Contents available at: www.repository.unwira.ac.id

ARTEKS
JURNAL TEKNIK ARSITEKTUR

https://journal.unwira.ac.id/index.php/ARTEKS

Research paper doi: 10.30822/arteks.v8i2.1979

Spatial configuration by rules: an experimental parametric shape rules by shape grammar method

Case study: Adolf Loos's residential works

Alva F. P. Sondakh, Aswin Indraprastha*

School of Architecture, Planning, and Policy Development, Institut Teknologi Bandung
Jl. Ganesa No. 10 Coblong, Kota Bandung, Jawa Barat, Indonesia 40132

ARTICLE INFO

Article history:
Received December 27, 2022
Received in revised form March 19, 2023
Accepted May 30, 2023
Available online August 01, 2023

Keywords:
Parametric shape rules
Rupa
Shape grammar
Spatial configuration

***Corresponding author**: Aswin Indraprastha
School of Architecture, Planning, and Policy Development, Institut Teknologi Bandung, Indonesia
Email: aswinindra@itb.ac.id
ORCID: https://orcid.org/0000-0002-7507-6953

ABSTRACT

This study investigated the method of Shape Grammar to develop a computational technique to generate spatial configuration at the early stage of the design process. In synthesizing the spatial configuration and its variants, the user (i.e., architect) can use this tool as a heuristic way to select and choose a spatial configuration of geometric shapes to represent functional requirements and their relationships. The aim of this study is to propose a new technique for spatial configuration based on parametric rules of a specific style of an architect. Our research methodology is explorative and experimental based on the Shape Grammar theory and analysis of the existing models of architects' works as a case study. We studied the example of Adolf Loos's residential works as our basis for the parametric shape rules and the tool developed in the form of a modular cluster component in the Visual Programming Language Grasshopper in a Rhinoceros 3D modeling software. This cluster component, termed "Rupa," is parametrically modeled and consists of two main algorithms: 1) two-dimensional and three-dimensional rules-based shape generation and 2) the transformation rules of translation, rotation, reflection, dilatation, and tension. Using this component, the user can create and transform a shape configuration by its parameters, constrained only by the characteristics of Adolf Loos's residential works design rules as the underlying principles behind the component. Although limited only to three of Loos's residential works, this experiment has successfully generated residential spatial configuration designs based on the constraint rules of Adolf Loos's residential works. The result shows the potential of this technique and tool in aiding architects to create design variants and select those that are the most optimal.

Introduction

Designing spatial configuration is among the core activities in architectural design. During this creative process, architects formulate a geometric composition of shapes in response to functional requirements, environmental conditions, and other design criteria. At a smaller scale of complexity, the process highly depends on the experience and knowledge of the architect. However, the creative process becomes increasingly complicated along with the scale of complexity of a large design project.

This study aims to propose a new technique for spatial configuration based on parametric rules and schemes of a specific style of an architect. This attempt was conducted by investigating the possibility to generate a specific spatial

configuration footprint of an architect by analyzing the existing works and further develop parametric rules as a scheme for generating spatial configuration.

Our study addressed one of the basic problems in the architectural design process and bridge programming and designing stage which is generating spatial configuration. Scholars have been investigating computational methods that can assist architects in generating alternatives and variants of spatial configuration design based on a set of rules. Among the methods is shape grammar, which is a part of the heuristic deterministic design synthesis methods (Kalay 2004). shape grammar is a method developed by Stiny and Gips (1972) to create shapes starting from an initial shape and a specific set of shape rules. Unlike a procedural method that generates all the possible outcomes from a rule set, the shape grammar method is more deterministic because of the more narrowly defined set of rules based on a specific grammar (Kalay 2004).

Previous research studies have been completed analyzing bodies of design works, identifying and formalizing their shape grammar. Among the examples are such works of Palladian houses (Stiny and Mitchell 1978), Mughul gardens (Stiny and Mitchell 1980), a Japanese Teahouse (Knight 1981), Yunnan houses (Yan 1992), Siza's houses in Malagueira (Duarte 2005), and Baltimore row houses (Khrisnamurti and Yue 2015). Another critical research study on shape grammar is that of Koning and Eizenberg (1981), who managed to formalize a shape grammar and implement it in creating design derivatives of Frank Lloyd Wright's Prairie Houses the importance of this research is due to it being successful in showing the real usage potential of the shape grammar method during the architectural design process.

Numerous software has been developed by major software developers to support the architectural design process and are widely available to architects. However, software development in synthesizing design solutions using a specific method such as shape grammar is limited to experimental projects or targeted to a specific project (Chau et al. 2004). One of the first programs that implemented the shape grammar method was developed by Gips (1975), which analyzed two shapes to identify the likeness between both. Gips (1975) also developed a program that generates shapes based on grammar for painting. Furthermore, he developed other programs that have implemented shape grammar theory in collaboration with other scholars such as in Gips (1999); Chase (2002); Chau et al. (2004); Hoisl and Shea (2011); McKay et al. (2012); and Mandić and Tepavčević (2015).

Gips (1999) suggested four types of tasks that a shape grammar implementation program can perform as follows: 1) shape interpreter, 2) shape parser, 3) shape inferrer, and 4) shape grammar design assistant. The fourth type is more than a shape interpreter as its primary purpose is to construct the grammar itself and in the form of a plug-in for existing computer-aided design (CAD) software. This type of implementation program is the object of this experimental research.

Major CAD software has included functionalities in the form of a programming/scripting language interface that enables the construction of a set of instructions to design a shape grammar. Previous research studies (Pak et al. 2006; Yazar and Colakoglu 2007; Al-Jokhadar 2007b; Duarte et al. 2007; Benros and Duarte 2009; Hoisl and Shea 2011; Benros et al. 2012; Ruiz-Montiel et al. 2013; and Wang et al. 2017) have managed to implement shape grammar in various scripting languages in CAD software. However, a Text-based Programming Language (TPL) is used to complete these implementations, which, according to Leitão et al. (2012), requires a considerable amount of programming language skills. Other than TPL, there is another programming method known as Visual Programming Language (VPL), which according to Ferruci et al. (2002), allows users to communicate with the computer system by visually arranging programmable objects on the computer screen. Figure 1 shows a simple shapes composition in Rhinoceros 3D generated by an algorithm in grasshopper, shown in figure 2.

