Out of the box design: Pedagogical approach on generative teaching
van der Zee, A.; de Vries, B.

Published in:

Published: 01/01/2011

Document Version
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

- A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Out of the box design

Pedagogical approach on generative design teaching

Aant van der Zee¹, Bauke de Vries²
¹Eindhoven University of Technology, the Netherlands, ²Eindhoven University of Technology, the Netherlands.
¹a.vd.zee@tue.nl, ²b.d.vries@tue.nl

Abstract. A traditional design studio is organized round tutors who give the students an assignment, more precisely a design problem which the students have to solve in 12 weeks. Since 2006 we run at our University a design studio which is focused on a new way of thinking in and about architecture. In many aspects the organization differs greatly from the more traditional organized design studios.

In the first part of the paper we will discuss the pedagogical organization of ‘our’ studio and how this new way of generative design is used in architectural training.

In the second part we will show and discuss some students work, one project will be discussed in more detail.

Finally, we will summarize our experience with this design studio and provide some guidelines for successful implementing Generative Design in architectural design teaching.

Keywords. Generative design; algorithmic design; teaching.

Introduction

Recently there is a renewed interest in the use of computer software during the form finding phase in the architectural design process. Terzidis (2008) refers to this renewed interest as the shift from computerization to computation. According to Terzidis (2008) computerization is the act of “entities or processes that are already conceptualized in the designer’s mind are entered, manipulated, or stored on a computer system.” In contrast, computation or computing, shapes aren’t manipulated by the designer but generated by the computer. Mouse-based operations on objects, as done in well-known CAD applications, such as AutoCAD, Revit, ArchiCAD etc, aren’t computations but belong to the ‘act’ of computerization. By using the embedded scripting languages of CAD-applications like VBA or AutoLisp in AutoCAD, the designers go beyond the limitations of the built-in functions.

Lately the use of the generative paradigm and hence mathematics in architecture has regained a lot of interest and this time from leading designers (eg. Frank Gehry, Kas Oosterhuis, Greg Lynn) closely followed by the architectural students. This new group of designers uses modern CAD technology for generating shapes often based on classical algorithms and gave this new impulse in Architecture also new names, as “Generative Design”, “Algorithmic Architecture” or “Design Computing”. These architects use visual based software like Generative Components or Grasshopper or they use the embedded scripting languages.
It is our belief, to understand the visual approach better, it is necessary to have a basic understanding of programming and designing an algorithms. Scripting is an easy way to learn both. So we focus more on software tools with an embedded scripting language. The 'text' based approach also makes the generative approach clear and the generation of alternative solutions is very down to earth.

A traditional design studio is organized round tutors who give the students an assignment, more precisely a design problem which the students have to solve in one semester. The problems are most of the time “design a ………..” Out of supervision-efficiency all the students in the design studio get the same design assignment. The tutor leads the students by the hand till the end. Along the path markers are placed (deadlines) which must be reached in time. So the whole path is well paved (or well structured). The tutor uses a “red pencil” to mark what is wrong and the student follows the given direction. Students who attend our studio often say: "the other studios are always the same, only the subject differs, we don't learn anything new". Besides that in the other studios all students have the same design problem, during the meetings there aren’t any new insights or subjects to learn. They learn a lot about the design problem at hand but don't learn to express themselves on unfamiliar design problems. With mixed problems, students get triggered by the solutions of other problem; the meeting becomes a challenge and an opportunity to get new insights.

Computational design isn’t a design process in the traditional way, but it is the use of a combination of different arithmetic methods (directly or indirectly) in order to generate a set of different alternative solutions for the design problem at hand. In this way it becomes possible to find solutions for complex problems which can’t be found in a traditional way of problem solving.

For most architects and architectural students mathematics is not heir favorite subject and we don’t even mention software programming. It is our opinion that the education of the new generations of architect needs a shift. In our design studio we educate the students this new way of looking at architecture. We link in our studio this traditional way of visual modeling and the generative approach. To establish this link we use CAD scripting languages. The problem rises however is twofold:

1. Scripting languages are meant to extend the system functionality and not to easily generate shapes;
2. Most architectural students have no programming skills at all.

Therefore dedicated training and support is necessary to get the first results in a short time, before the student gets frustrated over the programming burden.

Since 2006 we run at the Technical University of Eindhoven a 3rd year design studio which is focused on this new way of thinking in and about architecture. In many aspects the organization differs greatly from the more traditional organized design studios.

In the next paragraph we will discuss the pedagogical organization of ‘our’ studio and how this new way of generative design is used in architectural training.
In the following paragraph we will show and discuss some students work, one project will be discussed in more detail to show the path the student followed and the happiness that he experienced at the end.

Finally, we will summarize our experience with this design studio and provide some guidelines for successful implementing Generative Design in architectural design teaching.

**Generative Design Studio**

In this paragraph we discuss the design studio in more detail. To do so we use the work of a student. He was a typical architectural student and attended our studio because he was interested in.

The design studio lasts one semester, with a general meeting every week, at the first meeting there is a small course to explain generative design and what it stands for. Most of the times it comes as a shock, when the students are told that they need to formulate their own problem.

In the next two weeks the students perform a literature research on the topic, to trace their field of interest. They have to bring for the next session 10 imagines of photos. They have to pick those photos at first glance, not rethinking about why the liked the subject. We advise also to use the faculty-library and not only 'Google' pictures. During this period the students phrase their own design problem, which has to be solved with use of scripting. Together with the supervisors, CA(A)D software and the initial design problem are chosen. Because of the familiarity most students have with AutoCad, this software is mostly used together with the VBA scripting language which is implemented in AutoCad, but also Generative Components, Rhinoceros or Max-Design are used.

