Designing with generative systems: a creative-processes-oriented discussion

Projetando com sistemas generativos: uma discussão orientada a processos criativos

Caio Barrocal, FAU-USP
caio.barrocal@gmail.com

Clice de Toledo Sanjar Mazzilli, FAU-USP
clice@usp.br

Abstract

The goal of this paper is to provide the theoretical foundations for the study of the creative processes involved in designing with generative systems. To do so, this discussion is sustained by a two-layered literature review. First, creativity, generalist creative processes and creative processes in design are approached. Then, we resort to specific models and concepts of generative design, computational-mediated design processes and computation thinking. In the end, the contribution of this work takes shape as a synthesis of the main aspects learned about the process in question, such as the important role of computational thinking and the implications of a characteristic layer of abstraction that pervades it.

Keywords: creative processes, generative systems, autonomous systems, creativity, computational thinking.

Resumo

O objetivo deste artigo é fundamentar o estudo dos processos criativos envolvidos no design com sistemas generativos. Para tanto, esta discussão é sustentada por uma revisão bibliográfica em duas camadas. Primeiro, aborda-sea criatividade,os processos criativos generalistas e os processos criativos em design. Depois, recorre-se à modelos e conceitos específicos sobre design generativo, processos criativos mediados pela computação e pensamento computacional.Ao fim, a contribuição deste trabalho toma forma nasíntesedos principais aspectosapreendidos sobre o processo criativo estudado. Dentre eles, o importante papel do pensamento computacional e as implicações de uma característica camada de abstração que o perpassa.

Palavras-chave: *processos criativos, sistemas generativos, sistemas autônomos,criatividade, pensamento computacional.*

Introduction

The use of computers as means of creative production is a relatively recent phenomenon. The first relevant experiments were executed in the 1960s, mainly in Europe and the USA, when programmers and artists began exploring algorithms and machines to generate shapes and sounds. One of the first formal opportunities for establishing the computer as a creative setting amongst artists and designers happened when Max Bense (1910-1990) — an influential German philosopher and academic of the fields of aesthetics and semiotics — invited the mathematician Georg Nees (1926-2016) to expose his computer-generated graphics in an experimental concretist exhibition in 1964. In the following year, Nees and Bense published the booklet “rot 19. Computer-Grafik”, one of the primary publications that presented the computer as a means of creative production (NAKE, 2018).

The work with algorithms motivated several professionals to explore computing as a fruitful creative medium, extrapolating the uses delimited by available proprietary software. Whether due to the unpredictability of results, possibilities for parameterization and optimization, or the ability to support artifacts that respond in real time, several designers, programmers and artists have been relying on generative creation to deliver artifacts with instigating aesthetic and functional appealing. In design, for example, generative systems can be used in the production of flexible visual identities, objects and packaging, fonts and infographics, and interactive artifacts.

Formally, generative creation consists of the design of systems or processes, which are put into execution with a certain degree of autonomy contributing to or resulting in a complete work (GROß et al., 2018; GRÜNBERGER, 2019; GALANTER, 2003). Such autonomy can be guaranteed in several ways: through the writing of instructions that respond to randomness; by designing an interface that has an expected general behavior but delivers a different experience depending on unpredictable inputs captured by the computer; or even through complex mathematical models such as the capital market and the weather.

The critical point is that computational intelligence will be used as an active participant in the creative process and not only to support the decisions made (MOUNTSTEPHENS;TEO, 2020). In design, the now added computational complexity promotes a fundamental change in the creative process as designers are no longer executors of tasks, but conductors. Role that Groß et al. (2018, p.5) consider to be that of an “orchestrator of decision-making processes”. Essentially, this type of creation gives up total control, which is partially conducted by the computational intelligence of the chosen system.

Motivated by a vibrant production and intrigued by the proposed fundamental changes on designers’ work, this paper intends to build a theoretical foundation to contribute to the analysis of the creative processes in question. As an attempt to delimit and illustrate its scope, the following goal has been pursued: **To present relevant aspects of the creative processes undertaken by professionals when designing with generative systems**.

To do so, this paper is structured in two topics:in the first one — creative processes in design,this discussion is appropriately framed as an investigation of creative processes in design. Accordingly,definitions and models regarding the creative activity and its specificities when in a design process are presented.Then,in the topic ongenerative creation and computational thinking in design,we draw a conceptual parallel between designand generative creation. Pertinent definitions regarding generative designare presented, and its differences are introduced compared to other design approaches. We also discuss the possible intersection between computational thinking and design thinking.

