Journal of Systemics, Cybernetics and Informatics (2024) 22(5), 23-30
https://doi.org/10.54808/JSCI.22.05.23

# A Computer-Supported Framework for Form Analysis

**Buthayna EILOUTI**

Department of Architectural Engineering
American University of Ras Al-Khaimah
Ras Al-Khaimah, UAE

## ABSTRACT

Although there are several frameworks for the synthesis of form, and for the automated development of geometric shapes, studies in the systematic analysis of architectural forms and their automation are still underrepresented. This paper describes a framework for computer-aided geometric form analysis. It illustrates the framework's applicability by an example from precedent architectural design.

The form analysis framework is developed theoretically and further translated into a computer code that is written in the AutoLISP language to fit its graphic operation system. Application of the framework and its computerized code demonstrates its power as a promising analytical tool that can help designers analyze precedents' morphological structure more efficiently.

**Keywords**: Design Automation, Systematic Design, Design Knowledge, Form Processor, Shape Grammar, Top-Down Approach, Parametric Design

## INTRODUCTION

Form derivation is a major task in most design disciplines. It consists of multiple components that are processed through various phases. Although many phases require intuitive and heuristic-based processes, several steps can still be systematized. Similarly, while some aspects of product design can be assessed visually, several others can be analyzed using structured and systematic methods. There is multiple software that help model various types of engineering products. However, to date, there is no satisfactory explicit and systematic method to identify and describe a framework to automate the design analysis process. This paper describes a theoretical and a computational framework for design analysis that helps analyze geometric forms explicitly and systematically.

The theoretical foundation of the proposed framework uses a sequence of form analysis that reverses the form generation phases. These move typically from the schematic level to the spatial one and end with the architectural articulation level.

The bases of form generation use a hierarchical system of the assembly of rudimentary elements to derive design products. This process can be traced back to Alexander's effort of pattern language [1] and synthesis of basic blocks [2]. These efforts coupled with efforts of systematic design methods such as Archer's [3], Jones' [4], Lawson's [5] and Cross' [6] form the point of departure for the theoretical structure of the framework presented in this paper.

The major goal of this research is to establish a framework of design analysis automation. The research has three objectives. The first is to demonstrate one way in which the process of form analysis can be systematized. The second is to introduce a new direction for architectural informatics. This direction is based on a reverse engineering study of design products. The third is to translate the theoretical construct into an applicable framework that can be used by architects to help them automate their design product studies. In terms of the research methodology, the theoretical hierarchical design is used for framework building, and computational method. Case studies are used for framework testing. In hierarchical design methodology a design is divided into systems each of which recursively constructs more detailed sub-systems until it ends up with the final product. A major benefit of applying this methodology is to simplify the management of complex systems and to facilitate design component recycling. This research is expected to have pedagogical, theoretical, and practical consequences. It is expected to help educators and students experiment with an explicit and replicable method of analyzing architectural compositions. It is also expected to enhance the acquisition of knowledge about architectural form derivation and analysis. In this sense, the research contributes to the body of testable theories in architectural research. In architectural practice, the research is expected to facilitate the creation and use of a database of rudimentary architectural elements. Such elements are often used as starting points for designing buildings. The research will also demonstrate a method for developing sophisticated ideas in simpler and faster ways than manual and traditional computer-aided methods. In addition, it completes previous efforts of design process communication (e.g. [7-19]).

The research employs concepts and tools which are used to process strings in formal languages to process shapes. However, this is done with the realization that an architectural form language cannot be formulated directly from a string language. For example, while a single set of vocabulary elements suffices in the latter; multiple sets are needed in the former, corresponding to the major stages of design processing. In addition, a set of rules is introduced to translate each stage into another. Rule development uses concepts of shape grammars to articulate component replacement in the various stages of design processing.

