ITU A|Z • Vol 17 No 2 • July 2020 • 199-208

An assessment method for a designerly way of computational thinking

Elif Belkıs ÖKSÜZ[1], Gülen ÇAĞDAŞ[2]
[1]eeoksuzz@gmail.com • Department of Architecture, Faculty of Architecture, Istanbul Technical University, Istanbul, Turkey
[2]cagdas@itu.edu.tr • Department of Architecture, Faculty of Architecture, Istanbul Technical University, Istanbul, Turkey

Received: February 2020 • Final Acceptance: March 2020

Abstract

The article presents an assessment method for the designers' use of abstraction skills in the process of CT. Starting with questioning how abstraction partakes in design and computer sciences, the study focuses on the impacts of making conceptual and procedural abstractions in CT. For that, it offers an assessment method to explore whether a visual thinker's ability to make abstractions has any impact on their process of visual computing. As a concept, CT is considered as a mental activity for formulating a problem to admit a computational solution by combining the intelligence of humans and machines. It is addressed as a collection of mental tools and concepts that are borrowed from computer sciences. Within this regard, architecture is one of the fields that require careful consideration of these cognitive aspects towards CT. Although both computer and design sciences value abstraction in similar ways, its introduction to the design field slightly differs from its introduction to computer sciences. Considering the differences in their conceptual background and reflective practices, it can be said that the abstraction of a visual thinker may not always constitute the way that CT requires. Based on a two-stage experiment in a CAD modeling framework, the developed methodology revealed that the designers' abilities to make abstractions at a procedural level partake a significant role in their visual computing. While the first experiment is conducted with 3 sophomore architecture students, the second was conducted with the participation of 3 non-designers along with the same architecture students.

Keywords
Procedural abstraction, Computational thinking, Soft skill, CAD modeling, Assessment method.

doi: 10.5505/itujfa.2020.86729

1. Introduction

We are currently facing a paradigm shift towards a computing and coding culture in education. From digital tools to artificial intelligence, computational technologies are visibly affecting our everyday life and shaping our cognitive development. While the paradigm we are heading towards requires technologies to be 'brain-friendly' enough to complement our cognitive processes (Dror, 2011), it also needs these processes to be similar to the series of internal states like carrying out a program (Weisberg & Reeves, 2013). Alternatively, as Colin Ware puts it (2008), "(the) Real-world cognition increasingly involves computer-based cognitive tools that are designed to support one mode of thinking or another; and as time passes, these tools change how people think" (p:181). Either way, we are introduced to different modes of thinking in education; and without a doubt, Computational Thinking is the most in-demand. Over a decade, it has been promoted as a cognitive tool, which helps develop an understanding of computational technologies.

Despite its early use through programming practices, the concept of Computational Thinking (CT) is now offered as a soft skill, as a combination of certain cognitive features, which can be blended in any disciplinary education curricula. With its cognitive features, CT is acknowledged as a mental activity for formulating a problem to admit a computational solution by combining the intelligence of humans and machines (Wing, 2017; p:8). It is addressed as a collection of mental tools and concepts that are borrowed from computer sciences (NRC, 2010; p:10). For its integration to disciplines outside computer sciences or incorporation into different subjects or courses, the concept of CT is usually introduced with its core cognitive features, which are abstraction, pattern recognition, decomposition, and algorithmic thinking [Figure 1].

On the other hand, thinking computationally is not about having good knowledge of these features. As a soft skill, it requires putting the knowledge into action, using, transferring, and adapting it into different problems or situations. For that, addressing the conceptual counterparts of these features is essential. As Guzdial (2008) posits, "paving the way for computational thinking from computer sciences to other academic fields may require adapting this mode of thinking to match the needs of 'novices' and non-majors." Because, "if students don't learn the material or any knowledge well in the first place, they can't possibly transfer it to new situations" (Guzdial, 2010; p:5). In this case, careful consideration is required for the designer's use of abstraction towards CT.

Even though both computer and design sciences consider the cognitive value of abstraction in similar ways, differences occur in their conceptual approach to this skill and reflect on their practices. Therefore, the abstraction of a visual thinker may not always constitute the way that CT requires. Yet, little attention has been paid to the difference in the designer's use of abstraction in visual computing. However, as Wing emphasizes, the process of CT is mostly about making good abstractions; it involves defining abstractions, working with multiple layers of abstractions, and understanding relations between them (Wing, 2008; p: 3718).

