Olabe, J.C., Basogain, X., Olabe, M.Á. (2023). *Computational Model of the Mind: How to Think About Complex Systems in Nature*. In: Rezaei, N. (eds) Brain, **Decision Making and Mental Health**. Integrated Science, vol 12 553–573 Springer, Cham

COMPUTATIONAL MODEL OF THE MIND: HOW TO THINK ABOUT COMPLEX SYSTEMS IN NATURE

Juan Carlos Olabe
Christian Brothers University
Memphis, USA
jolabe@cbu.edu

Xabier Basogain
University of the Basque
Country Bilbao, Spain
xabier.basogain@ehu.eus

Miguel Ángel Olabe
University of the Basque
Country Bilbao, Spain
miguelangel.olabe@ehu.eus

"Our ignorance can be divided into *problems* and *mysteries*. When we face a problem, we may not know its solution, but we have an insight, increasing knowledge, and an inkling of what we are looking for. When we face a mystery, however, we can only stare in wonder and bewilderment, not knowing what an explanation would even look like."

Noam Chomsky

Abstract: The collaboration between the areas of computation and the biological sciences in the last two decades has changed in essential ways the scope and methods of research in their shared field. In this chapter we describe two fundamental principles: Computational Equivalency; and the Suboptimal Paradigm of Evolution. These principles permeate current research in computation and biology. We illustrate these principles with three examples: animal behavior; plant behavior; and human interaction. These examples are described formally by means of computational languages. These descriptions are designed to provide the reader with a concrete framework where these and similar ideas could be studied, tested, and developed. The computational descriptions of these systems are simple and easily accessible to readers. This accessibility makes appropriate the introduction of these ideas in the traditional school curriculum. Our current students are the future custodians of our society. They need novel ideas to respond to the novel challenges of the modern society.

- Keywords: Computational Models, Structured Complex Systems, Theory of the Mind, Animal Behavior, Plant Behavior, Human Interaction, STEAM Education

1 - Introduction

This chapter reviews some of the fundamental ideas behind the close cooperation between the formal sciences of theoretical computer science and numerical computation, and the biological sciences of plant science, neuroscience and behavior. It is intended as a general overview for an inquiring reader with a general scientific background. It places special interest in the description of real-living systems by means of computational systems. These systems are designed to be of easy comprehension by readers that are not necessarily trained in computational languages. For this purpose, we have selected two computational languages, Snap and Scratch, which use a simple graphical interface [1-4.]

The simplified representation of living processes, and the description of their behavior with simple computations, makes possible the integration of these models in the curriculum of our schools. Our

society has recognized the need to update the current educational system in order to prepare our citizens to the challenges of the modern world. One important group of academic initiatives plans to integrate in the classroom, along with the traditional core subjects of mathematics and sciences, the new areas of technology, engineering and the arts. These initiatives are grouped under the labels of STEM or STEAM initiatives [5-9.]

The chapter begins with a section dedicated to the introduction of two general principles of computation and evolutionary biology. These principles will serve as guides in the study and assessment of other novel ideas revealed in the chapter. These principles are: the *Computational Equivalency of Systems;* and the *Suboptimal Paradigm of Evolution*.

The rest of the chapter is organized into three sections. The first case involves the simulation of the motion patterns of a small worm called c-elegans. The objective of this simulation is to identify the relationship between its simple neural system and the resulting complex behavior. The second example explores the study of the complex distribution of the sunflower seeds and its relationship with two concepts: the Fibonacci sequence, and the golden ratio. The objective of this study is to identify how a simple living structure can create a complex geometric pattern. Finally, the third example studies the simulation of the complex patterns in the dynamics of a pandemic using a small set of computational systems based on the intuitive ideas of how pandemics appear, expand, contract and disappear.

In this chapter we will use several computational models to implement computational ideas and three living systems. We will use two programming environments: MIT's Scratch, and Berkeley's Snap. These two programming environments are the direct result of STEAM initiatives in the US. Although the ideas and systems of this chapter will be implemented with these programming environments, it is not necessary to have a background in programming to have a general understanding of these ideas and systems. Those with background in computer science will be able to follow in detail the technical aspects of the implementations.

Snap and Scratch will be used to illustrate the concept of Computational Equivalency, some artificial c-elegance worms, a dynamic sunflower, and a collection of 100 citizens under study that will help device optimal policies to combat the spread of a pandemic. All programs referenced in this chapter are accessible in the links provided throughout the chapter. In addition, these links include animations of their dynamic processes. In this chapter will include static figures that are screen shots of these dynamic processes. These static figures cannot, of course, show the complete dimension of these dynamic systems.

These programming environments use a graphical representation of the code. This allows readers to easily understand the processes involved in the behaviors of the living processes simulated. In addition, these computational environments provide researchers the opportunity to test, manipulate, and verify their ideas.

2 – Master Guidelines in the Computational Study of Evolutionary Systems

In this section we briefly introduce two principles that are relevant in the study of the computational behavior of living systems. We call the first principle the *Computational Equivalency of Systems*. Living systems exhibit behaviors of an apparent great external complexity. This leads to the intuition that a

complex internal control system must be required to explain this phenomenon. This intuition is often incorrect and leads researchers in wrong directions. As we will see later in the examples of the motion patterns of the c-elegans, or the seed distribution of the sunflower, their external complex behavior seem to lead us to search for complex control structures. However, the simple neural system of the c-elegans precludes the existence of such complex computational control. Even more dramatically, the absence of a neural system in the sunflower elevates to the level of mystery the process by which a plant can compute the location of the seeds in order to create complex patterns of radial spirals.