To demonstrate the efficiency of the proposed framework, computer-based simple software is formulated and tested. The software allows its users to derive three-dimensional architectural artifacts using a subset of spatial terminals (e.g., cube) and a subset of architectural terminals (e.g., walls, floors, pediments, roofs, columns, and openings). While these subsets define the scope of the proposed research, future studies can extend them to provide a comprehensive database that can describe any architectural composition. The interactive code produced by the computational framework may enable users to explore different compositional alternatives. Moreover, it allows users to analyze an architectural artifact at different levels of abstraction. However, the aspects of form analysis on the advanced levels will not be discussed in this paper. Using a language that is not designed specifically for graphic applications such as FORTRAN, C, or C++ suggests building computer graphic routines from scratch. Most of the basic graphic functions are already well established and developed since the 1960's. A major consideration of the proposed code is to begin where other graphic software ended, rather than to restart and redo what they did. Another alternative was TKL. It works well for two-dimensional graphic applications, but as effective for the intended 3D applications. OpenGL is a language that handles 3D graphic applications. However, it is not found efficient -by the author- in the set operations (union, subtraction, and intersection) of solids. AutoLISP is the language used for AutoCAD graphic software. It uses AutoCAD's display environment and routines. It handles solid modeling and set operations effectively. Also, this author is experienced in using AutoCAD in multiple applications. These considerations made AutoLISP the strongest candidate to implement the final code. The AutoLISP language is derived from its parent LISP language. The latter is an acronym for **LIS**t **P**rocessing. It is the second-oldest (the oldest is FORTRAN) high-level programming language still used. It was developed in the late 1950s by John McCarthy and a group of researchers at the Massachusetts Institute of Technology.

LISP permits succinct programming of complex problems. Since LISP's creation, it has been used by computer scientists in the field of language development. It is also characterized by its flexible nature when compared to other programming languages.

AutoLISP is the version of LISP that is designed specifically for interactive graphic applications. AutoCAD is the operating system[1] for AutoLISP. AutoLISP first appeared in AutoCAD Version 2.18, released in January 1986. In addition to the general characteristics of LISP, it is relatively easy to construct and modify data structures in AutoLISP. Both LISP and AutoLISP are powerful in supporting recursion and direct looping for iterations. Recursion is the concept of a subroutine calling itself. This capability is extremely important in the manipulation of structured and linked lists[1]. Since strings of formal languages are usually stored and processed as lists, this latter characteristic strengthens AutoLISP as a candidate for implementing the final code.

## THE THEORETICAL FRAMEWORK

This research applies a morphology-based model. Many scholarly works are concerned with the morphological analysis of the built form in architectural literature. Examples of these include a critical analysis of modern architecture [20], a provocative study of Palladio's plans [21], a comparative procedural interpretation of Palladio villa designs and a comparative analysis of the design schemes of Palladio and Sinan in their sacred buildings [22, 23]. In addition, there are related studies about precedent-based design paradigms, such as developing new approaches that use knowledge recycling from precedent-based problem solutions.

In terms of the method applied in this study, reverse engineering represents a top-down case-based approach to designing. It is the process of extracting knowledge from an existing manmade product, and reproducing a new product -similar to the original or inspired by it- based on the extracted knowledge. Resulting knowledge gained through the reverse engineering processes can be applied in a subsequent bottom-up forward-engineering approach to design similar products. Hence, reverse engineering functions not only as an analytical tool but also as a technique of problem solving, or an approach to the systematic analysis or synthesis of design products. It has applications in many fields such as the bio-medical, chemical, mechanical and civil engineering, computer programming, and manufacturing and industrial applications. Although indirectly implemented in case analysis, its applications in architecture are still underrepresented and under-utilized. The framework introduced in this paper contributes to knowledge in this area.

[1] An operating system is defined as a program which causes the hardware to operate as a system.