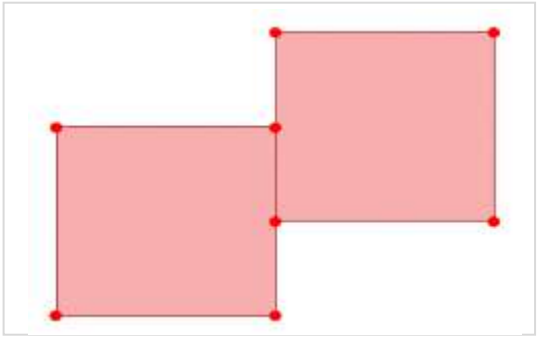


Figure 1. A simple shapes composition

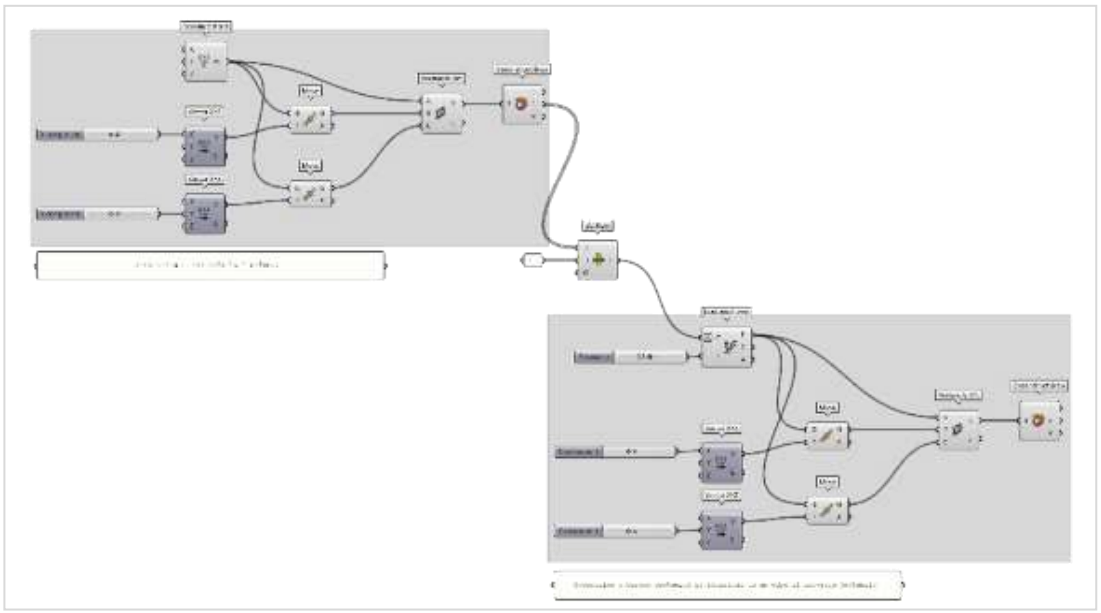


Figure 2. A simple visual algorithm in a grasshopper environment to generate points and curves

Although the composition is simple, the algorithm to generate it has a few components that, to an inexperienced programmer, would be pretty challenging to arrange. The ideal, as shown in figure 3, is an architect that, as an inexperienced programmer, only focuses on the creative process of defining the shape rules and constructing the shape grammar method and allows the generation of shapes and their transformation procedures to be automated process through a set of modular and parametric instructions.

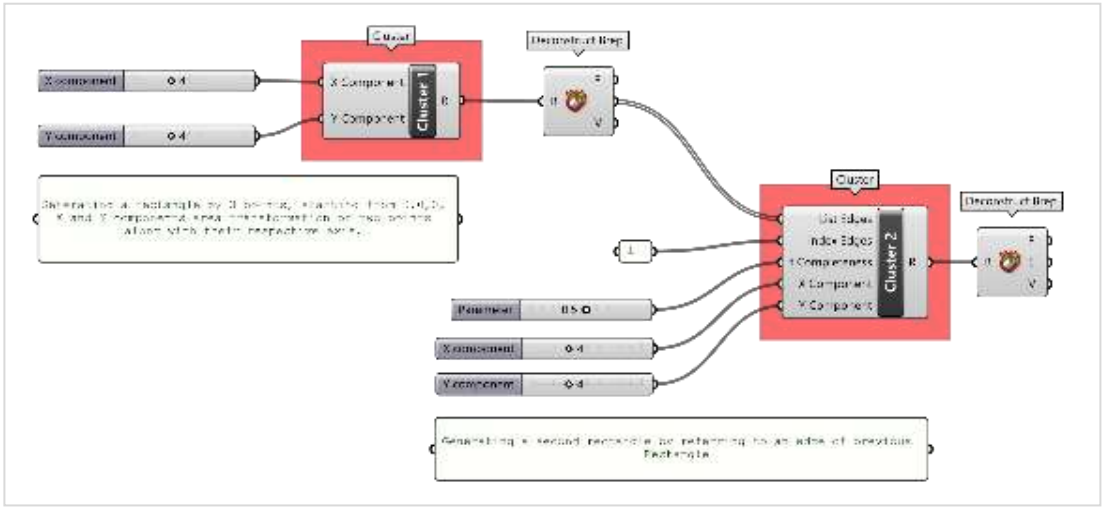


Figure 3. A preferred modular scheme of an algorithm

Theoretical basis

Our study provides literature expositions to show the theoretical basis of this research which mainly consists of Shape grammar theory and methodology. Shape grammar is a shape generation system and structure similar to a language grammar system. Shape Grammar was first formulated by Stiny and Gips (1972), who proposed a collection of rules or grammar associated with shapes generated composition of shapes by taking an initial shape and repeatedly or recursively implementing specific shape rules. The generation process stops when there are no more rules in the grammar to be implemented.

The set of shapes generated from a Shape Grammar is termed a shape language. Stiny (1980) showed four components forming a shape grammar as follows:
1) S, a bounded set of shapes.
2) L, a bounded set of symbols.
3) R, a bounded set of shape rules in the form of α and β where α is a labeled shape within (S, L) + and β is a labeled shape within (S, L) *; and
4) I, a labeled shape within (S, L) + termed an initial shape.