Regarding the gap in the literature, this study presents an assessment meth-

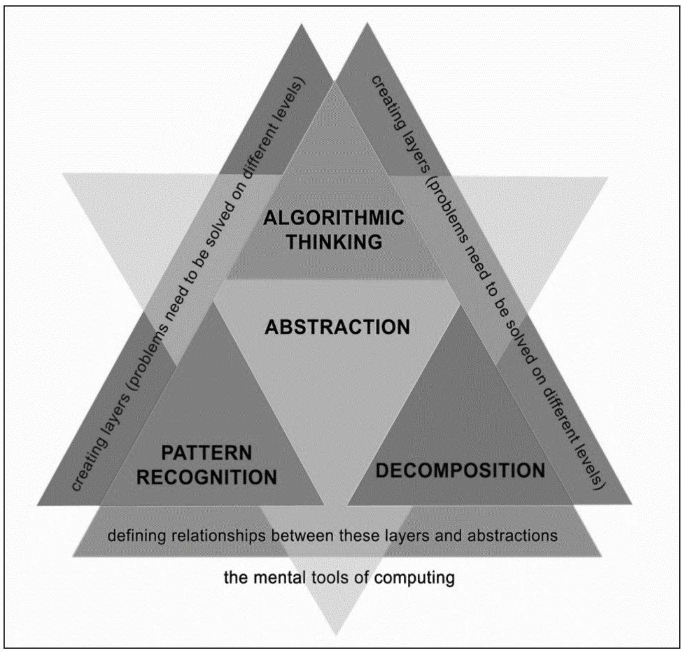

creating layers (problems need to be solved on different levels)
creating layers (problems need to be solved on different levels)
ALGORITHMIC THINKING
ABSTRACTION
creating layers (problems need to be solved on different levels)
PATTERN RECOGNITION
DECOMPOSITION
defining relationships between these layers and abstractions
the mental tools of computing

Figure 1. *The mental tools of computing, adapted from Wing (2008).*

od for the designers' use of abstraction skill in visual computing. Starting with questioning how abstraction partakes in design and computer sciences, the study focuses on the impacts of different levels of abstraction in CT. For that, it offers an assessment method to explore whether a visual thinker's ability to make abstractions has any impact on their process of visual computing. For the study, the assessment method was used in two experiments, which were conducted with non-designers and designers working in a CAD modeling framework. In the end, the developed method is discussed towards the future possibilities of its use in designer's education.

2. Background

The conception of 'abstraction' or 'abstract thinking' is one of the fundamentals skills in both computer and design sciences, to be acquired in their early years of education. It is taught to deal with different forms of complexities in simpler ways. From the perspective of a computer scientist, Jeannette Wing (2008) summarizes the use of abstraction as follows: "An abstraction is the ability to generalize and transfer a solution from one problem to other similar problems…our abstractions are extremely general because they are symbolic…they are for representing and processing the data to extract the knowledge buried within or spread throughout the data" (Wing, 2008; pp:3717-3720). Similarly, Gabriela Goldschmidt (2011) proposes the same concept 'as a key prerequisite for the successful use of external sources to aid design creativity that helps to distance oneself from the source properties and transfer only the essential relationships' for the design sciences (p: 97).' Nonetheless, both fields adopt this skill with certain similarities and differences towards their conceptual framework and endorse it with their distinct tools and strategies; so, the differences in their conceptual approach eventually reflect on their behavior pattern.

On the other hand, as Sengupta et. al (2013) highlights, Wing's definition of abstraction skill is highly similar to John Locke's view. For Locke (1979), making abstraction is the process in which "ideas were taken from particular beings become general representatives of all of the same kind." To abstract, we take ideas received from objects and separate them "from all other existences and the circumstances of real existence, as time, place, or any other concomitant ideas" (Arnheim, 2004; p: 154). For Arnheim (2004), Locke's understanding of abstraction is to be free from any perceptual collateral, which would be viewed as an impurity; yet, instead of relying on sensory experience, this kind of abstract thinking was supposed to take place in words (p: 154). Nonetheless, this isolated approach may not be fruitful as much as expected for visual thinkers, or design students for that matter. As opposed to a scientist's procedural approach to abstract thinking, a designer's approach can be highly conceptual, and rely on the embodied experience.

For a visual thinker, abstraction requires dealing with multiple forms and genres of representations beyond drawing, sketching, modeling; and it usually occurs while translating between these forms of representations. Hence, it requires making perceptual interpretations. For instance, architecture students are trained to make visual interpretations, visualize their thoughts through different forms of representation. And keeping up with such practices, they develop their visual interpretation, which eventually determines their designerly ways of abstract thinking. Even though nothing is more concrete than color, shape, and motion in visualization, there is always a possibility in which a designer's abstraction may not reflect the necessary information for a scientist to proceed with their work. As an artist or a designer, a visual thinker owns the potential of mentally adding simple patterns to their work to test possible design changes before making any changes to it (Ware, 2010; p: 170). Notably, in creative stages of design, designers seem to repeatedly change their ways of seeing (Schön, 1987). According to Lawson (2006), "Architects are taught through a series of design studies; they are not asked to understand problems or analyze situations. By comparison, scientists are taught theoretically and are taught that science proceeds through a method which is made explicit and which can be replicated by others." In other words, scientists and architects follow different strategies when approaching a problem and use their abstraction skills towards that. As opposed to scientists, architects develop their abstraction skills by practicing with solution-focused strategies.

