PARAMETRIC DESIGN PROCEDURES: A NEW APPROACH TO GENERATIVE-FORM IN THE CONCEPTUAL DESIGN PHASE.

Abdullah, H. K and Kamara, J. M[1].
hardibarznji@yahoo.com and john.kamara@ncl.ac.uk
School of Architecture, Planning and Landscape, Newcastle University, Newcastle upon Tyne, NE1 7RU

ABSTRACT

The conceptual design stage often involves a compound set of objectives and constrains such as abstract notions of function and aesthetic, performance, project requirements, site constrains and construction costs. To respond to these complexities, a number of design instances and alternatives need to be developed and assessed against predefined criteria. While this process requires human imagination, computational generative systems are increasingly being used in this stage of the design process. However, some of these approaches have limitations in the ability to make modifications within an interactive environment, requiring a model to be recreated with different attributes and parameters if changing geometry configuration or topology are needed. This research introduces a new approach – Parametric Design Procedures (PDPs) – which combines the techniques of Design Procedures and Parametric Modeling to address the limitations of existing systems. PDPs offer possibilities to explore a particular design instance after a model is constrained through the generation of an infinite number of design instances which can be considered in the evolution of parametric design instances. The rational for, and features of PDPs are described. The viability of this approach is explored through a prototype implementation in Grasshopper. The brief for an architectural design competition is used as the basis for the prototype development. The paper concludes with suggestions for further research and development, for example in the use of other software and other design phases to test the implementation and viability of PDPs.

Keywords: conceptual architectural design, generative systems, parametric design procedures,

INTRODUCTION

The term conceptual architecture is used to characterize a particular design or process that uses conceptualism in architectural design. It is an abstraction that filters unnecessary details and simplifies the object while elements of components and relations among them are determined. The representations at this stage should support various interpretations of design elements while simultaneously allowing them to be adjusted through the use of multiple methods (Emdanat, 1998).

[1] Corresponding author

Basically conceptual design is considered as a challenging stage of the design process in which architects often face a compound set of objectives and constrains. These include abstract notions of function and aesthetic, ecological performance, project requirements and construction cost. Some of the requirements of the conceptual design phase are:

- Generate and explore a huge number of possible design solutions,
- Test and evaluate generated solutions based on predefined criteria,
- Overcome human mental imaginary restrictions,
- Create imaginative forms and creative ideas
- Productivity with less time consuming.

To resolve these complexities, a number of alternatives need to be generated and tested against predefined criteria in order to select the most appropriate option(s) for further design development (Gane and Haymaker, 2007). While this process always requires human imagination, computational generative systems are increasingly being used in this stage of the design process. Such systems include: Transformations of Shape Grammars, Algorithmic design and parametric generative-design. However, all these systems have advantages and limitations regarding generative capabilities, support for complex design and system control by designers. The advantages include the development of satisfactory enhancement of parametric modeling. The limitations however, include the inability to respond to the needs, difficulties of the conceptual design stage, and the requirements for parametric modeling.

This paper reports research that was aimed at demonstrating parametric design procedures as an emergent computational methodology to form generation, complex form finding and formal explorations in the conceptual design phase. This approach was assessed based on the requirements of the conceptual design stage as well as the goals of parametric modeling. The specific objectives of the research were:

- To define needs and difficulties of the conceptual design phase.
- To define goals and objectives of parametric modeling.
- To review current approaches and identify problems as well as limitations.
- To develop an application that demonstrates the use of this technique.
- To assess and validate the capability of PDPs based on defined above needs and goals

The rest of the paper includes a literature review to determine needs and problems of the conceptual design phase and to define the goals and objectives of parametric modeling. Current approaches of generative systems and recent attempts to resolve the limitations of parametric modeling as also reviewed. The concept of PDPs is defined and its application demonstrated through a prototype application using Grasshopper. The paper concludes with a discussion of the findings and recommendations for further research.

GENERATIVE DESIGN SYSTEMS AND PARAMETRIC DESIGN

In their paper on interactive generative systems, Eckert et al. (1999) argue that generative systems can be considered as an artificial intelligence aid to support humans in achieving creativity especially when such systems are used in an interactive way with designers.