Stiny (1980) provided an example of a simple shape grammar with its shape generation process based on this shape grammar, as shown in figure 4 and figure 5.

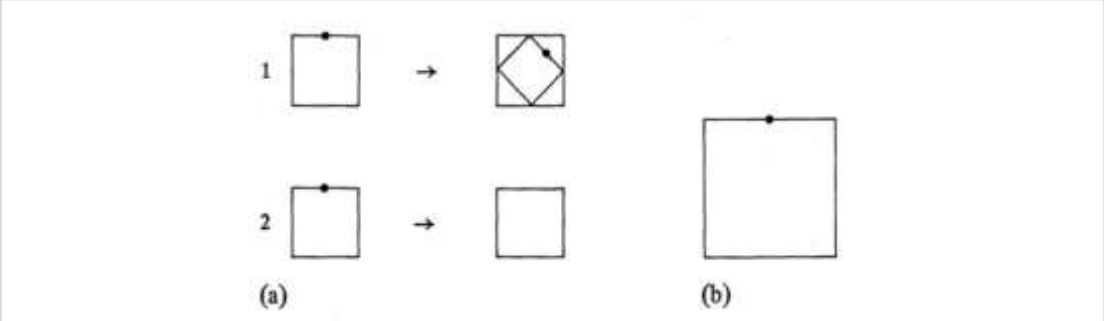


Figure 4. A simple example of shape grammar: (a) shape rules and (b) initial shape
Source: (Stiny 1980)

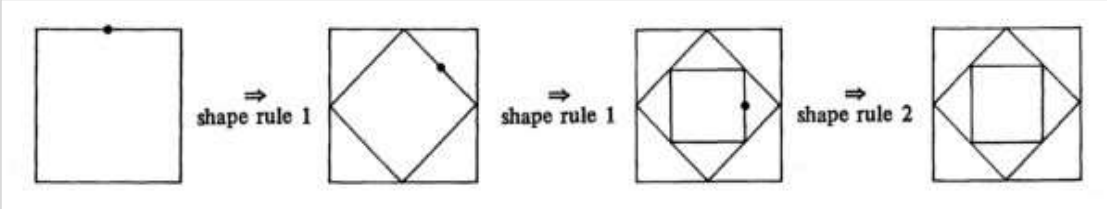


Figure 5. Shape generation process based on a simple shape grammar
Source: (Stiny 1980)

Stiny (2011, p. 17) noted that shape grammar could use any rules associated with specific shapes, but there are specific constraints of rules for a starting point. One part of the scheme shows uclidean transformation rules, consisting of translation, rotation, reflection (Stiny 1980), and dilatation (Boyd et al. 2008, p. 496). Another part of the scheme is a linear transformation consisting of strain and shear transformation (Martin 1982).

An exposition of a VPL is also provided in the following. Grasshopper is a visual algorithm editor within the rhinoceros software environment. It allows the user to construct an algorithm by placing and connecting functional blocks of instructions in a program editor (“canvas”). The data flows between these functional blocks ("components") through a "wire" that represents the flows of the program. The components and parameters take input values that are then processed to generate output values. In a more complicated program, parameters and components on the canvas become massive, complicated, and sometimes bewildering. To overcome this condition, grasshopper provides a method to combine a group of parameters and components within a cluster component. A cluster is a component of components that has a function to become a modular component to be used many times in a definition (e.g., algorithm), minimizing a messy and cluttered in the canvas.

Proposed parametric shapes rules
Conceptual model

The general concept of our proposed shape grammar parametric shape rules, named Rupa, automatically arranges the process of shape generation and transformation. The following are glossary used to define the variables and parameters of the program:

1. Geometry_refference (Geom_ref): initial rectangular shape.
2. Point_Refference (PR): a point located on Geom_ref designated to be the appointed point from Geom_Ref.
3. Point_Origin (PO): a point located inside Geom_Ref and has a specific rule derived from PR.
4. Point_X (PX): a point located inside Geom_Ref and has a specific rule derived from PO in the X-axis.
5. Point_Y(PY): a point located inside Geom_ref and has a specific rule derived from PO in the Y-axis.
6. Geometry_Result (Geom_res): a rectangular shape formed by three appointed points: PO, PX, and PY.

Shape transformation is completed by changing the parameters of these three appointed points (PO, PX, PY). The generated shape is two-dimensional, but the designer can input the height parameter to create a three-dimensional shape as well. The generated shape can be used as a geometric reference for the next iteration. These generated shapes and connected to one another will form a shape grammar that can be used for generating space configuration alternatives.

The conceptual process of simple shape grammar generation using the Rupa model is shown in figure 6 and figure 7.

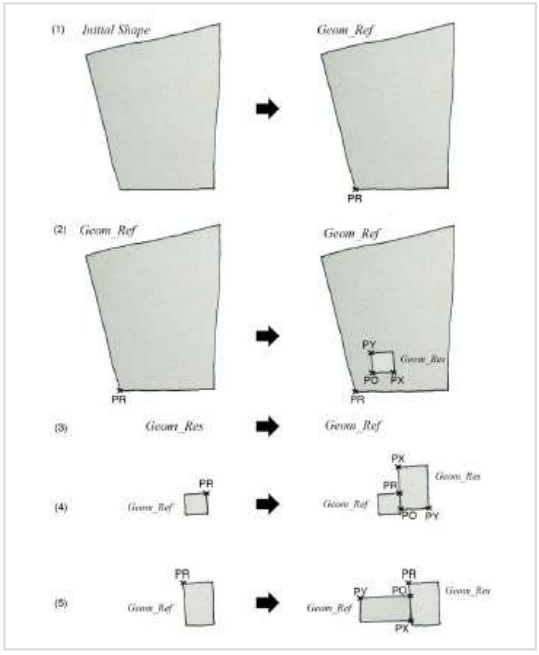


Figure 6. Shape grammar rules for a two-dimensional shape and its shape configuration