However, working with computational design tools and devices requires thinking quite differently than working with traditional tools (Erhan, Youssef & Berry, 2012). The computational operations behind design technologies endorse designers to utilize CT as a mental tool. All forms of computing, from the use of electronic devices to programming, are different forms of computational operations (Blackwell, 2002). In most cases, the workflow behind computer-aided design technologies requires making abstractions at a procedural level as computer scientists do. It demands to practice with different levels and forms of abstraction. This should be taken into account when promoting computer-aided design technologies, especially when teaching visual thinkers, such as architecture and art students.

Developing skills to make procedural abstractions is an essential part of CT. When programming, avoiding repeating code, making abstractions and generalizations for the repeating procedures, makes projects easier to program and maintain. Therefore, an assessment method to learn one's abilities, strengths, and weaknesses for the concepts of CT can be quite fruitful to construct individualized learning and teaching strategies to sharpen their CT skills. Developing a better understanding of a designerly approach to CT can also lead to better learning and training techniques suitable for the design technology curriculum at all levels. In this regard, the study aimed to determine the designer's ability to make abstractions at a procedural level in visual computing. Regarding these differences in the use of abstraction and questioning their impact on visual thinkers, two experiments were conducted to assess the cognitive processes of visual thinkers within a systematic framework.

3. Method

3.1. The assessment of a visual thinker's use of abstraction towards computational thinking

Since this is a novel analysis for the designers' use of abstraction skills towards a CAD modeling framework, a vital first step is to establish a coding scheme that captures behaviors at a broad enough level to be applicable to related research and specific enough to identify behaviors that are unique to visual thinkers. In this regard, the assessment method in the study was adapted from one of the existing educational tools for CT.

Because of the differences in the subjects' level of design expertise, the experiments were conducted in two stages. While the first one was conducted to evaluate designers' use of abstraction in visual computing, the second was conducted to compare their performances to non-designers. The objective of both experiments was to explore the modeling processes of the subject that have created the same geometry. For a task, the subjects were asked to model particular shapes with the limitation of specific commands and tools in the AutoCAD interface. In the experiments, the ability of visual thinkers to establish and maintain the relationship between different levels of abstraction was evaluated in the context of the shape making practices, which were developed through a series of pilot studies with architecture students.

3.2. Participants

Regarding the needs of a systematic analysis of the subjects' cognitive actions, the number of participants was limited to 6 participants; 3 sophomore architecture students with a mean age of 22 and 3 non-designers from the law, medical sciences, and engineering disciplines with a mean age of 24. For both experiments, the subjects were chosen on a voluntary basis and were told not to inform the other volunteers to keep their identities confidential. Since the content of the experiments requires the use of spatial and motor skills, all subjects were expected to pass a mini cognitive test before their tasks. Questions were directed to evaluate subjects' competences of using spatial abilities on a computer screen. To extend the differentiation between the participants' levels of design expertise, the architecture students were selected from the sophomore years. Also, it was preferred that all participants had gained an experience of problem-solving in their disciplinary education.

3.3. Experiment implementation

The workflow behind computer-aided design technologies involves different sets of computational operations and requires to think computationally. Just as working with visual programming tools, working with traditional CAD modeling programs also involves practicing CT skills for architects (Senske, 2014; Denning, 2017). Especially, the use of modeling tools and application of commands for the creation of geometric forms requires to make procedural abstractions. Keeping that in mind, studies in architecture education, which involve practicing with computer-aided design technologies, usually attempt to evaluate the designer's process of CT towards the practice of some of the well-known visual programming tools for designers. Though, the workflow behind these programming tools could be very problematic for clearly addressing the designer's cognitive actions in CT. According to Aish & Hanna's (2017) study, working with Grasshopper, Dynamo, and Generative Components software for visual programming requires dealing with different forms of representation, and eventually, abstraction barriers and convoluted workflows would appear in the work process. Also, for an inexperienced user, the learning process of the specified tools and adapting to their user interface would take significant time. Hence, considering the handicaps of using such programming tools for the research objective, the experiment was carried out in a traditional CAD modeling framework.

To provide a convenient user interface for the subjects from different backgrounds, and determine the content of the modeling tasks, a series of pilot studies with architecture students were held before the experiments. In these studies, several well-known programs for traditional CAD modeling were tried out by sophomore architecture students, including AutoCAD, SketchUp, 3DsMax. The content of the modeling tasks was refined in terms of graphic representation and task objectives. Regarding the outcomes from the pilot studies, the experiment setup was conducted. For that, AutoCAD was chosen as a CAD modeling platform with several adjustments in the user interface; and two solid object modeling tasks were defined within certain command limitations and repetitive actions. While the content of the first task was determined to compare the architecture students among themselves, the second task was determined to compare these students with non-designer subjects. The content of the modeling tasks for both experiments was given in [Figure 2].

