de Oliveira and Celani, 2019).

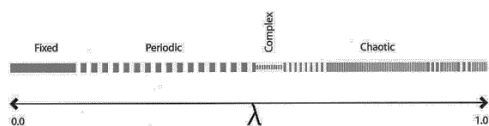


Figure 7. Behavior of CA from fixed to random proposed by Chris Langton (Flake, 2000)



Figure 8. Eight primary combinations of CA (Wolfram, 2002)

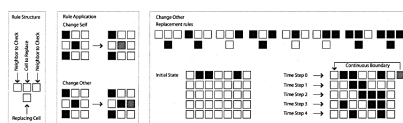


Figure 9. Combinations of CA (Wolfram, 2002)



Figure 10. CA in envelope design (El-Khaldi, 2007)

e. Generative Design System (L-System)

Aristide Lindenmayer developed the L-System generate shapes using strings, alphabets, rules, and repetitions (Lindenmayer, 1968). They provide a formalism for simulating plant growth (Prusinkiewicz et al., 2018). They can represent complex branching structures and organ differentiation, enhancing the realism of plant simulations (Loi, and Cournede, 2008). Timed, parametric L-systems enhance their ability to model dynamic phenomena like morphogenesis and mechanical models (McCormack, 2004). Each generation replaces previous data, enabling the generation of new structures without retaining prior configurations (Št'ava et al., 2010) and algorithms are executed in parallel. The alphabet growth representation creates a tree-like grid Figure 11, (Prusinkiewicz and Lindenmayer, 2012). Letters are the smallest units of the system. Figure 12, shows the application of L-system in envelope design (El-Khaldi, 2007). They consist of a grammar that includes an axiom, which is expanded into complex strings through defined rules Table 1, (Ashlock, Gent, and Bryden, 2005). They can visualize complex design patterns, such as those found in urban planning (Yu and Min, 2022).

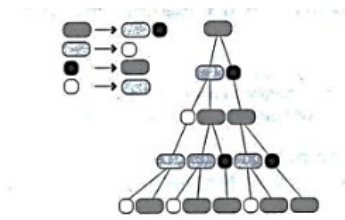


Figure 11. Tree network, L system (Prusinkiewicz and Lindenmayer, 2012).

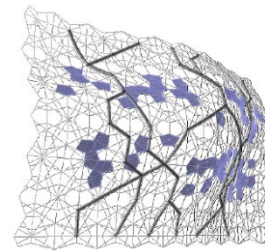


Figure 12. Application of L system in envelope design (Alfaris, 2009).

Table 1. The rules, the initial string and subsequent generations (El-Khaldi, 2007).

Rule	R \rightarrow L	L \rightarrow RR
Initial string		R R L R
Generation 1		L L RR L
Generation 2		RR RR LL RR
Generation 3		LL LL RR RR LL

f. Generative design system (Fractal)

The creation of natural shapes by architects relies heavily on geometric notions (Pérez García and Gómez Martínez, 2010). Euclidean geometry is limited to smooth curves and surfaces, which do not reflect the irregularities present in natural objects (Banerjee, Easwaramoorthy, and Gowrisankar, 2021). Fractal geometry is suitable for developing nature-inspired architectural designs (Mandelbrot, 1982). Fractals are complex geometric forms, pushing the boundaries of traditional architectural design (Ediz and Çağdaş, 2007) with self-similarity, meaning that its constituent parts are similar to one another. Self-similarity allows for the replication of patterns at different scales, which can be observed in historical architectures like Gothic cathedrals and Indian temples (Lorenz, 2011). To create a fractal, you must specify an initializer and rules for replacing copies of the initializer with smaller versions (El-Khaldi, 2007). Fractal geometry has been used in a variety of fields, including the natural sciences (Peitgen et al., 2004; Contini, 2007), engineering (Leung, Wu, and Zhong, 2004) and in medicine (Bankman, 2008). Fractal geometry is used in architecture to visually view buildings (Bovill and Bovill, 1996; Ostwald, 2001; Rian et al., 2007) and cities (Batty and Longley, 1994). Fractals can create new aesthetics (Patuano and Tara, 2020). Greg Lynn used fractals to create the Cardiff Bay Opera House (Addison, 1997). They lack a smallest unit because they are based on recursive models because they iteratively decompose components and replace them with new algorithms (El-Khaldi, 2007). Fractal geometry has been used to develop and study properties of innovative planar truss configurations (Rian and Sassone, 2014) and created new free and complex shell structures (Stotz, Gouaty, and Weinand, 2009; Vyzantiadou, Avdelas, and Zafiropoulos, 2007). The Sierpinski triangle (Ettestad and Carbonara, 2018) and the Koch curve (Purnomo et al., 2019) are two well-known instances of fractal geometry (Figure 13,14,15). Albrecht Dürer's pentagonal tile pattern was an early example of fractal design Figure 16. In fractal systems such as the L system, inheritance is not possible because data is constantly replaced (El-Khaldi, 2007). A fractal system creates objects with similar components that appear at different sizes Figure 17. They have been used to create a porous roof Figure 18, filtered sunlight and allowed air circulation Sakai et al., (2012) and to design a non-smooth covering surface Figure 19, that can transmit sound (Cox and d'Antonio, 2016). They have been used in computer models of tree column topologies (Rian, Callegary, and Spinelli, 2015). However fractals and other mathematical concepts do not teach us how to create; Nevertheless, they can help improve the design process (Rian and Asayama, 2016). Digital tools facilitate the application of fractal geometry in design (Ediz and Çağdaş, 2007) and is increasingly used in architecture to create unique and structurally optimal designs, inspired by natural forms and mathematical principles (Mayatskaya et al., 2022).

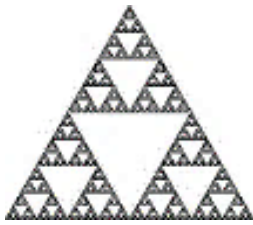


Figure 13. Sierpinski's triangle
Source: (El-Khaldi, 2007).

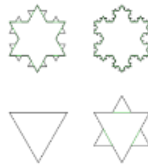


Figure 14. Koch curve Source: (El-Khaldi, 2007).

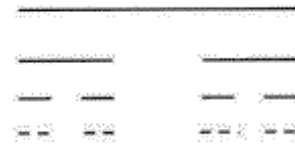


Figure 15. Contour set Source: (El-Khaldi, 2007).



Figure 16. Fractal pattern, Albrecht Dürer Source: (El-Khaldi, 2007).

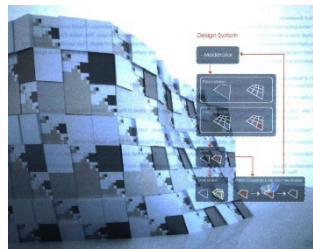


Figure 17. Fractal envelope
Source: (El-Khaldi, 2007).



Figure 18. Fractal pattern roof
(Rian and Asayama, 2016).



Figure 19. Non-smooth covering fractal surface (Sakai et al., 2012).

g. Generative design (Shape Grammar)

Shape Grammar (SG) is a generative design system (Caetano et al., 2020). Stiny and Gips were pioneers in this discipline (Stiny and Gips, 1971). It integrates visual observation with computational processes, enabling designers to engage in a form of "visual calculating" that enhances creativity (Stiny, 2022) and they are series of recursive transformations performed on an original shape to produce new shapes (Toussi, Etesam, and Mahdavinjad, 2021). SG develops an endless number of designs with just a few rules and it can decompose complex structures into basic components and create complex shapes from simple shapes (Stiny and Gips, 1971). Figure 20, demonstrates a SG rule (El-Khaldi, 2007). By combining SG with parametric

design, designers can simulate energy performance, (Granadeiro, Duarte, and Palensky, 2011). SG enable inheritance because replacement rules can be applied to certain parts of components while leaving others without transformation (El-Khaldi, 2007). Stiny categorizes units as point, line, plane and solid Table 2, (Stiny, 2006). Stiny and Mitchell used a parametric SG to create Palladio's villa designs (Tepavčević and Stojaković, 2012). Furthermore SG was used to analyze Frank Lloyd Wright's houses and the vernacular Japanese teahouses, traditional Taiwanese houses, Mongolian garden (Chiou and Krishnamurti, 1995; Stiny and Mitchell, 1980; Knight, 1981). SG could describe the historical development of styles in the creation of new design (Knight, 1981). SG is used to optimize daylight in the building envelope (Ashrafi and Duarte, 2017). Truss structures were created using performance-based optimization and SG (Haakonsen, Rønnquist, and Labonnote, 2023). It was also used to generate compositions Figure 21, (Eilouti, 2019). The Gothic minaret was designed using SG Figure 22, (Knight, 2000) and Frank Gehry used SG algorithms to justify envelope manufacturability Figure 23, (Shelden, 2002). City Engine, a software program that uses SG to autonomously create models based on a set of rules is creating virtual cities using 2D road networks (Müller et al., 2006). The Figure 24, shows how the program was used to create photos of Pompeii (Tepavčević and Stojaković, 2012). In architecture schools they are often used in design lessons (Haakonsen, Rønnquist, and Labonnote, 2023). CD systems gives users a tool to achieve goals (Haakonsen, Rønnquist, and Labonnote, 2023). The concept of a system is important in CD (Alfaris, 2009). This concept is examined in the next part.

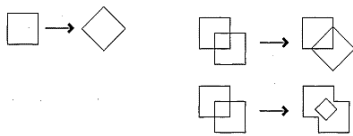


Figure 20. An example of a shape grammar (Stiny, 2006)

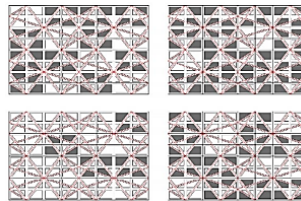


Figure 21. A shape grammar composition (Knight, 2000)

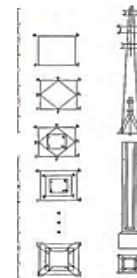


Figure 22. Minaret design (Knight, 2000)

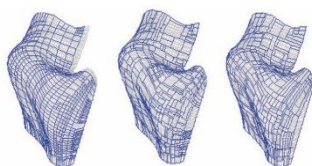


Figure 23. Shape Grammar envelope (Shelden, 2002)



Figure 24. Modeling the Pompeii city using shape Grammar software (Tepavčević & Stojaković, 2012)

Table 2. Shapes grammar units (Stiny, 2006).

Basic element	Dimension	Boundary	Content
Point	0	None	None
Line	1	Two points	Length
Plane	2	Three or more	Area
Solid	3	Four or more	Volume

System concept and computational design

The system concept influenced the architectural design and CD system. System is a collection of units and institutions that work together to achieve a common goal (Schmidt and Taylor, 1970). They have systematic structure (El-Khaldi, 2007). In these systems, components are hierarchically structured and interact to achieve goals such as envelope design and energy optimization (Granadeiro et al., 2013). As an example, the bottom-up approach of CA allows for self-organization, where local interactions lead to global patterns (Liu, Corcoran, and Feng, 2020) or in algorithmic design systems, multiple components described by rules, work together to achieve design goals (El-Khaldi, 2007). CD systems were used to create an integrated architectural design and Subsystems have an interaction with each other (Alfaris, 2009; Fasoulaki, 2008). In this status there is a balance between shape exploration and performance, for example it can be used to design high-rise building based on structural, lighting, zoning and aesthetic criteria (Fasoulaki, 2008).

A. The concept of the system and its components

The concept of system has penetrated to architectural design and CD systems (Alfaris, 2009). The Cambridge dictionary defines a system as “a set of connected things or devices that operate together.” Systems thinking encourages a holistic view, allowing architects to consider interactions within complex environments, leading to more effective design outcomes (Furtado, 2012). Systems are characterized by their goals such as service-oriented (airport, stadium), product-oriented (car factory) and process-oriented (oil refinery) (DAG and Ethic, 2000). Systems consist of numerous components that can adopt various configurations (Wong et al., 2023). Architectural research is concerned with systems in design to design the building envelope and predict energy consumption (Granadeiro et al., 2013). Systems analysis can help us better understand goals, constraints, risks, costs, opportunities and resources (Alfaris, 2009). System characteristics include integration, correlation, input/output, hierarchy, interaction, change and adaptability (Littlejohn and Foss, 2010). System tasks are completed in response to inputs (Papalambros and Wilde, 2000). The system's ability to form patterns is not solely dependent on local interactions but also on the broader context of the system's environment (Middya and Luss, 1994). Diagram 1, shows the boundary of the office building system as influenced by its

surroundings and environmental components (site conditions, soil quality, weather and urban environment) and it affects energy consumption, structural stability and working conditions (Alfaris, 2009).