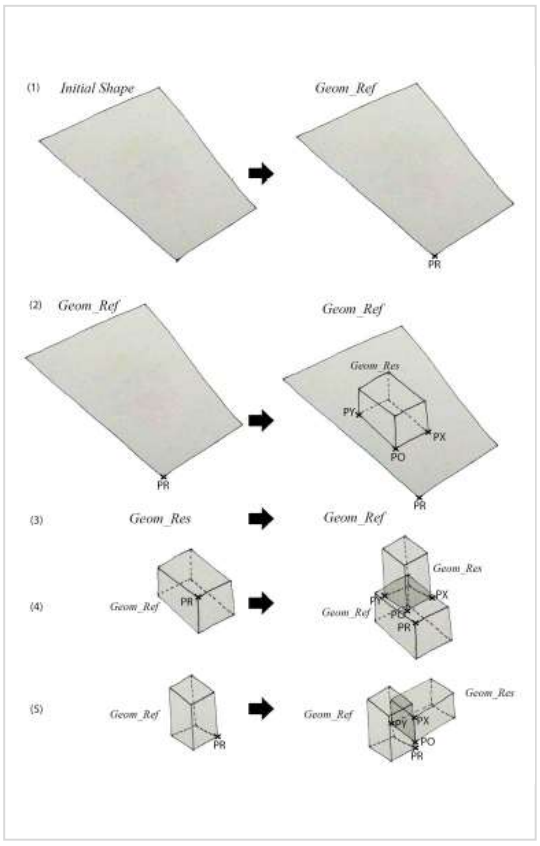


Figure 7. Shape grammar rules for a three-dimensional shape and its shape configuration

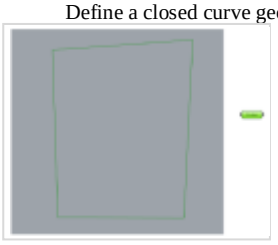
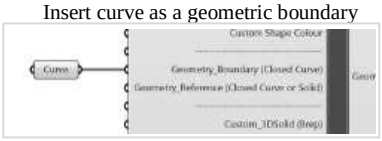
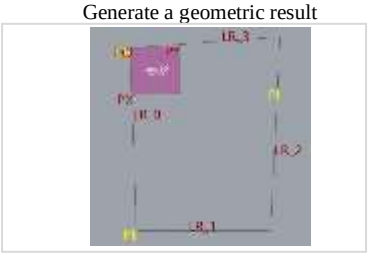
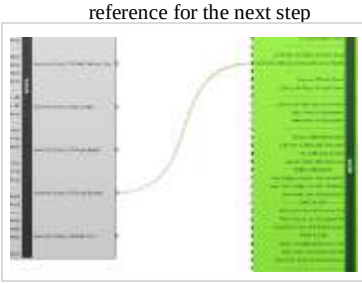
As seen in the conceptual sketches above, the parametric shape rules model considers rectangular configurations based on the rules of their respective vertices. The rectangular shape represents the functional dimensions required to compose a spatial configuration during a design process.

Parametric model in VPL grasshopper

Automation of the shape generation process by the shape grammar method is completed by using the Rupa clusters in the grashopper VPL editor. This cluster is a modular component to generate a transformed shape based on the input parameters and the shape grammar method. Figure 8 shows the Rupa cluster as in the grasshopper VPL editor and the parameter value inputs that can be changed later and the parameter value outputs that can be used for the next generation shape. The designer can change the value inputs to obtain a preferred shape configuration as seen in figure 8, table 1, and table 2.

Figure 8. A Rupa cluster in the grasshopper VPL Editor with the input and output parameters

Table 1. Step-by-step implementation of Rupa in two-dimensional geometries

Step	Two-dimensional
1	Define a closed curve geometry 
2	Insert curve as a geometric boundary 
3	Insert curve as a geometric reference on the next 'RUPA' Cluster 
4a	Generate a geometric result 
4b	(A two-dimensional geometry does not need to be extruded (height=0)
5	Insert the geometric result as a geometric reference for the next step 

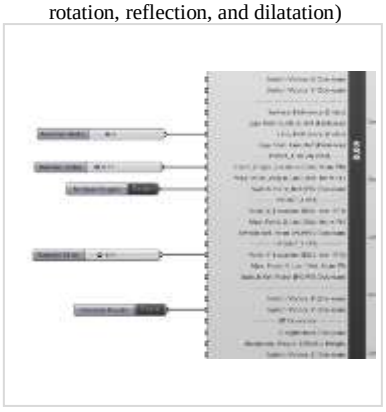
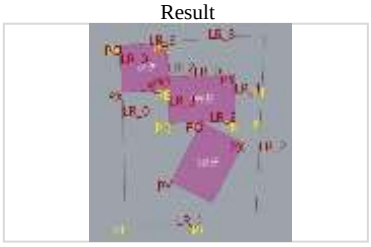
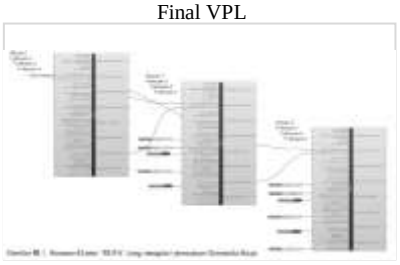
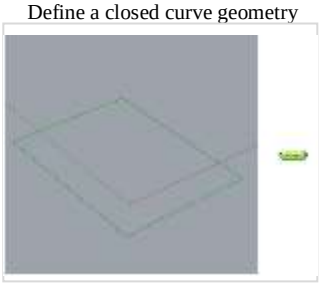
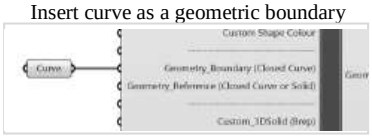
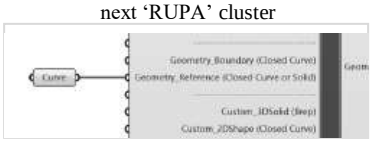
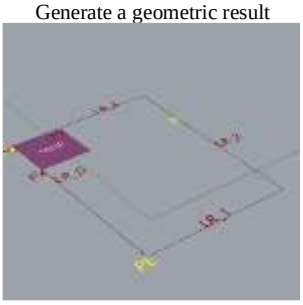
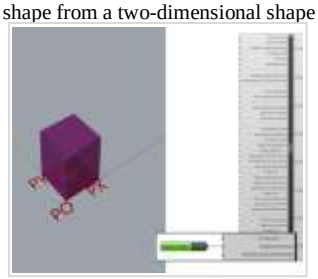
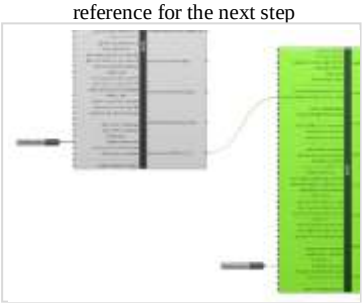
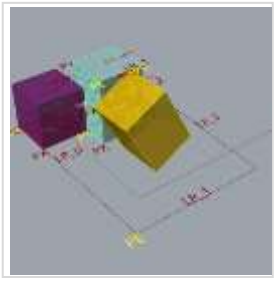
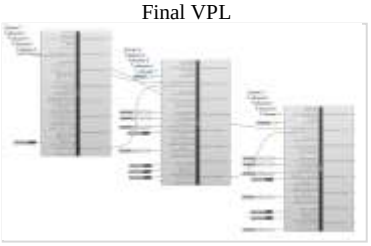
Step	Two-dimensional
6	Input transformation values (translation, rotation, reflection, and dilatation) 
7	Result 
8	Final VPL 