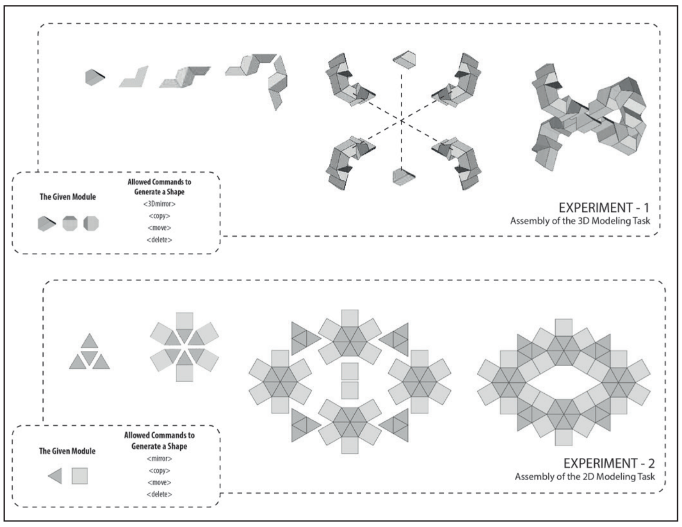

The Given Module
Allowed Commands to Generate a Shape
<3Dmirror>
<copy>
<move>
<delete>
EXPERIMENT - 1
Assembly of the 3D Modeling Task
The Given Module
Allowed Commands to Generate a Shape
<mirror>
<copy>
<move>
<delete>
EXPERIMENT - 2
Assembly of the 2D Modeling Task

Figure 2. *The content of the modeling tasks for both experiments.*

As seen in [Figure 2], the given shapes were created with the repetitive use of specific commands' move, copy, 3D/2Dmirror, and delete' on the given geometric modules. Furthermore, these geometric modules were chosen symmetrical and close enough to toy blocks to be simply generated by the allowed commands, but angular enough to offer surprises and allow for a variety of spatial relationships. While in the first experiment, three designer subjects were given a 3D modeling task by generating modules using <3Dmirror> and <copy> commands;

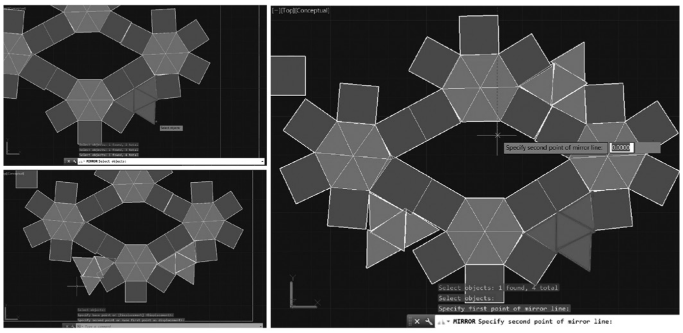
in the second experiment, along with three non-designers, the same subjects were given a 2D modeling task by generating modules using <mirror> and <copy> commands; in the same CAD modeling framework. The subjects were expected to think about and perform computational operations with the allowed commands and modules to accomplish their tasks.

It was preferred that all design students complete the tasks with self-regulated training strategies. Thus, before the experiments, all subjects were given an opportunity to familiarize themselves with the customized AutoCAD interface and the task materials. They were given enough time to practice with the tools, commands, and modules. It was noteworthy that even non-designers did not find familiarizing themselves with the commands and tools troubling. All participants completed the practice phase thoroughly. However, with the comparison of the time spent on the tasks and the accuracy of the subjects' set of operations during the tasks, it was found that the designers were remarkably better than the non-designers [Figure 3].

For the systematic analyses of the subjects' cognitive actions, their interface interactions were recorded to be encoded by using Tobii ProLab 30-day Trial Software. The content of the data collected from the experiments includes video-screen recordings, eye movements, command history scripts of the subjects' modeling processes. Following the completion of all the experiments, the assessment method was used on the subjects' analyzed data.