The behavior of a system varies over time and System status is a set of variables that represent a specific characteristic of the system (DAG and Ethic, 2000). For example, variables such as aircraft waiting times and available parking spaces can be used to monitor the airport system (Alfaris, 2009). Effective variable selection should ensure that the chosen variables are relevant and significant (Mulaik, 2009). For example, if an airport wants to improve the passenger experience, parking lot modeling may be necessary. However, parking spaces may not improve safety (DAG and Ethic, 2000). Every system has an architecture that determines its behavior (Whitney et al., 2004). Hierarchy is a key concept in systems and Diagram 2, shows a hierarchical organization of system (Alfaris, 2009). Complex systems are typically organized into layers, where each level represents different scales and interactions among components (Wu, 2013). Hierarchy enables inheritance, which means that traits are passed from parents to children (El-Khaldi, 2007). Each system can be a subsystem of a larger system (Alfaris, 2009). The system concept distinguishes between two types of architectural artifacts: modular (Kazemi, 2019) and integrated (Miraglia, 2014). In the modular architecture Diagram 3, function and physical elements are inextricably linked (Eppinger and Ulrich, 1995) and each component can be developed separately. In integrated architecture Diagram 4, the connection between function and physical elements is complicated (Ulrich, 1995). It is difficult to determine the mutual impact of components on performance (Ulrich and Eppinger, 2016). Integrated systems prioritize a cohesive design that enhances operational efficiency (Miraglia, 2014). While some theories advocate modular architecture, real-world examples show that designs with integrated functions can achieve greater success and goals (Ulrich and Seering, 1990; Whitney, 1996). As Figure 25, shows, the modular design of a nail clipper does not always outperform an integrated nail clipper Figure 26, (Ulrich, 1995). In addition, Figure (27,28), illustrates two types of building envelopes (modular and integrated envelope).

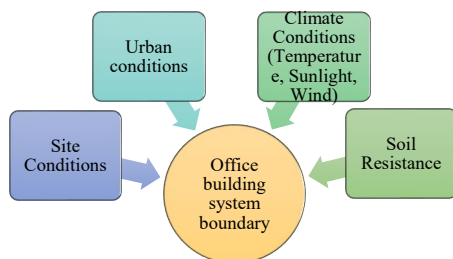


Diagram 1. The boundary of the office building system
(Source: authors).

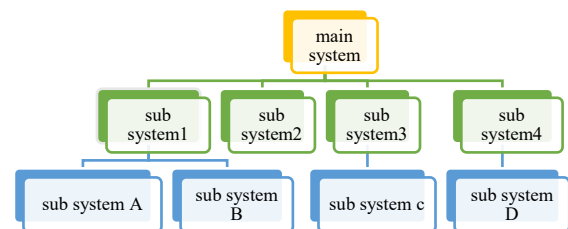


Diagram 2. Hierarchy in system (Alfaris, 2009).

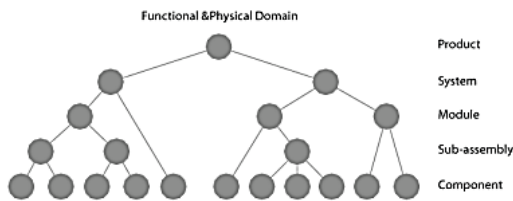


Diagram 3. Hierarchy and connection of physical elements in modular architecture (Alfaris, 2009).

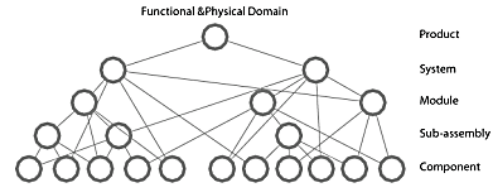


Diagram 4. Hierarchy and connection of physical elements in integrated architecture (Alfaris, 2009).

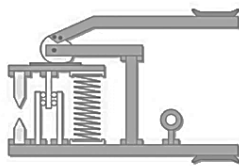


Figure 25. A Modular nail clipper (Ulrich, 1995).

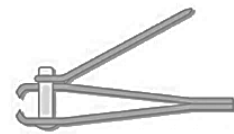


Figure 26. An integrated nail clipper (Ulrich, 1995).

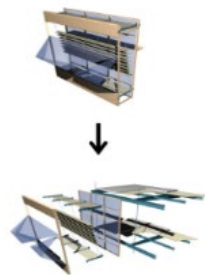


Figure 27. Modular façade design (Alfaris, 2009).

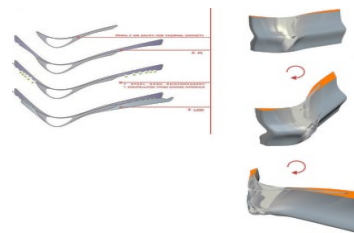


Figure 28. Integrated façade design (Alfaris, 2009).

Computational design process

In the late 1950s and 1960s, design models were created to reflect creative problem solving in design through phases such as synthesis, analysis, and evaluation (Alfaris, 2009). The design process involves a sequence of analytical, synthetic, and evaluative steps, allowing for iterative problem-solving and solution refinement (Vande Zande, 2006). Prescriptive models provide guidelines for implementation to achieve specific goals in design processes (Fernstrom, 1988) have algorithmic or systematic structure (Alfaris, 2009) and descriptive models capturing the actual process and patterns, identifying innovation opportunities (Zhang et al., 2012). Teaching prescriptive modeling alongside descriptive techniques enhances students' ability to implement design intent effectively (Gogolla and Selic, 2020). CD systems have algorithmic or systematic structure (Alfaris, 2009). Prescriptive models include the Archer (Archer, 1984), Eggert (Eggert, 2005), Asimov (Asimow, 1962), Marcus (Markus, 1969), and Mawer (Maver, 1970) models. Various prescriptive design models provide organized methods for the design process. These models outline the stages of a project from inception to completion, ensuring systematic progress

and quality control (Maher, 1990). Archer 1984, identified six design tasks Diagram 5, including programming, data collection, analysis, synthesis, development, and communication. The Eggert 2005 model Diagram 6, is divided into four phases: formulating problem, generating alternatives, analyzing alternatives, and evaluating alternatives (Eggert, 2005). As another prescriptive model, Asimov's model Diagram 7, is vertically structured and extends from needs description to production, including feedback loops to monitor and resolve difficulties. Asimov's horizontal model consists of repeated decision cycles: analysis, synthesis, evaluation and communication (Asimow, 1962). Marcus and Mawer's design model Diagram 8, provides a decision-making sequence that includes analysis, synthesis, evaluation, and decisions at various design levels (outline proposal to detail design) (Markus, 1969; Maver, 1970). MIT researchers proposed a Prescriptive performance-based CD process consisting of six phases including decomposition, formulation, synthesis, analysis, evaluation and optimization Diagram 9. Decomposition as a first step, breaking the problem into components. Formulation (the second phase) identifies component relationships (Alfaris, 2009). Alexander initiated the study of these ideas (Chermayeff and Alexander, 1963). Synthesis assembles recognized components according to desired principles and uses CD systems and offers a variety of design solutions (Alfaris, 2009). Computational design synthesis is a research area focused on approaches to automating synthesis activities in design (Campbell and Shea, 2014).

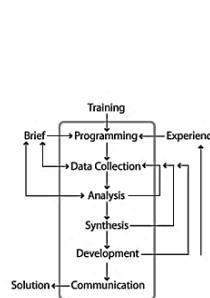


Diagram 5.
Archer's design
model (source:
Archer, 1984).

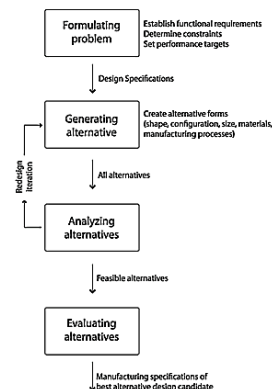


Diagram 6. Eggert
design model (source:
Eggert, 2005).

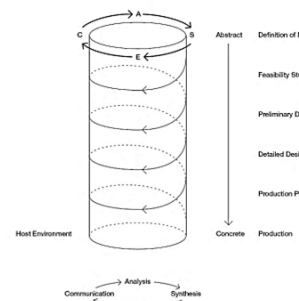


Diagram 7. Asimov
design model (source:
Asimow, 1962).

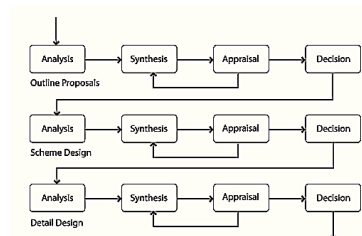


Diagram 8. Tom Marcus and
Tom Mawer's design model
(source: Markus, 1969;
Maver, 1970).



Diagram 9. Performance based computational design process (Alfaris, 2009).

Decomposition

In computational contexts, tasks can be broken down into sequential sub-tasks, which can simplify the design process (Fried et al., 2018). Decomposition is essential since learning individual components leads to a greater comprehension of the whole system (Alfaris, 2009). Alexander, (1964) broke down design problems based on customer needs as a network Diagram 10. Vertices represent functional requirements, while edges illustrate their connections and the degree of interaction. Shorter edges mean more interactions (Alfaris, 2009). This grouping allows interactions to be mapped (Alexander, 1964b). Models such as decomposition help in structuring design knowledge, facilitating better problem-solving and innovation in design (Maher, 1990). Two hierarchical methods can be used in decomposition (1. tree hierarchy Diagram 11, and 2. network hierarchy Diagram 12). Decomposing 3D models into architectural elements enhances comprehension of their structure, allowing for better analysis and representation (Kobyshev et al., 2016) Figure 29, shows decomposition of school floor plan into sub-problems (environmental, structural and circulation sub problems) and Figure 30, shows how the outer envelope is decomposed into its components (Alfaris, 2009).

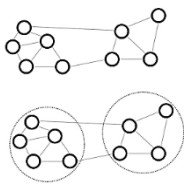


Diagram 10.
Decomposition of a
problem (Alexander,
1964a).

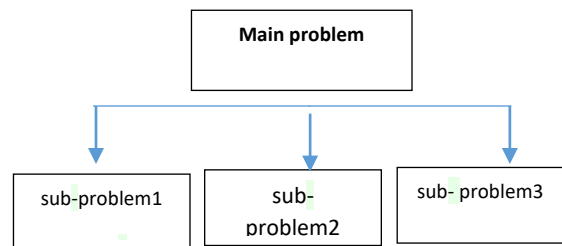
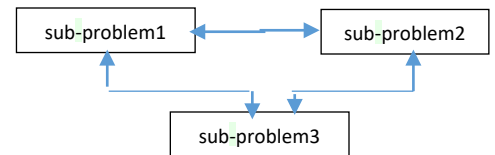
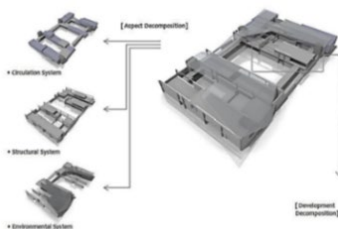


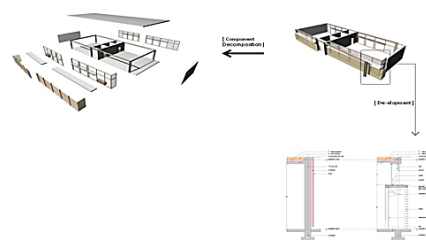
Diagram 11. Decomposition (tree hierarchy)
(Alfaris, 2009).



**Diagram 12. Decomposition (network
hierarchy) (Alfaris, 2009).**



**Figure 29. Decomposition of the school plan (Alfaris,
2009).**



**Figure 30. Decomposition of the (source: Alfaris,
2009).**

Formulation

The next step (formulation) in the CD model is to understand the relationships between the various components Diagram 13 (Alfaris, 2009). CD utilizes mathematical languages to define relationships between components, enabling sophisticated design processes that are otherwise unattainable (Koyama, 2021). Components in CAD systems are often sized and positioned based on their relationships with other components, ensuring that designs are coherent and functional, which necessitates a clear understanding of these relationships (Amadon, Rajkumar, and Kumar, 2021). Chermayeff and Alexander (1963) pioneered structural formulation techniques and they outlined the links between these problems Diagram 14. Alexander focused on patterns in his work Pattern Language (Alexander, 1977). Design Structure Matrices (DSM) is used in systems engineering to represent component interactions (Samson and Peterson, 2010). Diagram 15, shows an activity-based DSM for the creation of a soda bottle (McCord 1993). Reading across rows identifies the other activities on which a given action depends for information. Black squares represent the transmission of information or activity interdependence (Grady, 1994). The interactions that occur in DSM differ from one project to the next and provide a taxonomy for system element interactions based on four categories: spatial, energy, information, and material (as illustrated in Table 3). They also provide a quantification scheme for these interactions, where the square marks are replaced by numbers or colors Diagram 16, (Pimmler and Eppinger, 1994).

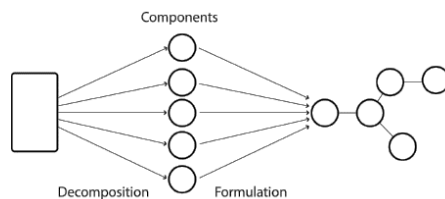


Diagram 13. Decomposition and formulation (source (Alfaris, 2009).

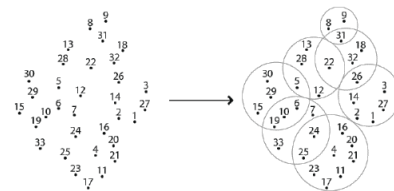


Diagram 14. Issues share many connections are grouped together (Chermayeff and Alexander, 1963).

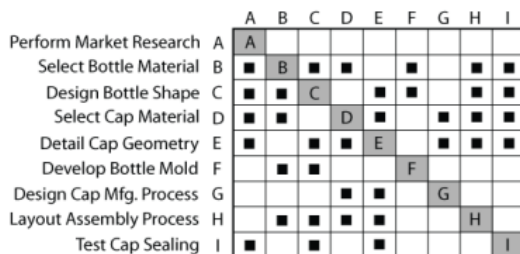


Diagram 15. An activity-based DSM for a soda bottle (McCord 1993).

