Learning to Know and Think: Computing for Architecture Course

Aswin Indraprastha^{1,*}

¹Architecture Program, Institute Technology Bandung, Indonesia

Abstract. Computational technologies for solving design problems become increasingly important in architectural practice. In responding, architectural education has encouraged the use of this tool and method in the curricula. As the technology and environment are ever changing, the curricula should be evaluated and updated to adapt and to teach method and skills necessary to the students. Over the last four years, Architecture Program of ITB has inserted new computational courses into undergraduate levels. These courses are the mix of a skill focused computational workshop that is compulsory for the second-year students and introduction of computational design as an elective course that can be enrolled both by third and fourth year students. This paper delivers a report of our methods and findings from our continuing study of the courses including analysis of student outcomes, student evaluation of the course structures, assignments, and feedback as well as computational abilities after completion of the courses. The aim of the study is to have a grounded validation of computational courses in architecture curricula and to improve courses goal based on the evaluation. The result of our study reveals challenging issue to teach computational thinking undergraduate level rather than only providing them a set of computer skills for production and presentation techniques of the design.

Keywords: computational thinking, computational course, architectural design

Introduction

Computational technologies for solving design problems become increasingly important in architectural practice. In responds, architectural education has encouraged the use of this tool and method in the curricula. The needs and expectancy for innovative architectural design that is aesthetically pleasing and responsive to the environmental loads and energy (i.e. green design, energy-efficient design) meets with new methods and tools to front-loading¹ design processes in order to achieve higher performance design through means of a digitally-assisted system (Kalay, 2004).

Computing as the act of calculating which has root in mathematics plays significant roles in shaping today's design methodology to make sense the design output that potentially bring back architect to be master builder as once was.

Among the first notable critical works of mathematics in architecture occurs in the Notes of Synthesis of Form (Alexander, 1964) and The Geometry of Environment (March & Steadman, 1971). Those works made important contributions to the design method, particularly on what we know as the relational theory, one of the basic principles of

architectural computing. In 1996, William Mitchell wrote ideas to develop architectural formal language by dissecting and developing inter-relation among building functions and elements in *The Logic of Architecture* (Mitchell, 1996).

Since then, the discourse over computation involvement with the architectural design process has been an interplay among scholars, designers, architects, engineers, even programmers, biologist, among others (Burry, 2010), (Terzidis, 2006), (Kolarevic, 2005), (Aish, 2013), (Menges, 2017), (Meredith 2007). By some, architectural design became a testbed for playing with formal yet liberating methodology called programming. John Maeda pioneering this approach to explore flexibility and platform-free of programming in design. Later, he involved in the development of *Processing*, one of the most popular programming platform today for non-programmers (Maeda, 2001).

This issue and trend left a challenge for architectural education both in pedagogy, or method of teaching and subject that involve technology that is always growing and changes constantly. There are some findings pointed that computation and architecture education (and architecture students as well) are hard to blend (Varinlioglu, 2004), (Colakoglu, 2007), (Abell, 2013). Abstracting and thinking in terms of formal, explicit and procedural way is hard to learn, especially when by nature, in architectural education creativity as an act to discover, is learned through many methods and less in a formal way. This opposition also contributes to the fact that many architectural students are hesitant to learn computation

¹A method in the workflow of design and construction phase where some processes which previously carried out in linear and sequential fashion now can be completed in parallel and move forward to schematic design phase (e.g. design analysis, schematic, quantification and scheduling).

* Corresponding author: aswin@itb.ac.id

because it is hard and they cannot envision how it will help to become a better designer (McCollough, 2006 in Senske, 2013).

On the other hand, computation as a *lingua franca* for all digital apparatus that assist design-construction process nowadays transforms itself into software and hardware that constantly developed and changes. Programming then, become an essential skill to master these new systems. McCollough in 2006 suggested that programming has been rediscovered in architectural design mainly by two reasons: first, there are more incentive to express design based on machining process. Second, this form expression is increasingly informed by other domain of knowledge that already familiar with the notion of the beauty of generative algorithmic: namely Biology, where harmonies, proportion, recursion can be codified into algorithmic and parametric formula (McCollough, 2006).

By this fact, we believe there is a potential to include the new way of thinking and working into architecture curricula. This paper delivers report of our development incorporating computation-based courses and the result in form of feedbacks to assess the course outcomes and the effectiveness of its execution.

Background

As part of First Common Year program, in 2008 ITB has introduced an introductory course on programming that is mandatory for all students. This course presumably planned to introduce basic computational logic through series of lectures and workshops emphasizing on basic programming skills using C++ programming language.