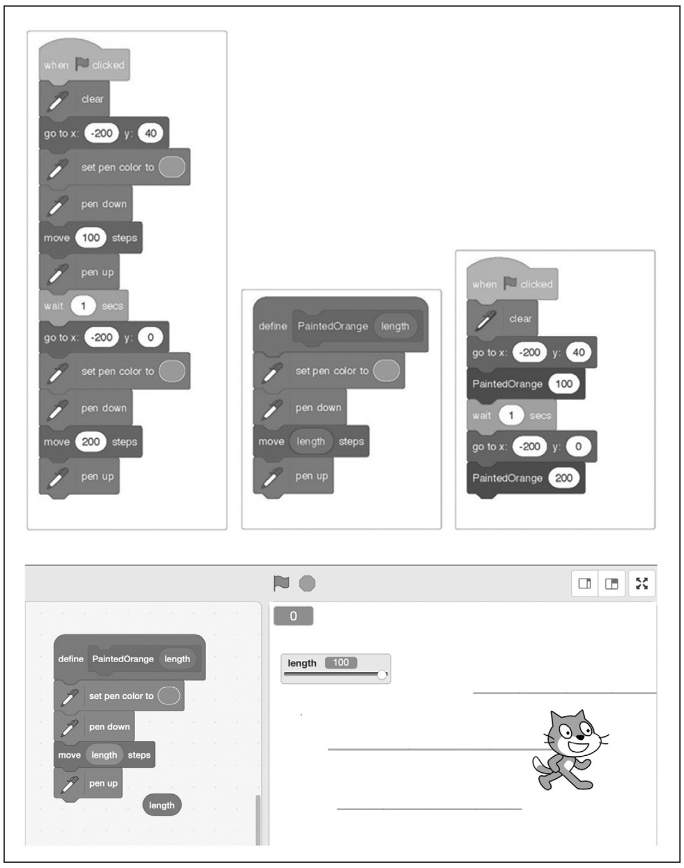
3.4. Assessment method

The offered assessment method for the subject's procedural abstraction skill was adapted from Dr.Scratch. In a nutshell, Dr.Scratch is a web tool that analyzes and scores the content of the Scratch projects. It grades the Scratch users' programming skills in terms of abstraction, parallelization, logical thinking, synchronization, flow control, user interactivity, and data representation concepts. For the assessment of the abstraction skill, this tool analyzes the repeating block patterns in the Scratch user's code and scores



```
Select objects: 1 found, 4 total
Select objects:
Specify first point of mirror line:
MIRROR Specify second point of mirror line:
```

Figure 3. *Still images from a non-designer's use of AutoCAD interface.*



```
when [flag] clicked
clear
go to x: -200 y: 40
set pen color to
pen down
move 100 steps
pen up
wait 1 secs
go to x: -200 y: 0
set pen color to
pen down
move 200 steps
pen up
```

```
define PaintedOrange length
set pen color to
pen down
move length steps
pen up
```

```
when [flag] clicked
clear
go to x: -200 y: 40
PaintedOrange 100
wait 1 secs
go to x: -200 y: 0
PaintedOrange 200
```

Figure 4. *Different abstractions of 'draw a line problem' codes that were made with Scratch, a block-based programming tool (Abstraction, Dr. Scratch, 2019).*

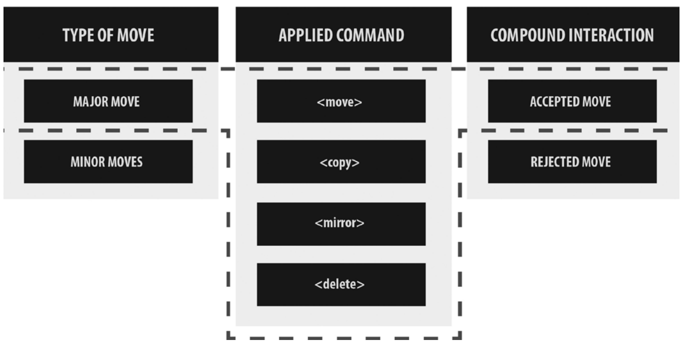


Figure 5. *The categorization of the subjects' interface interactions.*

it down if it finds any unnecessary repeats [Figure 4].

Similar to Dr.Scratch's method, the subjects' repetitive actions in their modeling processes were taken into account for the assessment of their abstraction skills. By tracking down the subjects' interface interactions and identifying their selection of modules and commands to make a model, it was aimed to extract the hidden abstraction patterns from their modeling processes.

3.4.1. Defining coding scheme

After synchronous analysis of the collected data, a coding scheme for the subjects' interactions with the commands and modules, and the segmentation method towards that were defined in order. For the systematic analysis of the subjects' modeling processes, the content of their interface interactions was identified and color-coded under three categories [Figure 5]. Then, the segment chunks were created based on the similarities in these color-coded interactions.

As a first step, within the synchronous analysis of the video-screen recordings and command history scripts, the interactions of the subjects were divided into 'major' and 'minor' moves. This intervention was held to differentiate the major interactions of the modeling process from the minor ones. While the major moves address the actions that are taken to generate a new shape from an existing shape, minor moves address any other actions between two major moves.

As a second step, the major moves of the subjects were identified by their command-shape affiliations. For that,

- the selected command for the shape generation,

- the selected shape to be generated, and

- the generated shapes were taken into account [Figure 6].

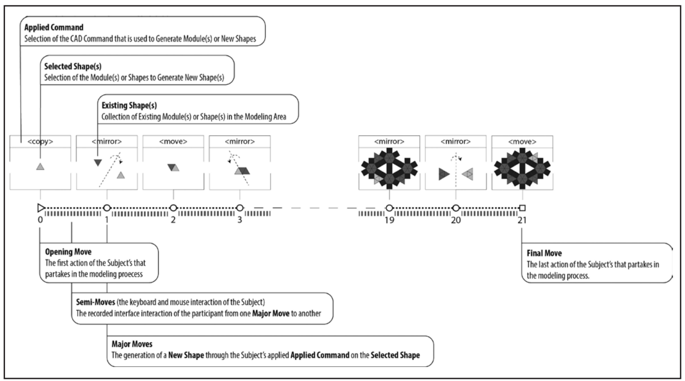
In order to identify the shapes and commands for that, the eye movements of the participants on the selected shape and command history scripts were used.