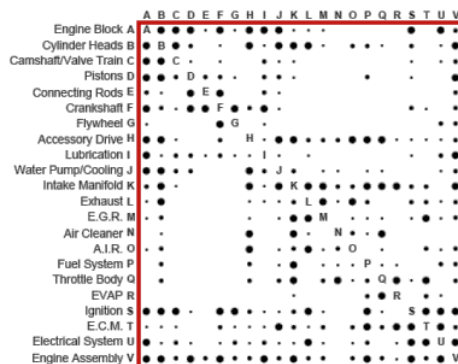


Diagram 16. DSM (Pimmler & Eppinger, 1994).

Table 3. Taxonomy for system element interactions (Pimmler and Eppinger, 1994).

Spatial	Associations of physical space and alignment; needs for adjacency of orientation between two elements.
Energy	Needs for energy transfer/exchange between two elements (e.g. power supply).
Information	Needs for data or signal exchange between two elements.
Material	Needs for material exchange between two elements.

Synthesis

Online Cambridge dictionary, defined synthesis as” the act of combining different ideas or things to make a whole that is new and different from the items considered separately.” Synthesis involves the use of abductive reasoning, which generate innovative concepts (Fei, 2019). It involves decisions about arrangement, connections, forms (Papalambros and Wilde, 2000) and the creation of physical and informational structures (Suh, 1990). (Eder, 2009) discusses the cyclical nature of design engineering, where analysis and synthesis are interlinked processes that inform the development of technical systems. Computational design synthesis has also championed the use of generative design grammars as a means to simultaneously provide structure and design freedom during synthesis (Campbell and Shea, 2014). Synthesis models should have a generative mechanism, typically performed using parametric or algorithmic descriptions (Alfaris, 2009). A design algorithm expresses a strategic approach to tractable problems or a stochastic search for intractable problems (Terzidis, 2006). The connection between form and performance should be included in the representation formalism. This provides restrictions on permitted designs and ensures that the rules discard designs that do not comply with constraints (Alfaris and Merello, 2008). Synthesis models require a geometric representation (Alfaris, 2009). Advances in function-based and analogy-based synthesis have expanded the range of potential solutions (Chakrabarti et al., 2011). As shown in the Diagram 17, the synthesis model uses the original design parameters to generate a variety of design solutions through internal operations. For example, in a curve or surface equation, parameters can be changed to represent a family of curves or surfaces Figure 31, (Alfaris, 2009). In the conceptual design phase, synthesis is key to explore and define design concepts. It allows architects to invent transitions that lead to the description of artifacts (Kotsopoulos, 2005; Hartmann et al., 2018). By generating a wide range of design alternatives, synthesis supports the innovation process and enhances creativity in architectural design. It allows for the exploration of new forms and solutions, contributing to the evolution of architectural practices (Helms and Shea, 2012). Relationships (enable communication between the components), constraints (conditions that must

be met) and rules (to verify the logic) determine the behavior of the synthesis model. Methods such as L-systems, CA and SG can be considered for capturing design relationships (Alfaris, 2009).

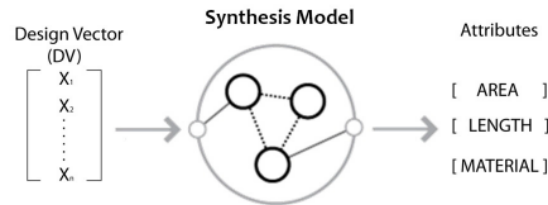


Diagram 17. Expected input and output of the synthesis model (Alfaris, 2009).

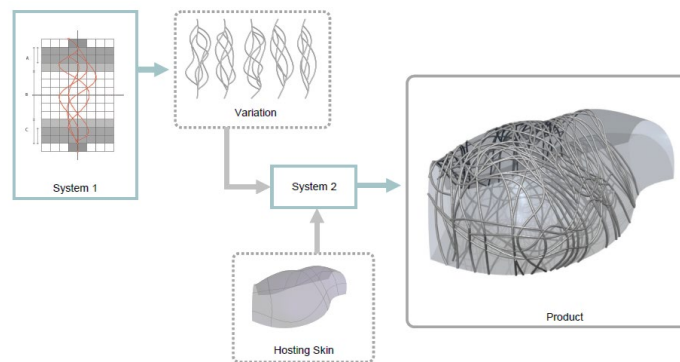


Figure 31. Parametric equations in geometry define curves (Alfaris, 2009).

Analysis, Evaluation and Optimization

The analysis model determines the behavior associated with each design and the evaluation model attempts to take into account the multi-objective criteria of the design problem. Optimization models are then used to determine the best designs (Alfaris, 2009). Online Cambridge dictionary, defined analysis as “the act of studying or examining something in detail, in order to discover or understand more about it.” Alexander, (1964) describes analysis as determining how effectively a solution achieves its stated goals. Design challenges sometimes involve numerous disciplines, each with its own analytical model. For example Figure 32, (Averill, 2006), Figure 33, shows analysis models for examining the quality of light in different spaces and the air flow around the building (Alfaris, 2009). Analysis models have different input requirements and output accuracies Diagram 18, (Alfaris and Merello, 2008). Outputs can include energy efficiency, structural integrity, and environmental impact, which are derived from the analytical model's computations (Mitchell and Molloy, 2020). Analytical models are formal representations that support reasoning and understanding in design processes (Jackson, 2009). In architectural education, analytical models assist students in grasping design principles by organizing elements hierarchically (Azmy, 2010). These models are represented in abstract

mathematical form by variables, parameters, equations and algorithms (Jacoby and Kowalik, 1980). Evaluation models make it easier to select good design by creating and comparing alternatives Diagram 19, (Alfaris, 2009). Online Cambridge dictionary defined evaluation as “the process of judging or calculating the quality, importance, amount or value of something.” Real-world problems sometimes involve multiple, possibly conflicting goals. This results in a collection of equivalent solutions rather than a single optimal solution (Abraham and Jain, 2005). Evaluation models aid decision making in multi-objective design challenges. If decision-making is delayed, the evaluation model becomes part of the optimization process (Alfaris, 2009). Online Cambridge dictionary defined Optimization as “the process of making something as good or effective as possible.” The chosen solution is determined by additional restrictions or objective functions that integrate the search goals (Gries, 2004). Gauss invented algorithm, which gave rise to the term optimization. It serves as the basis for the science of optimization (Gray, 2018). Optimization requires identifying performance criteria to maximize or minimize, such as cost or efficiency, while adhering to constraints like physical laws and manufacturing limitations (Lam and Chen, 2019). Optimization is the process of refining or fine-tuning a design or system based on one or more performance criteria (Papalambros, 2000). The optimization process is continuous, as architects must adapt designs based on evolving requirements and feedback throughout the project lifecycle (Davis, 1997). An optimization model generates a new design vector, which is then used as input to the synthesis model Diagram 20, (Alfaris, 2009). Multiple objectives in design can be inherently conflicting, such as minimizing control effort (Sardahi, 2016). This shows that there are numerous optimal solutions and not just one model. Multi-objective optimization integrates functional constraints, such as accessibility, into layout designs, ensuring that components are both operational and maintainable (Song et al., 2023).

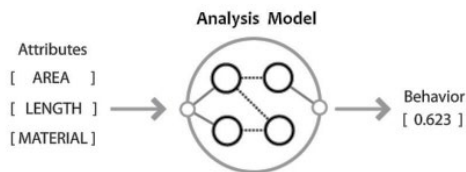


Diagram 18. Expected input and output of the analysis model (Alfaris, 2009)

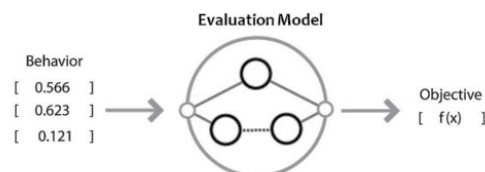


Diagram 19. Expected input and output of the evaluation model (Alfaris, 2009)

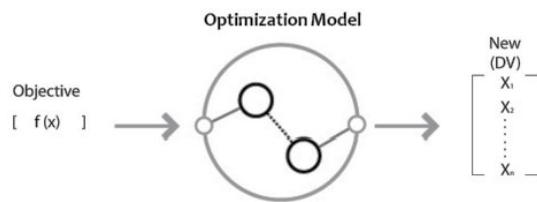


Diagram 20. Expected input and output of the optimization model (Alfaris, 2009)

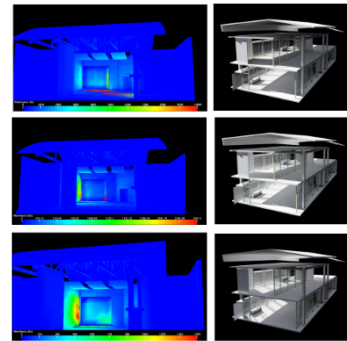


Figure 32. Analysis model for daylight to assess the quality of light in different spaces (Averill, 2006)

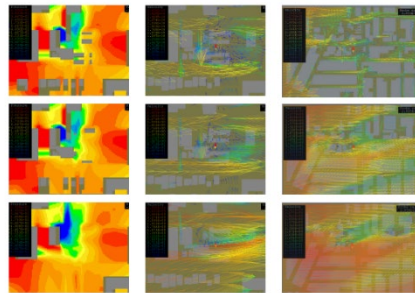


Figure 33. Analysis model to study of air flow around the building (Alfaris, 2009).

Methodology

The purpose of this research is to propose a comprehensive theoretical CD training framework. The selected method is based on the Design Research Method (DRM) by (Blessing and Chakrabarti, 2009), and research method conducted by (Vazquez, 2024) that merely provided a restricted program in parametric design training. However comprehensive CD training program had not been presented in that research. The DRM method is a framework in four stages: 1. Research Clarification 2. Descriptive Study I 3. Prescriptive Study and 4. Descriptive Study II. This paper proposed 3 research stages and the prescriptive study is followed by a second descriptive study that aims to implement and test the proposed approach will be done in further studies.

Research clarification and descriptive study which consists of a literature survey, followed by a prescriptive study, in which an instructional method is proposed. The descriptive study, is conducted through literature review on CD training and CD knowledge studies. The survey is conducted by searching in several databases with the following terms: (“teaching method” OR “pedagogical approach” OR “teaching strategy”) AND (“digital design” OR “computational design” OR “parametric design” OR “generative design” OR “algorithmic design”). After identifying the main articles in the area, an analysis was conducted. The outcome of the

descriptive study revealed that there is no specific comprehensive CD theoretical training framework prior to its usage in the design studio, and each study focused on some aspects of CD and concentrated on the use of software and coding (Austin and Qattan, 2016; Ostrowska-Wawryniuk, Strzala, and Słyk, 2022; Shtepani and Yunitsyna, 2023). However, learning CD necessitates theoretical knowledge (Caetano et al., 2020) that extends beyond software and programming. Additionally library resources analyzed and important topics in this field identified (DD and CD concepts, CD systems, system concept, CD system elements and CD design process (Caetano et al., 2020; Michelle and Gemilang, 2022; El-Khaldi, 2007; Alfaris, 2009; Fasoulaki, 2008). Finally in third stage (prescriptive study), with the goal of overcoming the deficiencies of the current educational program, this research will propose a comprehensive knowledge-based training framework. The training framework consists of two phases: 1. Learning CD principles 2. Learning an analysis of CD principles. In the first phase, topics such as DD and CD definition, types of CD systems (Algorithmic, Parametric and Generative design), system concept, prescriptive models and CD process were examined. Understanding CD concepts is necessary but not sufficient. Therefore, in the second phase, CD systems and their components were analyzed to identify differences. Additionally, integration of prescriptive models and CD process were examined. Finally training framework is proposed. Research methodology is presented in Diagram 21.