Generative design can generate forms automatically; therefore, designers can generate a huge number of design solutions and explore them in such huge design space within a minimum time as well as evaluate their performance based on a predefined framework. Consequently, generative design will become an evolution of exploring forms which is dramatically essential in the conceptual design phase. There are several methodologies that provide generative designs such as Genetic Algorithms, Shape Grammar and Parametric Techniques. D'Arcy Thompson (Thompson, 1961) in his book 'On Growth and Form' discusses the study of form which can be descriptive or analytical. The method of Cartesian transformations originated from the method of Co-ordinates which was previously used as way to translate the form of a curve into numbers and then into words. Thompson's concept on the study of forms lies in the comparison of related forms instead of a mathematical definition of each deformation. He thought that a form can be better understood by observing it as a deformation of another form, thus two forms can be compared by Cartesian net; this provides a visual framework for the morphologists to understand the relations between forms and transformations as they occur (Figure 1).

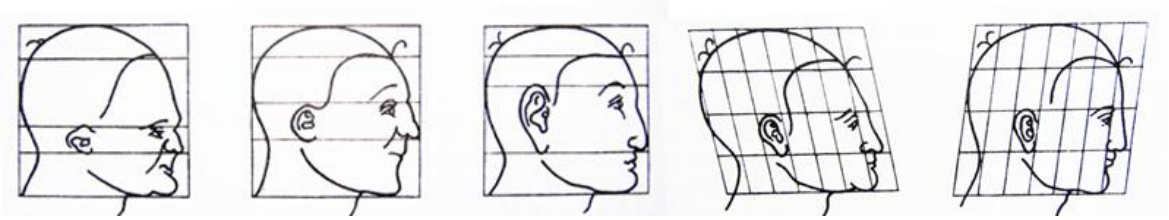

Figure (1): Variations generated using co-ordinates method of Cartesian Transformations (Source: Thompson, 1961).

Transformations of shape grammars are also generative techniques which were originally invented more than three decades ago by Stiny and Gips. Shape Grammars are shapes, computation and languages of design; they are also considered as one of the earliest algorithmic tools in which design can be created and understood directly via computations with shapes instead of indirectly via computation with symbols and text (Stiny, 1976). According to Terry Knight (Knight, 2000), "*a shape grammar is a set of shape rules that apply in a step-by-step way to generate a set, or language, of designs. Shape grammars are both descriptive and generative. The rules of a shape grammar generate or compute designs, and the rules themselves are descriptions of the forms of the generated designs*." The computations start with an initial shape and then applied rules onto the initial shape creates a new design. However, shape grammars look deterministic in restricting rules, thus generated instances might be predictable. It also does not allow transformations at the level of geometry and topology, to generate totally different design solutions.

Originating in other design related fields such as product design, aerospace and automotive design, parametric design has become prevalent in the Architecture, Engineering and Construction (AEC) industries, and architects have implemented parametric design in architectural design (Eastman et al., 2011). Parametric design is also called "associative geometry" (Burry and Murray, 1997 cited in Minh, 2009, p. 1), "Variational design, constraint based design or relational modeling" (Monedero, 1998, p. 158) controlled by parameters and constrains via assembly of associative operations.

When architects alter parametric values to explore various alternative solutions for a particular problem, the model will respond to modifications through automatically updating itself without deleting or remodeling any elements (Stavric and Marina, 2011). According to Burry and Murray (1997, p.1) "parametric modeling software is invaluable for both preliminary and developed design where there is a need for the definition, manipulation and visualization of complex geometry". Some of the most significant goals of parametric modeling are: flexibility; adaptability; modification without the need to delete or remodel; providing solution spaces to be explored; less time consuming; quick in responding to changes and updating of the whole model; and working with the historical based system where designers can come back at any stages.