For the final step, these major moves were divided based on their compound interactions. If the generated shape held for any reverse action, it was considered as a denied move, or else it was accepted.

3.4.2. Determining segment values

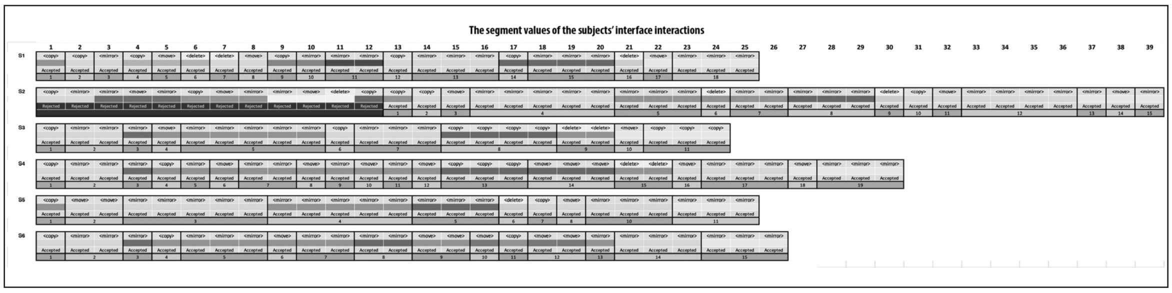
The segment values of the subjects' modeling processes were determined from the final position of their color-coded interactions. And, determining the unnecessarily repeating moves of the subjects by their content has revealed the hidden abstraction patterns in their modeling process. As seen in [Figure 7], the major moves of the subjects were turned into segment chunks based on their content.

For the comparison of subjects' use of abstraction skills, these segment chunks were compared to their major moves. Yet, this comparison would reveal their strengths and weaknesses of making abstractions at different levels.



Applied Command
Selection of the CAD Command that is used to Generate Module(s) or New Shapes
Selected Shape(s)
Selection of the Module(s) or Shapes to Generate New Shape(s)
Existing Shape(s)
Collection of Existing Module(s) or Shape(s) in the Modeling Area
<copy>
<mirror>
<move>
<mirror>
<mirror>
<mirror>
<move>
0
1
2
3
19
20
21
Opening Move
The first action of the Subject's that partakes in the modeling proecess
Final Move
The last action of the Subject's that partakes in the modeling process.
Semi-Moves (the keyboard and mouse interaction of the Subject)
The recorded interface interaction of the participant from one Major Move to another
Major Moves
The generation of a New Shape through the Subject's applied Applied Command on the Selected Shape

Figure 6. The coding scheme for the subjects' interface interactions.



The segment values of the subjects' interface interactions

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
S1	<copy>	<copy>	<mirror>	<copy>	<move>	<delete>	<delete>	<move>	<copy>	<mirror>	<mirror>	<mirror>	<copy>	<mirror>	<mirror>	<mirror>	<copy>	<mirror>	<mirror>	<mirror>
	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted
	1	2	3	4	5	6	7	8	9	10	11		12	13			14	15		
S2	<copy>	<mirror>	<mirror>	<move>	<mirror>	<copy>	<move>	<mirror>	<mirror>	<move>	<delete>	<copy>	<copy>	<copy>	<move>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>
	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted
													1	2	3	4				
S3	<copy>	<mirror>	<mirror>	<mirror>	<move>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<copy>	<mirror>	<mirror>	<mirror>	<copy>	<copy>	<copy>	<copy>	<delete>	<delete>
	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted
	1	2		3	4	5					6	7			8				9	
S4	<copy>	<mirror>	<mirror>	<mirror>	<copy>	<mirror>	<move>	<mirror>	<mirror>	<move>	<mirror>	<move>	<mirror>	<move>	<copy>	<copy>	<copy>	<move>	<move>	<move>
	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted
	1	2		3	4	5	6	7		8	9	10	11	12	13			14		
S5	<copy>	<move>	<move>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<delete>	<copy>	<move>	<mirror>
	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted
	1	2		3					4					5			6	7	8	10
S6	<copy>	<mirror>	<mirror>	<mirror>	<copy>	<mirror>	<mirror>	<mirror>	<move>	<mirror>	<mirror>	<mirror>	<mirror>	<move>	<move>	<move>	<copy>	<move>	<move>	<mirror>
	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted
	1	2		3	4	5			6	7		8		9		10	11	12		13

	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
S1	<delete>	<move>	<mirror>	<mirror>	<mirror>														
	Accepted	Accepted	Accepted	Accepted	Accepted														
	16	17	18																
S2	<mirror>	<mirror>	<mirror>	<delete>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<delete>	<copy>	<move>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<move>	<mirror>
	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted
	5			6	7		8			9	10	11	12				13	14	15
S3	<move>	<copy>	<copy>	<copy>															
	Accepted	Accepted	Accepted	Accepted															
	10	11																	
S4	<delete>	<delete>	<move>	<mirror>	<mirror>	<mirror>	<move>	<mirror>	<mirror>	<mirror>									
	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted									
	15		16	17			18	19											
S5	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>														
	Accepted	Accepted	Accepted	Accepted	Accepted														
			11																
S6	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>	<mirror>													
	Accepted	Accepted	Accepted	Accepted	Accepted	Accepted													
	14			15															

Figure 7. The segment values of the subjects' interface interactions.