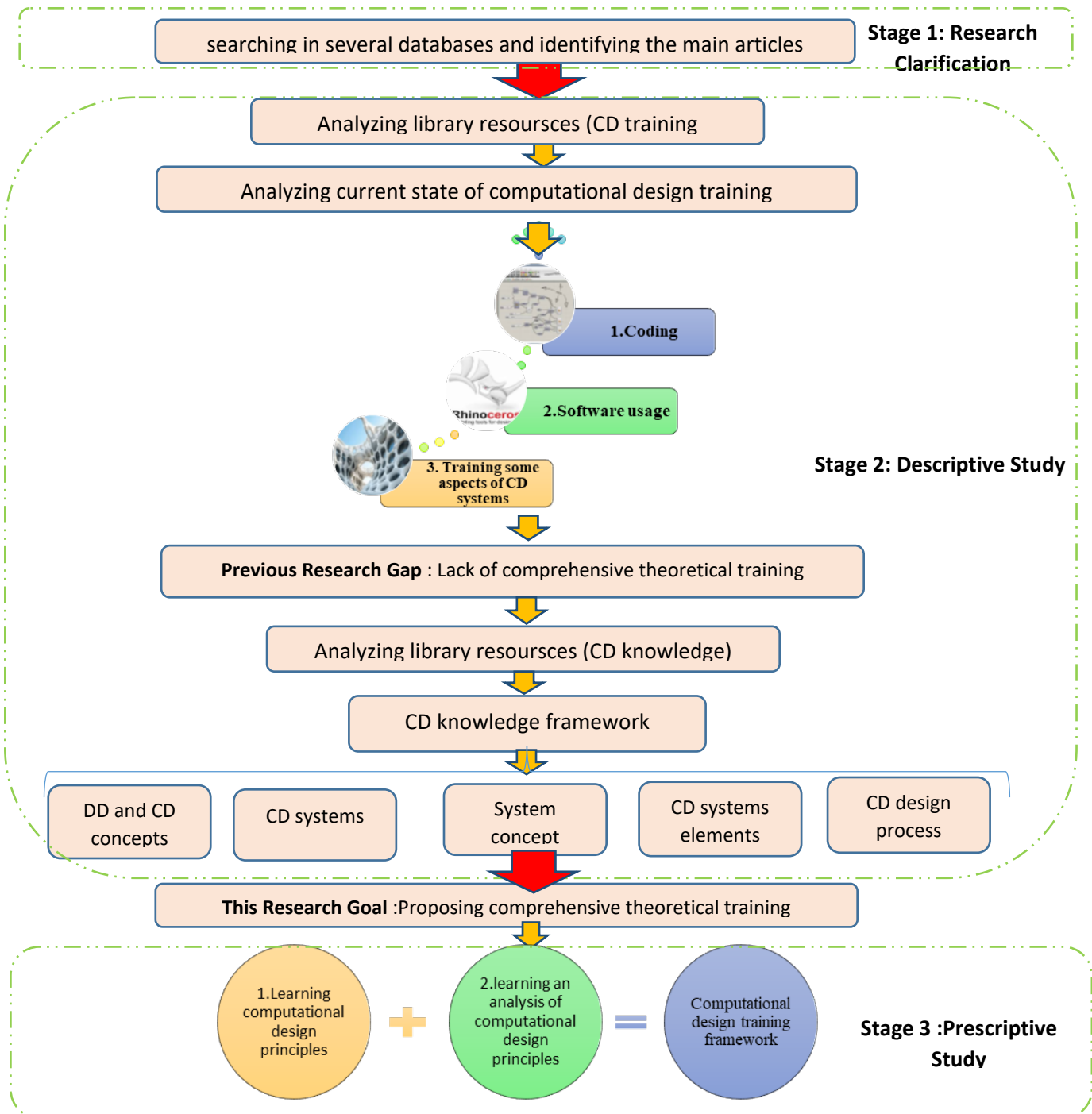


Diagram 21. Research methodology (analyzing current state and proposing CD training framework)
(Source: Authors).

Results and Discussion

Research clarification and Descriptive study

Analysis of the current state of CD education and CD concepts

According to the previous research (Vazquez, 2024; Ostrowska-Wawryniuk, Strzała, and Słyk, 2022; Lakhanpuria and Naik, 2023), as mentioned in Table 4, CD was partially taught in the design studio, and the emphasis of these studies were on coding and software usage. A review of internal articles (Mahmoudi and Naghizadeh, 2010; Poursistany et al. 2016; Asefi and Imani, 2017; Ahmadi Tabatabaie and Moosav, 2024), revealed that only the impact of information technology and digital tools on architectural education have been examined and comprehensive CD training program had not been studied. However architecture students should have a thorough understanding of CD theory before applying it (Caetano et al., 2020). Some previous research focused on GD (Bianconi and Filippucci, 2018) and PD (Lakhanpuria and Naik, 2023), however it is obvious that students need to be familiar with AD (as fundamental concept) (El-Khaldi, 2007). A review of Research like (Caetano et al., 2020), (Michelle and Gemilang, 2022)) demonstrates the importance of CD and DD concepts. However, research in the Table 4, shows that CD training research has generally focused on AD, PD, and GD methodologies, and the underlying concept of CD itself has received little attention. CD methodologies have a systematic framework ((El-Khaldi, 2007; Alfari, 2009; Fasoulaki, 2008). However, this concept has not been considered in previous education research. Additionally there are differences in CD systems, as evidenced by CD-related research such as (Michelle and Gemilang, 2022; El-Khaldi, 2007; Fasoulaki, 2008) and previous CD training studies did not address distinctions. Prescriptive models such as Archer (Archer, 1984) have been proposed and demonstrate the importance of the design process. The architectural theorists, Alexander and Chermayeff studied design stages such as decomposition and formulation (Alexander, 1964a; Chermayeff and Alexander, 1963). Academic research, particularly at MIT, has also proposed a model for the CD process, which includes steps such as decomposition, formulation, analysis, evaluation and optimization (Alfari, 2009; Alfari and Merello, 2008). But these concepts have been missed in previous CD training research described in the Table 4. Based on the findings and in order to fill research gap (the lack of a comprehensive CD training program), a comprehensive theoretical training framework on Learning CD principles has been proposed in this research. These concepts have been incorporated and examined in the upcoming section of the curriculum and main CD concepts collected from literature analysis represented in Table 5.

Table 4. Previous (CD) training research (Source: Authors).

Research	CD type	Research	CD type	Research	CD type	Research	CD type
1 (Fischer, 2002)	GD	7 (Yavuz & Çelik, 2014)	GD	13 (Lakhanpuria & Naik, 2023)	PD	19 (Vazquez, 2024)	PD and AD
2 (Karzer & Matcha, 2009)	PD	8 (Huang & Xu, 2015)	GD	14 (Nazidizaji & Safari, 2013)	AD	20 (Mahmoudi & Naghizadeh, 2010)	Information technology
3 (Gürbüz, Çağdaş, and Alaçam, 2010)	GD	9 (Agkathidis, 2015)	GD	15 (Austin & Qattan, 2016)	AD	21 (Eynifar & Hosseini, 2014)	Digital technology
4 (Guidera, 2011)	GD and PD	10 (Abdelmohsen et al., 2017)	GD	16 (Peteinarelis & Yiannoudes, 2018)	PD and AD	22 (Poursistany et al. 2016)	Digital technology
5 (Knight, 2012)	GD	11 (Bianconi & Filippucci, 2018)	GD	17 (Vrouwe, et al., 2020)	PD	23 (Ahmadi Tabatabaie & Moosav, 2024)	Teaching software
6 (Abdelmohsen, 2013)	GD	12 (VAZ & CELANI)	GD	18 (Agirbas, 2022)	PD		

Table 5. CD concepts in pervious CD research

CD and DD concepts	Cd and DD concepts and their distinctions	(Michelle & Gemilang, 2022), (Caetano et al., 2020)
CD systems	CD systems (AD, PD, GD) concepts and their application	(Michelle & Gemilang, 2022), (Caetano et al., 2020), (El-Khaldi, 2007); (Fasoulaki, 2008)
System concept	System concepts and its application in CD, system components	(Alfaris, 2009), (Alfaris & Merello, 2008)
CD systems elements	Units, smallest units, rules, inheritance, algorithm execution	(Caetano et al., 2020), (El-Khaldi, 2007), (Fasoulaki, 2008)
CD design process	Prescriptive models and CD design process	(Alexander, 1964a), (Chermayeff and Alexander, 1963)

Prescriptive study (Training program)

a. First training phase (Learning computational design principles)

This research training program has two phases (1. Learning CD principles 2. Learning an analysis of CD principles). During the initial training phase Diagram 22, it is critical to grasp the CD principles, such as definitions of CD and DD (Caetano et al., 2020; Michelle and Gemilang, 2022), recognition of all types of CD systems (El-Khaldi, 2007; Michelle and Gemilang, 2022; Fasoulaki, 2008). AD requires knowledge of algorithms, decomposition, and how to propose solution for each part (Terzidis, 2004; Fried et al., 2018). PD requires knowledge of parameters,

variables and algorithms (Gu, Yu, and Behbahani, 2021). GD requires an understanding of algorithms and the recognition of generative systems (Caetano et al., 2020). CD systems have systemic structure (Alfaris, 2009; Alfaris and Merello, 2008) and understanding its concept and components is critical. Recognizing design models, especially prescriptive models and CD design process (Alfaris, 2009; Alfaris and Merello, 2008), is also necessary in the first phase of training.

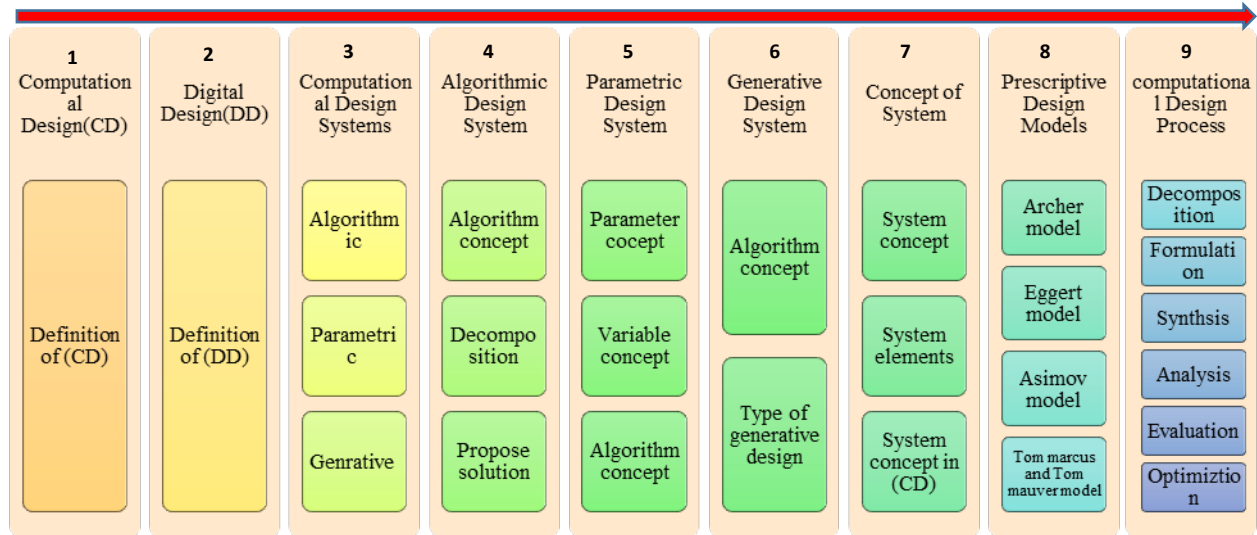


Diagram 22. First training phase (Source: Authors).

b. Second training phase (learning an analysis of computational design principles)

The second phase of CD training Diagram 23, is the analysis of CD systems. Studies such as (Caetano et al., 2020) examined CD and differentiated it from DD. As a result, knowing this distinction is critical as the first step. Furthermore, research like (El-Khaldi, 2007; Michelle and Gemilang, 2022; Fasoulaki, 2008; Caetano et al., 2020) examined CD methodologies and their constituent elements and structures, emphasizing the distinctions between them. As a result, it is critical to familiarize students with these distinctions and their basic components (application, algorithm execution type, unit, rules, smallest unit and inheritance). CD follows a systematic structure (El-Khaldi, 2007; Alfaris, 2009). As a result, knowing the systematic structure of CD and comparing it to the concept of system is important in the following step. The next step is to consider integrating prescriptive design models with CD process (Alfaris, 2009) in order to produce a comprehensive design model. Both of them (prescriptive model and CD process) have a systematic and algorithmic framework and can overcome each other's shortcomings (Alfaris, 2009).

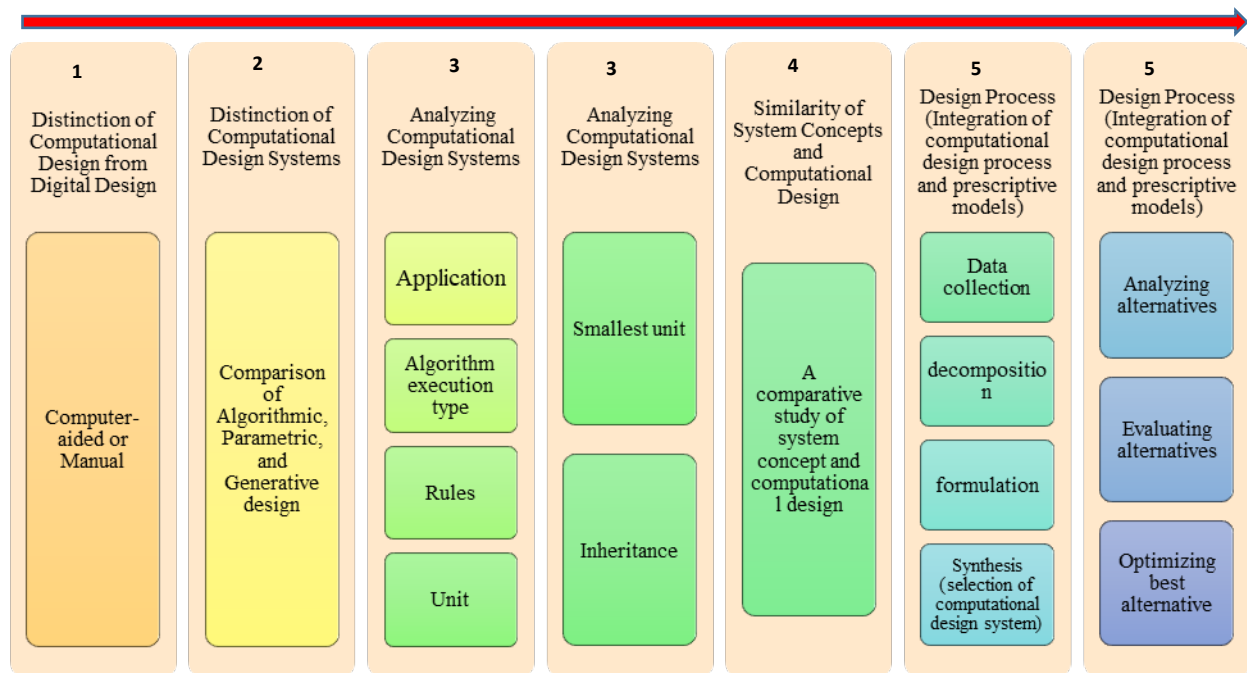


Diagram 23. Second training phase (Source: Authors).

c. *Proposing a comprehensive theoretical training framework*

By combination of first and second phase, CD training program is proposed Diagram 24. CD training program includes CD principles and their analysis. As Diagram 24, shows, Students should comprehend CD and DD and their distinction 2. CD systems (algorithmic, generative, parametric) and their distinctions 3. Analyzing CD systems (recognizing elements of CD systems such as hierarchy, inheritance, rules) 4. the concept of system and the similarity of system and CD 5. Design process (integration of Prescriptive Models and CD Process).