The concepts outlined in this chapter have their origins in ideas developed during the twentieth century in the areas of computation and information. First, we find the Theory of Computation, developed by Alan Turing. It established that complex tasks can be implemented through computation, and that computation can be formally studied and developed [10.] Secondly, in the area of Information Theory, Claude Shannon developed concepts and procedures to formally describe the tasks of communication and computation in terms of information [11.] A third area relevant to the topics of this chapter includes the Computational Theory of the Mind, developed by Hilary Putnam and his colleagues [12.]

The principle of Computational Equivalency establishes that two systems are equivalent if they are *functionally* equivalent. If two systems behave equally under similar circumstances, regardless of their internal computational complexity, they are equivalent.

Let us apply this principle to the patterns of the seeds in the sunflower. We are interested in researching the nature of the sunflower and the systems that controls it; we call this real system S0. We can first create a complex computational system, S1, because we assume that the real system is complex. This complex system could be based on a distribution of radial spirals with complex mathematical equations that reproduces the pattern of the real sunflower. Alternatively, as we will see later in the chapter, we can create a simple computational system, S2, because we assume that the real system could be simple. Let's assume that the simple system also is able to reproduce the pattern of the real sunflower. Computationally, all three systems, the unknown real system S0, and the artificial systems S1 and S2, one complex, the other simple, are equivalent: the three systems produce the same pattern of seed distribution. Because the system we are trying to model is a sunflower, a plant, its computational capabilities are limited. Therefore, it is more likely that the simple artificial system S2 resembles more closely the real system S0, than the alternative complex artificial system S1.

The following example introduces the computational capabilities of simple living systems. The goal of this example is to show that complex arithmetic operations, such as exponentiation and multiplication, could be implemented by very simple systems. We will limit the capabilities of these systems to count up, or increment, and count down, or decrement. We will note that it is unlikely that living systems implement exponentiation operations; the goal of this discussion is to introduce methodologies that show that it is possible.

Returning to the idea of Computational Equivalency, we can observe that a plant, a tree for example, has some means of computation. For example, it can keep track of its age by adding each year one layer to its trunk. Humans can determine the age of the tree by counting the rings. We call this operation of adding one to an accumulated value '*the operation increment*.' In a similar way we could imagine natural processes that would implement '*the operation decrement*.' Increment and

decrement are basic arithmetic operations. We could call them '*counting up*,' and *'counting down.'* Children learn these two operations first: they are able to recite 1, 2, 3, etc., or … 3, 2, 1.

Let's now think of other more complex arithmetic operations, such as addition, multiplication and exponentiation. Figure 1 shows three examples where these operations are implemented by systems that can only increment or decrement, count up, or count down. These three short scripts transform recursively a complex task into many simple operations. (In Snap and Scratch, only the color green blocks implement any arithmetic operations.) We can see that in these scripts, the only green blocks include the operations comparing, adding one, and subtracting one.

This principle of computational equivalency helps us understand how it is possible to observe complex behaviors in simple systems. It also points us in the direction of where we may find the answers to our research when analyzing the complex behavior of simple systems.

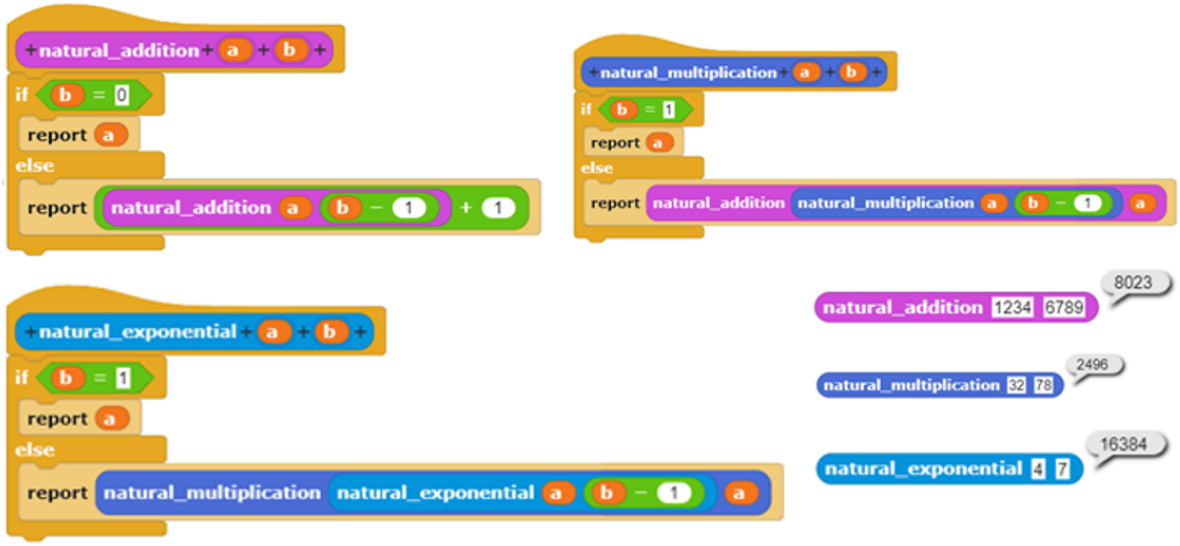

Figure 1. Three complex arithmetic operations implemented with simple Increment and Decrement

The second principle, the Suboptimal Paradigm of Evolution, establishes that the systems created in nature by the process of evolution by natural selection, are by nature suboptimal. Evolution proceeds invariably forward. New systems, new solutions, are obtained by modification and selection of old ones. Evolution cannot return to the past and start anew. This is the reason the forearm of vertebrates contains two main components, radius and ulna. A better solution would have been to have only one component, such as the humerus of the upper arm. The blind spot of the retina and many other examples illustrate this principle of the paradigm of suboptimal solutions [13, 14.]