Creative Processes in Design

The first modern studies about creativity were executed during the 1950s and 1960s, as leaders began recognizing it as an essential element for economic success and the solution of complex social problems. In the beginning, influenced by a predominant behaviorist tradition, many researchers studied the lives and practices of artists, scientists, and designers socially recognized as exceptional creators, in order to identify general aspects and traits that could somehow explain the conditions and paths through which a person should go in order to *be* creative (SAWYER, 2012, p. 63).

Many traits were indeed identified as related to a creative personality but none that could determine it. It is true that certain types of personalities are more inclined towards creativity, however this inclination seems to be the result of a very complex mixture of different traits(SAWYER, 2012, p. 63). Accordingly, the rise of cognitive psychology changed the focus of creativity research from studying personality traits to understanding mental processes shared by all individuals in their act of creation. These “representational structures of the mind, their interconnections, and the mental processes that transform them”, as referred to by Sawyer (2012, p. 87), became the center of research in the field and also the foundations of a definition for the creative process.

As stated by Lubart (2018, p. 3), “the creative process can be defined as a sequence of thoughts and actions that comprise the production of work that is original and valuable”. ‘Sequence’ meaning the chain of events that “unrolls over time, with a beginning and, potentially, an end”, and ‘thoughts and actions’ representing “both internal and external operations that contribute to the emerging production”. Such a statement implies that the many methods and tasks involved indesigning are also an expression of a creative process.

One of the first models for the creative process was proposed by psychologist Graham Wallas in 1926, as he aimed to explain creativity through four sequential steps: **preparation, incubation, insight, and verification** (SAWYER, 2012, p. 89). However, as attention to intelligence and problem solving in the mid twentieth century grew, researchers of the field seeked to go beyond the “popular four-stage description of the creative process” (LUBART, 2018, p. 7), which resulted in valuable contributions about the cognitive operations and problem-solving skills behind creative activity.

By examining many past contributions of theories and models to creativity, Sawyer (2012, p. 88-89) created the following “integrated framework” that describes the eight stages of the creative process. Like Lubart (2018, p. 9) — that suggests that “the creative process is an orchestrated symphony of more specific processes that play together, or in sequence as part of the whole" —, Sawyer (2012, p. 88-89) agrees that despite being represented as a linear process with specific steps, there is a consensus in the field that creativity is not resulted by a single, unitary mental process, but from many different ones that could be associated with the following stages:

1. **Find and Formulate the Problem**
2. **Acquire knowledge relevant to the problem**
3. **Gather a broad range of potentially related information.**
4. **Take time off for incubation.**
5. **Generate a large variety of ideas.**
6. **Combine ideas in unexpected ways.**
7. **Select the best ideas, applying relevant criteria.**
8. **Externalize ideas using materials and representations.**

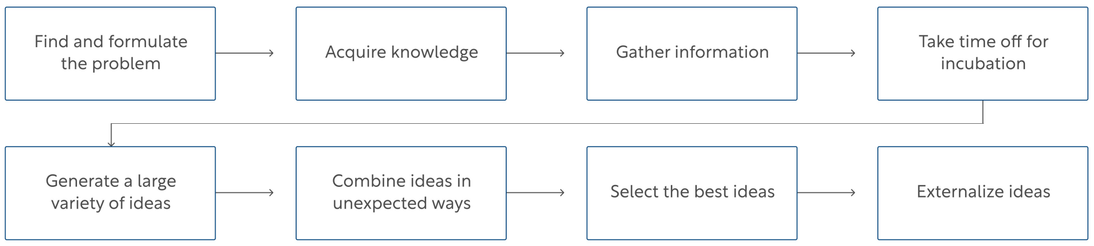
Find and formulate the problem
Acquire knowledge
Gather information
Take time off for incubation
Generate a large variety of ideas
Combine ideas in unexpected ways
Select the best ideas
Externalize ideas

Figure 1: Illustration of Sawyer's model (2012). Created by the author.

Because creativity is "at the heart of design, at all stages throughout the design process" (BAXTER, 1995, p.61) and because designers are individuals, studies and models about it might contribute to a much greater understanding of the processes undertaken by those tackling design problems (LUBART, 2018, p. 13). But is problem-solving in design different from any other problem-solving at all?

We can understand design challenges as problems that "are both goal-oriented and constrained and which depend upon a designer's perception of the context of the problem" (KELLY; GERO, 2021, p. 3). A statement that allows us to distinguish these specific problems from other ones, say, those that may not be goal-oriented and constrained — such as some artistic activities involving self-expression, and those in which all variables are known beforehand, such as some math puzzles "in which the designer's perception of the context is irrelevant to the solution". Lawson and Dorst (2009, p.28), for example, refer to design as "a mixture of creativity and analysis" because despite having to creatively develop propositions, a designer's creativity "is not unrestricted". The project must achieve a certain set of goals, create value for both prospective users and the client, and so on.

