International Journal of Advanced Research and Review

www.ijarr.in

THE CONVERGENCE OF COMPUTATIONAL THINKING AND ITS COGNITIVE ASPECTS IN ARCHITECTURAL EDUCATION

Elif Öksüz*, Gülen Çağdaş

Istanbul Technical University
Corresponding Author: Elif Oksuz, Email: eoksuz@itu.edu.tr. Istanbul Technical University, Faculty of Architecture, 34437, Istanbul/Turkey.

ABSTRACT
This article offers a nuanced understanding of Computational Thinking (CT) as a soft skill for non-computer scientists by addressing the conceptual differences in the learning experience of its cognitive aspects. In contemporary education, the cognitive contribution of CT is expected to be more than computer literacy. Over a decade, CT has been addressed as a '21st Century Skill' outside the Computer Sciences and promoted in the early stages of compulsory education. However, challenges remain in the existing culture of teaching CT in architectural education to better meets the 21st Century learners' needs. For that, we present a curated collection of existing knowledge from art, design, and computer sciences. The article does not aim inventing a set of new practices of teaching CT but shedding light to the culture to use new pedagogies determined as effective teaching/learning in architectural education. It is expected to be one of the preliminary studies that contribute to design education literature.

Keywords: architectural education; computational thinking; abstraction; soft skill; pattern recognition; algorithmic thinking, decomposition

INTRODUCTION
We are currently facing a paradigm shift toward computing and coding culture. From digital tools to robotics to artificial intelligence, computational technologies are visibly affecting our everyday life and shaping our cognitive development. In this framework, we acquired new fundamental skills to work in better collaboration with these technologies. Or as Colin Ware summarizes (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 designed tools change how people think" (181). The paradigm we are heading towards needs the technology to be 'brain friendly' to complement our cognitive processes (Dror 2011, 2) and our mental processes to be similar to the series of internal states as if the computer carries out a program (Weisberg and Reeves 2013, 17). Hence, we are imposed on employing and collaborating in different modes of thinking; and in this paradigm, Computational Thinking is the most demanding among others. In contemporary education, Computational Thinking (CT) is recognized as "a mental activity in formulating a problem to admit a computational solution combining the intelligence of humans and the intelligence of machines" (Wing 2017, 8). So the way that CT contributes to our cognitive development is expected to be more than computer literacy. It is neither considered as a machinery skill to be performed only by computers nor as a disciplinary skill only for computer scientists. Over the last decade, it has been recognized as one of the fundamental aspects of our cognitive development outside the Computer Sciences and promoted in the early stages of compulsory

education. From K-12 to collegiate level, this cognitive aspect has been inclusively affirmed under different teaching methods and strategies.

Since CT has been addressed as one of the twenty-first-century skills, its pedagogical alignment to outer disciplines requires a comprehensive evaluation of its cognitive aspects. In fact, for design disciplines like architecture, where the advanced technologies are intensely used under the visual thinking processes, developing a better understanding of computational thinking is particularly essential. As a mental process, this mode of thinking has the potential of engaging the human mind and technology on common ground with a systematic framework. For future designers, becoming competent with digital tools and working in interactive and collaborative platforms require having computational skills towards a disciplinary perspective.

However, challenges remain in the existing culture of teaching computational thinking in architectural education to better meets the 21st Century learners' needs. With few exceptions, most architecture schools still disregard the attributes of computational thinking for a designer's cognitive development and employ it as a technical skill, focus on mere forms of computerization. Yet, these non-contextualized or over-contextualized forms of computational thinking prevent the adoption by the first year. Instead of asking 'When do architecture students need CT?', 'How does CT contribute to a designer's learning experiences?', 'What hinders/helps students' learning CT?' and 'In which ways CT can be constructed for a designerly way of thinking?' most of the studies in architectural education focus on the advanced computational models; and therefore, miss out how novice students can benefit from this mode of thinking.