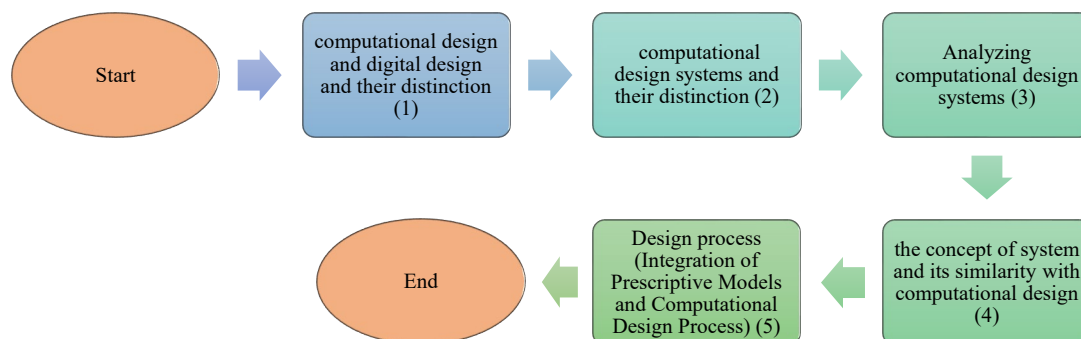


Diagram 24. Proposing a computational design training program (Source: Authors).

Computational Design, Digital Design, Computational Design Systems and their Distinction

DD and CD are widely used and Knowing their differences (Caetano et al., 2020) is helpful in understanding them better and using them in education. Digital design requires computer tools. CD requires calculations to develop designs and can be performed with or without computers Diagram 25, (Caetano et al., 2020). It is important to understand the differences between CD systems which are critical to maximizing their application. Algorithms are applied in algorithmic and generative design systems but in generative design system, the relationship between the algorithm and the output is difficult to discover. Parametric design systems are used when a number of parameters affect the final design and algorithms can be used in parametric design Diagram 26, (Caetano et al., 2020; El-Khaldi, 2007; Michelle and Gemilang, 2022; Fasoulaki, 2008).



Diagram 25. Distinction of (DD) and (CD) (Source: Authors).

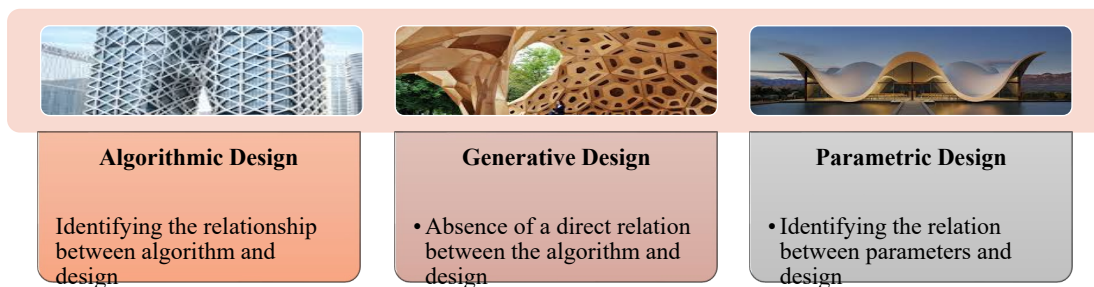


Diagram 26. Distinction between computational design systems (Source: Authors).

Analyzing computational design systems

The concept of decomposition can be used to examine the structure of CD systems. This may focus on the use of CD systems, rules, constituent units, the smallest unit, and inheritance (El-Khaldi, 2007). Various design problems are solved and simulated through algorithmic design (Terzidis, 2006) and parametric design (Schumacher, 2008; Tabadkani et al., 2019). Generative systems use L-systems (Prusinkiewicz and Lindenmayer, 2012), CA (Adem and Çağdaş, 2020), fractals (Mandelbrot, 1982; Patuano and Tara, 2020), and SG (Stiny and Gips, 1971; Tepavčević and Stojaković, 2012; Eilouti, 2019). CD systems are rule-based (Caetano et al., 2020). AD (Caetano and Leitão, 2021) and PD use numerous rules and generative systems use the substitution rules. Units used in CD systems are different. Algorithmic and parametric units are diverse, while L-systems and cellular automata use symbols. Fractals and shape grammar use

symbols, numbers and shapes. AD and PD use various smallest units, but L-systems, CA, and SG use alphabet, cell, and basic architectural elements. Fractals do not have a smallest unit due to substitution. Algorithmic and parametric design include inheritance, but L-systems, CA and fractals do not include inheritance. However, shape grammar can use replacement rules to change elements or parts while keeping the rest, allowing inheritance (El-Khaldi, 2007; Alfari, 2009; Michelle and Gemilang, 2022; Fasoulaki, 2008; Caetano et al., 2020). The characteristics of each system (AD, PD and GD) such as their implementation, rules, units, smallest unit and inheritance are presented in Diagram 27, 28.

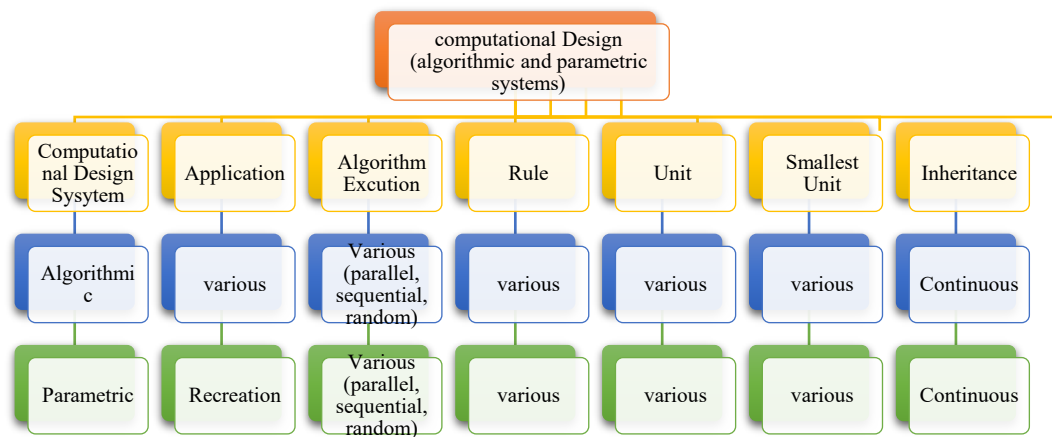


Diagram 27. Computational design systems analysis (algorithmic and parametric design system) (Source: Authors).

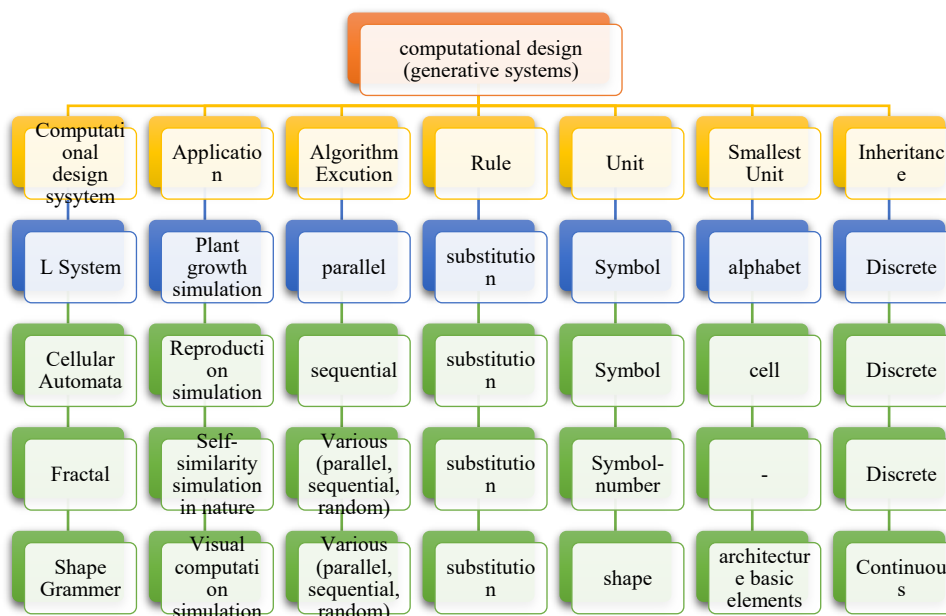


Diagram 28. Computational design system analysis (generative systems) (Source: Authors).

Concept of system and its similarity with computational design

Systems are collections of units and institutions that work together to achieve common goals. This concept could be used in CD systems (El-Khaldi, 2007, Alfari, 2009). As showed in Table 6, System and CD have comparable principles such as component relations, hierarchy, rules, and system execution (parallel, sequential, and random).

Table 6. Similarity of system and computational design (Source: Authors).

Concept	System	Computational design
Relation between components	✓	✓
Hierarchy	✓	✓
Rules	✓	✓
System execution (parallel, sequential and random)	✓	✓

Design process (Integration of Prescriptive Models and Computational Design Process).

The algorithmic and systematic structure of the CD process includes decomposition, formulation, synthesis, analysis, evaluation and optimization (Alfari, 2009; Alfari and Merello, 2008). In addition to the phases mentioned, the prescriptive models also include phases like planning, data collection Table 7. By combining prescriptive models and the CD process, shortcomings of these models can be minimized and a complete design process can be proposed Diagram 29. These steps are not sequential and can be performed and repeated as the designer considers (Alfari, 2009).

Table 7. Similarity of prescriptive model and computational design process (Source: Authors).

	Prescriptive models				Computational design process
	Archer	Eggert	Asimov	Tom Marcus and Tom Mawer	Computational design process
Programming	+				
Data collection	+				
Identifying needs		+	+		
Formulating the design problem (specifying goals and constraints)		+			
Feasibility studies	+				
Data analysis (decomposition)	+	+	+	+	+
Synthesis		+	+	+	+
Analysis of design alternatives		+	+	+	+
Evaluation of design alternatives		+	+	+	+
Optimization	+				+

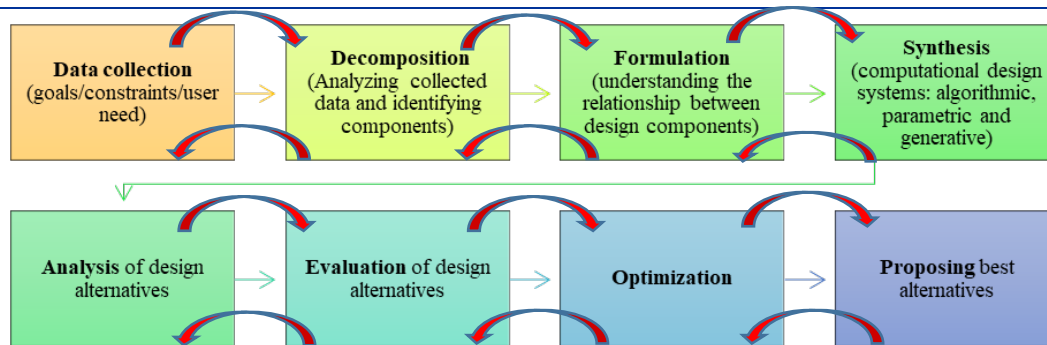


Diagram 29. Design process (combination of prescriptive models and computational design process)
(Source: authors).

Decomposition is an important concept in algorithms and CD systems. Decomposition can be used to decompose both the design product and the process. The Diagram 30,31, shows decomposition of the office building components (envelope, structure, space organization, facilities). The next step is to understand how they interact (formulation) (Alfaris, 2009). For example, when designing a sport stadium, the relationship between material, structure, envelope and space organization is crucial Figure 34, and Diagram 32. This makes it clear that the building does not consist of independent individual components, but is a networked system of subsystems in which all components interact with each other and create a mutual effect (Fasoulaki, 2008). The synthesis phase is crucial in CD because it combines components to provide design possibilities. By Using algorithmic, parametric and generative design methods, designers create a variety of alternatives (Alfaris, 2009). In contrast to traditional methods, CD offers a wider range of solutions (Agkathidis, 2015). The diagram shows an example of using a CD system to make design decisions in various areas such as structure, envelope and floor plan design Diagram 33.