4. Process analysis and results

The process analysis of each experiment offers a fruitful discussion on the designer's use of abstraction in visual computing and reflects their tendencies of making abstraction at different levels. While the first experiment revealed that making abstractions at different levels are inevitable parts of the visual computing process, the second experiment showed the intricate relationship between the different levels of abstraction partakes a significant impact in CT. Within this regard, the findings of the experiment analyses can be discussed under two terms.

4.1. Levels of abstraction in visual computing

In the first experiment, all architecture students (S1, S2, and S3) developed different strategies to complete their 3D modeling tasks. While S1 and S2 started the process by analyzing the assembly rules in the given shape, S3 directly started the process by assembling the modules. Compared to others, the subject completed the task with more moves, at the latest [Figure 8].

However, as the number of moves and task completion time of the subjects was compared among themselves, S3's performance of interface interaction was slightly better than S2.

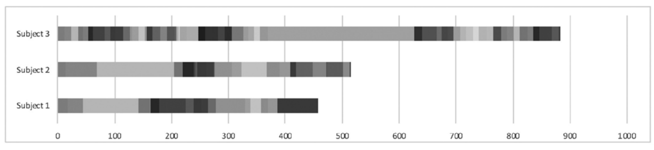
The comparison of the segment values in [Figure 8] shows that the performance of S3 was rather weak as compared to the others. By looking at S3's modeling process, it can be said that their fixation on a single module increased his number of moves and extended the task completion time. And the reason behind this kind of fixation might be related to the subject's tendency to make a conceptual abstraction. Instead of focusing on the function of the commands and tools, S3's fixation on a single module refrained him from making abstractions at a procedural level. Also, the same fixation problem for S3 was seen in their 2D modeling task.

4.2. Designer's level of expertise on the use of abstraction skill in visual computing

Different than the first experiment, the second experiment was presented in more detail to compare the subjects' use of abstraction skills. In this experiment, the performances of the architecture students were significantly better than the non-designers. For the comparison of the subjects' abstraction skills' the subjects' task completion time, number of moves, and segment values are given in the [Table 1].

By looking at the differences in the subjects' segment chunks, it can be said that no correlation has been found between the subject's level of design expertise and their use of abstraction skills. Even though S2 completed the task in the shortest time, S1 was the fastest subject in terms of interface interactions. As S2's performance is compared in terms of task completion time and the segment values, her performance was remarkably better than the other participants.

[Table 2] shows the different assem-



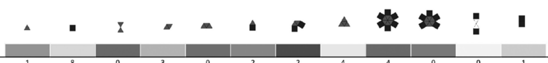
Subject's Name	Number of Minor Moves	Number of Major Moves	Task Completion Time (ms)	Task Completion Time/Minor Moves	Task Completion Time/Major Moves
S1	456	18	455901	999.78	25327.83
S2	51424	24	865468	16.83	36061.17
S3	882	63	1347151	1527.38	21383.35

Figure 8. *Comparison of the segment values in the first experiment.*

Table 1. *Task completion time, number of moves, and segment values of the subjects for the second experiment.*

Level of Design Expertise	Designers			Non-designers		
Subject's Name	S1	S2	S3	S4	S5	S6
Number of Minor Moves	722	442	691	633	511	968
Number of Segment Chunks	18	15	11	19	11	15
Segment Chunks with Repeating Moves	4	5	6	6	6	8
Number of Major Moves	25	27 (39-12)	24	30	25	26
Task Completion Time (ms)	390722	379540	837312	701779	492863	1467107
Total Number of Segment Chunks in Number of Major Moves	0.72	0.56	0.46	0.63	0.44	0.58

Table 2. *Shapes that are generated with the allowed commands in the second experiment.*



Subject's Name	Level of Design Expertise													Total Major Moves
S1	Designer	1	8	0	3	0	2	2	4	4	0	0	1	25
S2		10	15	3	0	0	0	0	3	4	0	3	1	39
S3		8	7	0	0	1	0	0	4	4	0	0	0	24
S4	Non-designer	6	12	0	0	0	0	0	3	3	0	0	1	25
S5		12	7	0	0	1	0	0	4	0	0	0	0	30
S6		11	6	0	0	1	0	0	3	2	3	0	0	26
Average		8.00	9.17	0.50	0.50	0.50	0.33	0.33	3.50	2.83	0.50	0.50	0.50	
Total		48	55	3	3	3	2	2	21	17	3	3	3	

blies of the modules in the second experiment that were generated and used by the subjects. By looking at their number of occurrences in the major moves, it can be said that the non-designers had become fixated with single modules more than the designers had. On the other hand, the creation of unique shapes can be seen in both groups.