The more we rely on CT as a technical skill, the longer that the pedagogical gap remains for the technology endorsement in architectural education. Thus, in order to bridge this gap, the current pedagogy of architectural education may require a transformation of some of the computational skills and concepts, and the vocabulary of computers referred to computational thinking towards its conceptual aspects. And the implementation of these aspects to architectural education curricula, therefore, requires a systematic approach to their conceptual counterparts. Because a better grasp of computational thinking can widen the reach of creative computation more generally and can empower designers to face the challenges of contemporary design education and practice in so many ways. In this regard, instead of presenting another case study to encourage CT over some of the well-known computational design models, we prefer focusing on the theoretical background of CT and address its cognitive contributions to novice designers as a soft skill.

Starting from its conceptual development, our article discusses the pedagogical alignment of this mode of thinking in architectural education; and the convergence of its conceptual counterparts in design literature. For that, it offers to develop a nuanced understanding of computational thinking as a soft skill for non-computer scientists; and addresses the conceptual similarities and differences in the learning experience of its cognitive aspects with a systematic approach. And so, the aim of this article is not about inventing a set of new practices for teaching CT but shedding light to the culture to use new pedagogies determined as effective teaching/learning in architectural education. We offer a curated collection of existing knowledge from art, design, and computer sciences. The article presents a comprehensive literature review of CT's core skills (abstraction, algorithmic thinking, decomposition, pattern recognition) and their conceptual counterparts in design literature. Within this regard, it is expected to be one of the preliminary studies that contribute to design education literature.

The Conception of Computational Thinking and Its Core Concepts

Although the terminology of computational thinking exists in education literature for a long time under Computer Sciences, its introduction to outer disciplines as a cognitive skill is relatively new. In contemporary literature, CT is considered as "a thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Cuny, Snyder, and Wing 2010). As a mental process, this concept grabbed our first attention through Cognitive Sciences with Herbert Simon and Alan Newell's 'general problem-solving model.' With the analogy of "mind as a computer," Simon and Newell (1971) introduce CT at a functional level as if a human carrying out a program (Weisberg and Reeves 2013, 17). And its former reflection in the education literature appears in Alan Perlis's 'introductory programming' and Seymour Papert's 'procedural thinking' models. While Perlis's focus was on 'teaching how computers work by adopting problem-solving methods' (Perlis 1962), Papert's aim was 'introducing procedural thinking to K-12 students with LOGO programming language' (Papert 1980). However, in both models, the cognitive approach to computational thinking was not taken any more than a Turing-based problem-solving model. Gradually, the emergence of new computational technologies and a transition from 'hard computing' to 'soft computing' models throughout the years have changed the reputation of CT for outer disciplines (Dodig-Crnkovic 2011) and helped the understanding of "computation = programming" to become obsolete (Czerkawski, Lyman, and,Ä¶ 2015). In contemporary literature, CT is taken far more than a Turing-based approach (Wing 2010). In "Computational Thinking" in the Association of Computing Machinery (2006), Jeannette M. Wing reintroduces this concept for "solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science" (33). With her definition, the former Head of Microsoft Research Lab and academic is known as the first and most influencing person who carried CT to an interdisciplinary level in the education literature.

As a mental process, this mode of thinking has the potential of engaging the human mind and technology on common ground with a systematic framework. And from an educational perspective, students and teachers are the former agents of widening the impacts of this skill and developing better ways of teaching/learning CT in terms of educational technology (Czerkawski and Xu 2012, 2608). For Mishra and Yadav (2013), CT has a potential "to foster creativity by allowing students to not only be consumers of technology but also to build tools that can have a significant impact on society" (11). Alternatively, "learning CT enables individuals to become more effective problem solvers by teaching them to recognize computable problems and approaching the problem-solving process skilfully" (Czerkawski, Lyman, and 2015, 57). Therefore, one of the major purposes of endorsing CT in the early stages of education is introducing the logic of computing and encouraging its use as a way-out in different forms of problem-solving or problem defining tasks. Aside from that, its cognitive impacts on problem-solving ability, what CT may present to individuals is listed as follows (Barr and Stephenson 2011):

confidence in dealing with complexity,
persistence in working with difficult problems,
tolerance for ambiguity,
the ability to deal with open-ended problems and
the ability to communicate and work with others to achieve a common goal or solution (51).