On the other hand, parametric modeling only allows variations, which allow the generation of related forms within the same family of forms; it does not allow topological and geometrical transformations to generate an infinite number of design solutions. It is also limited in its flexibility to allow the generation of sophisticated forms and curvilinear surfaces. The development of *'Design Procedures'* has been proposed to overcome some of the limitations of parametric modeling. However, this approach also has limitations, such as restricting some kind of transformations. Furthermore, the need for scripting knowledge to more fully exploit the benefits of design procedures is usually beyond the designers (architects) who are involved in the conceptual design phase. Design procedures must also be designed to solve a specific problem, for example, the generating columns of the Sagrada Familia church, rod symmetry and twisted towers (Barrios, 2006). As Barrios (2005, p. 25) observes, "…design can be described as a step by step process, where some things can occur over and over again. This can be interpreted as a procedural way of making design." This suggests that design procedures is a step by step process to be followed and it is a kind of generative system which the designer does not have full control over; the events and the design process should be flexible so that designers can start from anywhere or any step and come back to previous steps when needed. These problems make such approaches less viable to more fully address the needs of designers in the conceptual design phase. All of the reviewed methodologies are attempts to provide architects with a flexible and powerful environment. The development of a new approach which combines the advantages of various approaches is therefore a viable way forward.

PARAMETRIC DESIGN PROCEDURES (PDPs)

The nature and complexity of the conceptual design stage as well as the demand to generate various design solutions within the same model without the need for programming knowledge led to the idea of incorporating Parametric Design (PD) with Design Procedures (DP) and other significant generative methodologies to introduce a new approach in the name of Parametric Design Procedures (PDPs). This incorporation can be seen as taking viable aspects of PD and DP to overcome the limitations of parametric modeling and DP. PDPs use parameters (e.g. initial shapes, variables, operations, numbers and relationships) as inputs, and calculate them through an encapsulated mathematical process to interactively generate and explore solutions for the

design problem. Lecky-Thompson (2006) defines a procedure as "*a named block of code, like a subroutine, but with some additional features. For example, it can accept parameters, which might be input, output, or pass-through*." PDPs use encapsulated codes in the form of visual features without the need to use scripts. Unlike DP, which only supports generative forms which are designed for, PDPs support all kind of designs which means that a parameterized model can be used for many formal explorations. In spite of using initial shapes as parameters, in PDPs shapes have the 2D and 3D transformation capability of other shapes, including non-closed shapes 'which is a condition in Design Procedures that shapes must be closed'. This addresses topological and geometrical transformation limitations of PD and DP. Architects can also switch between operations such as extrusion, rotation, scaling, twisting, etc. to generate completely different instances within the same model, and explore more options – an essential requirement of the conceptual design phase. PDPs works with a historical based platform, and a designer can come back to any particular step to do further modifications. Moreover, Architects have the entire control over the generation procedures thus the generation procedure will allow the combination of automated and non-automation computational procedures. In other words, design instances can be generated interactively via altering parametric values from the beginning to the end of the design process. This approach supports the generation of all kinds of non-Euclidean forms and curvilinear surfaces, which are needed for architectural design in this digital age. Although, the PDPs approach is a computation methodology, the conceptual design process usually starts with some initial steps which are rather manually performed. These include the identification of problems and goals, and initial sketching of 2D shapes and 3D forms which can be moved into a digital environment using the PDPs approach to generate and explore forms. In this context, these constrains can be considered as inputs to the computational framework. Figure (2) shows the system of PDPs as a computational methodology to form generation.