Although the course is based on university evaluation to strengthen basic STEM (Science, Technology, Engineering, Mathematics), it is found that the majority students from SAPPD (School of Architecture, Planning and Policy Development) had difficulties to grasp the most important goals intended for this course namely, to understand how to think computationally through procedures (algorithm) and to understand how to execute program through systematically written syntaxes in form of codes. This condition has apparent significant reason and confusion among students as some students asked why they learn to code and how it will be relevant in their future career.

Nonetheless, at the second year in architecture program, the freshmen will be taught computational design studio (3 credits), which is designed to provide basic skills and knowledge of CAD, 3D modeling and BIM. Production of drawing, transformation and creative elaboration of 3D objects, and component-based modeling are the goals in which spread into three modules that conducted in sequence. This course in principle was designed to be a supporting course for their architecture design studio project in subsequent years.

The computational design studio however, did not adequately equipped students with computational thinking or computing abilities. They know how computational processes work through series of exercises using various software and could produce solution over certain geometrical problems, but they cannot explore beyond was taught in the lab session. This formal method usually limits student's creative thinking down to only the function and features that already been taught.

The computational design studio has another issue that fails to recall knowledge and understanding of computational logic they learned in the first year. From the lecturer's perspective, this condition is understandable given the fact that learning to use multiple software requires time and in this case, there was no room to teach computational logic.

Furthermore, in third and fourth year, we offer two elective courses with topic of algorithmic design and BIM respectively. These elective courses named Algorithmic Approach to Architectural Design (2 credits) and BIM Approach to Architectural Design (2 credits) are offered for the third and fourth-year students. Both courses designed to be basic introduction courses to architectural computing using two distinct approaches.

Workshop I: Computational Design Studio

Computational Design Studio (AR2250) is a mandatory course for the fourth semester of second- year architecture students that was conducted through three subsequent modules:

1. Production of technical drawing using 2D CAD
2. 3D geometry exploration and modeling
3. Component-based modeling of virtual building using BIM

The challenge was to provide the same level of abilities to all students of CAD, 3D modeling and BIM, having no prior experience in CAD, nor 3D modeling and BIM. We advised that at the second year, the appropriate learning method is through two stages:

1. Learning to draft: drawing and modeling by imitating a reference. Limited flexibility to expand skills through design. The computer is being used as the tool for drafting.
2. Learning to expand skills and abilities: further step after imitating is expanding abilities by modeling details, adding meta-information of elements, or producing documents. The computer is being used as a tool for modeling.

This workshop meets once a week, first with a lecture session and then followed by two hours lab session. We still conducted the scheme of the lecture-lab sequence as oppose to lab-lecture i.e. to work with concrete form first then learn the abstract later (Bransford et al., 2000).

The partial result showed in this Figure 1.

Fig. 1. (Top) CAD Drawing; (Middle) 3D modeling; (Down) BIM Modeling.

In the lab session, our method of teaching is following our hands-on training material and facilitate an active process where each student works at their own pace, the tutors are available to answer any questions or any technical difficulties.

The assessments for this workshop were designated on par with the Architectural Design Studio II where modeled objects are a single and small house. In 3D modules particularly, we advised students to expand their ability to model object as detailed as possible.

Serving as the basic introductory studio to learn to use software, most students felt that the course was important to support their study and their future career. Among other feedbacks gathered through a post-class survey, we discussed our course evaluation as it is presented in the later section of this paper.

Workshop II: Algorithmic Approach to Architectural Design

The Algorithmic Approach in Design (AR4121) is an elective course (2 credits) designated for the third and fourth year students and conducted in the fifth semester. This course was planned and designed to be an introductory course for architectural computation that includes computational design thinking and the methods of formal languages. The goal is set specifically towards computational methods of formal design exploration and analysis through algorithms.

Since an elective course, the syllabus, assessment and course evaluation was tailored and evaluated each year based on student feedbacks and their performance.

The course was conducted using learning by doing approach, starting with lecture and followed by programming session, in two hours in a week.

On early implementation of this course (2013-2014), overall topics of the syllabus for 15 sessions is as follows:

- A. Fundamental
 - Introduction to computational design, parametric design, computational thinking.
 - Parametric functions, mathematics, and logic, set, operators, variables using Rhinoceros & Grasshopper as visual algorithm editor.
 - Data structure, tree and manipulation
 - Vector, Curve and Surface parameterization and manipulation
- B. Application
 - Parametric patterns
 - Parametric analysis: structure, tensile, optimization and environmental analysis.