Table 2. Step-by-step implementation of Rupa in three-dimensional geometries

Step	Three-dimensional
1	Define a closed curve geometry 
2	Insert curve as a geometric boundary 
3	Insert curve as a geometric reference on the next 'RUPA' cluster 
4a	Generate a geometric result 
4b	Insert 'True' value for 'Heightened (Boolean)' to generate a three-dimensional shape from a two-dimensional shape 
5	Insert the geometric result as a geometric reference for the next step 
6	Input transformation values (translation, rotation, reflection, and dilatation)

Step	Three-dimensional
	
7	Result 
8	Final VPL 

Rupa as a parametric shape rules model developed and will be implemented to generate spatial configuration according to the geometric rules of specific floor plan design. In this study, we used a manual identification and analysis of geometric configuration over case studies. This manual approach includes the categorization of functional spaces (i.e., room), identification of space and circulation relationships, and identification and analysis of geometric configuration.

Methodology and mechanism of experiment

As indicated in the previous section, the following process is implementing the Rupa cluster to analyze floor plan designs based on the shape grammar method. We conducted this process over selected Adolf Loos's houses design as case studies the identification of Adolf Loos's residential design as objects for our experiment is based on the study by Sakurai (2018), who categorized Adolf Loos's residential design into four periods: 1) Early urban residences, 2) Residences in a natural environment, 3) Raumplan, and 4) The period of maturity in raumplan. Raumplan is a definition created by Heinrich Kulka to show Adolf Loos's method in generating space configuration inside a building (Sakurai, 2018, pg. 72). According to Sakurai (2018), loos designed a building as a volumetric object with optimal area and height. Each room is configured akin to a puzzle that fits with the volumetric object.

Several designs were chosen for this study which leads to the three final Raumplan methods. The chosen designs are as follows: 1) Cubic house, 2) Villa Muller, and 3) The last house. The shape grammar analysis of these objects was completed via two necessary steps:

1. Shape grammar identification from each analyzed design, and
2. Shape grammar configuration and rules determination

Phase 1: Shape grammar identification and analysis

On each floor plan design, we categorized functions (i.e., rooms) based on Lawrence's Three Binary Classification as follows: 1) clean/dirty; 2) day/night; 3) public/private and furthermore designated a color coding for each function. By observing the position of each room relative to other rooms that are connected by direct circulation, we can define and develop Shape grammar configuration rules manually. On each floor plan, circulation is rationally identified using arrows.

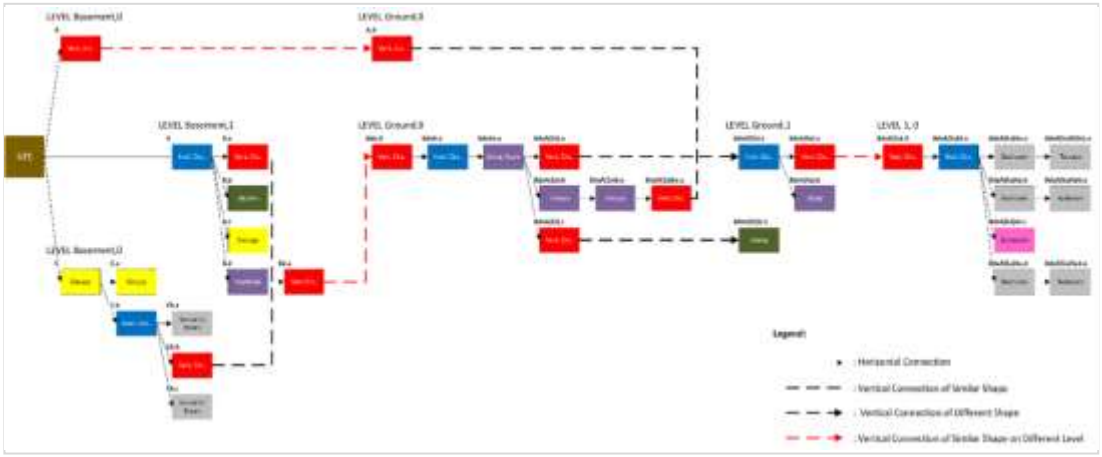


Figure 9. Shape configuration diagram of cubic house

The route taken for this circulation definition is coded. The outer room is coded with an uppercase letter (e.g., A, B, and C), and the inner room with a lowercase letter (e.g., a, b, and c). An example of diagrammatizations of floorplan is depicted in figure 9.

Lawrence's binary classification is used to determine hierarchical information of architectural functions. The result of this shape configuration diagram will be used to generate shape grammar-based rules by the Rupa cluster (table 3).