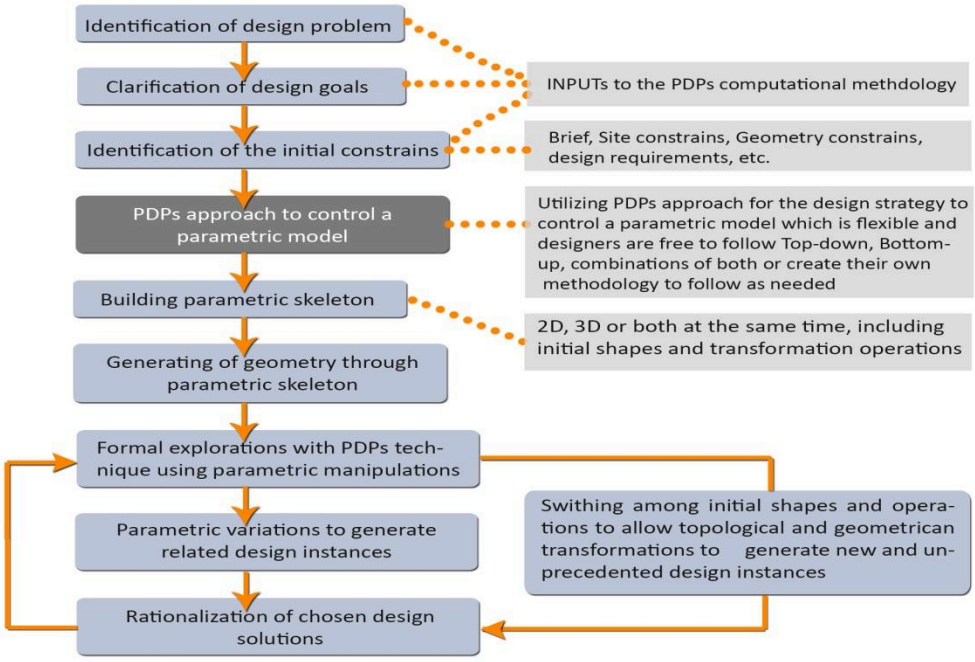

Figure (2): System of PDPs as a computational methodology to generative-forms of the conceptual design.

RESEARCH METHODOLOGY

The aim of this research was to explore the extent to which parametric design procedures (PDPs) can be used as a computational methodology to generative form in the conceptual design stage. The specific questions were:

- How can PDPs be used as a computational design generative system?
- How can PDPs allow designers to formalize and generate solution spaces that can be explored?

To answer the research questions an application was developed and evaluated with respect to the needs and goals of the conceptual design stage and parametric modeling. Grasshopper 3D was chosen as the software to design and develop the application of PDPs. This was because currently there is no suitable software like Grasshopper to support the PDPs approach. To ensure that the evaluation reflected a realistic scenario with respect to site, aesthetic and other criteria, the brief (program) for architectural competition (*USA: The 2nd Annual International Student Tall Building Design Competition)* was selected as the context. Furthermore, a real site context was selected to design and locate the competition on. The use of self-evaluation in the testing of the application has its limitations, but it was not possible during the timeframe of the research to involve other specialists in the validation process. However, the process adopted has yielded useful insights into the potential benefits of PDPs.

IMPLEMENTATION OF PDPs USING GRASSHOPPER

The competition brief required imaginative ideas which basically need formal explorations. The goals of this project was to design a tower which could be used as a multipurpose building considering aesthetic, flexibility, adaptability, technology, imaginative ideas, materials and digital revolution. Therefore it was necessary to generate and explore forms to respond to these requirements. For the chosen site context there are basically some geometrical constrains including the built area of tower which could be constrained between 225-625m^2, and number of floors and the height of the tower are constrained between 20-50 floors (70-175m height), height of each floor is constrained between 3-4m. Within these constrains the values can be parametrically manipulated later. The work started with thinking about initial shapes, 3D forms and concepts considering the site context and brief requirements. The initial concepts are used as inputs to the computational generative steps in the PDPs system. Computational works start with choosing Grasshopper software (*Grasshopper is a visual programming language originally developed by David Rutten at Robert McNeel & Associates*) and the setting up of parametric values with initial shapes, operations which are parametrically controlled and can be switched to any other shapes or operations (within the same model and definition) when searching for various alternatives. Figure (3) shows generated initial shapes with 2D transformations on the level of topological and geometrical using the same model without remodeling or deletion. The generating process is very easy and quick so that designers generate infinite numbers of design solutions in very short time. Moreover, the designed Grasshopper definition can be easily used to generate forms of other projects which mean the designer becomes a half programmer at the same time.

