Design thinking and computational thinking: An ontology for metacognitive approaches to problems

Nick Kelly and John S. Gero

Introduction

The term *design thinking* is widely discussed in design literature, and has its foundations in research on how designers know, act, and think.[1] A second form of 'thinking', *computational thinking* has not to date been theorised in relation to design thinking, where computational thinking has its foundations in research on how computer scientists know, act, and think.[2] Each form of 'thinking' has found widespread popularity, as indicated by the international adoption of both terms within formal education systems,[3] and, in the case of design thinking, within business and government.[4]

The relationship between design and computational thinking lacks clarity. Neither term has an accepted definition that provides clear boundaries around the parts of human cognition that could be labelled as either design thinking or computational thinking. As a result, even basic questions about this relationship have no established answers: are design thinking and computational thinking discrete, orthogonal, or overlapping notions? Consider, for example, the case of a software engineer trying to understand a client's needs. Such activity might readily be understood through the lens of either design thinking or computational thinking. In another example of the blurred boundaries between these terms, Jeanette Wing has suggested that computational thinking is "a creative process" that "relies on human ingenuity, flashes of insight and taste in design".[5] These cognitive abilities are typically taken to be markers of design thinking, yet they equally seem to be applicable to some definitions of computational thinking.

This paper aims to compare and contrast these two forms of thinking to provide some clarity about the relationship between them. The significance of the work is to position design

thinking in relation to another form of thinking that seems, intuitively, to be counterposed to it, contributing to the theoretical foundations of design thinking. The work is *ontological* in that it aims to specify representational terms that describe a domain[6]. The paper highlights the significance of *framing* for both design thinking and computational thinking and contrasts the class of outcomes that each type of thinking appears to produce. This leads to a proposal for two orthogonal variables that frame metacognitive approaches to problems. In the discussion section we suggest that this space has relevance for how design thinking and computational thinking are taught, both inside and outside of formal education. In the rest of the paper, we will refer to the person engaged in either design thinking or computational thinking as *the thinker*, for consistency. Throughout, we will refer to 'problems' and 'solutions' that are arrived at through design thinking and computational thinking; again, this is done for consistency, while recognising that much design activity does not have clearly defined problems and that design solutions do not necessarily 'solve' problems but address them in a designerly way.

Design thinking

Design thinking has become an overloaded and ambiguous term but has its roots in the scientific study of design cognition and design methods. Design thinking is the knowledge that has been developed relating to how designers reason,[7] also described as "designerly ways of knowing, thinking, and acting".[8] Disambiguation is needed, because there are currently multiple different understandings of this widely used and overloaded term, as described in two papers reviewing the use of the term. Kimbell identifies three definitions of design thinking in the literature, where the one that we intend here is 'design thinking as a cognitive style that is observed in designers'.[9] Johansson-Sköldberg et al. identify five definitions, where the one that we intend here is 'design thinking as a way of reasoning/making sense of things that is used by designers'.[10] Design thinking is often widely used in the public sphere

as shorthand for the suite of concepts and tools that are well-suited for teaching non-designers how to approach complex problems in a designerly way.[11] This is not what we refer to here.

The core of design thinking is an understanding that *a designer creates the frame within which design activity is undertaken*.[12] This idea of a frame has multiple origins. One can be found in *situated cognition* (where *frame* is referred to as the *situation*), in the empirically-supported argument that knowledge (e.g., concepts) cannot be abstracted from the situation (or frame) within which it is used and learned.[13] Another can be found in Minsky's suggestion that *frames* can be used as a way to specify the top-level of organisation in a system for structuring knowledge in the context of artificial intelligence.[14]

The common thread and relevance for design is that while a designer might 'know' a great deal about the world, that knowledge is not stored in discrete, abstract chunks waiting to be 'deployed' during design activity. Rather, when faced with a design problem, a designer cognitively, but unconsciously, constructs a complex assemblage of interrelated knowledge—what we will refer to as the frame. This frame forms the lens through which the object of attention—in this case, the design problem—comes to be understood and within which design actions that might be taken are available.