Cross (2006, p.99) argues that design is also different from conventional problem-solving approaches due to the specific ways in which problems are framed and solutions are generated throughout a project. Such idiosyncratic "knowledge that has been developed relating to how people reason when engaging with design problems" as referred to by Kelly and Gero (2021, p. 2) is what is called design thinking.

When framing a problem, a designer cognitively, yet unconsciously, builds a complex aggregation of interrelated knowledge (the frame) that becomes the lens through which the design problem is understood and within which design actions are envisioned (KELLY; GERO, 2021, p. 3). In other words, when faced with a design problem, designers appear to constantly build and rebuild the frame as they are looking for the best ways to organize relevant aspects of both the problem space and the solution space. According to Cross (2006, p. 102), when framing, "designers select features of the problem space to which they choose to attend and identify areas of the solution space in which they chose to explore". Because this frame is constantly changing, and since "the problem cannot be fully understood in isolation from consideration of the solution", designers tend to coevolve these two spaces by using solution conjectures as the means to increase their understanding of the problem until a satisfactory result is produced.Framing as a characterizer of design thinking explains:

- Why wicked problems are good candidates for this type of thinking;
- Why designers benefit from appropriately researching users, the context, and similar design solutions — strategies that will contribute to the elaboration of the frame;
- And why design solutions tend to be very specific to the design problem tackled (KELLY; GERO, 2021, p. 3).

To capture the idiosyncrasies discussed so farand provide a comprehensive — yet generalist — model for creativity in design, many researchers relied on studies about creativity and reflected on designers' practices to propose models for the design process.Lawson and Dorst (2009, p.50), for example, proposed that a successful design endeavorinvolves skillsrelated to the five categories of practical and theoretical initiatives below:

- **Formulate**(identify and frame the problem)**;**
- **Represent**(externalize ideas, thoughts, and concepts)**;**
- **Move**(make design propositions);
- **Evaluate**(solution alternatives)**;** and
- **Manage**(the process).

Naturally, depending on the nature of the product being designed and theprocedure behind it, such categories will be expressed differently.The same applies for the methods and tools employed.

As a pertinent example, many researchers interested inthe implications of computers in human creativity have been investigatinghowthesemachines change the design process when taken as co-creators.In this regard, contributions of the young fieldofhuman-computer co-creativityseek to undestand the many ways this partnership can happen. One of them — and also the one we are most interested in — being the computer as a collaborator, generator and colleague(KANTOSALO, 2019). The following topic takes advantage of these ideas.

Generative creation and computational thinking in design

In 1964, Karl Gerstner (1930-2017) released his book Designing Programmes, one of the first texts dedicated to systematically approaching design. At that time, the author argued that problems could be more efficiently solved with a matrix of possible solutions — programs, instead of unique propositions. Essentially, Gerstner (1964, p. 21) considered impossible for designers to delimit, in an absolute way, all the relevant solutions of a design problem since conditions and the context might change fast and often. Still at the beginning of the computational era, Max Bense (1910-1990) also relied on research about computing and systematization as he was looking for ways to formalize aesthetic processes. Exploring a hypothetical concept of calculability of all elements, he worked centrally on the role of quantitative and objective analysis in the creations of shapes and on a design methodology strongly influenced by algorithms (NEVES, 2015, p.533).

As computers became more accessible during the XX century, computationalpractices reached many industries and society's more popular layers. For designers, even though the adoption of these tools caused differences in the design process, it first represented a transposition of the usual media and routines to a digital context. The focus was on gaining precision and efficiency and not exploring new processual and aesthetic possibilities (REAS;MCWILLIAMS, 2010, p. 27). Behind the scenes, though, algorithmic explorations continued to advance due to scientists, designers, and artists who sought to extrapolate the most famous software tools and take computing as a fruitful creative material.

In this context, computational creation transcended a purely automating algorithmic approach and reached another, in which systems started to be employed in the production of artifacts and experiences beyond traditional means. In other words, with its unique characteristics, the computer came to be seen as a creative medium in its own right and used to explore novelties (REAS;MCWILLIAMS, 2010, p. 31; RICHARDSON, 2016, p. 16).

Furthermore, design has been relying on generative creation not only to explore aesthetic possibilities but also to rethink the conventional process. Since the early 1970s, when generative design began to be studied by pioneers such as architect John Frazer, alike approaches have attracted the attention of many academic researchers interested in design theory and of the CAD(Computer-Aided Design)industry— interested in selling parametric and optimization solutions (KRISH, 2010, p. 90).