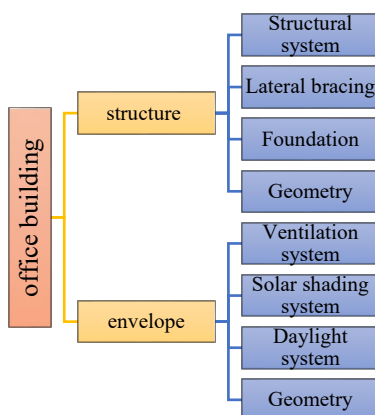


Diagram 30. Analysis of the envelope and structure of an office building (Source: Authors)

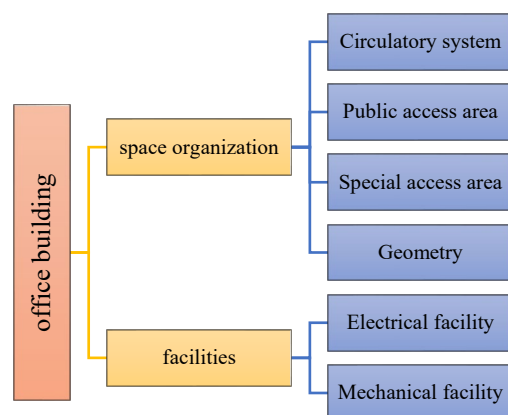


Diagram 31. Analysis of spatial organization and facilities of an office building (Source: Authors)

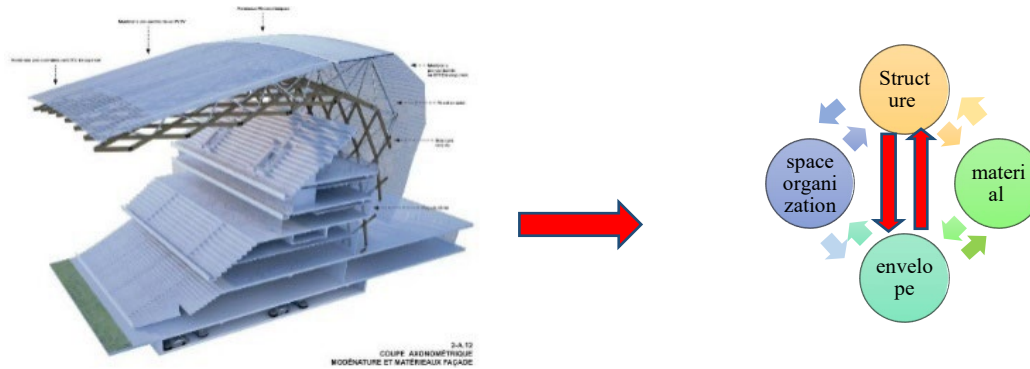


Figure 34. Allians Riviera stadium (Source: Archdaily).

Diagram 32. Relationship between structure, material, envelope and space organization in stadium design (Source: Authors).

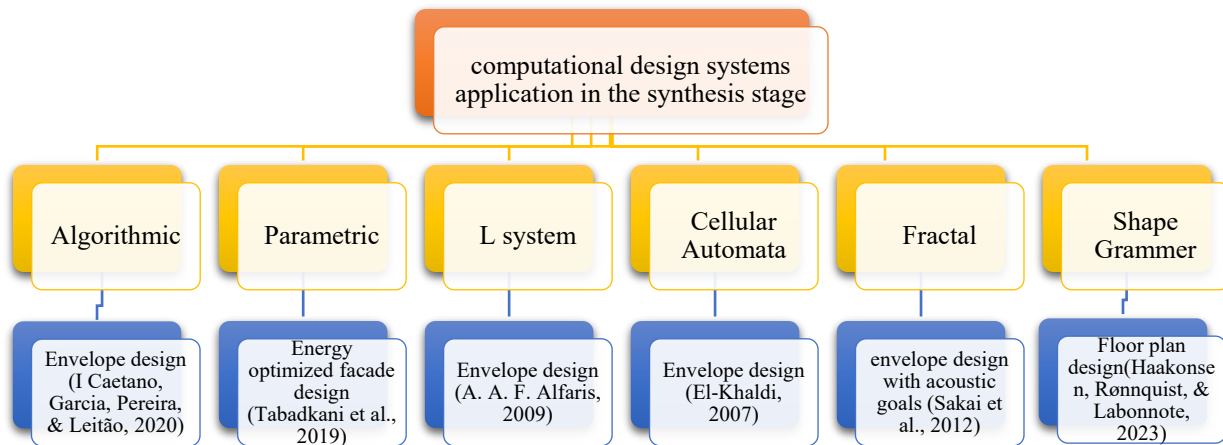


Diagram 33. Application of computational design systems in the synthesis phase (Source: Authors).

Before proceeding, all design decisions must be analyzed and evaluated (Alfaris, 2009; Alfaris and Merello, 2008). Students should be able to analyze and evaluate design criteria such as goals and restrictions. CD can be performed with or without a computer (Caetano et al., 2020). When analyzing a building, accurate revision of plans, elevations, sections, openings, site plans, installation problems, form analysis, function, structure, climate, acoustics, topography, analysis of economic, social and cultural factors, historical records, obstacles and legal restrictions should be taken into account (Ching, 2023). After analyzing and evaluating the design alternatives, the selected alternative may need to be optimized (Alfaris, 2009). Optimization can be used in a variety of goals including energy consumption, column spacing, and space organization. Training in analysis, evaluation and optimization is presented in Table 8.

Table 8. Training in analysis, evaluation and optimization in CD (Source: Authors).

1. Understanding the concept of analysis and evaluation
2. Ability to examine design constraints and objectives in design alternatives
3. Recognizing analysis and evaluation software
4. Understanding the concept of optimization
5. Ability to optimize the selected alternative based on design criteria
6. Recognizing optimization software

Conclusion

Computational design (CD) systems (Algorithmic, Parametric and Generative design systems) have been widely used in architectural education during the last decade. Examining previous research indicates that separate comprehensive framework for its training has not been proposed. Previous research focused on programming, the use of software and some aspect of CD systems in the design process. However, learning CD necessitates theoretical knowledge that extends beyond software and programming. An extra course on theoretical topics can improve its use. Therefore, this research proposes a framework for training theoretical knowledge in this field. In the first and second stage of the research (research clarification and descriptive study), the existing state of its training was reviewed and analyzed, and its deficiencies and shortcomings were identified through the use of library resources. Additionally important concepts for comprehending CD knowledge were identified, and in the third stage (prescriptive study), with the goal of overcoming the deficiencies of the current CD training, a comprehensive framework including two phases of 1. Learning CD principles 2. learning an analysis of CD principles is proposed. This framework consists of topics ranging from basic to advanced, including: 1. computational design and digital design and their distinction 2. CD systems (Algorithmic, Generative and Parametric) and their distinctions 3. Analyzing CD systems (recognizing concepts such as hierarchy, inheritance, rules and units in CD) 4. The concept of system and the similarity of system and CD 5. Design process (integration of Prescriptive Models and CD Process). This framework can gradually familiarize students with principles of CD systems and their analysis. Results of this study can be applied as a framework for CD training.

Author Contributions

All authors contributed equally to the conceptualization of the article and writing of the original and subsequent drafts.

Data Availability Statement

Not applicable

Acknowledgements

The authors would like to thank all participants of the present study.

Ethical considerations

The study was approved by the Ethics Committee of the Islamic Azad University, Sav.C. The authors avoided data fabrication, falsification, plagiarism, and misconduct.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare no conflict of interest.

References

- Abdelmohsen, S., Massoud, P., Tarabishy, S., & Hassab, A. (2017). Rule-based vs. intuition-based generative design: an inquiry into the digital chain concept in undergraduate architectural education. *International Journal of Parallel, Emergent and Distributed Systems*, 32(sup1), S199-S209.
- Abdelmohsen, S. M. (2013). Reconfiguring architectural space using generative design and digital fabrication: A project-based course. Paper presented at *the Proceedings of the 17th Conference of the Iberoamerican Society of Digital Graphics*.
- Abraham, A., & Jain, L. (2005). Evolutionary multi objective optimization. In *Evolutionary multi objective optimization (pp. 1-6)*: Springer.
- Adem, P. Ç., & Çağdaş, G. (2020). Computational Design Thinking through Cellular Automata: Reflections from Design Studios. *Journal of Design Studio*, 2(2), 71-83.
- Agirbas, A. (2022). A teaching methodology on the combination of architectural tradition and parametric design: A case study with birdhouses. *International Journal of Islamic Architecture*, 11(1), 149-168.
- Agkathidis, A. (2015). Generative design methods. In *Proceedings of eCAADe* (pp. 47-55).
- Ahmadi Tabatabaie, S. M. A., & Moosav, S. M. (2024). The Impact of Software Pedagogy on Architectural Creativity: Finding the Appropriate Method and Time for Teaching Software to Architecture Students. *Technology of Education Journal (TEJ)*, 18(1).
- Alexander, C. (1964a). *Notes on the Synthesis of Form* (Vol. 5). Harvard University Press.
- Alexander, C. (1964b). *Notes on the Synthesis of Form* (Vol. 5): Harvard University Press.
- Alexander, C. (1977). *A pattern language: towns, buildings, construction*. Oxford university press.
- Alfaris, A., & Merello, R. (2008). The generative multi-performance design system. *ACADIA 08 ò Silicon+ Skin ò Biological Processes and Computation*, 448-457.
- Alfaris, A. A. F. (2009). *Emergence through conflict: the Multi-Disciplinary Design System (MDDS)*. (Doctoral dissertation, Massachusetts Institute of Technology).
- Amadon, G., Rajkumar, P., & Kumar, M. (2021). *U.S. Patent No. 11,093,661*. Washington, DC: U.S. Patent and Trademark Office.
- Archer, L. B. (1984). Systematic method for designers. *Design*, 56-59.
- Ashlock, D. A., Gent, S. P., & Bryden, K. M. (2005). Evolution of l-systems for compact virtual landscape generation. In *2005 IEEE Congress on Evolutionary Computation* (Vol. 3, pp. 2760-2767). IEEE.
- Ashour, S. F., & Gogo, S. M. (2024). Boosting the Design Process Using a Proposed Methodology Based on Computational Design. *ERJ. Engineering Research Journal*, 47(2), 207-215.
- Ashrafi, N., & Duarte, J. P. (2017). A shape-grammar for double skin facades. *Sharing of Computable Knowledge*, 471.
- Asimow, M. (1962). Introduction to design. (*No Title*).

- Austin, M., & Qattan, W. (2016). I'M A visual thinker: Rethinking algorithmic education for architectural design. In *CAADRIA 2016, 21st International Conference on Computer-Aided Architectural Design Research in Asia-Living Systems and Micro-Utopias: Towards Continuous Designing*.
- Averill, M. L. (2006). Simulation Modeling and Analysis with Expertfit Software. In *USA: Mc Graw Hill International*.
- Azmy, A. M. (2010). An Analytical Model for Teaching Architectural Building Design. In *2010 Developments in E-systems Engineering* (pp. 101-106). IEEE.
- Banerjee, S., Easwaramoorthy, D., & Gowrisankar, A. (2021). *Fractal functions, dimensions and signal analysis*. Cham: Springer.
- Bankman, I. (Ed.). (2008). *Handbook of medical image processing and analysis*. Elsevier.
- Batty, M., & Longley, P. A. (1994). *Fractal cities: a geometry of form and function*. Academic press.
- Bianconi, F., & Filippucci, M. (2018). Generative education: thinking by modeling/modeling by thinking. In *Congreso Internacional de Expresión Gráfica Arquitectónica* (pp. 1009-1020). Cham: Springer International Publishing..
- Blessing, L. T., & Chakrabarti, A. (2009). *DRM, A design research methodology*. London: Springer London.
- Bovill, C., & Bovill, C. (1996). Fractal geometry in architecture and design.
- Caetano, I., Garcia, S., Pereira, I., & Leitão, A. (2020). Creativity Inspired by Analysis: an algorithmic design system for designing structurally feasible façades. In *25th International Conference on Computer-Aided Architectural Design Research in Asia, CAADRIA 2020* (pp. 599-608). The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)..
- Caetano, I., & Leitão, A. (2021). Mathematically developing building facades: an algorithmic framework. In *Formal Methods in Architecture: Proceedings of the 5th International Symposium on Formal Methods in Architecture (5FMA), Lisbon 2020* (pp. 3-17). Cham: Springer International Publishing.
- Caetano, I., Santos, L., & Leitão, A. (2020). Computational design in architecture: Defining parametric, generative, and algorithmic design. *Frontiers of architectural research*, 9(2), 287-300.
- Campbell, M. I., & Shea, K. (2014). Computational design synthesis. *AI EDAM*, 28(3), 207-208.
- Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N. V., & Wood, K. L. (2011). Computer-based design synthesis research: an overview.
- Chen, C. Y. (2020). *Algorithmic design for residential housing concept: Cologne-Mülheim: generating design plan and floor plan in four steps: transform, select, determine and extrude* (Doctoral dissertation, Wien).
- Chermayeff, S., & Alexander, C. (1963). *Community and privacy: Toward a new architecture of humanism* (Vol. 474): Garden City, NJ: Anchor Books, Doubleday.
- Ching, F. D. (2023). *Architecture: Form, space, and order*. John Wiley & Sons.