Assignments were given in two stages: the first project is to be carried out individually. Student required to design, develop or explore scripts within GH VAE (Grasshopper Visual Algorithm Editor) with an architectural application such as a pattern of the wall, roof shelter, or industrial products such as lampstand. The output is a poster depicting algorithmic process and its output. The second project is a group project to design and construct a scale model of a shelter, bench, or art installation, based on parametric and algorithmic process.

The teaching pedagogy we experimented during the past four years is combining a theoretical and a top-down style lectures on the logic and principles of computation, followed by hands-on exercise on the logic of the workflow of the GH VAE, its structures and syntaxes, data management and manipulation, and geometric parameterization and manipulation.

After self-evaluation, the topics of this course during 2015-2017 are arranged as follows:

- A. Basic Understanding
 1. Introduction to Rhinoceros and Grasshopper: workflows, principles, navigation, commands.
 2. Program logic and workflow, syntaxes/components, network.
 3. Data structure and manipulation.
 4. Point, vector and curve.
 5. Surface and polysurfaces.
- B. Control and Manipulation
 1. Folded, grid and porosity logic
 2. Force field and image mapping

3. Dynamic relaxation
4. Optimization methods

During last three years of the course, the weekly course-lab subjects and assignments have been continuously re-evaluated and refined. Part of the result of assignments is depicted in Figure 2.

Fig. 2. AR4121 Student Works on Algorithmic Approach.

We faced multiple issues mostly from pedagogical aspects and learning methodologies. These issues can be summarized as follows:

1. It is a mandatory to provide a ground and fundamental lecture combined with hand-on exercises on computer programming and logic of computing during short period of the course. It is found that majority of Architecture Students had high expectation on the parametric design side without knowing that they had to learn how to program.
2. In consequence, students were trying hard to grasp basic principles and understanding some fundamental concepts in computing: functions, operator, set, etc. it is found that the programming principles they had in their first year, has a little trace in their third and fourth year.
3. Among other aspects, it is found that some students failed to expand or to explore, computing principles beyond pre-cooked program or definition that has taught. They know the workflow, they understand the principles and methods but they feel the hardest part is to transform real life problem i.e. concrete problem into computational problem.

Workshop III: BIM Approach to Architectural Design

The BIM Approach in Design (AR4222) is another elective course (2 credits) intended to equip students the advanced knowledge, skills related with Building Information Modeling. This course is divided into two modules:

1. Architectural BIM: advanced parametric modeling, component-based prototyping: hosted and hosted parametric component, schedule and quantification.
2. Collaborative BIM: introducing coordination methods, clash detection, introduction to scheduling and technical documentation.

As with Algorithmic Course, during the past four years of its implementation, we evaluated and refined the course subjects, assignments based on student performance and feedback.

Fig. 3. Student Works on Adaptive Component using BIM Application.

The BIM course is supposedly providing student method and focused on developing fluency BIM software. However, we believe that in addition to skills and rather using software as tool, computational thinking is the most significant aspect that should change their mindset to use computation well, regardless its role in design process.

During the first module, architectural BIM for instance, there is a potential to develop computational thinking using parametric family (component) modeling in family. By BIM software, in this case we use Autodesk Revit, the platform and pipeline are already in place so that any parametric object can be used to generate drawings, quantified, and analyzed.

However, our experience suggested that was not the case. Two hours per week sessions in the total 15 weeks is inadequate to cover learning curve of 3D BIM. Although students starting to rely on BIM software for their Architectural Design studio projects, the necessity to think that BIM is a platform for architectural computation and a tool for form and design exploration is a far fetch. They still used BIM modeling for pragmatic purpose.

This condition may be forgiving if we consider for undergraduate students the knowledge and skills of architectural computation are for practical purposes. The goal of the undergraduate program of architecture is to prepare knowledge, skills, and values required for professional practice in architecture.

In 2017, we advised another approach to develop computational skill using BIM by conducting a collaborative project with Universitas Pelita Harapan (UPH). In this joint

workshop, our students were required to design a dynamic and adaptive component while students of UPH developing and constructing kinetic system using either digital or analog mechanism. Each team composed of ITB and UPH students then collaboratively construct and present their kinetic installation.

This workshop was effective to teach computational processes from design inception, parametrization of a component of a system, fabrication, and prototyping in addition to work as a team in different places. Students on both universities engaged in works process where they can learn from each other (Figure 4).

Fig. 4. Collaborative Workshop on BIM and Kinetic Installation.