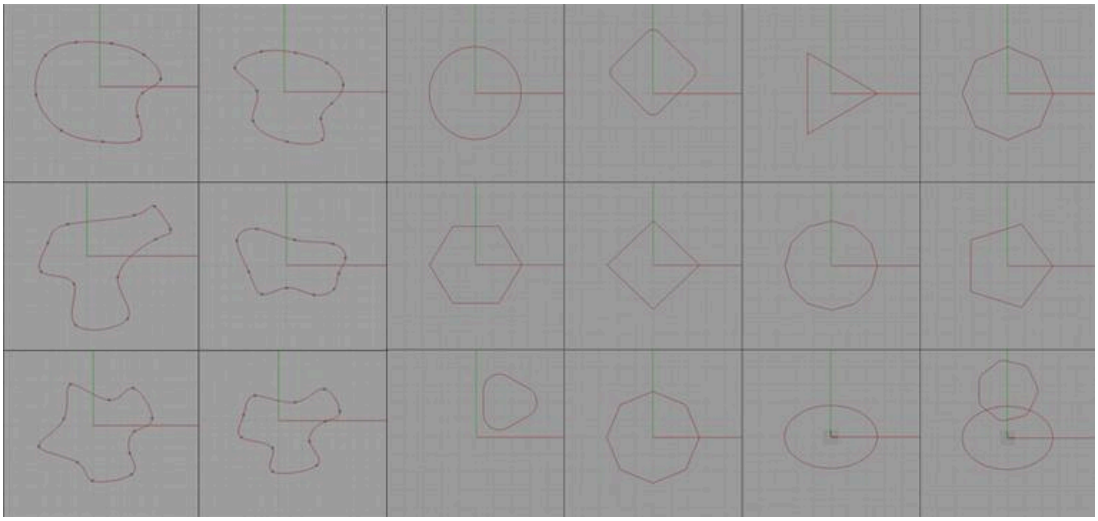

Figure (3): Parameterized values and generated initial shapes using the same model.

Figure (3) shows the capability of PDPs in switching from one shape to another, including any type of curves (e.g. NURBS curves, Polylines etc.), and also allowing closed non-closed to be processed. The shape can be controlled based on plane coordination so that the whole model can be easily moved around. The next step is setting up parametric skeleton, which are fully controlled parametrically. Here we can get flexible initial shapes, number of floors, height of floors and the whole building with all other parametric features including values and operations. The skeletons are generated within the same model and can be used to generate forms at a later time.

After building a geometry on the skeleton, other design instances can be generated via variations, and changing the entire configurations of the model, which is required in the conceptual design phase to assess as many solutions as possible. The design instances in Figure (4) show the capability of PDPs in generating various design instances within the same parameterized model without any deletion through switching among different initial shapes and operations which allow topological and geometrical transformations.

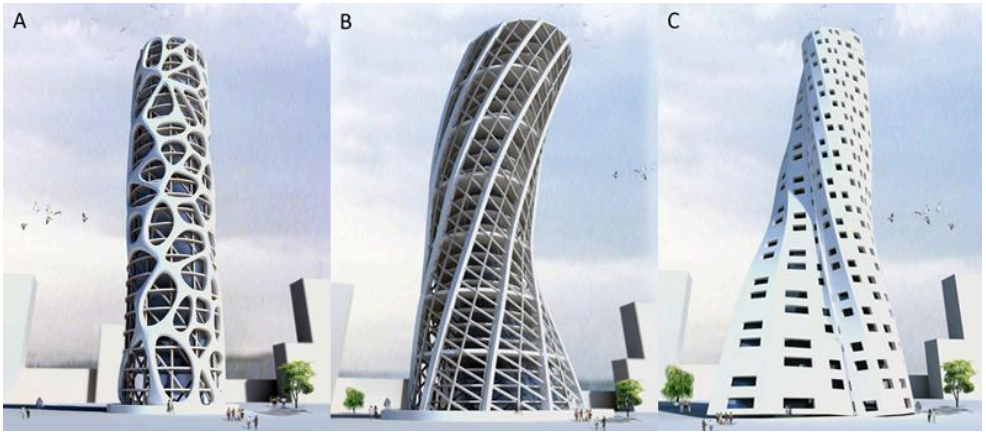

Figure (4-A, B and C): Show three different design solutions generated within the same model.

DISSCUSSION

The results suggest that PDPs can overcome limitations of previous approaches. Designers can start with an infinite number of initial shapes to create a parametric skeleton and create a parameterized model on the skeleton to start with generating and exploring potential design solutions. After that, the initial shapes, parametric values and operations can be easily and quickly modified to achieve the goals of parametric modeling and overcome restrictions of geometrical and topological transformations (Figure 5).