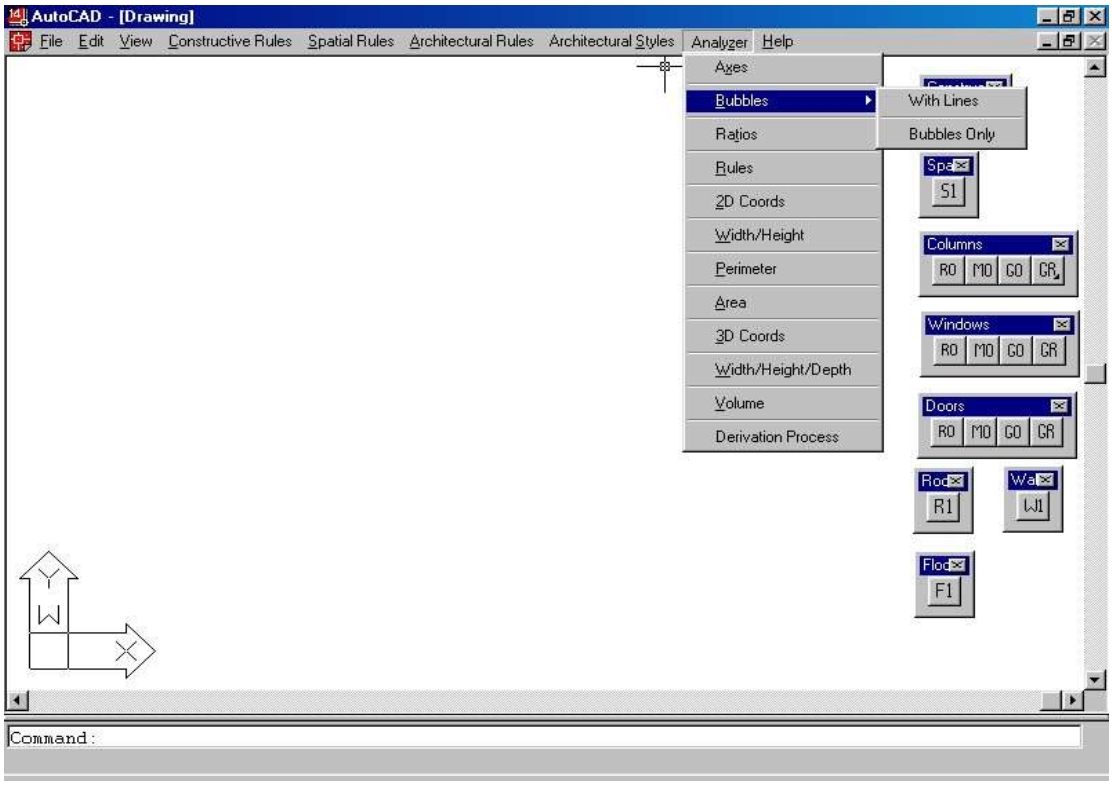


Figure 1 An Image of FormPro 1.0 Interface with the Analyzer Pull Down Menu Activated

## THE ANALYZER INTERFACE

This author developed a framework called FormPro 1.0 for form derivation and analysis. In this framework, the analysis pull-up menu displays 12 buttons (Figure 1). Pressing each one of them activates a command that helps analyze the drawing at hand. They consist of:

**1. Axes Command**: it displays the axes of the major spaces defined in a drawing. These are displayed with dotted lines to distinguish them from other construction lines.

**2. Bubbles Command**: This has two options: Bubbles with Lines, and Bubbles Only. The former displays the bubbles correspondent to major spaces in a drawing, along with other lines in a drawing. The latter filters all other lines and displays bubbles only.

**3. Ratios Command**: During the derivation process of a building form, the code keeps track and records the ratios used in a separate file. Activating this command displays on the screen a rewriting of this file. This command helps realizing which ratios repeat in a given style or for a given architect. It helps comparing differen buildings according to their proportional systems. This feature helps the user to emphasize one aspect of the morphological structure of a given form.

**4. Rules Command**: This command is similar in structure to the Ratios command. The drawing file stores all rules used in a drawing in a separate file. Activating Rules command displays a rewriting of this file on the screen.