This principle leads us to the conclusion that the formal structural study of natural systems, in particular the brain, may suggest the implementations of cognitive tasks in ways that are not optimal. The structural study of the brain is very complex. With almost 100 billion neurons, each with thousands of synapses, the complete structural description of the human brain is not near. However, the functional behavior of the brain at higher levels of abstraction, such as behavior, memory and symbol manipulation are much simpler tasks. Therefore, a functional description of the systems of the brain, rather than their structural compositions, can provide, in the immediate future, insights that are directly applicable to areas such as education, psychology and medicine. By exploring and describing the functional capabilities of the brain, we would be able to harvest the cognitive power

of systems designed by evolution to solve the problems of our evolutionary ancestors, and use them to solve our current challenges.

The following sections of the chapter present three case studies. These are examples found in nature. They will be described by means of computational models using the environments Snap and Scratch. Table 1 summarizes the computational concepts and principles, fields of science, and the programming languages used in their study.

Table 1. Summary of the case examples described in the chapter

Case study	**Science**	**Concepts & Computational Principles**	**Computational Language**
C. elegans	Animal behavior	Suboptimal Paradigm Neural activity	Scratch
Sunflower	Plant behavior	Paradox of Complexity Fibonacci sequence, and the golden ratio	Snap
Pandemic behavior	Human Behavior	Computational Equivalency Dynamics of a pandemic	Scratch

3 – Brain, Neural Activity and Behavior: analysis of the motion of the c-elegans

The first example of collaboration of computation and biology describes the connection between neural activity and the external behavior of a colony of c-elegans. It is inspired by the work of Dr. Cornelia Bargmann, 2012 Kavli Prize in Neuroscience. Dr. Bragmann and her team study the mechanisms used by c-elegans for the detection of volatile odorants, and how they affect the motion according to whether these chemicals are attractants, indicative of nutrition, or repellents, indicative of hostile environments [15-21.]

The programs and animations corresponding to this example are accessible in this link (https://www.ehu.eus/gmm/springer-nature#MaterialoftheExample1).

The motions used by c-elegans in its exploration of the environment are controlled by a simple neurological network. The simplicity of its connectome, the complete neuronal description of its brain, is the main reason of selecting c-elegans in this type of studies. However, the simple neurological control of its motion produces patterns of behavior, with turns and reversals with irregular frequencies that require the probabilistic analysis of these patterns.

A first set of ideas to be tested by this example consisted in providing ever simpler motion control systems to a simulated c-elegans, in order to explore the patterns of behavior produced and their similarity to actual living creatures.

In addition, some constraints were included in the prototypes. The simulated c-elegans would have only limited access to environmental conditions. In particular, for the purposes of moving in the environment, the model would be blind: it could only identify obstacles by collision. Initial prototypes provided results with very lifelike motion patterns that could be recognized as those of c-elegans or small insects.

A second set of ideas to be tested by this example consisted in creating complex environments that would provide information on how powerful the controls systems were in solving maze traversing tasks.

Figure 2 shows the block-type programming code that implements the motion control of two simulated c-elegans inside a maze. The complete control system consists of two parallel scripts. The script on the left controls the motion of the c-elegans in the absence of collision. It is based on a relatively short forward motion before randomly turning. The simple repetitive sequence of short forward motion interrupted by random left or right turns constitutes the complete control of motion in the absence of collision. The numerical values of the forward motion, 4 steps, and the random turns, +/- 30 degrees, were determined by trial and error. In addition, by simulating multiple c-elegans with different sets of parameters, one could compare their performance, and in a process analogous to evolution by natural selection, select those that outperformed their competitors.

The script on the right is the complete motion control for the case of collision. The strategy to follow when encountering a wall (represented by the blue-grey color) consisted in moving straight back a short distance from the wall, and turn a set angle. These two parameters were also set by trial and error and are candidates to modification and selection according to performance or fitness.

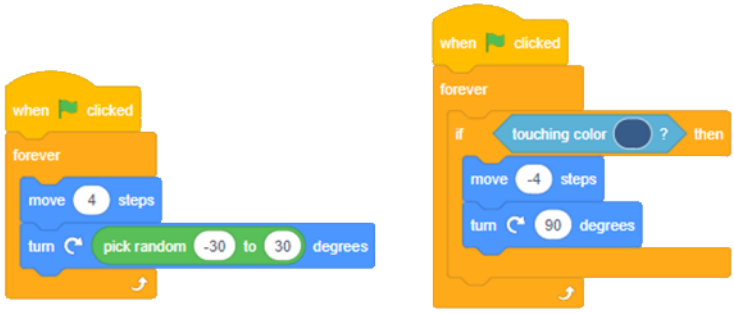

Figure 2. Control systems for the motion of simulated c-elegans

Figure 3 represents a computer screen shot of a simulation already in progress. Two c-elegans, represented by two small blue dots move inside a maze with 16 separate compartments. To aid in the analysis, two traces, blue and pink, were created to mark the progress of the simulated creatures. It can be observed that, perhaps against intuition, the simple motion control system provides great ability to navigate throughout the maze, often colliding against the walls, but also able to move from compartment to compartment in the maze.

These simulations allow us to visualize the complex external behavior created by a simple control system. It also allows the probabilistic study of parameter sets for the controls, and how these affect the overall performance: Is it possible to find all 16 compartments of the maze is a given time period? What are the tradeoffs between speed and collision? Etc.