Many insights into designerly ways of acting and knowing relate to the way that designers frame problems and the strategies that they have for acting such that the frame changes in a desirable way. One such insight is that designers typically have a conception of the understanding of the problem, a problem space, and a conception of possible solutions, a solution space, that both form a part of the frame. Designers, when observed, appear to be co-evolving these two different spaces, changing the understanding of possible designs and possible solutions in parallel and in an interdependent way.[15] Another frame-based design phenomenon, identified by Suwa, Gero, and Purcell, is that designers make unexpected discoveries within their own external representations (e.g., sketches) that change the

trajectory of the design process.[16] These insights fit with the description of design as something that occurs "within a context which depends on the designer's perception of the context".[17] This has broad implications; for example, that the same design problem approached by the same designer at two different times might be conceived in entirely different ways.[18]

Problems that are well suited to design thinking are problems in which frames that include a useful solution are not immediately available to a thinker—the solution space needs to be evolved in some way for a solution to become apparent. *Wicked problems* are a good example of the type of problem in which design thinking is needed, in which variables are unknown and the knowledge needed to address the problem is incomplete, such as the kinds of problems encountered in social planning.[19] A problem in which all of the variables are known at the outset (e.g., solving a tangram puzzle) is not a good candidate for design thinking. Skill in addressing this type of problem requires a capability for working with frames in a particular way. Expert designers have metacognitive skills that enable them to observe the frame that they have created for the design problem and, critically, have access within this frame to conceive of appropriate actions that might expand their understanding of the problem in a useful direction.

Many renowned examples of outstanding design—examples like Jørn Utzon's *Sydney Opera House*, Frank Lloyd Wright's *Fallingwater*—can be recast as tales of exceptional capability in framing and bringing things into or out of the frame. The Sydney Opera House was designed through a competition; where other entrants heeded the rules in their designs (keeping them in the frame), Jørn Utzon broke them in service to his vision for a design. Frank Lloyd Wright introduced many novel ideas in his design of Fallingwater, driven, in part, by his inclusion in the frame the idea that the design of any house should enhance the

landscape within which it is situated—leading him to challenge many of the requests of the client including even the location of the house.

In schools of design around the world, whether they are studying architecture, industrial design, engineering, fashion design, web design, etc., students are taught how to think about the *user* of their design, how to think about the *context/site/situation*, and how to do *research* about the design. All of these can be considered useful guides for actions to take when confronted with a novel design problem; they all also denote strategies that will implicitly change the frame in a direction that is useful for moving towards a design solution.

The outcome from the design process is design documentation, some kind of communicable representation of the design solution. A notable feature of design solutions is that they tend to be specific to the problem that they were created for—they cannot (typically) be taken and applied directly to other users or other design scenarios. For example, the architectural design of a house needs to be specific to the site where it is located—aspect, topography, landscape, surroundings, history—as well as its inhabitants—the specific needs of the people who will be living in it—to be considered an example of good design. The reuse of the design of a house from one design situation to another in cookie cutter fashion generally results in poor design outcomes. This tendency for design solutions to be specific to a design problem is common across different design disciplines.

Computational thinking

The term *computational thinking* has its origins in the recognition that computer science is a vital part of innovation and discovery in solving human problems in the modern world; and that there is thus a broad need for people in society to have the cognitive capabilities to do computer science.[20] Through computer science (and the engineering that has enabled technological progress) humans have been able to solve problems in such a way that, once solved, they can be replicated at scale and at very low cost. As a result, the fruits of

computational thinking now underpin much of modern life. Because of its importance to society and to economies,[21] there has been widespread uptake of computational thinking in educational systems worldwide. Countries such as Russia, South Africa, New Zealand, and Australia have already brought computational thinking into the K–12 curriculum, and there has been a move towards making computational thinking a part of compulsory education in many nations.[22]