Considering its attributions to our cognitive development and its support in the use of digital technologies, CT plays a strategic role in shaping the near future of educational technology.

From kindergarten to higher education, this mode of thinking is no longer acquired as a technical ability or programming skill. Instead, it is promoted as a soft skill, which can be developed through the deep learning experiences and practices of certain cognitive aspects.

In this respect, organizations like National Research Council (NRC) have already acknowledged the necessity of introducing CT in the early stages of education by claiming its potential on "preparing students to succeed and thrive in our increasingly technological society; supporting inquiry in other disciplines; and enabling personal empowerment to tackle complex problems" (NRC 2011). Following that, many other private and public organizations, such as National Science Foundation, British Royal Society, European Commission, Google for Education and Microsoft Research Lab, support and work on the dissemination of CT among different online/offline educational platforms. For a better understanding of CT and its transaction to outer disciplines, the terminology of CT is usually affiliated with certain cognitive skills and actions by these organizations as follows: parallelization, simulation, data analysis, data modelling, data collection, automation, problem-solving, pattern recognition, generalization, decomposition, debugging, algorithmic thinking, and abstraction (Table 1). So the educators and students may be able to develop their own methods and strategies or individual learning experiences for acquiring CT as a soft skill.

However, most of the affiliated terms are borrowed from Computer Sciences and on a demand for background experience in programming; and few of them are applicable as core concepts to its interdisciplinary terminology. As Chenglie Hu (2011) highlights, "Certain ways of thinking may not appear 'computational' despite their potential computational implications." And "paving a way for computational thinking from computer science to other academic fields may require adapting this mode of thinking to match the needs of 'novices' and non-majors" (Guzdial, 2008). As a result, the introduction of CT to non-computer scientists usually upholds its four prominent features among others: algorithmic thinking, abstraction, pattern recognition, and decomposition (e.g. Microsoft Education, Google for Education, and BBC's Introduction to Computational Thinking). For instance, Microsoft Education (2019) constructs the interdisciplinary terminology of CT over these four cognitive aspects and exemplified with real-life experiences as given below.

Following that, Wing (2008) summarizes what CT needs as a cognitive process in three stages: "Defining abstractions and working with multiple layers of abstractions and understanding relations among different layers" (3718). Thus, for computational thinking, these core concepts should not be seen as separate skills, but one is always linked to another. And for a better pedagogical alignment of CT to outer disciplines, the deep learning experiences of computational thinking should be constructed upon that and their conceptual counterparts.

The Pedagogical Alignment of CT in Architectural Education

Regarding CT’s core concepts, not only in terms of technology use but also in terms of its cognitive attributions, a novice designer can quite benefit from this skill. “Working creatively and effectively with computers, digital fabrication machines, and other devices require a new set of workflows and adaptations to academic and professional behaviours” (Doyle and Senske 2016, 207). Starting from the early stages of their education, architecture students are likely to become digital literates/performers. Systematically, some of the conventional methods in architectural education are about to be replaced with advanced computational methods. For instance, “while descriptive geometry is no longer required (or even offered) by most design curricula, no corresponding pedagogy has arisen that addresses its successor, geometric computation, in a way that finds persistent relevance in a design context" (Ko and Steinfeld 2018, 4). Additionally, 'the competence of using digital design technologies' has been acknowledged as one of the fundamentals of visual communication skills in Professional Architectural Accreditation in America (NAAB 2009). Thus, for the twenty-first-century designers, this high demand on computational technologies entails a good practice of CT from the very beginning. Not only for students but also for educators and technology developers, Architectural Education is in need of acknowledging CT more than technical proficiency. "CT is a way of thinking about problems, and solving problems with computational thinking can leverage the power of technology, like software" (Microsoft 2019). However, one of the major problems with encouraging computational thinking to novices is that architectural education seems to take such a narrow view of what CT is and how it can contribute to our cognitive development and design knowledge. "As new territories of digital design practice are emerging continuously and for the last twenty years, the mainstream introduction of computing, a host of digitally-inspired movements and aesthetic styles have come and gone – including bio-inspired design, digital fabrication, emergent design, and data visualization - each of which suggest a different set of foundational topics" (Ko and Steinfeld 2018, 4). Embedding the concept of computational thinking in such topics overshadows the cognitive support of CT as a mental activity and its adoption as a soft skill by a novice designer. As Kostas Terzidis (2006) mentioned earlier, "the dominant mode of utilizing computing in the architectural education curricula is that of computerization; entities or processes that are already conceptualized in the designer's mind: entered, manipulated, or stored on a computer system" (58). Within the endorsement of advanced digital design and modelling tools and platforms, these content-specific courses and implication of advanced modelling tools are usually upholding the employment of computational thinking as a hard/technical skill. Subsequently, the contribution of CT to the designer's cognitive development is not sufficiently covered by the current architectural education curricula. It does not give enough weight to the deep learning experiences towards CT's core aspects. As Brian Johnson (2017) posits "with the exception of a few schools where 'digital' or 'paperless' studio experiments were undertaken, the advent of digital analysis and media options has been addressed with specialized course content focused on acquiring skills with the new tools, each one an option on top of the traditional structure" (186-187). Because of that, most educators in the first-year of architectural education are sceptical on introducing the conceptions of computing and computational thinking to their pedagogical agenda.