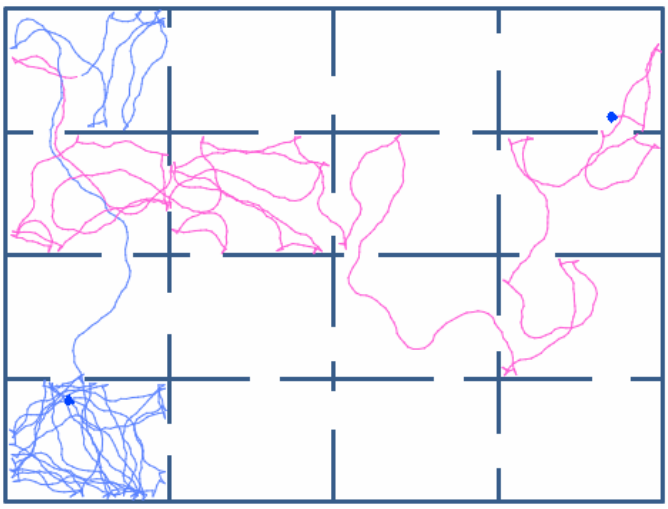

Figure 3. Two c-elegans in a maze: comparing the performance of motion parameters

4 – Computation without a Brain: Sunflower, Fibonacci sequence and Golden Ratio

The example described in this section studies the seed distribution of the sunflower, with its complex and aesthetic patterns, and its relationship to computational systems. Because the seed distribution of a sunflower is very complex and it has direct relation to two mathematical ideas (the Fibonacci sequence, and the golden ratio) it will be used to illustrate the phenomenon called the Paradox of Complexity. This paradox establishes that complex behavior is often the result of simple controls systems [22-28.]

The programs and animations corresponding to this example are accessible in this link (https://www.ehu.eus/gmm/springer-nature#MaterialoftheExample2).

Figure 4 shows the pattern of seed distribution of a sunflower. It can be seen by simple inspection two sets of spiral curves, one turning right, the other turning left. A closer inspection also shows that the patterns of spirals appear with different curvatures. In addition to the aesthetic symmetry of the patterns created, the numbers of spirals, left and right, follow the sequence of the Fibonacci sequence: 21, 34, 55, etc.

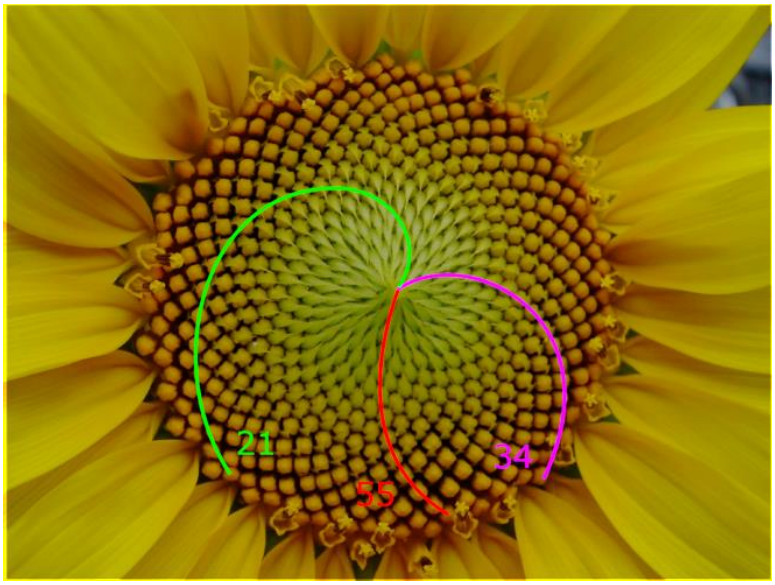

Figure 4. Sunflower patterns and count of different types of spirals (21 green, 55 red, 34 pink)

The Fibonacci sequence follows a simple rule: each number in the sequence is obtained by adding the two preceding numbers. The first two numbers of the sequence are zero and one. This simple rule generates the sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, etc.

The distribution of the seeds in the sunflower also presents a particular geometric property that involves the golden ratio. The golden ratio is an irrational number with the value 1. 61803…

The history of the golden ratio [29-31] is full of interesting facts. It involves the study on the part of the ancient Greeks of the dimensions of the pentagon and the dodecahedron. It also represents an ideal ratio between quantities in architecture, human proportions, etc.

The golden ratio is also equal to the limit of the ratio of two consecutive numbers of the Fibonacci sequence. The higher in the sequence (21/13, 34/21, 55/34, 89/55, etc.) the closer the ratio approximates the irrational golden ratio.

As final commentary on the golden ratio, it is the most irrational number of all possible irrational numbers. An irrational number cannot be expressed as a ratio of two integers, or as a finite sum of ratios, but it can be approximated by them. The more elements of the approximation that are included, the better the approximation. In this sense, the golden ratio is the irrational number that requires more elements of the approximation for the same accuracy.

All these commentaries regarding the patterns of distribution of seeds in the sunflower and their relationship to the Fibonacci sequence and the golden ratio seem to anticipate a very complex system controlling the sunflower in order to create such patterns.

We present first a computational implementation of a simulated sunflower. Figure 5 shows the complete, and very simple, computational system that distributes 800 seeds in a pattern the replicates natural sunflowers. The complete control includes only two scripts. The script on the left creates in a loop 800 distinct seeds. The only relevant process is that after one seed is created, a simple rotation of 360/1.61803 degrees is implemented, and the process of creation and rotation is repeated. The expression is presented in this form for clarity, to signify that the turn is a complete

circle divided by the golden ratio. This first script distributes the stems from which the seeds will grow.

The second script, which is even simpler, is the pattern of movement of each seed after it has been created. The motion is a simple constant movement forward. One can think that this motion represents the motion of each stem as it is pushed outward when the seeds grow in size.

Figure 5. Complete computational control of the seed distribution in sunflowers

Figure 6 shows a screen shot of the process of seed distribution of the sunflower. All seeds are created at the center of the flower. The orientation of each seed is determined by a turn of 222.5 degrees (360/golden ratio) from the previous seed. After the seeds have been created, their motion consists of a simple uniform and rectilinear forward movement.