Both commands, Ratios and Rules commands, are structured to keep records of the constructive and spatial levels only. This keeps the records simple, of low memory requirement, and easy to follow.

**5. Width/Height Command**: It displays the numeric value of the length of the two perpendicular sides of a highlighted rectangle.

**6. Perimeter Command**: It displays the numeric value of the perimeter of a given rectangle.

**7. 2D Coords Command**: It displays the four-point coordinates for a highlighted rectangle.

**8. Area Command**: Given a rectangle, this command displays the numeric value of its area in square units.

**9. 3D Coords**: Activating this command prompts the user to point to two opposite corners in space of a cuboid. It displays the coordinates for all eight corners of that cuboid.

10. **Width/Height/Depth Command**: Highlighting a cuboid by its two opposite corners, this command displays the numeric value of the length of its sides.
11. **Volume Command**: Given a cuboid, this command displays the numeric value of its volume in cubic units. It prompts the user to provide the two opposite corners of a cuboid in space.
12. **Derivation Process Command**: Built in the structure of the constructive and spatial commands is a routine that records an image of the drawing at the end of the execution of each command. At the end of the process, activating the Derivation Process command plays a slide presentation for the step-by-step sequence of the form derivation.

## CASE STUDY THE ANALYSIS PROCESS OF THE UDENE HOUSE

The FormPro 1.0 has a derivative part that records all derivation steps of forms. In addition to the analytical layers impeded in the derivation process itself, FormPro 1.0 provides several analytical operations. These rules are applied on a well-known architectural precedent, that is the Udene House (Figure 2).

These include recording all the rules and ratios used to derive a building, coordinates, side lengths, perimeters, areas and volumes of architectural components. The analytical commands of FormPro 1.0 also enable users to display the major axes used in a drawing. In addition, a bubble diagram that corresponds to the major spaces in a building can be displayed (Figure 3).

The major axes of the form can also be analyzed (Figure 4). Finally, FormPro 1.0 stores a slide image for each step in the process. This enables the deriver and others to watch a slide show of the derivation process after its termination.

In addition to the analytical capabilities of FormPro 1.0 that are demonstrated in Figures (3 and 4), the program enables experimenting with other imaginary scenarios of the building appearance. For example, it is possible by one command to display a beam and column structural skeleton for the Udene house as shown in Figure (5). Similarly, FormPro 1.0 facilitates examining how a basic spatial configuration may appear using various architectural style articulations. However, the role of the framework in assigning various styles is outside the scope of this research.