So starting from the first year, encouraging novice designers to think computationally towards their disciplinary background can help close the remaining gaps in the design pedagogy. As Mark Guzdial points out (2010), "If students don't learn the material well in the first place, they can't possibly transfer it to new situations" (5). In order to complement the designer's cognitive development, the deep learning experiences of CT can be built upon that. For that, we should not aim for the centrality of software/programming itself but for the foundational concepts that underpin it. In this regard, developing better learning experiences would require a systematic approach to the affordance of these concepts. Following from this, starting from the conception of abstraction, let us address in what ways these cognitive aspects may require a reconsideration for a designerly way of computational thinking so that for a novice designer's education they would be encouraged within an embodied framework. Thus, instead of addressing certain computational models in design education, our study lays the groundwork to enable the reader to engage in all of them. To this end, we emphasize a specific combination of these core concepts as the key to understanding design computation.

The Convergence of Computational Thinking and Its Cognitive Aspects in Architectural Education towards Embodiment

Understanding what CT requires as a mental process can be useful for understanding its causal relations to design education, and therefore empowering CT in ways that enhance the designer's cognitive process. However, a better understanding of its cognitive aspects is also necessary for addressing the conceptual differences or resemblances between these two fields. Encouraging such convenient skills under Computational Thinking can deepen a novice designer's conceptual understanding of digital technologies with disciplinarily grounded knowledge and different forms of design representations. For the convergence of CT towards a disciplinary perspective, the conceptual understanding of its cognitive skills should be acknowledged properly.

In contrast to science disciplines, with an artisan education background, starting from the very first year, architecture students are likely to acquire the cognitive aspects of computational thinking with more intuitive approaches. With its complementary design courses, the first year of Architectural Education involves an intense focus on visual thinking and visual communication through different tools and environments, such as drawing, sketching, modelling, etc. And the skills and techniques involved in such practices are mostly converged from art and design fields. So, even though the core cognitive aspects of CT, which are abstraction, pattern recognition, decomposition, and algorithmic thinking, are no alienated terms to Architectural Education, the ways they are introduced to designer's cognitive framework slightly differ from the way they are engaged in computational thinking. And these subtle differences can change the meaning and the affordance of these computational thinking skills thoroughly. As Hu (2011) highlights, "One may acquire the very same thinking skill from a very different learning experience; thus, it might be a philosophical challenge to stress the importance of computational thinking today while its products might have existed even before modern computers were born." Thus, in case of architectural education, instead of mere acknowledgment of computing as the production of a set of axiomatic computational abstractions, computational thinking may require a discursive, perspectival, material and embodied experiences (Sengupta, Dickes, and Farris 2018, 3). Within this framework, we address the conceptual counterparts of CT's core cognitive aspects and highlight the similarities and differences found in the practice of these skills for a visual thinker, so the better learning experiences for future designers can be constructed upon them.