Traditionally, a design project depends on the central role of designers (or the team) to execute it from its initial stages to the finalization of the proposal. Constraints are continually and directly articulated in order to be contemplated by the designed product — a type of creative process that relegates the computer to the role of executing human decisions. In generative design, artifacts are produced by algorithms or autonomous systems, albeit partially, which can be fully automated or refined through interventions made by human designers. The critical point is that computational intelligence is used as an active participant in the creative process and not only to support previous choices (MOUNTSTEPHENS; TEO, 2020, p.1). Nevertheless, due to the added computational complexity, there is a fundamental change in the creative process: the designer is no longer an executor of tasks but a conductor. A role that Groß et al. (2018) refer to as an "orchestrator of decision-making processes". According to the authors, this is what generative design is about: "iteratively developing different processes and then selecting those that produce the most […] compelling results".

To illustrate this fundamental change, Groß et al. (2018) proposed a model for the creative process in generative design characterized by an emphasis on abstraction and on a different role for designers (Figure 2). The main change is that traditional craft recedes to the background, and abstraction and information become the protagonists. Thus, the relevant question is no longer ‘How do I draw?’, but ‘How do I abstract?’ (GROßet al., 2018, p. 504).

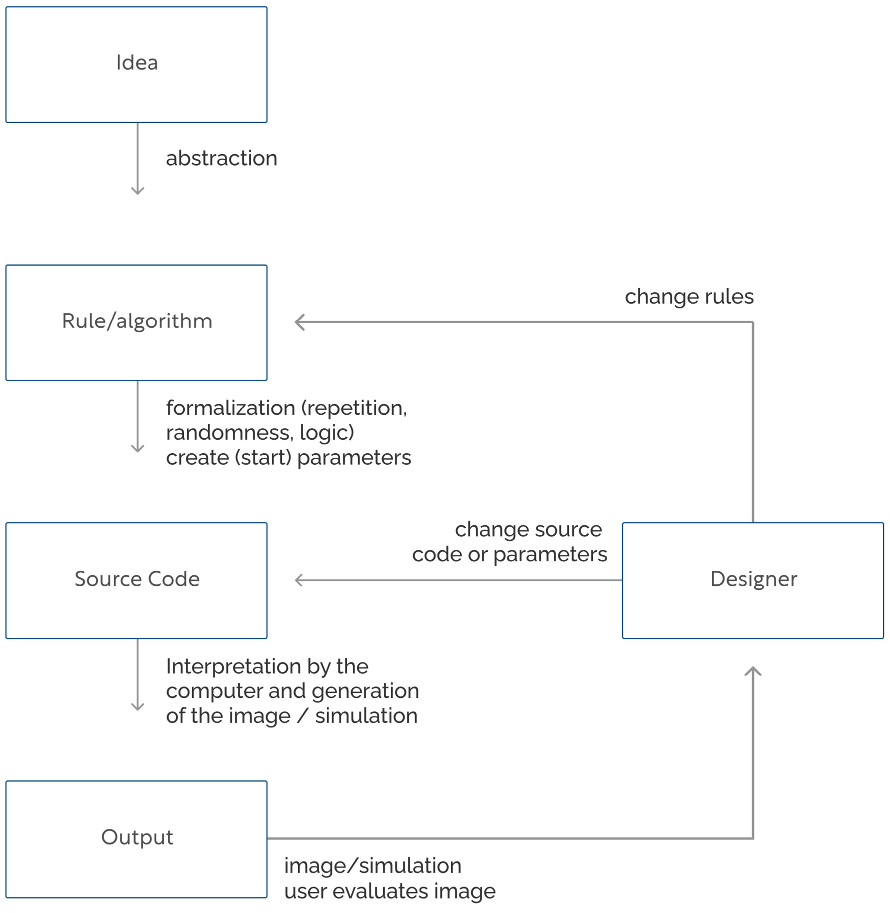
Idea
abstraction
Rule/algorithm
change rules
formalization (repetition, randomness, logic) create (start) parameters
Source Code
change source code or parameters
Designer
Interpretation by the computer and generation of the image / simulation
Output
image/simulation user evaluates image

Figure 2: Illustration of the model proposed by Groß et al. (2018). Created by the author.

In opposition to the conventional process in which designers implement an idea directly, when generative creation is employed, a process of abstraction is undertaken to **transform the idea into an algorithm** which is **translated into a source code**, or program, to **run and generate results**. Being an "indirect" design process, as referred to by Omine (2014, p. 85), designers participate through activities that encompass the planning and creation of algorithms, writing of code, the evaluation of results, and the consequent refinement of programs.As this is an unpredictable way to create, Groß et al. (2018, p. 507) also highlight the role of human designers in evaluating results and in the refinement of algorithms as the basis for an incremental and necessary improvement.