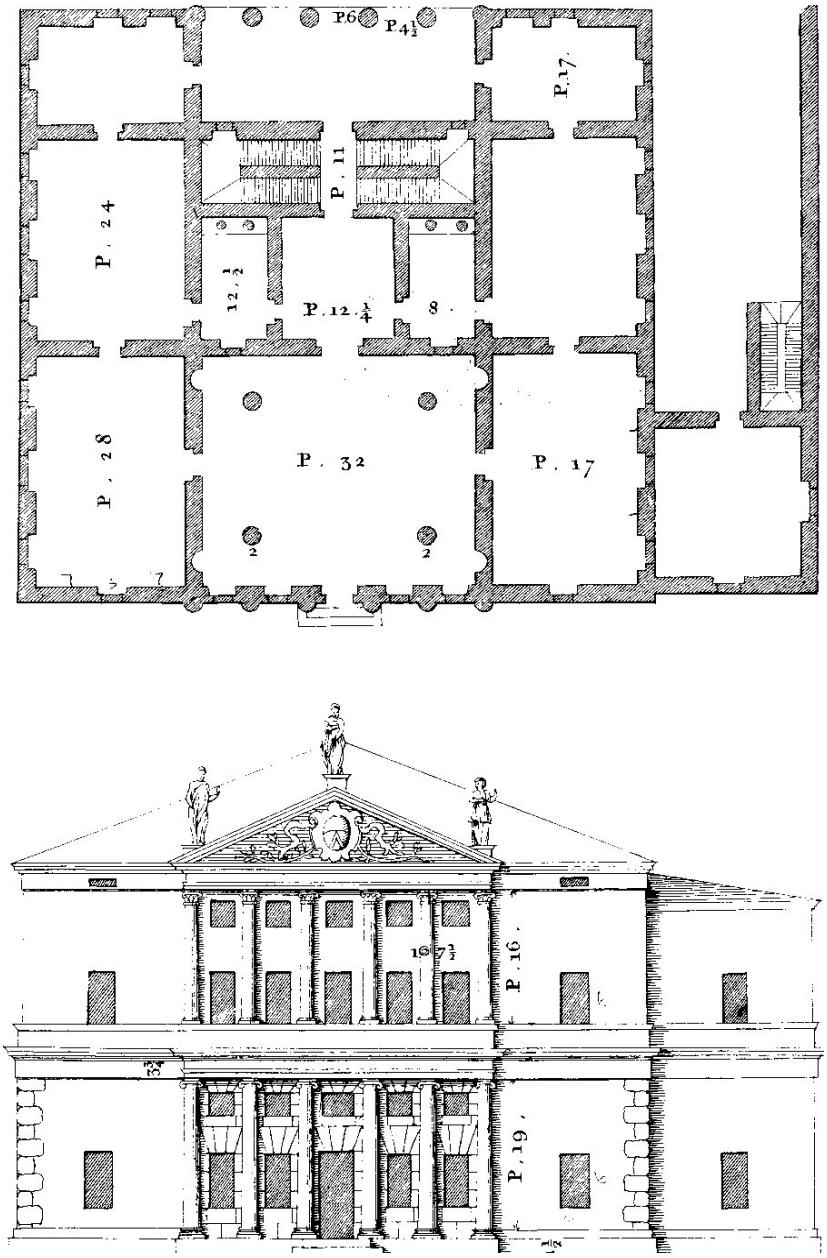


Figure 2: The Plan and Front Elevation of the Udene House

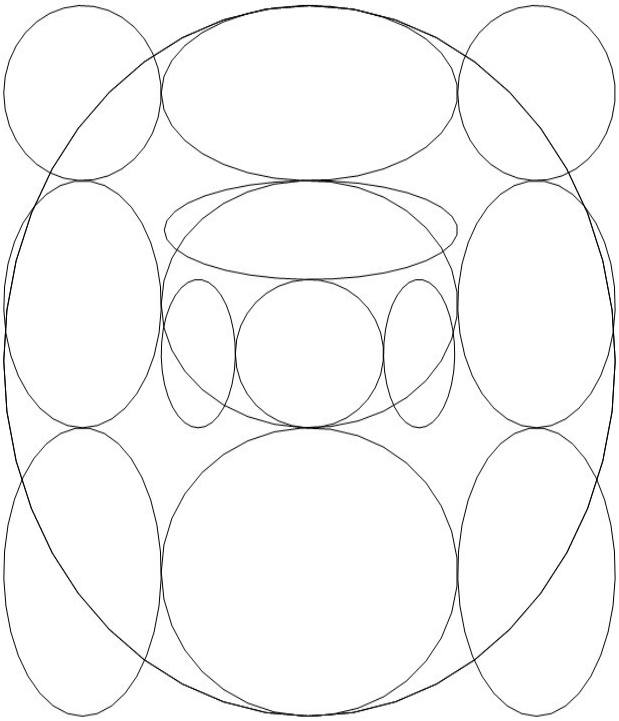

Figure 3 The Bubble Diagram Correspondent to the

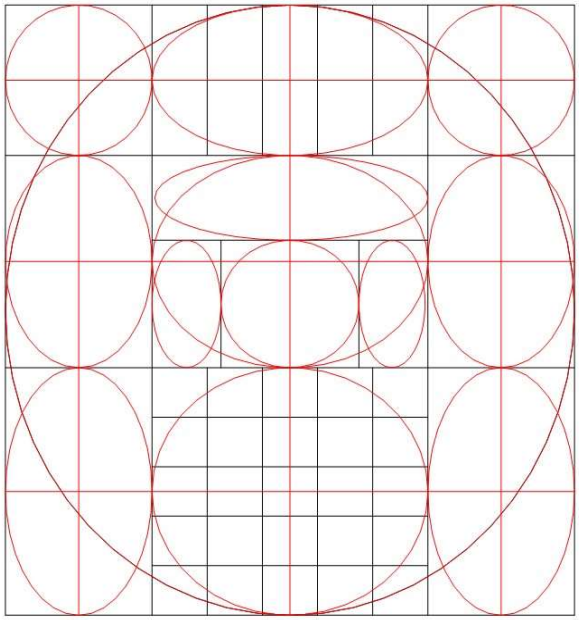

Figure 4: Major Axes and Spaces of the Udene House

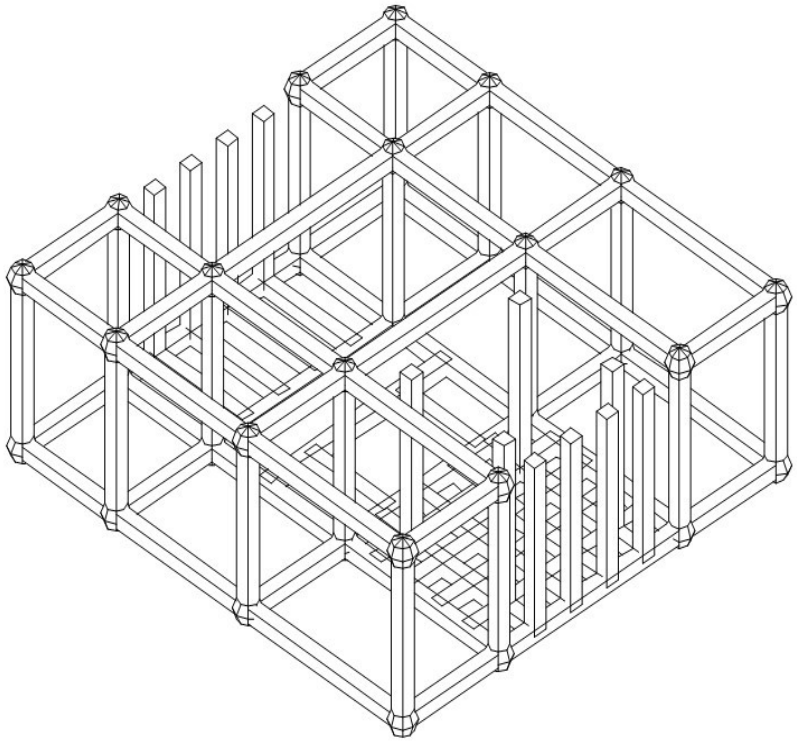

Figure 5: An Image of an Imaginary Column and Beam Skeleton for the Udene House

## CONCLUSION

Although literature in the analytical studies of form and computer-aided design studies recorded some research in the analysis of geometrical shapes [e.g., 24], most examples were concerned with the technical issues of building design. This research represents a systematic structure for the analysis of architectural forms in terms of their morphological constructs. It introduces a framework for the morphological analysis of built forms. The theoretical framework is further developed into a computer software that helps analyze architectural forms faster than manual analyses. A future extension of this research involves computer vision to recognize pixel images and extract their morphological attributes to analyze a wider scope of graphic representations.