Although the vehicle in this context is BIM, the pre-workshop lectures and exercises on parametric modeling gave students fundamental principles of Computation.

Some of the result of student projects are depicted in Figure 3 that reflect their abstraction of real/concrete problem, to be solved using computational method.

Feedbacks

We conducted a post-class survey and collecting feedbacks for all courses. This survey is mandatory by the university for all students and included multiple choice with Likert Scale (1-5; 1=unlikely, 5=likely) and short written answers. The result of this survey is presented in the table in Figure 4. The statement of questionnaires mainly focused on course's outcome and operation/process as shown below:

Fig. 5. Student Feedbacks.

1. Outcomes or Goals are achievable.
2. I obtain all knowledge and information needed to accomplish the outcome
3. The lecture was designed and conducted so that students can accomplish goals and assessments.
4. I can master determined goals for this course.
5. The workshop is well prepared.
6. The workshop was well-organized.
7. Lecturer and tutor have good communication
8. Lecturer and tutor care about students being able to grasp knowledge and skills to accomplish the goals or outcomes.
9. Lecturer was being fair to the students.
10. The credit of this course is questionable compare with the assessments and lab activities.
11. Lab and infrastucture is adequate for conducting workshop.
12. There are other additional resources to support this course.

On AR2250 as we may inspect at the chart in Figure 5, students felt that the credit (3) is too small for such workshops with assessments, lab sessions and lectures while in average, this course was well-received. More than 90% of the students agreed that outcome of this course was achievable (average 3.42 in 2013-2017).

of mastering this course. Unfortunately, we cannot determine how they understand computational thinking through their projects and network of components in GH VAE.

On AR4222, our finding suggested that course credit (question #10) and course preparation and organization (question #6) had low average among others (average 3.08/5.00 and 2.90/5.00 respectively). The BIM course over year from 2013 to 2016 always invited practitioner as external tutors for the module of Collaborative BIM. In this module, we simulate workflow of collaboration among users to modify, inspect and create development of a project. This requires better technical preparation of IT infrastructure and additional resources that our university need to plan it out in the future.

Aside from those results, it saves to note that the computation-based courses appear to be successful at teaching and provide knowledge and skills to many students (average participant for AR2250 is 100 students, while AR4121 and AR4222 usually 40-45 students). However, we are still curious on the course's goals whether we can improve on the outcome to increase student's performance in the computational thinking that it will lead to evaluation of our teaching and exercise methodologies.

Reflection and Discussion

This paper presents our self assessment and ongoing evaluation for the computational-based courses.

Over four years of implementation, there are several lessons that may provide insights improvements. We still believe that in our case, the computational-based course for undergraduate program focuses on the know-how aspect of pedagogy.

the main topics are parametric modeling and collaborative BIM.

It should be taken into consideration that the topics being refined, is based on experiences and evaluation of student's performance in the period of 2013-2017. Students characteristic of each batch may vary and have an impact on the overall class performance as well.

Fig. 6.Summary of Student Feedbacks on AR4121 and AR4222.

The goal is to know how to use, how to solve a certain problem using some set of tools or algorithms. Therefore, the basic principles of computation still relevant to be taught in the first year to provide a fundamental understanding of how computation works.

Given its significant impact on computational skill, the Computational Design Studio (AR2250) remain to be focused on the know-how aspect of learning various software, priority is given to the pragmatic solution of given drawing and modeling assignments. Another additional goal is to leverage computational skills among the students.

Algorithmic and BIM elective courses (AR4121 and AR4222) however, can be further sharpened into specific goals and lay a foundation for further exploration in the subjects of parametric design and analysis, and BIM applications. However, it needs further innovations and other supporting improvements.

As we may inspect in Figure 6 where both Algorithmic and BIM courses suffer in the aspects of the method of teaching, workshop execution and process, as well as additional resources.

a. Topic

Topics or object of learning both in Algorithmic and BIM courses focus on computational method. In Algorithmic Approach course the main topics are mathematics logic and functions, data structures, elements of geometry, and parametric applications. While in BIM Approach course,

b. Activity

Student activities during course should be carefully planned in order to give them both challenge and time to enjoy programming both in GH VAE or BIM platform. The additional joint workshop is preferable but should be planned and arranged well to maximise efficiency. One aspect important related with activity is it should be fun and enjoyable to do. Student assignments is a tricky part since it should balance between encouraging the student to push to their limit of creative exploration through programming and time required to complete them.

c. Additional Resources

Additional resources to support both courses is inevitable and should be improved. From lecturer's stand point, some topics both in Algorithmic and BIM courses can be enriched by digital apparatus such as CNC milling, 3D printers in order to give experiences of prototyping and hand-on with real material properties, and constructing real elements.