Besides being a central aspect of designing with generative systems, abstraction is also vital to computational thinking (or CT), a concept that will be explored as a valuable additional theoretical layer to this work.

Although there is no single definition for what this specific type of thinking is (SHUTE et al., 2017), informally, computational thinking describes the mental activities involved in formulating a problem in such a way that it admits computational solutions. Solutions that then can be performed by humans, machines, or both.

As an attempt to more formally define it, Kelly and Gero (2021, p.6) point out the existence of two distinct trends. In the first, authors seek to define computational thinking based on the type of reasoning involved in it. For instance, Furber (2012) defines it as the method of recognizing computational aspects in the world around us and applying computer science tools and techniques to understand and argue about natural or artificial systems and processes.

The second trend defines computational thinking through the solutions it can produce. Yadav et al. (2014) describe the concept simply as a mental activity used to abstract problems and formulate solutions that can be automated. Wing (2011), in a second more recent work, defines computational thinking as the "processes involved in the formulation of problems and solutions so that these solutions can be represented and, consequently, carried out by information processing agents". Moreover, focusing on the elements and abilities that compose it, Bocconi et al. (2016, p. 18) define computational thinking as the "mental processes involved in solving problems so that they admit a computational solution that involves **abstraction, algorithmic thinking, automation, decomposition, debugging and generalization**", which they understand as the core skills or abilities of computational thinking. Finally, going partially in line with this proposition, Beecher (2017, p. 7) adds the concepts of **logical thinking, modeling, and evaluation** to the list.

The most crucial cognitive skill in computational thinking is abstraction (KELLY; GERO, 2021, p.6; BOCCONI et al., 2016, p. 17-18). According to Grover and Pea (2013, p. 39), its importance as a key concept of computational thinking and as the element that differentiates it from other types of thinking is undisputed.

In essence, by abstracting, people express an idea in a specific context while suppressing details irrelevant to that context (BEECHER, 2016, p. 56). In this way, it is possible to say that the ability to abstract is related to the act of choosing the correct details to be removed from a problem so that it can be better understood or represented (CSIZMADIA et al., 2015, p. 7). By placing 'How do I abstract?' as the central question of the creative process in generative design, for example, Groß et al. (2018, p. 507) illustrate the existence of an abstraction layer that thoroughly mediates it. For them, one of the biggest challenges for designers in this approach is actually abstracting vague ideas into formal computer-interpretable instructions.

In any case, when heading towards formalization, the abstracted idea needs to be transformed into an algorithm that, in turn, can be understood as “a set of steps to accomplish a task.” (CORMEN, 2013, p. 1) or, more rigorously, as “a finite, deterministic, and effective problem-solving method suitable for implementation as a computer program” (SEDGEWICK, 2011, p. 4). In essence, this abstraction followed by organizing an idea into a set of well-defined steps can be understood as algorithmic thinking.

Although we are discussing algorithms in a specific generative context, we have already seen at the beginning of this topic that they are not a new concept for design. As put concisely by Omine (2014, p. 78-83), Karl Gerstner — one of the “most coherent theorists” of graphic design, promoted discussions on the subject since the middle of the 20th century, when he published his book Designing Programs (1964).

In contemporary works of many designers and artists, we can still find algorithmic thinking as an essential and characterizing element.In his book Analog Algorithm (2019), for example, the German illustrator and graphic designer Christoph Grünberger experiments with a series of grid-based procedures to define a clear rationale for each of his works. For him, a rational and logical approach to design is not opposed to creative freedom, but an accelerant for the creative process. Such a belief motivated him to reflect on his career and present many examples of how algorithmic creation enables the production of an universe of possibilities after the establishment of a sufficient foundation — the grid (GRÜNBERGER, 2019).

Algorithms, though, are only part of solving the problems we are interested in. Since computational environments manipulate data through lines of code, algorithms must be translated into programs so that they can, in fact, work. At this point, computational thinking reaches a very practical spectrum (RICHARDSON, 2016 p. 20).

In addition to the computational-thinking skills already discussed, programmers employfrequent debugging and generalization processes to create efficient and correct programs to deliver the desired quality of automation. When debugging, they systematically analyze and evaluate programs or algorithms through testing, reviewing logs, and the intuitive application of logical thinking — that allows results to be anticipated and verified. Through generalization, patterns are identified and used as the basis of the general strategy of what needs to be programmed. Finally, by decomposing a problem, or artifact, programmers can break it down into smaller elements and organize it in a way it can be understood and solved. This makes complex problems, whether related to writing a program or defining a design project, easier (CSIZMADIAet al., 2015, p. 7).