- Chiou, S. C., & Krishnamurti, R. (1995). The grammar of Taiwanese traditional vernacular dwellings. *Environment and planning B: planning and design*, 22(6), 689-720.
- Contini, A. (2007). Critical Phenomena in Natural Sciences. Chaos, Fractals, Self Organization and Disorder: Concepts and Tools. In *JSTOR*.
- Cox, T., & d'Antonio, P. (2016). *Acoustic absorbers and diffusers: theory, design and application*. CRC press.
- DAG, D. A. G., & Ethic, W. (2000). *Introduction to systems engineering*. State College, PA, USA: Citeseer.
- Davis, D. (1997). Design as a Process the Project Development Process. In *1997 Annual Conference* (pp. 2-130).
- de Oliveira, M. N. P., & Celani, M. G. C. (2019). Cellular automata: Towards possible applications in urban design education and practice. *Oculum Ensaios: revista de arquitetura e urbanismo*.
- De Souza, M. A. F., & Ferreira, M. A. G. V. (2002). Designing reusable rule-based architectures with design patterns. *Expert Systems with Applications*, 23(4), 395-403.
- Doe, R. (2018). Facilitating integration of computational design processes in the design and production of prefabricated homes. *Architectural Science Review*, 61(4), 246-254.
- Eastman, C. M. (2011). *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons.
- Eder, W. E. (2009). Analysis, synthesis and problem solving in design engineering. In *DS 58-2: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 2, Design Theory and Research Methodology, Palo Alto, CA, USA, 24.-27.08. 2009* (pp. 13-24).
- Ediz, Ö., & Çağdaş, G. (2007). A computational architectural design model based on fractals. *Open house international*, 32(2), 36-45.
- Eggert, R. (2005). *Engineering design*. Pearson/Prentice Hall.
- Eilouti, B. (2019). Shape grammars as a reverse engineering method for the morphogenesis of architectural façade design. *Frontiers of Architectural Research*, 8(2), 191-200.
- El-Khaldi, M. M. S. (2007). *Mapping boundaries of generative systems for design synthesis* (Doctoral dissertation, Massachusetts Institute of Technology,
- Elghandour, A., Saleh, A., Aboeineen, O., & Elmokadem, A. (2016). Using parametric design to optimize building's façade skin to improve indoor daylighting performance. In *Proceedings of the 3rd IBPSA-England Conference BSO*.
- Eppinger, S. D., & Ulrich, K. (1995). Product design and development.
- Ettestad, D., & Carbonara, J. (2018). The Sierpinski triangle plane. *Fractals*, 26(01), 1850003.
- Fasoulaki, E. (2008). *Integrated design: A generative multi-performative design approach* (Doctoral dissertation, Massachusetts Institute of Technology).

- Fatai, T. (2024). A Research on the use of Algorithmic Design Methods in the field of Architectural Design. *ScienceOpen Preprints*.
- Fei, D. (2019). Abductive thinking, conceptualization, and design synthesis. In *International Conference on Human Systems Engineering and Design: Future Trends and Applications* (pp. 101-104). Cham: Springer International Publishing.
- Fernstrom, C. (1988). Design considerations for process-driven software environments. In *Proceedings of the 4th international software process workshop on Representing and enacting the software process* (pp. 65-67).
- Fischer, T. (2002). Computation-universal voxel automata as material for generative design education. In *Proceedings of the 5th Conference and Exhibition on Generative Art* (pp. 10-1).
- Fischer, T., & Herr, C. M. (2001). Teaching generative design. In *Proceedings of the 4th Conference on Generative Art* (pp. 147-160). Politecnico di Milano University Milan..
- Flake, G. W. (2000). *The computational beauty of nature: Computer explorations of fractals, chaos, complex systems, and adaptation*. MIT press.
- Fried, D., Legay, A., Ouaknine, J., & Vardi, M. Y. (2018). Sequential relational decomposition. In *Proceedings of the 33rd annual ACM/IEEE Symposium on Logic in computer science* (pp. 42-441).
- Furtado, G. (2012). Dealing with Information, Complex Dynamics and Organizations: Notes on Architecture, Systems Research and Computational Sciences. *Nexus Network Journal*, 14(1), 3-15.
- Gogolla, M., & Selic, B. (2020). On teaching descriptive and prescriptive modeling. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings* (pp. 1-9).
- Grady, J. O. (1994). *System integration* (Vol. 5): CRC press.
- Granadeiro, V., Duarte, J. P., Correia, J. R., & Leal, V. M. (2013). Building envelope shape design in early stages of the design process: Integrating architectural design systems and energy simulation. *Automation in Construction*, 32, 196-209.
- Granadeiro, V., Duarte, J. P., & Palensky, P. (2011). Building envelope shape design using a shape grammar-based parametric design system integrating energy simulation. In *IEEE Africon'11* (pp. 1-6). IEEE.
- Gray, J. (2018). Gauss's Disquisitiones Arithmeticae. In *A History of Abstract Algebra: From Algebraic Equations to Modern Algebra* (pp. 37-47). Cham: Springer International Publishing.
- Gries, M. (2004). Methods for evaluating and covering the design space during early design development. *Integration*, 38(2), 131-183.
- Gu, N., Yu, R., & Behbahani, P. A. (2021). Parametric design: Theoretical development and algorithmic foundation for design generation in architecture. *Handbook of the Mathematics of the Arts and Sciences*, 1-22.

- Guerguis, M., Eikevik, L., Obendorf, A., Tryggestad, L., Enquist, P., Lee, B., . . . Biswas, K. (2017). Algorithmic design for 3D printing at building scale. *International Journal of Modern Research in Engineering and Technology*, 2(1).
- Guidera, S. (2011). Conceptual design exploration in architecture using parametric generative computing: a case study. In *2011 Asee Annual Conference & Exposition* (pp. 22-368).
- Gürbüz, E., Çağdaş, G., & Alaçam, S. (2010). A generative design model for Gaziantep's traditional pattern. In *Proceedings of the 28th Conference on Education of Computer Aided Architectural Design in Europe* (pp. 841-849). ETH Zurich.
- Gurcan Bahadir, C. G., & Tong, T. (2025). Computational approaches to space planning: A systematic review of enhancing architectural layouts. *International Journal of Architectural Computing*, 14780771241310215.
- Gürer, E., Alaçam, S., & Çağdaş, G. (2012). A Dynamic Methodology for Embedding Generative System Approaches in Architectural Design Education. In *ICONARCH International Congress of Architecture and Planning: Architecture and Technology* (pp. 368-374).
- Haakonsen, S. M., Rønquist, A., & Labonnote, N. (2023). Fifty years of shape grammars: A systematic mapping of its application in engineering and architecture. *International Journal of Architectural Computing*, 21(1), 5-22.
- Hartmann, C., Chenouard, R., Mermoz, E., & Bernard, A. (2018). A framework for automatic architectural synthesis in conceptual design phase. *Journal of Engineering Design*, 29(11), 665-689.
- Helms, B., & Shea, K. (2012). Computational synthesis of product architectures based on object-oriented graph grammars. *Journal of Mechanical Design*, 134(2).
- Herr, C. M., & Ford, R. C. (2015). Adapting cellular automata as architectural design tools. In *Emerging Experiences in the Past, Present and Future of Digital Architecture: Proceedings of the 20th CAADRIA conference* (pp. 169-178).
- Huang, W., & Xu, W. (2015). Generative Design Begins with Physical Experiment. In *Tsinghua University Forum*.
- Humppi, H. (2015). Algorithm-Aided Building Information Modeling. In *Complexity & simplicity—Proceedings of the 34th eCAADe conference* (pp. 601-609).
- Indraprastha, A. (2018). Learning to Know and Think: Computing for Architecture Course. In *SHS Web of Conferences* (Vol. 41, p. 05001). EDP Sciences.
- Jabi, W., Soe, S., Theobald, P., Aish, R., & Lannon, S. (2017). Enhancing parametric design through non-manifold topology. *Design Studies*, 52, 96-114.
- Jackson, M. (2009). Some notes on models and modelling. In *Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos* (pp. 68-81). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Jacoby, S. L., & Kowalik, J. S. (1980). *Mathematical modeling with computers*. Prentice Hall.

- Januszkiewicz, K., & Paszkowska-Kaczmarek, N. (2023). Generative and Evolutionary Models in the Design of Architectural Form-Insights from History. *Architecturae et Artibus*, 15(3), 11-38.
- Kalay, Y. E. (2004). *Architecture's new media: Principles, theories, and methods of computer-aided design*. MIT press.
- Karzer, R., & Matcha, H. (2009). Experimental design-build: teaching parameter-based design. In *Computation: The New Realm of Architectural Design [27th eCAADe Conference Proceedings]* (pp. 153-158).
- Kazemi, L. (2019). Application of modular system for innovation buildings architectural design.
- Khamis, A. A., Ibrahim, S. A., Khateb, M. A., Abdel-Fatah, H., & Barakat, M. A. (2022). Introducing the Architecture Parametric Design Procedure: From Concept to Execution. In *IOP Conference Series: Earth and Environmental Science* (Vol. 1056, No. 1, p. 012004). IOP Publishing.
- Knight, T. (2000). Introduction to shape grammars. In *Lecture Notes presented at the MIT, MIT/Miyagi Workshop*.
- Knight, T. (2012). Slow computing: Teaching generative design with shape grammars. In *Computational Design Methods and Technologies: Applications in CAD, CAM and CAE Education* (pp. 34-55). IGI Global.
- Knight, T. W. (1981). The forty-one steps. *Environment and planning B: planning and design*, 8(1), 97-114.
- Kobyshev, N., Riemenschneider, H., Bodis-Szomoru, A., & Van Gool, L. (2016). Architectural decomposition for 3D landmark building understanding. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 1-10). IEEE.
- Kotsopoulos, S. D. (2005). Constructing design concepts: A computational approach to the synthesis of architectural form.
- Koyama, Y. (2021). Introduction to computational design. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (pp.1-4).
- Lakhanpuria, H., & Naik, M. (2023). Incorporating Problem-Based Learning for Promoting Parametric Design Thinking in Architecture Studios: Insights from an Experiment in India. *Journal of the International Society for the Study of Vernacular Settlements*, 10(10), 379-392.
- Lam, R. H., & Chen, W. (2019). Process Design Optimization. In *Biomedical Devices: Materials, Design, and Manufacturing* (pp. 329-368). Cham: Springer International Publishing.
- Leeds, S. (1977). George Boolos and Richard Jeffrey. Computability and logic. Cambridge University Press, New York and London 1974, x+ 262 pp. *The Journal of Symbolic Logic*, 42(4), 585-586.
- Leung, A. Y. T., Wu, G. R., & Zhong, W. F. (2004). Exterior problems of acoustics by fractal finite element mesh. *Journal of sound and vibration*, 272(1-2), 125-135.
- Lindenmayer, A. (1968). Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of theoretical biology*, 18(3), 280-299.

- Littlejohn, S. W., & Foss, K. A. (2010). *Theories of human communication*. Waveland press.
- Liu, Y., Corcoran, J., & Feng, Y. (2020). Urban Cellular Automata.
- Liu, Y., & Herr, C. M. (2023). Cellular Automata as Design Tools for Artificial Ecologies. In *xArch—creativity in the age of digital reproduction symposium* (pp. 42-49). Singapore: Springer Nature Singapore.
- Loi, C., & Cournede, P. H. (2008). Generating functions of stochastic L-systems and application to models of plant development. *Discrete Mathematics & Theoretical Computer Science*, (Proceedings).
- Lorenz, W. E. (2011). FRACTAL GEOMETRY OF ARCHITECTURE: Fractal Dimension as a connection between Fractal Geometry and Architecture. In *Biomimetics--Materials, Structures and Processes: Examples, Ideas and Case Studies* (pp. 179-200). Berlin, Heidelberg: Springer Berlin Heidelberg.
- M Rocker, I. (2006). When code matters. *Architectural Design*, 76(4), 16-25.
- Maher, M. L. (1990). Process models for design synthesis. *AI magazine*, 11(4), 49-49.
- Mandelbrot, B. B. (1982). *The fractal geometry of nature*. NY: Freeman.
- Mark, E. (2008). Animated parametric rapid prototyping. In *the Proceedings of the 26th eCAADe Conference, Antwerpen, Belgium* (pp. 897-904).
- Markus, T. A. (1969). The role of building performance measurement and appraisal in design method. *Design methods in Architecture*, 6(7), 109-117.
- Maver, T. W. (1970). Appraisal in the building design process. Emerging methods in environmental design and planning. *MIT Press (Cambridge, MA)*.
- Mayatskaya, I., Yazyeva, S., Gatiev, M., Kuznetsov, V., Klyuev, S., & Sabitov, L. (2022). Application of Fractal Methods in the Design of Modern Structures. In *International Scientific Conference Industrial and Civil Construction* (pp. 414-422). Cham: Springer Nature Switzerland.
- McCormack, J. (2004). Generative modelling with timed L-systems. In *Design Computing and Cognition '04* (pp. 157-175). Dordrecht: Springer Netherlands.
- Michelle, B., & Gemilang, M. P. (2022). A bibliometric analysis of generative design, algorithmic design, and parametric design in architecture. *Journal of Artificial Intelligence in Architecture*, 1(1), 30-40.
- Middya, U., & Luss, D. (1994). Impact of global interactions on patterns in a simple system. *The Journal of chemical physics*, 100(9), 6386-6394.
- Miraglia, S. (2014). Systems architectures and innovation: The modularity-integrality framework. *Cambridge Service Alliance, Working Paper*.
- Mitchell, J. W., & Molloy, I. P. (2020). Complete energy analytical model building information modeling (BIM) integration. *U.S. Patent No. 10,628,535*. 21 Apr.2020.
- Mitchell, W. J., & Terzidis, K. (2004). *Expressive form: A conceptual approach to computational design*. Routledge.
- Moretti, L. (1971). Ricerca matematica in architettura e urbanistica. *Moebuis IV*, 1, 30-53.