A few of the pictures the architectural student presented here were: a balancing acrobat, a lamp hanging on the ceiling and a picture of a sailing boat.

During the discussion of the photos a few words constantly re-appeared: elegance, balance. The photo of the hanging-lamp (see Figure 2), the balancing acrobat (see Figure 1) and a sailing boat (see Figure 3) triggered those words. During the discussion the teachers tried to change topic to get to a better defined topic. A topic which could easily be translates into a mathematical description. Every time we tried to change topic or direction of the discussion, it returned to those words. After a long discussion we agreed maybe we could gain elegance and balance in a pile of discs. So we agreed to develop a script to build a tower of piled disks, each disk would have a random diameter and overhang.
The design problem is reviewed, because it has to be a problem in which “generative designing” can be fully explored. Students are introduced to scripting with help of simple exercises.

After a brief introduction to answer questions as: how to script in Autocad, how to open the script-editor, to load a file, we explain where the two help files are located, one for the Basic commands and the other for the Autocad commands. The former mentioned help-files helps to answer Basic syntax questions, and the later answers question which Autocad command has to be used and what the syntax of that command is. There is a brief introduction to VBA (= visual basic for applications) to explain: Why do you have to assign variables, what are integers and doubles. We keep these introductions to a minimum, by studying the Autocad examples the students learn in their own pace. This design studio is not about programming, it is merely an introduction to scripting. Scripting is a tool to learn thinking in a more structured manner about designing and parametric design.

These exercises are also a start of the design problem they want to script. For simplicity the initial exercise is chosen in 2D. During the course, the results and progress are weekly monitored, and students get support with their scripting tasks. This initial script is developed subsequently.

The initial assignment, for ‘our’ student, was to script a disk (see figure 7). The next script would be to pile a number of discs (see figures 4, 5, and 6). This assignment introduced the loop command. During the next weeks the script slowly gained in complexity. After a few weeks ‘branching of the tower’ was introduced, after a random number of stacked discs, the tower would branch into a second tower and this branch would become a contra weight for the first branch.

During all sessions we advise the students to experiment with their scripts to get a feeling of their script. They need to ask themselves what the limits of the variables are, are there singularities in the script etc. Most of the time this experimenting results in a better understanding of the script and new insights of which direction the development will go.
Last but not least the availability of the digital model allows the students to create a physical mock-up from their design without much extra effort. Transferring the digital model into a physical model raises questions about the materialization of the design, and therefore this is an important step in the design process. The technologies used such as the laser-cutter and a 3D printer, and the encountered constraints are discussed with the students. This step out of the virtual reality in to the tangible and the concrete reality is welcomed by the students and is a reward for the hard labor they did during the studio.

After finishing the scrip, our student made it ready for rapid prototyping (see figure 5, 6 and 7). Seeing the end result of his hard labor he was ecstasies. He called his project "Where Elegance meets Balance".
Some results from other students

**Madolis (student 1)**

Differentiation and unification in interaction with each other, these are the underlying principles of the Silodam project in Amsterdam, designed by MVDRV. “The final design left many question unanswered, as if it was the easy way out of this fascinating concept. What would be the effect if we use modern computers to generate complex but attractive alternatives, with use of a set of newly developed constrains.”

![Figure 11 Basic Madolis](image)

Madolis is an alternative solution to the Silodam project. The underlying concept is left intact, but the design is generated by the computer, figure 11. On the building site a preset number of buildings blocks is the starting point. A building block occupies a total floor area within certain limits. These building blocks will increase, with the smaller ones having a higher priority. Fulfilling the constraint area, the entree of each building block is created, to ensure that it is connected to the pier, figure 12.
Figure 12 Rendered Madolis

Chaos Pavilion (student 2)
The goal of this project was to generate “chaos” within a pre determined bandwidth. A small number simple steps results in a complex structure. This is done for a number of different scales levels.

Figure 13 Chaos Pavilion

Neo Amsterdamse School (student 3)
The goal of this project was to see what can be achieved by placing a number of boxes (bricks) along a curve. By applying to each box a small translation or rotation, different walls could be generated. In the end the student did some experiments with the alternative wall configurations and made with a laser cutter a prototype (see figure 14)
Conclusions

Students who participate in the design studio learn a new way of designing. They also learn that there is a hidden relation between many aspects of the design and they learn to think and look in a more abstract way.

After a slow start, the students learn quickly to program a script, and the end result is always astonishing.

By translating their own algorithm, with use of an embedded script language, into ‘a script’, the CAD-software can perform the task. According to Terzidis (2008, pp 65) “an algorithm is a computational procedure for addressing a problem in a finite number of steps. It involves deduction, induction, abstraction, generalization, and structured logic.”

He continues with “Algorithmic strategies utilize the search for repetitive patterns, universal principles, interchangeable modules, and inductive links.” Every one of the former mentioned ‘actions’ is part of a general problem solving strategy.

Learning to develop an algorithm from scratch as well learn how to program, serves a few purposes, namely:

- Learn to solve a problem in general way;
- Learn to master CAD software in a more fundamental way;
- Learn to surpass the limitation of the ‘out of the box’ CAD - software;
- Looking at a problem in a different way.

With the experience of our design studios we conclude that students adapt quickly to this new way of designing. Some students see the advantage of this way of thinking and the possibilities to generated new alternatives. Other students think it is a nice experience, and will probably not utilize the experience in the future.

References