Moreover, to effectively write the program’s source code, the most suitable programming language can be selected, among the thousands of options that exist. To make this decision, one can rely on criteria such as the operating system or device on which the program will run; the desired aesthetic quality; and personal preference(REAS; MCWILLIAMS, 2010, p. 17).

With a notable focus on programming and experimentation, Zhang and Funk (2021) proposed a four-stage creative process for projects in which computing is the primary “material” (Figure 3). The stages are **idea to visuals**; **composition and structure**; **refinement and depth**; and **completion and production**.

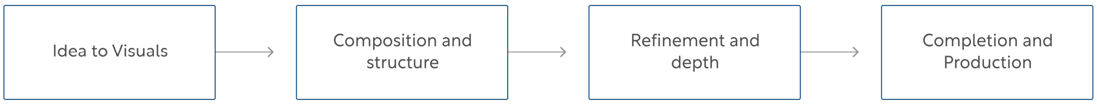

Figure 3: Representation of the model proposed by Zhang and Funk (2021). Created by the author.

According to them, the project begins with the **development of visual** concepts that will serve as a starting point for the project. To this end, instead of conventional approaches such as creating mood boards and drafts, Zhang and Funk suggest a very exploratory way, in which moves are made through **successive programming experiments**. Then, once the initial visual concepts are reasonably developed, the next step of the process is improving them by organizing the graphic elements and defining their interrelationships. At this point, it is possible to experiment with a series of compositional practices and concepts, such as repetition, variation, and randomness, and incorporate them into the generative algorithm.

In the **refinement and depth** stage, the intended elements and visual composition are properly programmed, as a structure for the code is also determined. Therefore, functions for moving and positioning elements, controlling sizes, coloring, and employing user input data to control certain aspects are written.

Finally, in the **completion** stage, the program is prepared to produce the expected artifact. This means that if a movie or animation is being produced, for example, the team will worry about correctly setting the frame rate per second and adjusting the program to synthesize the clips in the necessary resolution. On the other hand, if the outcome is a poster, they will probably be concerned about producing image files large enough to print with pixel-perfect quality. Moreover, at this stage, it might also be interesting to run the program on a mobile device, or over the internet, which will require specific adjustments.

Zhang and Funk (2021) also proposetwo different, yet complementary, approaches to ideation in generative design. In the first approach — **concept-based ideation** —, creative work begins conceptually, probably before computational tools are used. Since the main challenge in this approach is to turn an abstract idea, already in mind, into a satisfying computational result, one implication is that designers or programmers already have ideas and expectations about how the employed concept will be materialized. In the second approach — **material-based ideation** —, computing and programming are taken as the creative “material” we will be experimenting with to encounter concepts to be elaborated. More practically, this means that the creative work will gradually take a more concrete shape and direction only after a series of experiments with different software tools, and algorithms.

In a way, the above-discussed distinction in the two approaches for ideation inherits elements from an earlier debate concerning the computer’s role in creative activities such as design and art. As stated at the beginning of this topic, for a long time, computing was employed to gain precision and efficiency rather than exploring new procedural and aesthetic possibilities. A use, according to Reas and McWilliams (2010, p. 25), centered on the **production** of preconceived shapes and concepts. However, when employed with a **conception**-oriented perspective, the computer actively participates in developing the shape or concept in question.

For Agkathidis (2015, p. 14), for example, when used for conception, one of the main strengths of generative creation is its ability to offer new directions to design projects and break with predictable relationships between form and representation. Something that occurs because computational complexity is adopted as a co-drawer with whom designers "negotiate" creation.

Accordingly, Grünberger (2019, p. 13) understands generative design as a practice in which control is relinquishedin favor of results since the entire design process becomes subordinate to an autonomous logic or random seed. Because of this, the author states that designers start behaving not only as creators but mainly as interpreters and curators who must spontaneously evaluate results while ensuring that the process employed is adequate and compatible with the strategy of the project.

Finally, significant advances in computational power have enabled design programs to make essential contributions in solving complex problems — for which the space of design solutions is vast and dynamic (BUONAMICI et al., 2021, p. 144). Moreover, adopting computing as a means can also point out new procedural and aesthetic paths and potentialize human designers' creativity (BUONAMICIet al., 2021, p. 144).

In the end, such a paradigm shift clarifies the relevance of studying generative creation as an element capable of instigating novel practices and aesthetics to the field of design. Something also illustrated by the many designers and artists that have been engaging in political and experimental initiativesby relying on generative poetics (BRAIN; LEVIN, 2021).