There is a pattern for review articles about computational thinking to commence by noting its importance, its widespread uptake within education, but also its lack of robust definition.[23] The definitional confusion around computational thinking is summarised by Shute et al. who suggest that "[computational thinking] definitions vary in their operationalization of [computational thinking] in certain studies, and are not particularly generalizable".[24] There are two trends in attempts to define computational thinking. The first trend defines computational thinking based upon the types of reasoning that are used. An example of this is Wing's initial work in suggesting that "computational thinking involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science" (p. 33).[25] The second trend defines computational thinking based upon the types of solutions that it produces. Many papers in the literature refer to a definition developed by Wing and colleagues as "the thought processes 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"; where the idea is that an information-processing agent can be either human or computational.[26] Here we will first consider the cognitive markers of computational thinking, and then discuss the types of solutions that it produces. The term computational thinking is not used to suggest that humans reason in ways that are similar to computers (e.g., theories subscribing to

computationalism[27]); it is shorthand for referring to the problem-solving approaches that computer scientists make use of.

The primary cognitive ability required for computational thinking is abstraction and the competencies that support abstraction. Abstraction is concerned with "defining patterns, generalizing from specific instances,"[28] and the "value of abstraction as [computational thinking]'s keystone (distinguishing it from other types of thinking) is undisputed".[29] Abstraction is the type of reasoning that involves moving from specific instances to general patterns, keeping relevant information and discarding irrelevant data. Through abstraction people "glean relevant information (and discard irrelevant data) from complex systems to generate patterns and find commonalities among different representations".[30] A cliché in teaching computational thinking is to take the process of making toast, break it down into individual steps, and specify an algorithm for how anybody could make toast. This is a useful activity, because it demonstrates an algorithm as a sequence of steps whilst simultaneously making it clear that it is useful to specify certain things in the algorithm (e.g., putting toast in the toaster) while unhelpful to specify others (e.g., how to co-ordinate the hand to reach inside a bag of sliced bread). One of the core skills in computational thinking is learning to find the right abstraction.

Problems that require computational thinking are typically highly structured; or rather, the way that the problem is framed requires that a well-structured solution (e.g., an algorithm) be a part of that frame. They are also typically recurrent problems, problems that either occur in many places, or recur within the same place. The value in taking the time to solve a problem in such a way that a computer can carry out the process is so that the solution can be deployed in other circumstances with similar problems. For example, businesses often invest significant resources in developing specific software that automates workflow, as they know that eventually the software will pay for itself in time saved or in a competitive

advantage. Software developers can create a solution to a very specific problem in their immediate circumstance and then suddenly find that, by solving this problem better than anyone else, they have a global market of people interested in it.

Computational thinking requires thinking about problems in a way that enables solutions to be found—often, but not always, using computers—that apply to many other similar problems and where the steps required to solve those problems can be represented and used in applicable circumstances. For example, the problem of wayfinding within a city can be solved by the development of navigational software combined with GPS enabled smartphones and data about city geography (e.g., Google maps). Solving the problem in one city enables the solution to be deployed in another city with only a change of the data being used—the algorithms and hardware can remain unchanged.

The solution to a computational thinking problem is typically a representation of a solution at the appropriate level of abstraction to allow the solution to be applied in other similar circumstances, as typified by an algorithm. In this respect the solutions provided by computational thinking aim to be generally applicable. However, they also tend to have clearly stated conditions on applicability—such as the kind of variables that a function can accept—and solutions are transferable and repeatable to other situations in which these conditions apply.

Design thinking and computational thinking

Despite the popularity of both terms, there is little in the literature that compares design thinking and computational thinking to consider the relationship between the two—perhaps due to the lack of a consensus on definition for both terms. One exception is a brief comparison by Shute et al. who suggest that the main difference lies in the domains where each type of thinking operates, but do not discuss differences in the types of thinking utilised.[31] Additionally, there is theory within the design literature that recognises the nature

of design problems as being inherently embedded within complex systems (e.g., technical, political, economic, ethical, etc.), where the embedded systemic way of 'seeing the whole' can be juxtaposed with the need for designers to, at times, see component parts and the relationships between them.[32] This view is taken up in the Discussion section of this paper, in suggesting that design thinking and computational thinking can be considered archetypes within a spectrum of metacognitive approaches to problem solving.