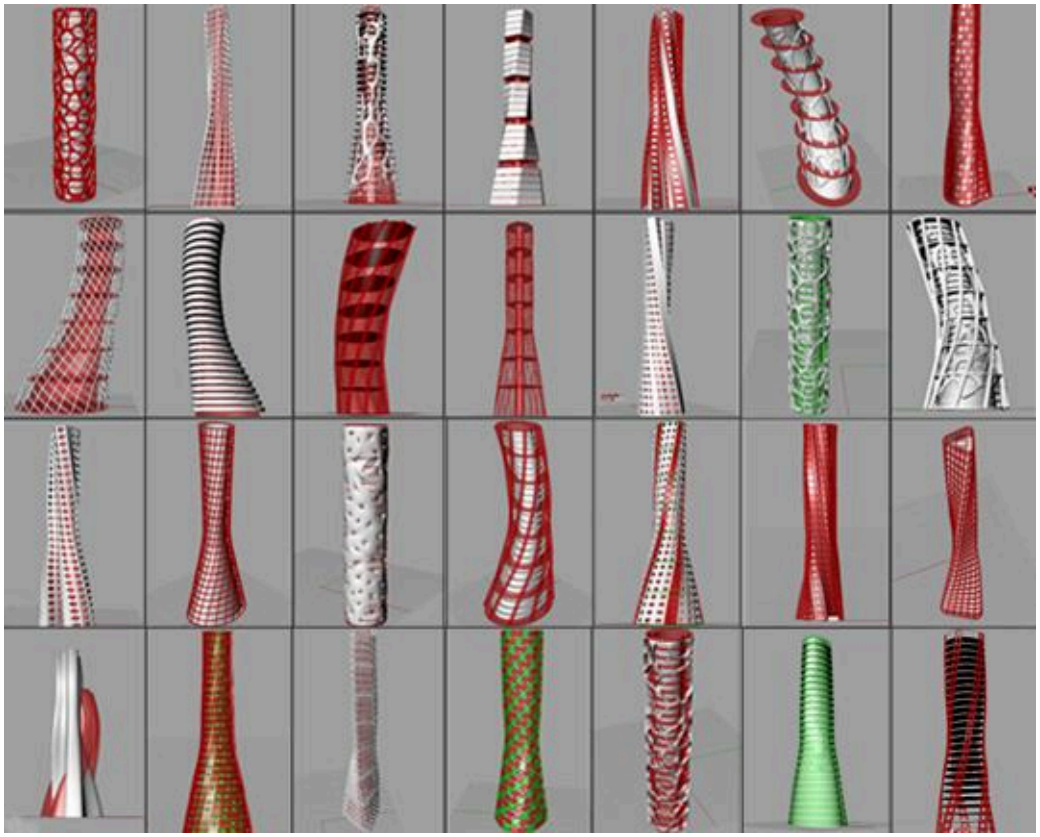

Figure (5): Different design instances generated using the same model.

Compared to other computational generative methodologies such as Transformations of Shape Grammars, the results suggest that PDPs were successful in providing an appropriate environment to generate unpredictable design solutions and designers have full control in a parametric way over all events. Compared to Design Procedures, PDPs provide designers another level of complexity which is easy to create and updating can be done automatically when changing any parametric feature (unlike DP which is very basic and does not allow designers to generate complex layers without writing a script). Also as the results indicate, PDPs generate all those design solutions using the same definition; using Design Procedure, designers must write codes to create any particular design. On the other hand, considering the difficulties and needs of the conceptual design phase, the

results demonstrate that PDPs have the potential of addressing the needs of this phase of the design process (e.g. generating, exploring, and testing multiple design instances using the same parameterized model).

PDPs were also successful in promoting parametric design through achieving goals of parametric modeling and overcoming current restrictions of parametric design. For example, in allowing 2D and 3D transformation at any time (potentially overcoming topological and geometrical limitations of conventional parametric design), allowing any additional layers at any time during the design process without breaking the built model, creating multiple number of complex design solutions, and allowing designers to interact with the entire design process.

It should however be pointed out that the assessment of the potential benefits of prototype, although based on the predefined goals of parametric modeling and the needs of the conceptual design phase, was based on self-evaluation; a more robust evaluation (e.g. by professionals and potential users) is required to ascertain the real benefits of the system. Another limitation was that only one development environment (Grasshopper) was used; the implementation of PDPs in multiple environments will provide a better assessment of its capabilities in meeting the objectives for which they were developed.