The Design Fundamentals for a Visual Thinker: Abstraction, Decomposition, and Pattern Recognition

For instance, the conception of abstraction or abstract thinking is fundamental to both Computer Sciences (Wing 2008) and Design Education (Goldschmidt 2011). From the perspective of a computer scientist, Wing (2008) summarizes what abstraction may entail for a computer scientist 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, 3717-3720). Moreover, as Sengupta et al. (2013) highlights, Wing's definition is very similar to John Locke's view on abstraction. For Locke (1979), 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 the particular ideas received from particular objects and separate them "from all other existences and the circumstances of real existence, as time, place, or any other concomitant ideas" (Arnheim 2004, 154). On the other hand, this isolated approach may not be fruitful as much as expected for visual thinkers, or design students for this matter. Rudolf Arnheim

(2004) argues that 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 (154). Hence, the abstraction of a visual thinker may not always constitute in the way that computational thinking requires. As opposed to a scientist’s approach to abstract thinking, the abstraction of a designer is very perceptual. As Mike Tovey (2015) puts it, “designing involves a peculiar and particular blend of thinking processes, which are the distinguishing characteristics shared by different sorts of designer. This could be labelled the designerly way of knowing, making use of various forms of intelligence, particularly visuospatial thinking” (58). Thus, for a visual thinker, dealing with abstractions requires engaging with multiple forms and genres of representations beyond drawing, sketching or modelling; it is often translating between these representations, which involves perceptual interpretations. Additionally, a high abstraction always has the risk of detaching itself from the wealth of actual existence (Arnheim 1974, 147). Even though nothing is more concrete than colour, shape, and motion in a novice designer’s visualizations, there is always a possibility in which a designer’s abstraction may not reflect the necessary information to proceed with their work. An artist or a designer, a visual thinker always can mentally add simple patterns to a sketch to test possible design changes before making any changes to the sketch (Ware 2010, 170).

Conversely, for a scientific abstraction, the ignorance or missing out the great majority of the information in such tasks may cause unexpected/unrelated outcomes. In this regard, two features become prominent in the process of abstract thinking: making generalizations and determining the level of abstraction while making these generalizations. Whether it is for a scientific or perceptual task, making abstractions always require making generalizations (Wing 2008, Arnheim 2004). It relies on finding the balance in making abstractions through decomposition and pattern recognition processes. Arnheim exemplifies that (2004), “true generalization is the way by which the scientists perfects his concepts and the artist his images. It is an eminently unmechanical procedure.” Nonetheless, for a visual thinker, generalization is not a matter of collecting a random or an infinite number of instances. Because “in many ways, the real power of visual thinking rests in pattern finding; the choice of pattern is important for the visual queries to be processed by the intended viewer” (Ware 2008, 174). For instance, Ware (2008) claims that “the perceptual tasks, such as reading a contour map, understanding a cubist painting and interpreting an x-ray image, involve specialized pattern recognition skills; and once they are built on more basic skills, it will be easier to acquire them in higher levels” (172).

In architectural education, one of the well-known models of visual calculation is called Shape Grammars, the rule-based systems for describing and generating designs by George Stiny and James Gips (1971). It is one of the highly recognized models in the first-year curricula of architectural students. What it offers is “computing directly with shapes in two or three dimensions, rather than with symbols, words, numbers, or other abstract forms” (Knight 1999, Stiny 2006). However, as seen in shape grammar model, focusing solely on form and visual calculation may not sufficiently cover the deep learning experience of pattern recognition and decomposition skills of CT. “Visual thinking consists of a series of acts of attention, driving eye movements and tuning our pattern-finding circuits. These acts of attention are called visual queries, and understanding how visual queries work can make us better designers” (Ware 2008, 3). The digital tools and interactive design platforms used in architectural education needs the concept of pattern recognition and decomposition skills to be more than dealing with mere shapes or geometric forms. First, “we tend to use visual images that are prototypical reminders of categories in the world, rather than accurate images of categories like chairs and movie stars” (Baars and Gage 2010, 359). Concurrently, although we are not nearly as good at identifying and remembering motion patterns as we are at remembering spatial patterns, “a way of responding to a pattern is also a pattern, and usually, one we have executed many times before. These response patterns are the essence of the skills that bind perception to action” (Ware 2010, 173-4). Therefore, we need to link these one another in the process of computational thinking is essential. Which takes us to the next step: algorithmic thinking.