Final Considerations

When implying that 'How do I abstract?' is the most relevant question in the generative design process, instead of "How do I draw?", Groß et al. (2018, p. 507) are not only illustrating the existence of a layer of abstraction that intermediates the whole creative process but also establishing a clear intersection between design and computing in which generative design is positioned — **something that suggests the need of concepts of both areas for a sufficient understanding of the creative process investigated**.

After being understood as one of the most important elements within the creative process in generative design, further research on the abstraction-based nature of it became necessary. This gap was filled by research on computational thinking, to which abstraction is also a central element (KELLY; GERO, 2021; BOCCONI et al., 2016). As stated by Shute et al. (2007), this form of thinking describes the mental activities involved in the formulation of a problem in ways through which it admits computational solutions to be carried out by humans, machines, or both.Through further research, Bocconi et al. (2016, p. 18) and Beecher (2017, p. 7) identified that the primary abilities involved in computational thinking areabstraction, algorithmic thinking, automation, decomposition, debugging and generalization, logical thinking, modeling, and evaluation.

Despite being the only two forms of thinking to gain prominence since the beginning of the XXI century, studies about how design thinking and computational thinking relate to each other are scarce — perhaps due to the lack of consensus when it comes to defining them, or to the assumed differences between the problems each one of them is more suitable for (KELLY; GERO, 2021, p. 1). Nevertheless, as Kelly and Gero (2021, p. 7) suggest, some tasks involve both computational and design thinking, such as a web designer addressing the design of a website for a client or an engineer during a design activity of setting up parametric models.

Imagine our web designer begins addressing a client's brief by following a conventional design thinking approach (e.g., the double-diamond), thus spending some time researching stakeholders' expectations and final users' needs. Of course, this would expand the frame through which the problem will be understood by arising additional knowledge related to these contexts. However, suppose that, at some point in the process, the web designer needs to develop reusable components for the current set of pages of the website being created and the possible future ones. This designer would be abstracting this problem into a set of design aspects of the website to bebe componentized —thus applicable to future pages without much effort — and which not. Similarly, when engineers need to establish parametric design systems, computational thinking is necessary, as these parameters need to be abstracted from the problem and the context to synthesize maybe hundreds of alternative solutions. However, for these solutions to be abstracted into a set of parameters capable of modulating them, relevant knowledge about the context — such as users' needs and environmental restrictions — must be obtained so that the team can understand which variables will feed these parameters' creation.

It appears that in generative design, designers are constantly dealing with these two forms of thinking, as well.

Finally, it is worth noticing that different points of view exist when it comes to defining generative design and describing how generative designers work — perhaps due to the complex conceptual network involved in writing about an approach that is both recent (as a more popular practice) and multidisciplinary. As suggested by Brain & Levin (2021, p. 4), "in computational art and design, many responses to the questions of what and why continue historical lines of creative inquiry centered on procedure, connection, abstraction, authorship, the nature of time, and the role of chance".

As a concise answer to this paper's goal, it is possible to say that according to the literature review:

- When designing with generative systems, designers have their creative process fundamentally altered as their role shifts from directly manipulating artifacts to curating and evaluating the computational processes that are supposed to produce them — albelt partially —, and as computational intelligence becomes an active participant in it;

- Such a creative process involves reaching an idea; abstracting this idea into an algorithm, translating the algorithm into a program to be executed; and obtaining and evaluating the output — what would possibly feedback the cycle;
- Due to the existence of a characteristic layer of abstraction — as designers need to constantly think in terms of algorithms to create desired artifacts (to move)—, there is a relationship between this specific creative process and computational thinking — to which abstraction is a characterizing aspect—, which implies that designers are required to rely on abilities related to design thinking and computational thinking;
- Therefore, whether if supported by material-based or concept-based ideation, designs will need to be formulated in ways that admit computational solutions, which implies thata sucessful generative design process will be incoporating CT-related abilities such as algorithmic thinking, automation, decomposition, debugging, generalization, logical thinking, modeling, and evaluation into designers'skillset, and raising technical concerns related to how the product will be implemented.

References

AGKATHIDIS, Asterios. **Generative Design**. 1.ed. London: Laurence King Publishing, 2015.

BAXTER, Mike.**Product design**: A practical guide to systematic methods of new product development. Boca Raton: CRC Press, 1995.

BEECHER, Karl. **Computational Thinking:** A beginner's guide to problem-solving and programming. Swindon: BCS Learning & Development Limited, 2017.

BRAIN, Tega; LEVIN, Golan. **Code as Creative Medium:** A Handbook for Computational Art and Design. Cambridge: MIT Press, 2021.

BOCCONI, Stefania et al. (2016). **Developing computational thinking in compulsory education**: Implications for policy and practice. JRC Science for Policy Report, Luxembourg, 2016.