There are clearly activities that are almost entirely focused on design thinking—say, the engineering design of a machine—with little or no computational thinking involved; and the inverse also, tasks such as sorting a list which require computational thinking and little or no design thinking. There are also tasks that appear to involve both computational and design thinking, such as a web designer responding to a client's brief. In all three tasks there is a thinker, a problem to be addressed, and a solution. On what basis can these three tasks be compared?

We identify two variables that can form the basis for differentiation between design and computational thinking as: (1) the generality/specificity of solutions; and (2) the generality/specificity of the frame.

Specificity of solutions

A synthesis of the descriptions of design thinking and computational thinking suggests that solutions from each type of thinking seem to fall at opposite ends of a continuum. A typical design solution is highly specific to the design problem that the thinker is addressing—specific to the users, the site, the context, etc. It is rare that a design solution from a prior problem can be directly transferred into a new problem without further design thinking being done to adapt it. In contrast to this, most computational thinking solutions can be applied to new problems without any further computational thinking being done—for some scholars,[26]

this is a definitional quality of computational thinking.

This leads to the proposal that one variable that explains the difference between design and computational thinking is the *specificity of the solution in relation to the problem to which it pertains*. In design thinking solutions tend to be specific to the problem. In computational thinking the solution tends to be more general than the problem that it was created to solve. Figure 1 depicts this as an ontological category with values ranging from specific to general.

Figure 1 The specificity of the solution in relation to the problem as an ontological category with values ranging from specific to general

Specificity of framing

A second variable that can be used to explain much of the difference between design thinking and computational thinking relates to the way that the thinker frames the problem. In design thinking, many of the activities—doing user research, playing with materials, researching theory and cases, etc.—are all oriented towards an expansion of the frame. For example, it is usually an indicator of good design if parts of culture relating to the design are given consideration by the thinker and are brought into the frame. In contrast, computational thinking involves abstraction—capturing what is the core relationship between information and processes and abstracting away what can be removed. To continue the previous example, in the context of computational thinking, culture tends to be abstracted away.

This leads to the proposal that a second ontological category that is used to explains the difference between design thinking and computational thinking is the *specificity of the*

frame in relation to the problem to which it pertains. Figure 2 depicts this as an ontological category with values ranging from specific to general.

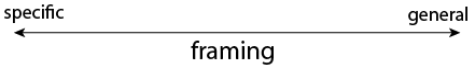

Figure 2 The specificity of the framing in relation to the problem as an ontological category with values ranging from specific to general

An ontology for reasoning about problems

These two categories, of solution specificity and frame specificity, are independent of one another. Given that these categories are orthogonal, a space can be created, which we propose is a useful ontology for how people reason about problems using design thinking or computational thinking, Figure 3. It is an ontology in that it specifies representational terms—the two axes—that are useful for specifying a domain of the ways in which humans reason about problems.

In the space created by these two axes, the upper left-hand quadrant is a good match with the characterisation of design thinking that is provided in the literature. Here the thinker is aiming for a very specific solution and is getting there by trying to gain a broad understanding of the problem. In contrast, the lower right-hand quadrant is a good fit with the characterisation of computational thinking. The thinker is producing general solutions and is getting there by honing a precise understanding of the problem in a form that has the most helpful abstraction possible.

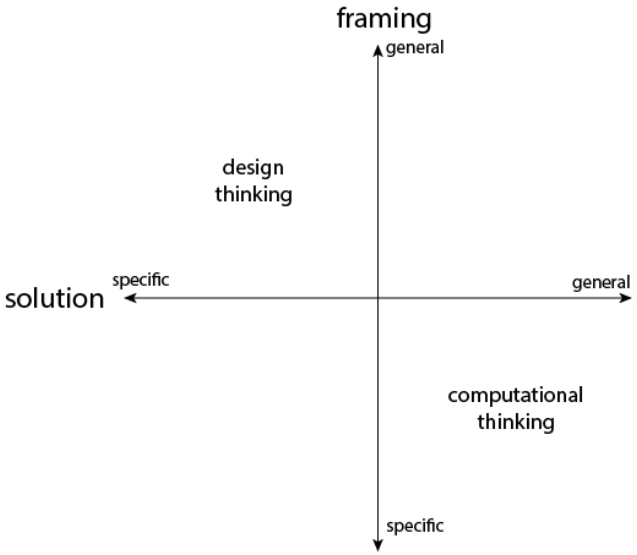

Figure 3 Space created by graphing the two orthogonal ontological categories, with design thinking and computational thinking located in the space.