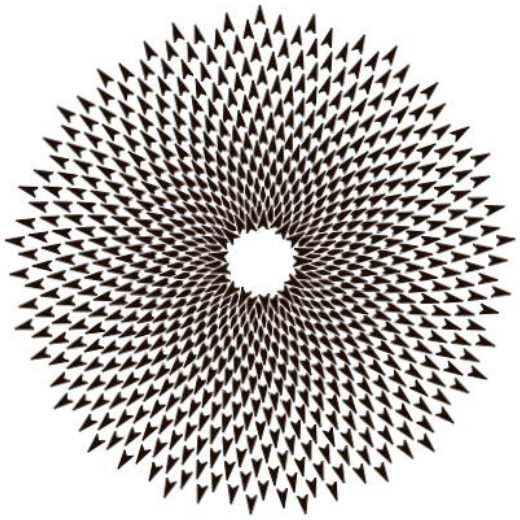

Figure 6. Screen shot of the dynamic distribution of the seeds of a sunflower

There are several conclusions that can be derived from this process. There are also some fundamental questions. The first conclusion is that it is possible to construct a dynamic distribution of seeds in a way that replicates the complex patterns of real sunflowers with a very simple computational system. This illustrates the Paradox of Complexity, or the counterintuitive fact that a complex behavior requires a simple generative system. The second conclusion is that this phenomenon of complex behavior with simple generative systems could be norm in nature, rather than the exception. This last conclusion may direct new research in novel directions where the hypotheses to be examined are based on simple control systems, rather than complex ones.

A fundamental question remains: how did the sunflower arrive to *'discover'* the golden ratio, the most irrational of all irrational numbers in the world of mathematics? To help address this question it may

be useful to turn to the fundamental principles of evolutionary biology. The 13-year and 17-year periodical cicadas also point to an apparent mathematical puzzle in nature [32, 33.] How did these cicadas arrive to *'discover'* the mathematically complex concept of prime numbers, such as 13 and 17? The evolutionary answer suggests that different cicadas evolved to live underground different number of seasons. Those who evolved to live underground a prime number of years found that, when they emerged in the springtime, encountered less competition, which favor their reproduction.

In a similar way, during evolutionary time, different types of sunflowers evolved to distribute their seeds with a different angle of turn. The angle of turn that provided an optimum size for growth produced the largest seeds. This may have attracted more birds that would feed on them, and carry in their digestive systems the seeds that would expand their geographical reach often and far.

5 – Complex systems ruled by simple norms: Pandemics and policies of confinement and social distancing

The worldwide pandemic caused by Covid19 has made familiar to the general population some of the strategies that can be used to delay and even reverse its expansion. Two of the most widely used strategies include confinement, requiring the citizens to remain isolated in their homes, and social distancing, eliminating contact and prolonged close proximity [34-38.]

The programs and animations corresponding to this example are accessible in this link (https://www.ehu.eus/gmm/springer-nature#MaterialoftheExample3).

Technical journals and websites have offered computer simulations that analyze the effects of such policies and their effects in the general population. Even the general media, such as the New York Times or the Washington Post, offered in their online publications simulation models that the readers could manipulate in order to compare the effects of alternative confinement and social distancing policies.

The familiarization of the general public with this type of computational tools is welcome because they describe the modern tools that experts use in assessing the effects of their decisions. The objective of this section is to promote the study of these tools in schools. The curriculum of our school systems continues teaching traditional subjects such as Euclidean and Cartesian geometry, arithmetic, basic algebra and trigonometry. However, the last decades have produced large sets of numerical methods that allow us to approach very complex systems, such as pandemics, that were outside the possibility of analysis only a few years ago. One could argue that these new mathematical tools are of great importance, but also of great cognitive complexity, and could not be studied in the school environment. In this section we describe one such system of analysis of pandemic and, with its illustration and description, we convey the message that its cognitive complexity is within the capacities of our students, and therefore it should be included in the current school curriculum. This philosophy to modernize the current school curriculum reflects the main goal of STEAM initiatives around the world.

We will illustrate these ideas by describing how a simple programming environment such as MIT's Scratch performs all the operations required to solve the apparent complex problem of a pandemic.

We begin by describing the major characteristics of the simulation. We will represent a community of 100 citizens under study as a set of 100 circles that may freely move on a stage that represents their community. We will color code the citizens in order to represent their health status: blue means healthy, pink means infected, and green means recovered. This is represented in Figure 7.

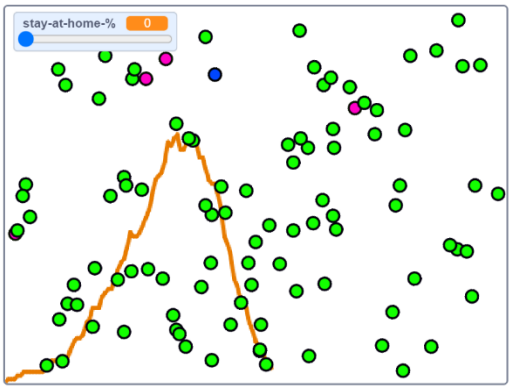

Figure 7. Screen-shot of the evolution of infected persons in a social group

Initially, 99 citizens will have a healthy status, blue color. One of the citizens will be infected with the virus, pink color. As the citizens move about their town, they may get in contact with each other. In that case, a healthy citizen will become infected by the virus if the contact is with a carrier person. From the moment a citizen becomes infected, it enters a period of recovery, and its status is changed to pink color. After the period of recovery is completed, the health status changes to recovered: green color. However, during the recovery period the infected citizen will transmit the virus to those citizens with whom they enter in contact. For simplicity, in this simulation, recovered citizens cannot get infected again, nor can they infect other people.