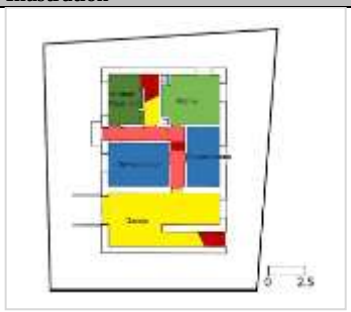
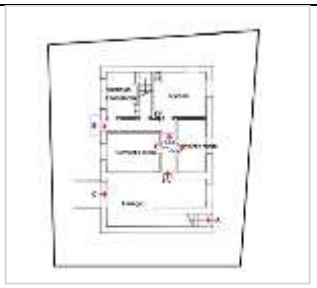
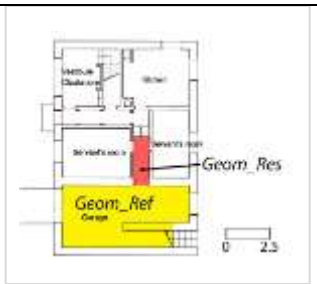
Table 3. Lawrence's binary classification

Room function	Colour
Vertical circulation (stairs and lift)	
Horizontal circulation	
Private/clean (e.g., bedroom)	
Private/dirty (e.g., bathroom)	
Public/clean (e.g., family room, library)	
Public/dirty (e.g., kitchen, dining room)	
Utility and service (e.g., garage, storage)	

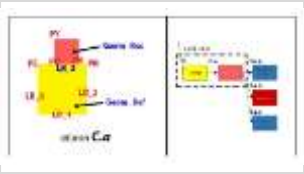
Example of implementation

An example of Rupa implementation to analyze the raumplan shape grammar and generate shape configurations is presented in table 4 and figure 10, a study case from the ground floor of the cubic house.

Table 4. Procedural steps to analyze a raumplan shape grammar.

Steps	Illustration
Shape grammar from the Lawrence (1981) binary classification	
Shape grammar configuration rule; code-named	
Geom_ref and *Geom res*, assignation rules	

Steps	Illustration
Shape grammar configuration	

Steps	Illustration
Shape configuration diagram	

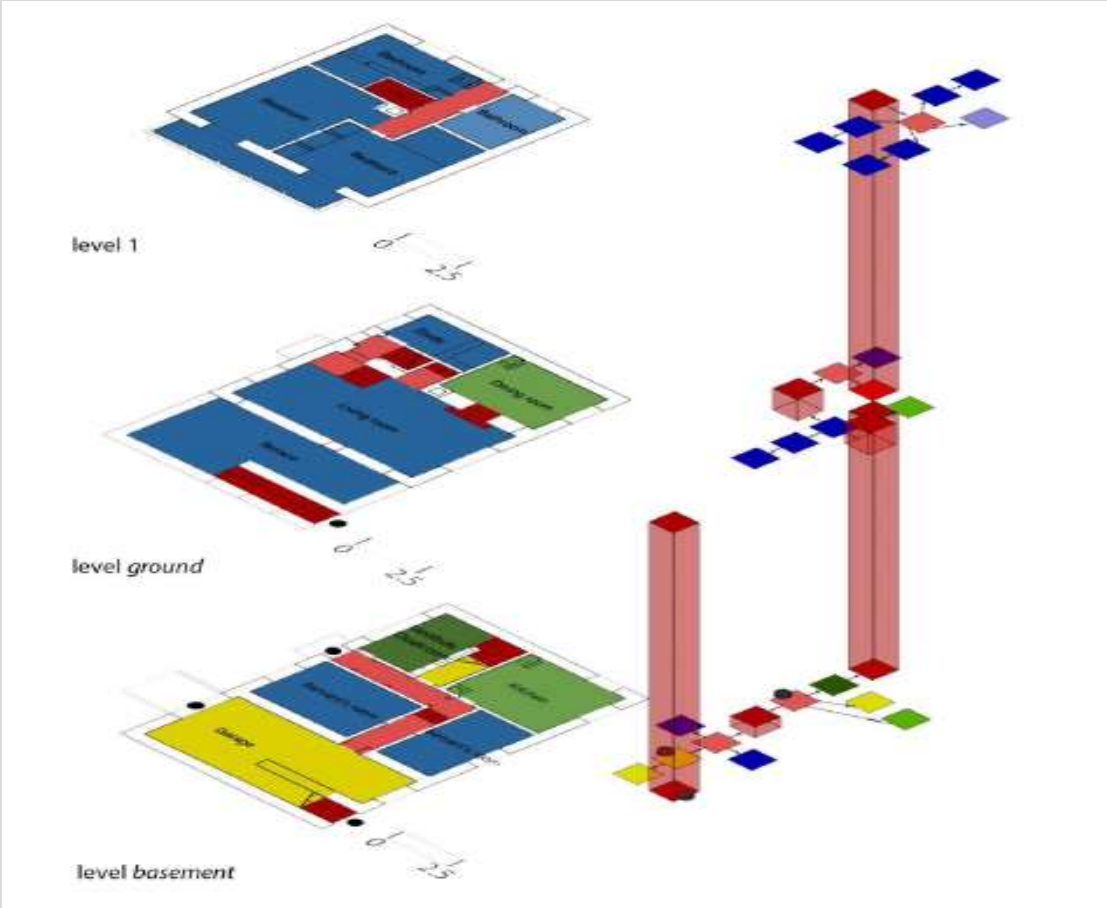


Figure 10. Shape grammar of the cubic house

Shape grammar configuration and rules determination in Rupa cluster

The result from the previous phase is used as a basis for shape grammar configuration and rules determination in Rupa cluster. This process is completed in two schemes based on the parameter value input methods as follows:

1. an independent scheme, in which each Rupa cluster input parameter does not have input from other Rupa cluster output parameters. In other words, the input parameter of each cluster is arbitrary data that will result in a more flexible shape configuration, and;
2. a dependent scheme, in which each Rupa cluster input parameter can be the input data from the output of the other Rupa cluster.

For the purpose of this study, we only examine dependent schemes to generate spatial configuration. Example results of the 3D shape configuration of the cubic house are depicted in figure 11.