Discussion and conclusions

Significance of the ontology for design thinking

The ontology identifies design thinking and computational thinking as 'archetypal' descriptions of approaches to problems a larger domain of such approaches. It is productive in the sense that there are two spaces in Figure 3 that do not map uniquely onto either design or computational thinking, and certain notions within the literature can be mapped onto these spaces. For example, the upper right quadrant is an area that suggests both general framing and general solutions. This maps well with the kind of thinking that can be seen in the creation of *design patterns*, where a design pattern is a way of capturing the essence of a design solution at a level of abstraction that allows the thinking behind it to be reused:

> Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.[33]

This is an attempt to pose enough information about a design solution such that it can be transferred to other situations; while also meeting the demands of the particular problem.

Similarly, there is a space in the lower left quadrant that denotes an orientation towards including less in the frame and of having solution that are not expected to be general. We suggest that an example of this kind of thinking is a good fit with that observed by designers engaged in *parametric design* approaches, such as is widely used in architectural design and in engineering design.[34] Here the aim is to create a specific abstraction of the problem—and thus an orientation of having less in the frame—but with the intention of using its output to address a very specific problem.

Understanding how thinkers think

This paper compares and contrasts the relationship between design thinking and computational thinking. In the proposed ontology these two forms of thinking represent two spaces within a larger space. This raises further questions about the remaining spaces and about how humans reason in response to problems. Does the preponderance of design thinking and computational thinking suggest that these two forms of thinking are more useful for humans than the two spaces that are not clearly defined? Or is it an invitation to give a clearer characterisation of the two remaining spaces?

There is also a lack of understanding about how humans move around within this space during their thinking. One way of talking about this movement between different types of thinking has been proposed by Goodyear and Markauskaite as *epistemic fluency*.[35] Epistemic fluency requires "being flexible and adept with respect to different ways of knowing about the world" (p. 1) and the term can be used to describe the need for a problem solver to have capacity to both 'think like a designer' and 'think like a computer scientist' and to know how to move between the two. Students need to develop the metacognitive

capacity for multiple ways of thinking alongside an awareness of which of these strategies for thought are most useful at a particular time in responding to a situation.

Implications for formal education

In recent decades there has been a revision of Bloom's taxonomy,[36] to include a second dimension for cognitive processes. In this revised taxonomy the highest educational objective is to create "Putting elements together to form a novel, coherent whole or make an original product".[37] The emphasis on cognitive processes within the context of learner expectations is also reflected in Webb's Depth-of-Knowledge schema where Level 4, extended thinking, correlates with Bloom's two highest levels.[38] These educational objectives target students' development of metacognitive skills, and computational thinking and design thinking both respond to these educational objectives. Given the suggested ontology, and in the context of epistemic fluency, it may be appropriate to teach these two forms of thinking as complementary ways of approaching problems—as opposed to the current status quo, in which they are taught and discussed largely in isolation from one another.

Conclusions

This paper has positioned design thinking in relation to computational thinking, and in doing so contributed to the theoretical foundations of design thinking. The proposed ontology places design and computational thinking in relation to each other as regions within a space of 'metacognitive approaches to problems' with axes of specificity of framing and specificity of solutions. There are spaces in the top-right and bottom-left of Figure 3 that are not yet well described; the presence of these spaces leads to further questions that have been discussed, and the responses to which lie beyond the scope of this paper. The paper raises new questions about the ways in which persons engaged in design activity move around within this metacognitive space. For example: What are the expectations of how different professionals

(e.g., designers, computer scientists) might move within this space when addressing problems? What are the implications for how design thinking and computational thinking are taught within formal education, at all levels?