Algorithmic Thinking: A Designerly Way of a Problem Formulation

Algorithms are used explaining how to do something in precise ways (Reas and McWilliams 2010, 13) and usually, they consist in finding a solution meant to be encoded in a computer (Combéfis, Van Den Schrieck, and Nootens 2013, 4). A typical algorithm is the finite list of instructions to be applied on an input to produce an output. What algorithms offer as a thinking mode slightly differs from our conventional way of thinking. Although constructing algorithmic thinking is usually recalled as a machinery skill, algorithms are quite pertinent for a human mind and can be formed in different ways, from describing a direction to knitting. Gerald Futchek (2006) defines the stages of algorithmic thinking as “a pool of abilities that are connected to constructing and understanding algorithms, which consists

- analysing given problems,
- specifying a problem precisely,
- finding the basic actions that are adequate to the given problem,
- constructing a correct algorithm to a given problem using the basic actions,
- thinking about all possible special and normal cases of a problem and
- improving the efficiency of an algorithm (160).

In computer sciences, the learning experience of algorithmic thinking usually involves practicing programming languages, dealing with codes or pseudocodes. However, since written algorithms may be too abstract for novices to understand the syntax of the language that describes algorithms” (Futschek and Moschitz 2010, 3), starting with the programming languages can be a real challenge for novice designers. Kostas Terzidis (2006) implies, “In studying the articulation of algorithms one may be able to discern similarities with design. While such a study may lead to the development of procedures that may be useful in design, more importantly, it may reveal certain clues about design as a mental process” (20). Like algorithms, design can be seen as a set of procedures that lead stochastically towards the accomplishment of a goal. In a way, algorithmic thinking may be considered as a designerly way of a problem formulation. And just as the way Terzidis (2006) suggested, algorithmic thinking can be quite supportive for a novice architect’s logical analysis process. Mike Tovey (2015) enlightens us about the designer’s mental process with the following words: “the process of creating a design needs two or more simultaneous tracks of thought covering both logical analysis and creative thinking; and for a number of types of designing the reframing of the context through analytical activities, such as data acquisition, problem definition and solution evaluation requires an explicit and objective manner”(54). Thus, in certain circumstances, a designer may be in need of a linear structure of the algorithmic process for problem formulations. And this possibility opens up a more intricate relationship between a designer’s perspective and computational thinking than has been previously possible. Instead of using algorithms to copy, simulate, or replace manual methods of design, they can be studied as methodologies that operate in ways similar, parallel, or complementary to that of the human mind (Terzidis 2009, 19-20). However, these instructions do not purely consist the objective manipulations of the inputs; they may also involve the subjectivity in creating meaningful relations and causalities between the inputs as performing entities. Considering designers’ unconventional ways and nonlinear paths in design activities, algorithmic thinking usually constitutes in problem formulation activities for architectural students. Constructing an algorithm may involve a high level of subjectivity and causality for design activities. Therefore, for novice designers, the deep learning experience of algorithmic thinking should be constructed in flexible ways to cover the designerly way of thinking. In order to enhance Futchek's algorithm definition (2006) for a designerly way of thinking, Derek Ham (2015) offers three new abilities as follows:

to analyse a problem that constantly changes,
to find new actions based on the basic actions and
to construct an algorithm flexible enough to respond to the change of the given problem (90-91).

CONCLUSION

Neither 'computation' nor 'computational thinking' is novel for architectural education. From design studios to computer programming/modelling courses, the adoption of computational thinking to architectural education has broad implications. Including the preparing of future designers for operating in different fields of design and this has implications inside and outside the architecture discipline. Linking our current teaching and scholarship with the foundational research about these concepts will help us reframe their designerly way of uses and foster CT's encouragement, starting from the first-year of architectural education. Nonetheless, whether through computational thinking or not, these skills are fundamental parts of a novice designer's learning objectives.