Another important improvement should be relying on the methodology of teaching computation for architecture students. We suggest that a visual-based algorithm editor such as GH is a good tool to teach such principles. The ability to understand and explore computational thinking is something that need constant learning through programming. We should encourage students to explore beyond class hours and provide insight on the computational works that they will face in their lines of work in the future.

In our brief conversation with WassimJabi² about teaching methodology of computation for architecture student, one of the approach is the *Deep Pool* approach, meaning that in order to gain more understanding of computing, students must do scripting in which they not only learn workflow of the program, they also learn how to think as a computer program works, learning to manage resources, learning to discover and to think rather than to know. This insight should be a great idea and already implemented in some architectural schools. In UNC Charlotte, North Carolina for example, learning programming in the early years of architectural education is mandatory to expose the student to computational thinking, which may help them learn software and other tools, motivates them to continue learning in the future (Senske, 2013).

Both cases are adequate and fit for their context and academic cultures. We are aware and fully understand that computational thinking and methods will extend beyond these courses. The potential, accessibility, and advancement of computation will democratize work of design, prototyping, and construction. It is our responsibility as academia to respond and to adapt to this inevitable phenomenon in order to stay relevant and innovative.

As Mike Silver pointed out in the editorial notes in Architectural Design, that that what computation now serve are the founding concepts, intuitions, and desires that can only emerge from a varied and creative practice. Computing should be regarded as an endeavor of creativity and arena of play, beyond any restrictions. That will be achieved through the development of computation culture in academia. It will require time, paradigm shift both for faculty members and students regarding computational thinking and further study, but we are preparing towards that direction.

References

Books

- Abell, John, H. (2013), Hand Crafting Computational Design Thinking in Basic Design Studio, Proceedings of the AAE Conference.
- Aish, Robert (2013), First Build Your Tools, in Architectural Design: Inside Smart Geometry- Expanding the Architectural Possibilities of Computational Design, Wiley, London.
- Alexander, Christopher (1964). Notes on the Synthesis of Form. Harvard University Press. New York.
- Burroughs, Mark and Jane Burroughs (2010), The New Mathematics of Architecture, Thames and Hudson, London.
- Colakoglu, Birgul and Tugrul Yazar (2007), An Innovative Design Education Approach: Computational Design Teaching for Architecture, Proceedings of the METU JFA Conference.
- Jabi, Wassim (2013), Parametric Design for Architecture, Laurence King, London.

- Kalay, Yehuda, E. (2004), Architecture's New Media: Principles, Theories and Methods of Computer-Aided Design, the MIT Press, Massachusetts.
- Kolarevic, Branco (2005), Architecture in the Digital Age: Design and Manufacturing, Taylor and Francis, New York.
- Maeda, John (2001), Design by Numbers, the MIT Press, Massachusetts.
- March, Lionel & Phillip Steadman (1971), The Geometry of Environment, the MIT Press, Massachusetts.
- McCollough, Malcolm (2006), 20 Years of Scripted Space, in Programming Cultures: Art and Architecture in the Age of Software, Architectural Design, July-August 2006Wiley-Academy, London
- Menges, Achim, et al. (2017), Advancing Wood Architecture, a Computational Approach, Routledge, London.
- Meredith, Michael, et al. (2007), From Control to Design: Parametric/Algorithmic Architecture, Ingoprint SL.
- Mitchell, William J. (1996), The Logic of Architecture: Design, Computation and Cognition, the MIT Press, Massachusetts.
- Senske, Nick (2013), Building a Computational Culture: A Pedagogical Study of a Computer Programming Requirement, Proceedings of ARCC 2013 Conference: Pedagogy: New Visions and Revisions in Architectural Education.
- Silver, Mike (2006), Programming Cultures: Art and Architecture in the Age of Software (Editor), Architectural Design, July-August 2006Wiley-Academy, London.
- Terzidis, Kostas (2006), Algorithmic Architecture, Architectural Press, Massachusetts.
- Varienlioglu, Guzen, et al. (2004), Computational Thinking and Architectural Curriculum: Simple to Complex or Complex to Simple, Proceedings of the 34th eCAADe Symposium: Design Tools.

²AssociateProfessorat the WelshSchool of Architecture, Cardiff University. Chair, Digital StrategyGroup.The conversation took place during ASCAAD Conference in London, November 2016.