Relevant to the aim of this paper is the space *between* computational and design thinking. This space between each of the labels of design thinking and computational thinking in Figure 3, close to the centre, is a region within which we can propose that design thinking might include computational thinking in some situations, and vice versa in others. We suggest that design thinking and computational thinking are not mutually exclusive—as might be implied by the lack of literature addressing the relationship between them—but rather are mirror images of each other in relation to the two ontological categories of solutions and framing.

1 E.g., Nigel Cross. *Design thinking: Understanding how designers think and work* (Oxford: Berg, 2011).
Kees Dorst, "The Core of 'Design Thinking'and Its Application," *Design Studies 32* no. 6 (2011), 521-532.

2 Valerie Shute, Chen Sun, and Jodi Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review 22* (2017), 142-158.
Jeanette M. Wing, "Computational thinking," *Communications of the ACM 49* no. 3 (2006), 33-35.

3 Shuchi Grover and Roy Pea. "Computational thinking in K–12: A review of the state of the field," *Educational Researcher 42* no. 1 (2013), 38-43.
Joyce Hwee Ling Koh, Ching Sing Chai, Benjamin Wong, and Huang-Yao Hong. *Design thinking for education: Conceptions and applications in teaching and learning* (Berlin: Springer, 2015).

4 Tim Brown. "Design thinking," *Harvard Business Review 86* no. 6 (2008), 84-94.

5 Jeanette M. Wing. "A Conversation about Computational Thinking," *Education: Future Frontiers* (2019), 5.

6 Thomas R. Gruber. "A translation approach to portable ontology specifications," *Knowledge Acquisition 5* no. 2 (1993), 199-220.

[7] E.g.,.Nigel Cross. *Design thinking: Understanding how designers think and work* (Oxford: Berg, 2011).
Kees Dorst, "The Core of 'Design Thinking'and Its Application," *Design Studies 32* no. 6 (2011), 521-532.
Bryan Lawson. *How designers think: The design process demystified 4th edn.* (New York, NY: Elsevier, 2006)

[8] Nigel Cross. "Designerly ways of knowing: design discipline versus design science," *Design Issues* 17 no. 3 (2001), 50.

[9] Lucy Kimbell. "Rethinking design thinking: Part I," *Design and Culture 3* no. *3* (2011), 285-306.

[10] Ulla Johansson-Sköldberg, Jill Woodilla, and Mehves Çetinkaya. "Design thinking: past, present and possible futures," *Creativity and Innovation Management 22* no. 2 (2013), 121-146.

[11] E.g., Shelley Goldman, Maureen P. Carroll, Zandile Kabayadondo, Leticia Cavagnaro, Adam Royalty, Bernard Roth, Swee Hong Kwek, and Jain Kim. "Assessing d. learning: Capturing the journey of becoming a design thinker," in *Design Thinking Research*, eds. Hasso Plattner, Cristoph Meinel, and Larry Leifer (Berlin: Springer, 2012), 13-33

[12] Kees Dorst. *Frame innovation: Create new thinking by design*, (Cambridge, MA: MIT Press, 2015).
John S. Gero. "Design prototypes: a knowledge representation schema for design," *AI magazine 11* no. 4 (1990), 26.
Donald A. Schön. *The reflective practitioner: How professionals think in action* (New York: Basic Books, 1983).

[13] William Clancey. *Situated Cognition*, (Cambridge: Cambridge University Press, 1997).
Lawrence W. Barsalou. "Grounded cognition," *Annual Review of Psychology 59* (2008), 617-645.
John S. Brown, Allan Collins, and Paul Duguid. "Situated cognition and the culture of learning," *Educational Researcher 18* no. 1 (1989), 32-42.

[14] Marvin Minsky. "A framework for representing knowledge," *MIT-AI Laboratory Memo 306* (1974).

[15] Kees Dorst and Nigel Cross. "Creativity in the design process: co-evolution of problem–solution," *Design Studies 22* no. 5 (2001), 425-437.
Josiah Poon, and Mary Lou Maher. "Co-evolution and emergence in design," *Artificial Intelligence in Engineering 11* no. 3 (2015), 319-327.