Well-constructed curricula for design complementary courses with a systematic alignment of CT's core concepts would give the result of improved student learning in many ways. And addressing the resemblances and differences of CT's core concepts in design would ease the implementation of these concepts upon a disciplinary perspective. Otherwise, adopting it as a technical skill or a type of programming skill, most of the studies in the literature do not give enough weight to the thought and experience relationship that CT presents for a designer. As a consequence, it limits the contribution of CT to architectural education to a technical gain and reflects the curriculum as a technical skill, a kind of computerization; between design and calculation training, pedagogical gaps arise.

Nevertheless, in architectural education, a deep learning experience of computational thinking can be constructed with or without machines. The key point is constructing this experience by encouraging CT's core components altogether in a contextual framework. For architectural education, this contextual framework should be built upon the embodiment. As visual thinkers, architecture students tend to think, experience, and learn through embodiment. They are encouraged to collect and evaluate data and visualize their thoughts by making different forms of abstraction. Thus, it is essential for novice students to develop their computational skills upon that. The use of abstraction towards computational thinking should recognize the perceptual differences in the education of abstract thinking for designers and artists.

REFERENCES

Arnheim, Rudolf. 2004. Visual Thinking. University of California Press.
Arnheim, Rudolf. 1974. Art and Visual Perception. University of California Press.
Baars, Bernard J., and Nicole M. Gage. 2010. Cognition, brain, and consciousness: Introduction to cognitive neuroscience. Academic Press.
Barr, Valerie, and Stephenson, Chris. 2011. "Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?." Inroads 2, no. 48-54.
Johnson, Brian R. 2017. Design Computing.

Combéfis, Sébastien, Virginie Van Den Schrieck, and Alexis Nootens. 2013. 'Growing Algorithmic Thinking Through Interactive Problems to Encourage Learning Programming'. Olympiads in Informatics 7: 3–13. https://www.mii.lt/olympiads_in_informatics/pdf/INFOL115.pdf.
Cuny, Jan, Larry Snyder, and Jeannette M Wing. 2010. 'Demystifying Computational Thinking for Non-Computer Scientists'. Unpublished Manuscript in Progress, Referenced in Http://Www. Cs. Cmu. Edu/~ CompThink/Resources/TheLinkWing. Pdf.
Czerkawski, Betul C, and Lyman, Eugene W .2015. 'Exploring Issues About Computational Thinking in Higher Education - ProQuest Social Sciences Premium Collection - ProQuest' 59 (2).
Czerkawski, Betul, and Xu. L 2012. 'Computational Thinking and Educational Technology'. Proceedings of World Conference on Educational, 2607–2610. https://www.editlib.org/p/41130/proceeding_41130.pdf.
Dodig-Crnkovic, Gordana. 2011. 'Significance of Models of Computation, from Turing Model to Natural Computation'. Minds and Machines 21 (2). Springer: 301–322. doi:10.1007/s11023-011-9235-1.
Doyle, Shelby, and Nick Senske. 2016. 'Exploring Learning Objectives for Digital Design in Architectural Education'.
Doyle, Shelby, and Nick Senske. 2016. 'Between Design and Digital: Bridging the Gaps in Architectural Education'.
Dror, Itiel E. 2011. Technology Enhanced Learning and Cognition. Vol. 27. John Benjamins Publishing.
Futschek, Gerald. 2006. "Algorithmic thinking: the key for understanding computer science." In International conference on informatics in secondary schools-evolution and perspectives, pp. 159-168. Springer, Berlin, Heidelberg.
Futschek, Gerald, and Julia Moschitz. 2011. 'Learning Algorithmic Thinking with Tangible Objects Eases Transition to Computer Programming'. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7013 LNCS: 155–164. doi:10.1007/978-3-642-24722-4_14.
Goldschmidt, Gabriela.2011. "Avoiding design fixation: transformation and abstraction in mapping from source to target." The Journal of creative behavior 45, no. 2, 92-100.
Google for Education. 2011. "Computational Thinking." 2019. Accessed February 10. https://edu.google.com/resources/programs/exploring-computational-thinking/#!ct-overview.
Guzdial, Mark. 2008. 'Education Paving the Way for Computational Thinking'. Communications of the ACM 51 (8): 25. doi:10.1145/1378704.1378713.
Guzdial, Mark. 2010. "Does Contextualized Computing Education Help ?" ACM Inroads 1 (4). ACM: 2008–2010.
Ham, Derek. 2015. "Playful Calculation: Tangible Coding for Visual Calculation." PhD diss., Massachusetts Institute of Technology.
Hu, Chenglie. 2011. 'Computational Thinking – What It Might Mean and What We Might Do About It'. Computer Science Education, 223–227.
Ko, Joy, and Kyle Steinfeld. 2018. Geometric Computation: Foundations for Design. Routledge.
Knight, Terry. 2000. "Shape grammars in education and practice: history and prospects." International Journal of Design Computing 2, no. 67.
Locke, John .1979. An essay concerning human understanding. New York: Oxford University Press.
Microsoft Education Community. 2018. "Computational Thinking and its importance in education." Last modified, February 13, 2019. https://education.microsoft.com/courses-and-resources/courses/computational-thinking-and-its-importance-in-education