- Moussavi, F. (2009). *The function of form*: Actar, Barcelona.
- Mulaik, S. A. (2009). *Foundations of factor analysis*. CRC press.
- Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). Procedural modeling of buildings. In *ACM SIGGRAPH 2006 Papers* (pp. 614-623).
- Ng, C. S., Chen, C. H., & Sathikh, P. M. (2024). A procedural approach based on cellular automata for the generation of spatial layout designs. *International Journal of Architectural Computing*, 14780771241299596.
- Ostrowska-Wawryniuk, K., Strzała, M., & Słyk, J. (2022). Form Follows Parameter: Algorithmic-Thinking-Oriented Course for Early-stage Architectural Education. *Nexus Network Journal*, 24(2), 503-522.
- Ostwald, M. J. (2001). "Fractal architecture": Late twentieth century connections between architecture and fractal geometry. *Nexus Network Journal*, 3(1), 73-84.
- Oxman, R. (2008). Digital architecture as a challenge for design pedagogy: theory, knowledge, models and medium. *Design studies*, 29(2), 99-120.
- Oxman, R. (2017). Thinking difference: Theories and models of parametric design thinking. *Design studies*, 52, 4-39.
- Ozkar, M. (2017). *Rethinking basic design in architectural education: foundations past and future*. Routledge.
- Papalambros, P. Y. (2000). Extending the optimization paradigm in engineering design. In *Proc 3rd Int. Symp. Tools Meth. Compet. Engineer. Delft*.
- Papalambros, P. Y., & Wilde, D. J. (2000). *Principles of optimal design: modeling and computation*. Cambridge university press
- Patt, T. (2015). Generative masterplanning inspired by cellular automata with context-specific tessellations. *EDUCATION AND RESEARCH IN COMPUTER AIDED ARCHITECTURAL DESIGN IN EUROPE*, 33, 461-466.
- Patuano, A., & Tara, A. (2020). Fractal geometry for landscape architecture: review of methodologies and interpretations. *Journal of Digital Landscape Architecture*, 5(10).
- Peitgen, H. O., Jürgens, H., Saupe, D., & Feigenbaum, M. J. (2004). *Chaos and fractals: new frontiers of science* (Vol. 106, pp. 560-604). New York: Springer.
- Pérez García, A. J., & Gómez Martínez, F. (2010). Natural structures: strategies for geometric and morphological optimization. In *Symposium of the International Association for Shell and Spatial Structures (50th. 2009. Valencia). Evolution and Trends in Design, Analysis and Construction of Shell and Spatial Structures: Proceedings*. Editorial Universitat Politècnica de València.
- Peteinarelis, A., & Yiannoudes, S. (2018). Parametric Models and Algorithmic Thinking in Architectural Education. In *Proceedings of the International Conference on Education and Research in Computer Aided Architectural Design in Europe*.

- Pimmler, T. U., & Eppinger, S. D. (1994). Integration analysis of product decompositions. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Vol. 12822, pp. 343-351). American Society of Mechanical Engineers.
- Prusinkiewicz, P., Cieslak, M., Ferraro, P., & Hanan, J. (2018). Modeling plant development with L-systems. In *Mathematical modelling in plant biology* (pp. 139-169). Cham: Springer International Publishing.
- Prusinkiewicz, P., & Lindenmayer, A. (2012). *The algorithmic beauty of plants*. Springer Science & Business Media.
- Purnomo, K. D., Sari, N. P. W., Ubaidillah, F., & Agustin, I. H. (2019). The construction of the Koch curve (n, c) using L-system. In *AIP Conference Proceedings* (Vol. 2202, No. 1, p. 020108). AIP Publishing LLC.
- Rian, I. M., & Asayama, S. (2016). Computational Design of a nature-inspired architectural structure using the concepts of self-similar and random fractals. *Automation in Construction*, 66, 43-58.
- Rian, I. M., Callegary, G., & Spinelli, A. (2015). Transforming Nature's Forest into Manmade Forest: Fractal-Based Computational Morphogenesis Approach for a Dendriform Pavilion Design. *Proceedings of the IASS 2015 Tokyo Colloquium on Bio-Based and Bio-Inspired Environmentally Compatible Structures*. Tokyo Denki University, Tokyo, Japan.
- Rian, I. M., Park, J. H., Ahn, H. U., & Chang, D. (2007). Fractal geometry as the synthesis of Hindu cosmology in Kandariya Mahadev temple, Khajuraho. *Building and environment*, 42(12), 4093-4107.
- Rian, I. M., & Sassone, M. (2014). Tree-inspired dendriforms and fractal-like branching structures in architecture: A brief historical overview. *Frontiers of Architectural Research*, 3(3), 298-323.
- Sammer, M., Leitão, A., & Caetano, I. (2019). From visual input to visual output in textual programming. In *Proceedings of the 24th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA)* (Vol. 1, pp. 645-654).
- Samson, F. P., & Peterson, T. A. (2010). A Systems Engineering and Integration Methodology for Complex Systems. In *Ground Vehicle Systems Engineering Technology Symposium* (pp. 1-8).
- Sardahi, Y. (2016). Multi-objective optimal design of control systems (Doctoral dissertation, University of California, Merced).
- Schmidt, J. W., & Taylor, R. E. (1970). *Simulation and analysis of industrial systems* (Vol. 20): RD Irwin.
- Schumacher, P. (2008). Parametricism as style-parametricist manifesto. *11th Architecture Biennale, Venice*, 14.
- Schumacher, P., & Krish, S. (2010). Teaching Generative Design Strategies for Industrial Design. *Design, July*, 1-4.
- Shelden, D. R. (2002). Digital surface representation and the constructability of Gehry's architecture. (Doctoral dissertation, Massachusetts Institute of Technology).

- Shtepani, E., & Yunitsyna, A. (2023). Application of 3D Printing for the Parametric Models Fabrication in the Architectural Education. In *1st International Conference on Frontiers in Academic Research* (pp. 155-161).
- Soliman, S., Taha, D., & El Sayad, Z. (2019). Architectural education in the digital age: Computer applications: Between academia and practice. *Alexandria Engineering Journal*, 58(2), 809-818.
- Song, X., Poirson, E., Ravaut, Y., & Bennis, F. (2023). Multi-objective optimization of layout with functional constraints. *Optimization and Engineering*, 24(3), 1849-1882.
- Št'ava, O., Beneš, B., Měch, R., Aliaga, D. G., & Křištof, P. (2010). Inverse procedural modeling by automatic generation of L-systems. In *Computer graphics forum* (Vol. 29, No. 2, pp. 665-674). Oxford, UK: Blackwell Publishing Ltd.
- Stiny, G. (2006). *Shape: talking about seeing and doing*. MIT Press.
- Stiny, G. (2022). *Shapes of Imagination: calculating in Coleridge's Magical realm*. MIT Press.
- Stiny, G., & Gips, J. (1971). Shape grammars and the generative specification of painting and sculpture. In *IFIP congress (2)* (Vol. 2, No. 3, pp. 125-135).
- Stiny, G., & Mitchell, W. J. (1980). The grammar of paradise: on the generation of Mughul gardens. *Environment and planning B: planning and design*, 7(2), 209-226.
- Stotz, I., Gouaty, G., & Weinand, Y. (2009). Iterative geometric design for architecture. *Journal of the International Association for Shell and Spatial Structures*, 50(1), 11-20.
- Suh, N. P. (1990). *The principles of design*: Oxford university press. New York, Oxford.
- Tabadkani, A., Shoubi, M. V., Soflaei, F., & Banihashemi, S. (2019). Integrated parametric design of adaptive facades for user's visual comfort. *Automation in Construction*, 106, 102857.
- Tepavčević, B., & Stojaković, V. (2012). Shape grammar in contemporary architectural theory and design. *Facta Universitatis-series: Architecture and Civil Engineering*, 10(2), 169-178.
- Terzidis, K. (2004). Algorithmic design: a paradigm shift in architecture. In *Architecture in the Network Society [22nd eCAADe Conference Proceedings/ISBN 0-9541183-2-4] Copenhagen (Denmark)* (pp. 201-207).
- Terzidis, K. (2006). *Algorithmic architecture*. Routledge.
- Touloupaki, E., & Theodosiou, T. (2017). Optimization of building form to minimize energy consumption through parametric modelling. *Procedia environmental sciences*, 38, 509-514.
- Toussi, H. E. (2020). The application of evolutionary, generative, and hybrid approaches in architecture design optimization. *NEU Journal of Faculty of Architecture (NEU-JFA)*, 2(2), 1-20.
- Toussi, H. E., Etesam, I., & Mahdavejad, M. (2021). The Application of Evolutionary Algorithms and Shape Grammar in the Design Process Based upon Traditional Structures. *The Monthly Scientific Journal of Bagh-e Nazar*, 18(95), 19-36.
- Ulrich, K. (1995). The role of product architecture in the manufacturing firm. *Research policy*, 24(3), 419-440.

- Ulrich, K. T., & Eppinger, S. D. (2016). *Product design and development*. New York: McGraw-hill.
- Ulrich, K. T., & Seering, W. P. (1990). Function sharing in mechanical design. *Design Studies*, 11(4), 223-234.
- Vande Zande, R. (2006). The design process of problem solving. *Academic Exchange Quarterly*, 10(4), 150-154.
- VAZ, C. E. V., & CELANI, M. G. C. Developing knowledge based design education method: using generative systems and ontology to teach landscape design.
- Vazquez, E. (2024). Teaching parametric design: fostering algorithmic thinking through incomplete recipes. *Open House International*, 49(4), 736-751.
- Vyzantiadou, M. A., Avdelas, A. V., & Zafiropoulos, S. (2007). The application of fractal geometry to the design of grid or reticulated shell structures. *Computer-Aided Design*, 39(1), 51-59.
- Wahbeh, W. (2017). Building skins, parametric design tools and BIM platforms. In *Conference Proceedings of the 12th Conference of Advanced Building Skins* (pp. 1104-1111).
- Whitney, D., Crawley, E., de Weck, O., Eppinger, S., Magee, C., Moses, J., . . . Wallace, D. (2004). The influence of architecture in engineering systems. *Engineering Systems Monograph, MIT Engineering Systems Division, March*.
- Whitney, D. E. (1996). Why mechanical design cannot be like VLSI design. *Research in Engineering Design*, 8(3), 125-138.
- Wolfram, S. (2002). A new kind of science (Vol. 5). *Wolfram media Champaign*, 80.
- Wong, M. L., Cleland, C. E., Arend Jr, D., Bartlett, S., Cleaves, H. J., Demarest, H., . . . Hazen, R. M. (2023). On the roles of function and selection in evolving systems. *Proceedings of the National Academy of Sciences*, 120(43), e2310223120.
- Wu, J. (2013). Hierarchy theory: an overview. *Linking ecology and ethics for a changing world: Values, philosophy, and action*, 281-301.
- Yavuz, A. Ö., & Çelik, T. (2014). Proposing A Generative Model Developed by Ecologic Approaches In Architectural Design Education. *Procedia-Social and Behavioral Sciences*, 143, 330-333.
- Yu, J., & Min, D. (2022). PL-System: Visual representation of pattern language using L-System. In *POST-CARBON-Proceedings of the 27th CAADRIA Conference* (pp. 201-210).
- Zhang, M. (2020). The applications of parametric design in green building. In *IOP Conference Series: Earth and Environmental Science* (Vol. 567, No. 1, p. 012033). IOP Publishing.
- Zhang, Q., Deniaud, I., Caillaud, E., & Baron, C. (2012). Descriptive model for interpreting innovative design. In *International Design Conference 2012* (pp. 343-353).
- Cambridge dictionary. Algorithmic. <https://dictionary.cambridge.org/dictionary/english/algorithmic>
- Cambridge Dictionary. Analysis. <https://dictionary.cambridge.org/dictionary/english/analysis>
- Cambridge Dictionary. Evaluation. <https://dictionary.cambridge.org/dictionary/english/evaluation>

Cambridge dictionary. Generative. <https://dictionary.cambridge.org/dictionary/english/generative>

Cambridge dictionary. Optimization. <https://dictionary.cambridge.org/dictionary/english/optimization>

Cambridge dictionary. Parametric. <https://dictionary.cambridge.org/dictionary/english/parametric>

Cambridge dictionary. System. <https://dictionary.cambridge.org/dictionary/english/system>

Cambridge dictionary. Synthesis. <https://dictionary.cambridge.org/dictionary/english/synthesis>

<https://www.archdaily.com/896433/morpheus-hotel-zaha-hadid-architects>