[16] Masaki Suwa, John S. Gero, and Terry Purcell. "Unexpected discoveries and S-invention of design requirements: important vehicles for a design process," *Design Studies 21* no. 6 (2000), 539-567.

[17] John S. Gero. “Design prototypes: a knowledge representation schema for design,” *AI magazine 11* no. 4 (1990), 26.

[18] John S. Gero. “Conceptual designing as a sequence of situated acts,” in *Artificial Intelligence in Structural Engineering*, ed. Ian Smith (Berlin: Springer, 1998), 165-177.
Nick Kelly and John S. Gero. “Situated interpretation in computational creativity,” *Knowledge-Based Systems 80* (2015), 48-57.

[19] Horst Rittel, and Melvin Webber. “Dilemmas in a general theory of planning,” *Policy sciences 4* no. 2 (1973), 155-169.

[20] Jeanette M. Wing. “Computational thinking and thinking about computing,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 366* no. 1881 (2008), 3717-3725.

[21] National Research Council. *Report of a workshop on the scope and nature of computational thinking* (Washington, DC: National Academies Press, 2010).

[22] Shuchi Grover, and Roy Pea. “Computational thinking in K–12: A review of the state of the field,” *Educational Researcher 42* no. 1. (2013), 38-43.
Joke Voogt, Petra Fisser, Jon Good, Punya Mishra, and Aman Yadav. “Computational thinking in compulsory education: Towards an agenda for research and practice,” *Education and Information Technologie*s *20* no. 4 (2015), 715-728.

[23] E.g., Karen Brennan and Mitchel Resnick. “New frameworks for studying and assessing the development of computational thinking,” *Proceedings of the 2012 annual meeting of the American Educational Research Association Vol. 1* (Vancouver, Canada, 2012), 25.
Valerie Shute, Chen Sun, and Jodi Asbell-Clarke, “Demystifying computational thinking,” *Educational Research Review 22* (2017), 142-158.
Shuchi Grover, and Roy Pea. “Computational thinking in K–12: A review of the state of the field,” *Educational Researcher 42* no. 1. (2013), 38-43

[24] Valerie Shute, Chen Sun, and Jodi Asbell-Clarke, “Demystifying computational thinking,” *Educational Research Review 22* (2017), 142-158.

[25] Jeanette M. Wing. “Computational thinking,” *Communications of the ACM 49* no. 3 (2006), 33-35.

[26] Jeanette Wing. “Research notebook: Computational thinking—What and why,” *The Link Magazine* (2011), 20-23

[27] Rescorla, M. “The computational theory of mind,” *The Stanford Encyclopedia of Philosophy (Spring 2020 Edition)*, Edward N. Zalta (ed.), retrieved from https://plato.stanford.edu/archives/spr2020/entries/computational-mind/ (2020)

[28] Shuchi Grover, and Roy Pea. “Computational thinking in K–12: A review of the state of the field,” *Educational Researcher 42* no. 1. (2013), 38-43.

[29] Ibid.

[30] Valerie Shute, Chen Sun, and Jodi Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review 22* (2017), 142-158.

[31] Ibid.

[32] Harold G. Nelson, and Erik Stolterman. *The Design Way: Intentional Change in an Unpredictable World* (MIT Press, 2012), 57-82

[33] Christopher Alexander. *A pattern language: towns, buildings, construction* (New York, NY: Oxford University Press, 1977).

[34] E.g., Robert Woodbury. *Elements of parametric design* (Oxford: Routledge, 2010).

[35] Lina Markauskaite, and Peter Goodyear. *Epistemic Fluency and Professional Education*. (Dordrecht: Springer, 2017).

[36] Benjamin Bloom. *Taxonomy of educational objectives Vol. 1: Cognitive domain* (New York, NY: McKay, 1956), 20-24.

[37] David R. Krathwohl. "A revision of Bloom's taxonomy: An overview," *Theory into practice 41* no. 4 (2002), 212-218.

[38] Norman Webb (1997). "Criteria for alignment of expectations and assessments on mathematics and science education," *NISE Research Monograph Number 6.* (Madison: University of Wisconsin–Madison, 1997).