## REFERENCES


[1] Alexander, Christopher, 1977. *A Pattern Language: Towns, Buildings, Construction*. London: Oxford University Press

[2] Alexander, Christopher, 1964. *Notes on the Synthesis of Form.* Cambridge: Harvard UP

[3] Archer, L. Bruce, 1965. *Systematic Method for Designers.* Council of Industrial Design, H.M.S.O.,

[4] Jones, John Christopher. 1970, *Design Methods.* New York: John Wiley & Sons

[5] Lawson, Bryan, 2006. How Designers Think. Oxford UK: Architectural Press/Elsevier

[6] Cross, Nigel, 2011. Design Thinking: Understanding How Designers Think and Work. Oxford UK and New York: Berg

[7] Eilouti, B.H, 2003. Three-Dimensional Modeling by Replacement. In M.H.Hamza (ed.), Proceedings of the IASTED conference on Modeling, Simulation and Optimization, pp. 94-99, Banff, Alberta, Canada

[8] Senescu, R., and J. Haymaker, 2011. Communicating Design Processes Effectively and Efficiently, Technical Report. Stanford University, Center for Integrated Facility Engineering

[9] Eilouti, B. and A. Al-Jokhadar. 2007. A Generative System for Mamluk Madrasa FormMaking, Nexus Network Journal Vol. 9, No.1, pp. 7-30

[10] Eilouti, Buthayna H., Al-Sha'ar, Jamil, 2012, Shape Grammars of Traditional Damascene Houses, International Journal of Architectural Heritage, Vol. 6 (4), Taylor & Francis, pp. 1-21

[11] Eilouti, B. and A. Al-Jokhadar. 2007. A Computer-Aided Rule-Based Mamluk Madrasa Plan Generator, Nexus Network Journal 9, 1, pp. 31-38

[12] Eilouti, Buthayna H., 2009. Design Knowledge Recycling Using Precedent-Based Analysis and Synthesis Models, Design Studies, Elsevier, Vol. 30, No.4, pp. 340-368

[13] Eilouti, Buthayna H., 2009. Knowledge Modelling and Processing in Architectural Design, The 3rd International Conference on Knowledge Generation, Communication and Management: KGCM 2009, Orlando, Florida, USA

[14] Schmitt, Gerhard et al, 2014. Case-Based Architectural Design, in Mary Lou Maher, Pearl Pu (ed.), Issues and Applications of Case-Based Reasoning to Design, pp 241-259

[15] Tovey, Mike (ed), 2015. Design Pedagogy: Developments in Art and Design Education, Ashgate Publishing, Ltd.

[16] Eilouti, Buthayna H., 2012. Enhancement of Systematic Design Processing by Diagrams. *Architectoni.ca,* Vol. 1 (1), pp. 83-94 , CCAAS, Canada

[17] Svanerudha, Patrik, 1999. Architectural Constraints for Design Automation of MultiStorey Timber Houses in Sweden, Automation and Robotics in Construction XVI, pp. 181-186

[18] Ichioka, Y., P. Teicholz, and P. Chinowsky, 1991. An Approach to Automated Architectural Floor Layout Generation with Case-Based Reasoning, Technical Report, Stanford University Center for Integrated Facility Engineering

[19] Jürgen K. Müller, 2003. The Building Block Method: Component-Based Architectural Design for Large Software-Intensive Product Families, Thesis, University of Amsterdam

[20] Frampton Kenneth, and Simone, Ashley. 2015. A Genealogy of Modern Architecture. Baden, Switzerland: Lars Müller Publishers

[21] Eisenman P., and Roman, M., 2015. Palladio Virtuel. New Haven: Yale University Press.

[22] Eilouti, Buthayna. 2012. Sinan and Palladio: Two Cultures and Nine Squares. *International Journal of Architectural Heritage* 6 (1): 1-18. DOI:10.1080/15583058.2010.495821

[23] Eilouti, Buthayna. 2017. Sinan and Palladio: A Comparative Morphological Analysis of Two Sacred Precedents. *Frontiers of Architectural Research* 6 (2): 231-247, DOI: 10.1016/j.foar.2017.03.003.

[24] Saha, A., Manna, N.K., Ghosh, K. et al. 2022. Analysis of geometrical shape impact on thermal management of practical fluids using square and circular cavities. Eur. Phys. J. Spec. Top. 231, 2509–2537. https://doi.org/10.1140/epjs/s11734-022-005938