5. Conclusion and future remark

Similar to how the use of computers in design defined the early research frameworks of design cognition, the use of computational thinking as a mental tool is now ready to challenge traditional notions of design cognition. Dealing with computational technologies requires to have CT as a skill, which also means to affiliate its cognitive features in the correct forms. For a visual thinker, this can be achieved by developing a better understanding of these features and their counterparts through a set of practices in different levels and forms of abstraction. Once these perceptual tasks are built on basic skills, it will be easier to acquire them at higher levels (Ware, 2008; p: 172).

By considering a designer as a visual thinker, this study shed light on a designer's use of abstraction in CT. It revealed that the designers tend to make abstractions at different levels in a CAD modeling framework, and the abstraction skill at different levels dominantly partakes in the process of visual computing. The assessment of subjects' cognitive processes has shown that the ability to make abstractions at a procedural level affects the process of CT. Regarding the subjects' performances in the CAD modeling tasks, it can be said that the better use of CT relies heavily on finding a balance between the different levels of abstraction, or in other words, making generalizations towards them. And a proper evaluation of a visual thinker's abstraction skill relies on determining their tendencies to make procedural and conceptual abstractions. Because, whether it is for a scientific or perceptual task, making abstractions always require making generalizations. As Arnheim exemplifies (2004), "true generalization is the way by which a scientist perfects his concepts and the artist his images. It is an eminently unmechanical procedure." For a visual thinker, making generalizations is not a matter of collecting a random or an infinite number of instances; in many ways, it is a matter of finding patterns.

In the future, the assessment method in the research is planned to be developed into an educational tool for visual thinkers. This type of assessment would provide constructive feedback about the students' use of CT and reflect their use of abstraction, pattern recognition, decomposition, and algorithmic thinking skills. For educators, who are looking for better ways of integrating computational technologies into the early years of design education or strengthen their educational efforts, this type of educational tool can be quite fruitful. With this tool, the learning experiences of students can be customiz ed by digging deeper into their cognitive strengths and weaknesses.

References

Aish, R., & Hanna, S. (2017). Comparative evaluation of parametric design systems for teaching design computation. *Design Studies*, *52*, 144–172. https://doi.org/10.1016/j.destud.2017.05.002

Arnheim, R. (2004). *Art and visual perception: A psychology of the creative eye, fiftieth anniversary printing* (Rev. ed.). Berkeley and Los Angeles.

Blackwell, A. F. (2002, June). *What is programming?*. In *PPIG* (p. 20).

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39. https://doi.org/10.1145/2998438

Dror, I. (2011). Brain friendly technology: What is it? And why do we need it?. *eLearn, 2011*(8), 4.

Dr.Scratch, (n.d.). *Abstraction*. Retrieved November 15, 2019, from http://www.drscratch.org/learn/Abstraction/

Erhan, H. I., Youssef, B. B., & Berry, B. (2012). Teaching Spatial Thinking in Design Computation Contexts: Challenges and Opportunities. In *Computational Design Methods and Technologies: Applications in CAD, CAM and CAE Education* (pp. 365-389). IGI

Global.

Goldschmidt, G. (2011). Avoiding design fixation: transformation and abstraction in mapping from source to target. *The Journal of Creative Behavior*, *45*(2), 92–100.

Guzdial, M. (2008). Education Paving the way for computational thinking. *Commun. ACM*, *51*(8), 25. https://doi.org/10.1145/1378704.1378713

Guzdial, M. (2010). Does contextualized computing education help?. *ACM Inroads, 1*(4), 4-6.

Lawson, B. (2006). *How designers think*. Routledge.

Locke, J. (1979). *An Essay Concerning Human Understanding*, ed. P. Nidditch (1975). Oxford: Clarendon Press.

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking.* National Academies Press.

Schön, D. A. (1987). *Jossey-Bass higher education series. Educating the reflective practitioner: Toward a new design for teaching and learning in the professions.* Jossey-Bass.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation : A theoretical framework. *Education and Information Technologies, 18*(2), 351–380.

Senske, N. (2014). Digital minds, materials, and ethics: linking computational thinking and digital craft. *Proceedings of the 19th International Conference on ComputerAided Architectural Design Research in Asia CAADRIA,* 841–850.

Ware, C. (2008). Visual Thinking: for Design (Morgan Kaufmann Series in Interactive Technologies).

Weisberg, R. W., & Reeves, L. M. (2013). *Cognition: from memory to creativity*. John Wiley & Sons.

Wing, J. M. (2017). Computational thinking's influence on research and education for all Influenza del pensiero computazionale nella ricerca e nell'educazione per tutti. *Italian Journal of Educational Technology*, *25*(2), , 7–14. https://doi.org/10.17471/2499-4324/922

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118