The simulation includes an orange graph, see Figure 7, that represents the total number of infected citizens throughout the period of pandemic. One of the parameters that is studied in this simulation is the percentage of citizens that is required to be in confinement (stay-at-home.) By changing the value of this parameter, the simulation will reflect the corresponding evolution, orange graph, of the number of infected people at any time. The goal of the confinement policy is to flatten the curve so that at any time the total number of infected people is below the resources available in hospitals and the health system in general.

Figure 8 shows the process of initializing all the citizens in the simulation. The first 99 citizens are created with a health status of healthy. Two additional variables are set: the time of being sick is initialized to zero, and time to recover in the event of becoming sick is set to 100. Finally, in this script, a single citizen is created with a health status of sick and the corresponding pink color. This citizen will be patient-0 and the origin of the pandemic.

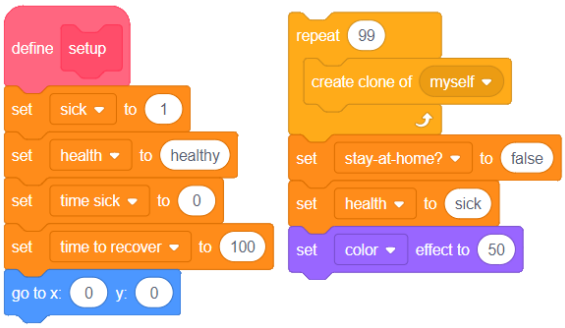

Figure 8. Code to initialize the program to simulate a pandemic

In Figure 9 we observe that when the 99 health citizens are created, they are positioned randomly somewhere in the geography of their city. Later, we select randomly whether they are confined or not, according to the level stablished by the analyst. This value is represented by the variable stay-at-home%.

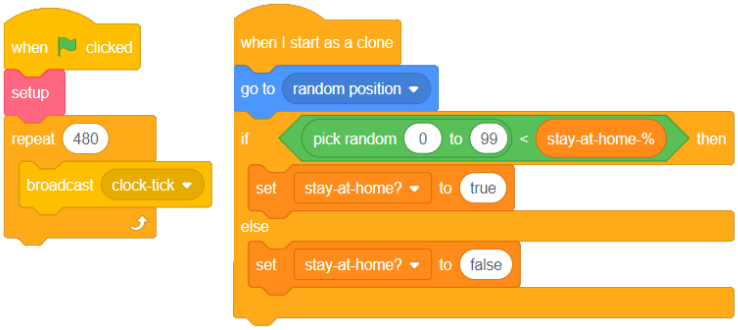

Figure 9. Code to setup the program and determine the persons that will stay at home

After this process of initialization, the life of the citizens is simulated as the interactions of 100 individual citizens that move in the geography of their city or stay at home. Figure 10 shows that each citizen follows a repetitive set of three phases. First, if the citizen is free to move (is not a stay-at-home person) it will follow the script to take a step, to move about. Then, the functions of check if healthy or check if sick are implemented.

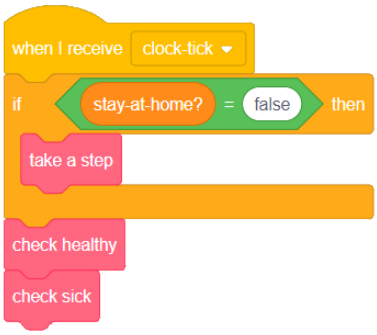

Figure 10. Simple three phase process of a citizen

The left script of Figure 11 shows that when a person is free to move, they will move with a random pattern (take-a-step function.) Free-to-move citizens are susceptible to be infected if in their paths

they get in contact with an infected person. A stay-at-home person also can infect others, or be infected by persons, that happen to visit him in his confinement. These possibilities and their consequences are controlled by the functions check healthy and check sick.

The center script of Figure 11 shows the process by which a healthy person becomes infected by the virus and its consequences. It was described earlier that only healthy persons can be infected in this simulation. That will occur when they get in touch with an infected person (touching color pink.) In that case, the health status of the person will be changed to sick, and its color will change from blue to pink. The period of recovery will then start. Also, the health authorities will receive information of this case (change sick by 1) and this will update the orange graph that represents the current number of infected citizens.

The right script of Figure 11 shows the process followed by the infected persons. Every period of analysis, the total time the person has been sick or in recovery is updated (change time sick by 1.) When the time being sick surpasses the recovery period, the health status of the person is changed to 'recovered'.

This is also represented graphically in the simulation by changing its color from pink to green. In addition, the health authorities will receive information of this case (change sick by -1) and this will update the orange graph that represents the current number of infected citizens.

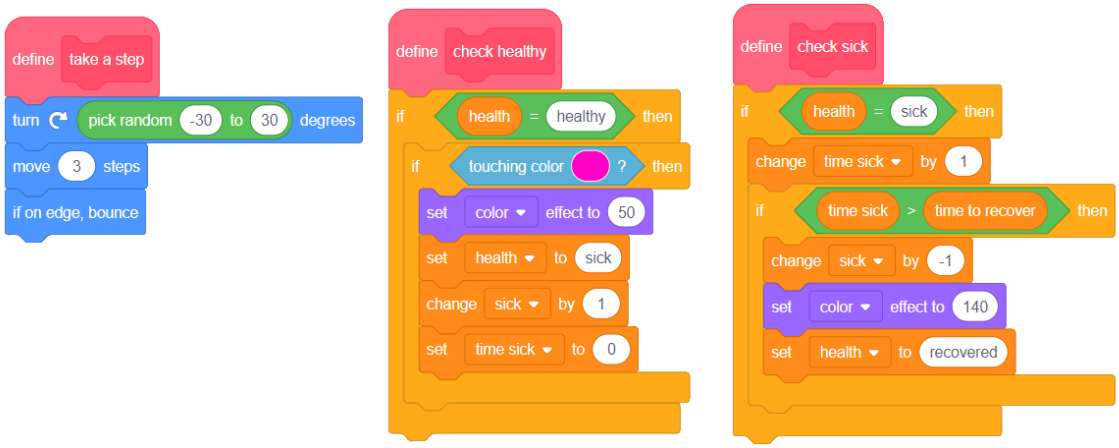

Figure 11. Definitions of the three main phases: take-a-step, check-healthy, check-sick

A feature of this program allows the user to modify the parameter of stay-at-home. The range of 0% (all citizens are free to move) to 100% (all citizens must remain at home) allows the user to see the consequences of the policy by assessing the graph of infected citizens. Figure 12 shows the different patterns of the evolution of the pandemic when the stay-at-home policy values are 20% and 70%.