CONCLUSIONS AND RECOMMENDATIONS

The research was aimed at developing a new computational methodology for form generation focusing on the conceptual stage of design process. This was pursued through an identification of the goals and objectives of parametric modeling, the particular features of the conceptual design phase, and the limitations of existing generative systems. Results showed that, PDPs have the potential of achieving the goals which they were designed for. PDPs provide designers with an environment that design solutions can be easily and quickly generated in a relatively short time using the same parameterized model. Although the assessment of the approach was based on predefined goals of parametric modeling and needs of the conceptual design phase in the design process, the assessment was a kind of self-assessment which can be seen as a limitation of the study. Another limitation was related to the implementation environment as currently only Grasshopper software supports this approach. The need to develop and assess PDPs using other software environments and tools is a logical step for future research.

As a recommendation, PDPs as a collaborative computational methodology for form-generation can be applied to parametric systems and software which are used by architects. Given the potential of PDPs in the conceptual design phase, it is reasonable to assume that it will be appropriate to other phases of the design process. However, further research is required to explore this.

REFERENCES

Barrios, C. (2005) "Transformations on parametric design models: A Case Study on the Sagrada Familia Columns. In: the 11th International CAAD Futures Conference, *Computer Aided Architectural Design Features.* Vienna, Austria, 20–22, Vienna: the Vienna University of Technology.

Barrios, C., (2006). Thinking Parametric Design: Introducing Parametric Gaudi. *Design Studies*, 27(3), pp 309-324.

Burry, M. and Murray, Z., (1997). Architectural design based on parametric variation and associative geometry. In*: the 15th eCAADe Conference, Challenges of the future.* Vienna, Austria, 17-20 September 1997. Vienna: Österreichischer Kunst- und Kulturverlag.

Eastman, C., Teicholz, P., Sacks, R. and Liston, K., (2011). *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contactors*. Hoboken, NJ: John Wiley & Sons, Inc.

Eckert, C., Kelly, I. and Stacey, M., (1999). Interactive Generative Systems for Conceptual Design: An Empirical Perspective. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. 13 (1999), pp. 303-320.

Emdanat, S. (1998). An Ontology for Conceptual Design in Architecture. *Proceedings of the Third Conference on Computer Aided Architectural Design Research in Asia*, (eds.) T. Sasada, S. Yamaguchi, M. Morozumi, A. Kaga, and R. Homma. Pp. 425-434. Osaka University, Osaka, Japan.

Gane, V. and Haymaker, J., (2007). Conceptual Design of Highrises with Parametric Methods. In: the 25th eCAADe Conference, *Predicting the Future*. Frankfurt, Germany, 26-29 September 2007. Frankfurt: Am Main

Koile, K. (2004). An Intelligent Assistant for Conceptual Design, Informed Search Using a Mapping of Abstract Qualities to Physical Form. JS Gero (ed), *Design Computing and Cognition'04*, 3-22. Dordrecht: Kluwer Academic Publishers.

Knight, T., (2000). *Shape grammar*. [Online] Available at: <http://www.mit.edu/~tknight/IJDC/frameset_introduction.htm> [Accessed 14 September 2000].

Lecky-Thompson, G., (2006). *Procedure, subroutine or function?* [online] Available at: <http://suite101.com/article/procedure--subroutine-or-function--a8208> [Accessed 25 October 2006].

Monedero, J., (1998). The role of the architect in the age of automatic reproduction. In: the 16th eCAADe Conference, *Computer Craftsmanship in Architectural Education*. Paris, France, 24-26 September 1998. Paris: eCAADe.

Stavric, M. and Marina, O., (2011). Parametric Modelling for Advanced Architecture. *International Journal of Applied Mathematics and Informatics*, 5(1), pp 9-16

Stiny, G., (1976). Two exercises in formal composition, *Environment and Planning.* B (3), p. 187-210.

Stiny, G., (1976). Two exercises in formal composition, *Environment and Planning.* B (3), p. 187-210.

Thompson, D. W., (1961). *On Growth and Form*. Abridged edition, edited by J. T. Bonner, foreword by S. J. Gould. Cambridge University Press: Cambridge.