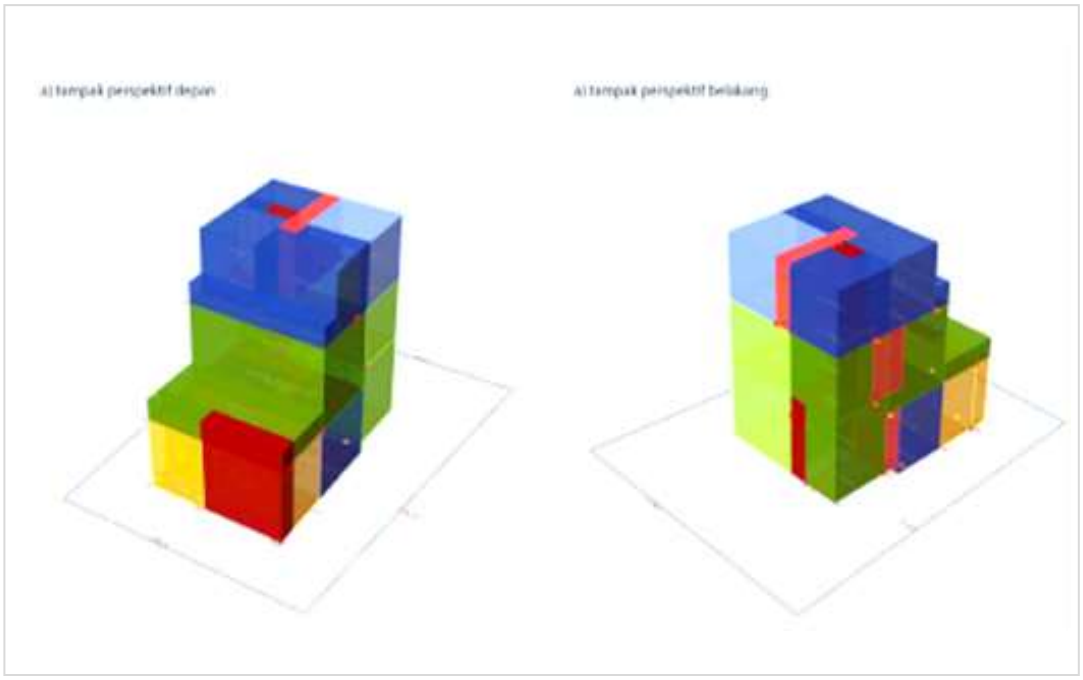


Figure 11. Example result of shape configuration of the cubic house by Rupa cluster

Design derivation generation based on the Raumplan shape grammar

The finished shape grammar arrangement can be used to generate a shape configuration of design derivation. The process can be completed in several stages and modes as mentioned; however, in this study, the strategy was to change the parameter value of selected key shapes inside a shape grammar that has been dependently created.

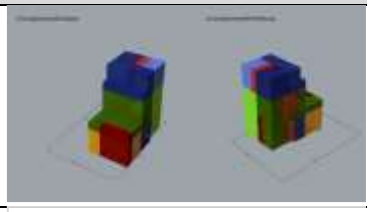
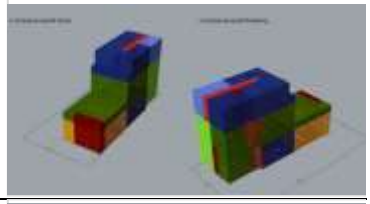
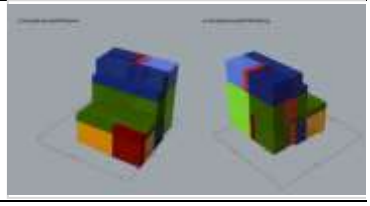
This strategy lessened the number of possible derivations and also ensured that the geometric proportion was similar to the original design. Otherwise, without any key shapes, the design would have a different proportion, and thus a disproportional shape.

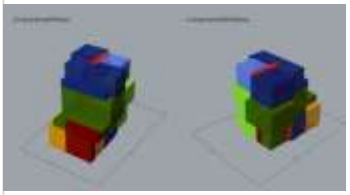
The design derivation is then categorized into three types of primary derivations based on on-site dimensions as follows:

1) minimum site area,
2) elongated site,
3) widening site, and
4) disproportional shape.

Table 5 shows design derivation generations in response to the specific type of site.

Table 5. Design derivation generation of cubic house

Case 1: Cubic house	
Site type	**Design derivation**
1 (Minimum area)	
2 (Elongated)	
3 (Widening)	

Case 1: Cubic house	
Site type	**Design derivation**
Disproportional Shape	

Results and discussion

Identification and analysis of floorplan using Rupa cluster

The first phase of utilizing Rupa as a parametric shape rules model is to give input according to the floorplan diagram. This labeled and structured information is the key to setting up a Rupa cluster and determining the arrangement of clusters in the definition of the algorithm in grasshopper VPL. Therefore, in our experiment, manual identification and analysis of the floorplan are needed to further differentiate the Rupa cluster according to the functions, levels, or combination of functions.

Shape grammar rules in Rupa cluster

As parametric shape rules, the Rupa cluster is used to generate shape configurations based on constraints or rules derived from the previous process. Here as a shape grammar design assistant role, composing Rupa clusters in the form of the algorithm in grasshopper VPL enables users to generate shape configuration constrained by the shape grammar rules embedded in Rupa clusters. Although this is a possible scenario, in theory, our experiment shows that the Rupa clusters configuration that accommodates shape transformations tends to grow messy (i.e., inefficiency in the algorithm). The more complex the floor plan is, the more complex the algorithm and complexity of Rupa cluster arrangements that it cannot perform effectively to generate spatial configuration.

However, as an experimental model for shape grammar design assistant, the Rupa cluster is developed to be responsive in an arbitrary closed curve (i.e., site boundary). It enables the user to generate various shape configurations within the boundary using various input parameters such as boundary setbacks, type of architectural functions, and the dimension of each function. This capability is significant for the early stage of the design process, where architects still develop conceptual design and ideation of spatial configuration.

Conclusions

This experimental study succeeded in creating a shape configuration based on the geometric rules of the shape grammar method. The analysis process succeeded in identifying the ahape grammar of three of Adolf Loos's residential works that were subsequently used to prepare the shape grammar-based method using the Rupa cluster. However, the analysis process required considerable time relative to the entire study time because the process must be manually completed in each Loos's residential work. The Rupa cluster enables users to create and transform two-dimensional and three-dimensional shapes by changing the provided parameter values and generating shape configuration parametrically.