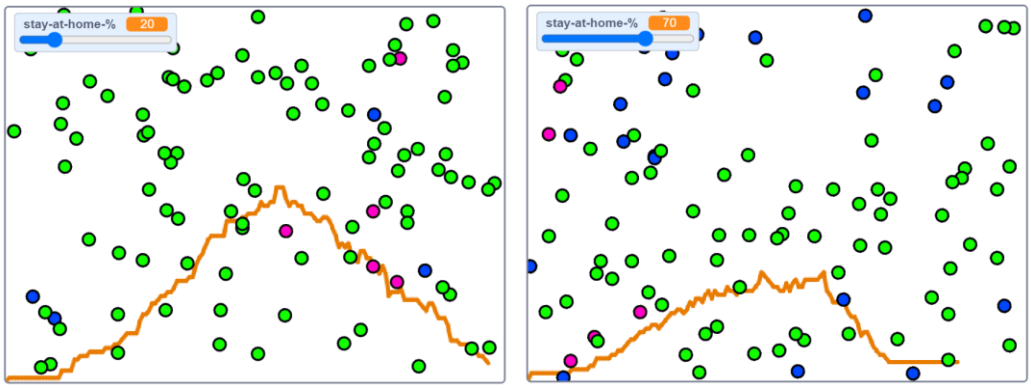

Figure 12. Simulation of the evolution of infected citizens for different of stay-at-home policies

6 - Conclusion

The study of nature and its biological systems has always been a priority endeavor of the human race. The creation of evolutionary biology in the 20th century provided a set of guidelines and paradigms to frame research questions in ways that would lead to successful results of investigation. Computational Theory, and the Computational Model of the Mind provided the tools to study the cognitive and computational processes of living creatures.

In this chapter we have presented two fundamental principles and three concrete examples that illustrate how the formal and biological sciences can collaborate to address old problems with novel approaches that will lead to successful discoveries.

A first conclusion is that, in areas of cognitive and brain research, there is greater possibility of reward in studying the functional nature of a systems than in the study of their internal neural structures. Evolution responds by nature to a paradigm of suboptimal solutions. Evolution constantly moves forward, even if this implies undoing early changes. If the motion of the c-elegans can be described by a simple computational process, which is functionally equivalent to that of the real c-elegans, such computational model should be adopted. The internal details of its implementation have no effect in the study of higher-level systems. An exponential operation can be obtained by many methods that are functionally equivalent. The details on how the system obtains the result have no consequence in a larger system of which it is a part.

A second conclusion of this work is that the computational complexity of natural processes will be determined by the complexity of its computational control, and not by the complexity of the behavior exhibited. This, in turn, suggests that research questions should be framed in terms of the assessed computational complexity of the living system. If a sunflower has little potential for computation, the processes involved in controlling its behavior will be simple, regardless of the overwhelming evidence of its external complexity.

Finally, complex, non-linear systems, with large number of interacting parts, are susceptible to formal study by abstracting the essential elements of each part. Traditional formal sciences were designed with the existing lack of computational power of their time. Having made computational power readily available at a very low cost, current formal sciences address the study of complex systems by focusing on their accurate description, assuming access to almost limitless computational power. The

study of a real pandemic with very large number of components, complex policies, and multiple possible alternatives is only a few levels of complexity above the simple model that was described in this chapter, which was designed to serve as an introduction to these paradigms.

Core Messages:

- The Computational Model of the Mind, and Evolutionary Biology have downgraded century-old mysteries to mere problems.
- The principle of computational equivalency helps us understand how simple systems create complex behaviors.
- The paradigm of suboptimal solutions promotes the study of the functional nature of a system.
- Natural processes should be defined by the complexity of their computational control, and not by the complexity of the behavior exhibited.
- Complex systems are susceptible of formal study by abstracting the essential elements of each part.
- Graphical programming environments allow the description of real-living systems with a computational format that is of easy comprehension by readers, where they can test, manipulate, and verify their ideas.

7 – References

[1] Harvey B, Garcia D, Barnes T, Titterton N, Armendariz D, Segars L, Paley J (2013) Snap!(build your own blocks). In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 759-759)

[2] Kahn KM, Megasari R, Piantari E, & Junaeti E (2018) AI programming by children using snap! block programming in a developing country. In: Vol. 11082. Springer

[3] Resnick M, Maloney J, Monroy-Hernández A, Rusk N, Eastmond E, Brennan K, Kafai Y, (2009) Scratch: programming for all. Communications of the ACM, 52(11), 60-67

[4] Maloney J, Resnick M, Rusk N, Silverman B, Eastmond E (2010) The scratch programming language and environment. ACM Transactions on Computing Education (TOCE), 10(4), 1-15

[5] Morrison J, (2006) Attributes of STEM education: The student, the school, the classroom. TIES (Teaching Institute for Excellence in STEM), 20, 2-7

[6] Byhee B (2010) Advancing STEM education: A 2020 vision. Technology and engineering teacher, 70(1), 30-35