BUONAMICI, Francesco et. al. **Generative Design:** An Explorative Study**.** Computer-Aided Design & Applications,Milton Park,v.18, n.1, p. 144-155, 2021.

CORMEN, Thomas. **Algorithms Unlocked.** Cambridge: MIT Press, 2013.

CROSS, Nigel. **Designerly Ways of Knowing**. Nova York: Springer, 2006.

CSIZMADIA, Andrew et al.**Computational thinking**: A guide for teachers. London: Computing at School, 2015.

FURBER, Steve. **Shut down or restart?**: The way forward for computing in UK schools. London: The Royal Society, 2012.

GALANTER, Philip. **What Is Generative Art?:** Complexity Theory as a Context for Art Theory, 2003.

GERSTNER, Karl. **Designing Programmes:** Programme as Typeface, Typography, Picture, Method. Zurique: Lars Müller Publishers, 2007.

GROß, Benedikt et al. **Generative Design:** Visualize, Program, and Create with JavaScript in p5.js. New York: Princeton Architectural Press, 2018.

GRÜNBERGER, Christoph. **Analog Algorithm:** Source-Related Grid Systems. Baden: Lars Müller Publishers, 2019.

KANTOSALO, Anna.**Human-Computer Co-Creativity:** Designing, Evaluating and Modelling Computational Collaborators for Poetry Writing. PhD Thesis. Helsinki : University of Helsinki, 2019.

KELLY, Nick;GERO, John. **Design thinking and computational thinking**: A dual process model for addressing design problems. Design Science, Cambridge,v. 7, n. 8, p. 1-15,2021.

KRISH, Sivam. **A practical generative design method**. Computer-Aided Design, Amsterdam, v. 43, n. 1, p.88-100, 2011.

LAWSON, Bryan; DORST, Kees. **Design Expertise**. Burlington: Architectural Press, 2009.

LUBART, Todd.**The Creative Process**: Perspectives from Multiple Domains. London: Palgrave Macmillan, 2018.

MOUNTSTEPHENS, James; TEO, Jason. **Progress and Challenges in Generative Product Design:** A Review of Systems**.** Malaysia: Universiti Malaysia Sabah, 2020.

NAKE, Frieder. **The Pioneer of Generative Art**: Georg Nees. Leonardo, Cambridge, v. 51, n. 3, p. 277–279, 2018.

NEVES, Isabel. **Abordagem científica ao Projecto no início da Era Computacional:** Hochschule fur Gestaltung of Ulm e a sua diáspora. Lisboa: Faculdade de Arquitetura da Universidade de Lisboa, 2015.

OMINE, Eduardo. **Design gráfico computacional: computação aplicada no projeto e na produção de imagens dinâmicas e interativas**. Master's Thesis. São Paulo: Faculdade de Arquitetura e Urbanismo, Universidade de São Paulo, 2014.

REAS, Casey; MCWILLIAMS, Chandler. **Form+Code in Design, Art, and Architecture**. 1. ed. Hudson: Princeton Architectural Press, 2010.

SAWYER, Keith. **Explaining Creativity:** The Science Behind Human Innovation. 2. ed. New York: OUP US, 2012.

SEDGEWICK, Robert; WAYNE, Kevin. **Algorithms**. 4. ed. Boston: Addison-Wesley Professional, 2011.

SHUTE, Valerie. **Demystifying computational thinking**. Educational Research Review,Amsterdam,v. 22, p. 142-158, 2017.

WING, Jeannette. **Research notebook: computational thinking**: what and why? The Link, 2011.Available from: https://www.cs.cmu.edu/link/research- notebook-computational-thinking-what-and-why. Accessed in Mar 05th, 2022.

YADAV, Aman. et al. **Computationalthinking in elementary and secondary teacher education**. ACM Transactions on ComputingEducation, v. 14, n. 1, p. 1–16, 2014.

ZHANG, Yu; FUNK, Mathias. **Coding Art**: The Four Steps to Creative Programming with theProcessing Language. New York: Apress, 2021.

About the authors

Caio Barrocal
Interested in the intersections between design and computing, Caio holds an MSc in Design and a BSc in Computer Science, both from the University of São Paulo (Universidade de São Paulo).Currently, heworks as a digital product designeratBabbel, in Berlin. Moreover, during the last two years he's been occasionally working as a design teacher and mentor for international young students.
https://orcid.org/0000-0002-0687-1881

Clice de Toledo Sanjar Mazzilli
Associate Professor at FAUUSP, where she's been teaching since 2001. She's the vice-coordinator of LabVisual and leads the research group Design, Environment and Interfaces (CNPq). Gathers experience in the following themes: design creation processes, experimental processes, and visual narratives.
https://orcid.org/0000-0002-6903-9099