Moreover, the shape grammar rules determined by Rupa clusters in the form of modules diagram in grasshopper VPL facilitate users to create design derivations of the spatial configuration. The users can change the parameter values on one or more key shapes to change and find the most suitable spatial configuration design.

The possibility of simplifying and automating the grammatical analysis of existing designs through artificial intelligence is a future research direction. A study that focuses on further interpretations of visual grammar due to the analysis using the Rupa cluster is also potentially possible. This further interpretation may be completed by reviewing special grammatical characters that are arranged by the Rupa cluster. The Rupa cluster provides sufficient information for this interpretative study in three-dimensional visual geometry, color information, labels, and relationships between clusters.

References

Benrós, D. & Duarte, J. P., 2009. An integrated system for providing mass customized housing. Automation in Construction, 18(3), 310–320.

Benrós, D., Duarte, J. P., & Hanna, S., 2012. A new Palladian Shape Grammar. International

Journal of Architectural Computing, 10(4), 521–540.

Blessing, L.T.M & Chakrabarti, A., 2009. DRM, a Design Research Methodology, Springer.

Broadbent, G., 1973. Design in architecture, John Wiley & Sons.

Boyd, C.J., Cummins, J., Malloy, C.E., Carter, J.A., dan Flores, A., 2008. Geometry, McGraw Hill/Glencoe.

Celani, G. & Vaz, C. E. V., 2012. CAD Scripting and Visual Programming Languages for Implementing Computational Design Concepts: A Comparison from a Pedagogical Point of View. International Journal of Architectural Computing, 10(1), 121–137.

Chau, H.H., Chen, X., McKay, A., & de Pennington, A., 2004. Evaluation of a 3D Shape Grammar implementation, in Gero, J.S., ed., Design Computing and Cognition'04, 357-376, Kluwer Academic Publishers.

Chase, S. C., 2002. A model for user interaction in grammar-based design systems. Automation in Construction, 11(2), 161–172.

Duarte, J. P., Rocha, J. M., & Soares, G. D., 2007: Unveiling the structure of the Marrakech Medina: A Shape Grammar and an interpreter for generating urban form. Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, 21(4), 317–349.

Eilouti, B. H. & Al-Jokhadar, A. M. I., 2007. A computer-aided rule-based mamluk madrasa plan generator. Nexus Network Journal, 9(1), 31–58.

Gips, J., 1999. Computer implementation of Shape Grammars, Workshop on Shape Computation, from: http://www.shapegrammar.org/implement.pdf. Accessed 29 January 29, 2018.

Hoisl, F. & Shea, K., 2011. An interactive, visual approach to developing and applying parametric three-dimensional spatial grammars. Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, 25(4), 333–356.

Kalay, Y.E., 2004. Architecture's new media, The MIT Press.

Knight, T., 1989. Color grammars: designing with lines and colors. Environment and Planning B: Planning and Design, 16(4), 417 – 449.

Knight, T. W., 1995. Constructive symmetry. Environment and Planning B: Planning and Design, 22(4), 419 – 450.

Knight, T. W., 1999. Shape Grammars: Six types. Environment and Planning B: Planning and Design, 26(1), 15–31.

Leitão, A., Santos, L., & Lopes, J., 2012. Programming Languages for Generative Design: A Comparative Study. International Journal of Architectural Computing, 10(1), 139–162.

Mandić, M. &n Tepavčević, B., 2015. Analysis of Shape Grammar application as a tool for urban design. Environment and Planning B: Planning and Design, 42(4), 675–687.

Martin, G.E., 1982. Transformation geometry: An introduction to symmetry, Springer-Verlag.

McKay, A., Chase, S., Shea, K., & Chau, H. H., 2012. Spatial grammar implementation: From theory to useable software. Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, 26(2), 143–159.

Pak, B., Ozener, O. O., & Erdem, A., 2006. Utilizing customizable generative design tools in digital design studio: Xp-GEN experimental form generator. International Journal of Architectural Computing, 4(4), 21–33.

Ruiz-Montiel, M., Boned, J., Gavilanes, J., Jiménez, E., Mandow, L., & Pérez-De-La-Cruz, J. L. ,2013. Design with Shape Grammars and reinforcement learning. Advanced Engineering Informatics, 27(2), 230–245.

Sakurai, Y. (Ed.)., 2018. Feature: Adolf Loos Residences. Architecture and Urbanism, 572.

Stiny, G., 1980. Introduction to shape and Shape Grammars. Environment and Planning B: Planning and Design, 7(3), 343–351.

Stiny, G., 1981. A note on the description of designs. Environment and Planning B: Planning and Design, 8(3), 257–267.

Stiny, G., 2011. What Rule(s) Should I Use? Nexus Network Journal, 13(1), 15–47.

Stiny, G. & Gips, J.,1972: Shape Grammars and the generative specification of painting and

sculpture. Information Processing 71 Proceedings of the IFIP Congress 1971, 2(71), 1460–1465, from: http://www.shapegrammar.org/ifip/SGBestPapers72.pdf. Accessed 5 February, 2018.

Stouffs, R., 2018b. Description grammars: Precedents revisited. Environment and Planning B: Urban Analytics and City Science, 45(1), 124–144.

The Grasshopper Primer, from: http://grasshopperprimer.com/en/index.html. Accessed 18 February, 2018.

Wang, J., Zhao, J., Wu, T., & Li, J., 2017. A Co-Evolution Model of Planning Space and Self-Built Space for Compact Settlements in Rural China. Nexus Network Journal, 19(2), 473–501.

Yazar, T. & Colakoglu, B., 2007. QSHAPER. in Predicting the Future: 25th eCAADe Conference Proceedings. Faculty of Architecture and Civil engineering, FH Wiesbaden.

Author(s) contribution

Alva F. P. Sondakh contributed to the research concepts preparation, methodologies, investigations, data analysis, visualization, articles drafting and revisions.

Aswin Indraprastha contribute to the research concepts preparation and literature reviews, data analysis, of article drafts preparation and validation.