Mishra, P. & Yadav, A. 2013. Rethinking technology & creativity in the 21st century. TechTrends: Linking Research & Practice to Improve Learning, 57(3), 10-14.
NAAB. 2009. “Substantial Equivalency”. Last Modified, February 11, 2019. ttps://www.naab.org/international/substantial-equivalency/
Stiny, George. 2006. Shape: talking about seeing and doing. MIT Press
Papert, Seymour. 1980. Mindstorms: Children, computers, and powerful ideas: Basic Books, Inc.
Perlis, Alan. 1962. The Computer in the University. Computers and the World of the Future. M. Greenberger. In: Cambridge, Massachusetts, The MIT Press.
Reas, Casey, and Chandler McWilliams. 2011. Form+ Code: in design, art, and architecture. Princeton Architectural Press.
Sengupta, Pratim, John S. Kinnebrew, SatabdiBasu, Gautam Biswas, and Douglas Clark. 2013. "Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework." Education and Information Technologies 18, no. 2, 351-380.
Sengupta, Pratim, Amanda Dickes, and Amy Farris. 2018. ‘Toward a Phenomenology of Computational Thinking in K-12 STEM’. Computational Thinking in STEM Discipline: Foundations and Research Highlights. doi:arXiv:1801.09258v1.
Stiny, George, and James Gips. 1971. "Shape Grammars and the Generative Specification of Painting and Sculpture." In IFIP Congress (2), vol. 2, no. 3.
Terzidis, Kostas. 2006. Algorithmic Architecture. Routledge.
Terzidis, Kostas. 2009. Algorithms for Visual Design Using the Processing Language. Wiley. John Wiley & Sons.
Tovey, Mike. 2015. Design pedagogy: Developments in art and design education. Ashgate Publishing, Ltd.
Ware, Colin. 2008. Visual Thinking for Design. Morgan Kaufmann.
Weisberg, Robert W, and Lauretta M Reeves. 2013. Cognition: From Memory to Creativity. John Wiley & Sons.
Wing, Jeannette M. 2006. ‘Computational Thinking’. Magazine Communications of the ACM 49 (3): 33–35. doi:10.1145/1118178.1118215.
Wing, Jeannette M. 2008. ‘Computational Thinking and Thinking about Computing’. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 366 (1881). The Royal Society: 3717–3725. doi:10.1098/rsta.2008.0118.
Wing, Jeannette M. 2010. Computational thinking: What and why? Retrieved from http://www.cs.cmu.edu/~CompThink/ resources/TheLinkWing.pdf.
Wing, Jeannette M. 2017. ‘Computational Thinking’s Influence on Research and Education for All Influenza Del PensieroComputazionale Nella Ricerca e Nell’educazione per Tutti’. Italian Journal of Educational Technology 25 (2): 7–14. doi:10.17471/2499-4324/922.