[7] Kim SW, Chung YL, Woo AJ, Lee HJ (2012) Development of a theoretical model for STEAM education. Journal of the Korean Association for Science Education, 32(2), 388-401

[8] Sousa D A, Pilecki T (2013) From STEM to STEAM: Using brain-compatible strategies to integrate the arts. Corwin Press

[9] Olabe JC, Basogain X, Olabe MA (2019) An Ontology of Computational STEAM: The Role of Educational Technology. A Closer Look at Educational, Nova Science Publishers, New York, p 1-27

[10]Turing AM (1936) On computable numbers, with an application to the Entscheidungsproblem. J. of Math, 58(345-363), 5
[11] Shannon C (1948) Claude Shannon. Information Theory, 3, 224
[12] Putnam H (1967) Psychophysical Predicates, in Art, Mind, and Religion*, W. Capitan* and D. Merrill (eds), Pittsburgh: University of Pittsburgh Press. Reprinted in Putnam 1975 as The Nature of Mental States: 429–440
[13] O'Malley MA, Powell A, Davies JF, Calvert J (2008) Knowledge-making distinctions in synthetic biology. BioEssays, 30(1), 57-65
[14] Nordmann A (2015) Synthetic biology at the limits of science. In Synthetic Biology (pp. 31-58). Springer, Cham
[15] Bargmann CI, Hartwieg E, Horvitz HR (1993) Odorant-selective genes and neurons mediate olfaction in C. elegans. Cell, 74(3), 515-527
[16] Gray JM, Hill JJ, Bargmann CI (2005) A circuit for navigation in Caenorhabditis elegans. Proceedings of the National Academy of Sciences, 102(9), 3184-3191
[17] Tian L, Hires SA, Mao T, Huber D, Chiappe ME, Chalasani SH, Bargmann CI (2009) Imaging neural activity in worms, flies and mice with improved GCaMP calcium indicators. Nature methods, 6(12), 875-881
[18] Kimura A, Onami S (2005) Computer simulations and image processing reveal length-dependent pulling force as the primary mechanism for C. elegans male pronuclear migration. Developmental cell, 8(5), 765-775
[19] Korta J, Clark DA, Gabel CV, Mahadevan L, & Samuel AD (2007) Mechanosensation and mechanical load modulate the locomotory gait of swimming C. elegans. Journal of Experimental Biology, 210(13), 2383-2389
[20] Pierce-Shimomura JT, Chen BL, Mun JJ, Ho R, Sarkis R, McIntire SL (2008) Genetic analysis of crawling and swimming locomotory patterns in C. elegans. Proceedings of the National Academy of Sciences, 105(52), 20982-20987
[21] Restif C, Ibáñez-Ventoso C, Vora MM, Guo S, Metaxas D, Driscoll M (2014) CeleST: computer vision software for quantitative analysis of C. elegans swim behavior reveals novel features of locomotion. PLoS Comput Biol, 10(7), e1003702
[22] Li C, Zhang X, Cao Z (2005) Triangular and Fibonacci number patterns driven by stress on core/shell microstructures. Science, 309(5736), 909-911
[23] Cooke, TJ (2006) Do Fibonacci numbers reveal the involvement of geometrical imperatives or biological interactions in phyllotaxis?. Botanical Journal of the Linnean Society, 150(1), 3-24
[24] Swinbank R, James Purser R (2006) Fibonacci grids: A novel approach to global modelling. Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography, 132(619), 1769-1793
[25] Vary A (2016) Fibonacci Sunflowers, Mathematical Objects & Cubic Electrons. Prespacetime Journal, 7(11)
[26] Swinton J, Ochu E, MSI Turing's Sunflower Consortium (2016). Novel Fibonacci and non-Fibonacci structure in the sunflower: results of a citizen science experiment. Royal Society open science, 3(5), 160091
[27]Kahneman D (2011) Thinking, fast and slow. Macmillan
[28] Dennett DC (2013) Intuition pumps and other tools for thinking. WW Norton & Company
[29] Akhtaruzzaman M, Shafie A (2011) Geometrical substantiation of Phi, the golden ratio and the baroque of nature, architecture, design and engineering. International Journal of Arts, 1(1), 1-22

[30] Bejan A (2009) The golden ratio predicted: Vision, cognition and locomotion as a single design in nature. International Journal of Design & Nature and Ecodynamics, 4(2), 97-104
[31] Heyrovská R (2009) The Golden ratio in the creations of Nature arises in the architecture of atoms and ions. In Innovations in Chemical Biology (pp. 133-139). Springer, Dordrecht
[32] Karban R, Black C A, Weinbaum SA (2000) How 17-year cicadas keep track of time. Ecology Letters, 3(4), 253-256
[33] Alexander RD, Moore TE (1962) The evolutionary relationships of 17-year and 13-year cicadas, and three new species (Homoptera, Cicadidae, Magicicada)
[34] Reluga TC (2010) Game theory of social distancing in response to an epidemic. PLoS Comput Biol, 6(5), e1000793
[35] Greenstone M, Nigam V (2020) Does social distancing matter?. University of Chicago, Becker Friedman Institute for Economics Working Paper, (2020-26)
[36] Caley P, Philp DJ, McCracken K (2008) Quantifying social distancing arising from pandemic influenza. Journal of the Royal Society Interface, 5(23), 631-639
[37] Kelso JK, Milne GJ, Kelly H (2009) Simulation suggests that rapid activation of social distancing can arrest epidemic development due to a novel strain of influenza. BMC public health, 9(1), 117
[38] Lewnard JA, Lo NC (2020) Scientific and ethical basis for social-distancing interventions against COVID-19. The Lancet. Infectious diseases, 20(6), 631

Juan Carlos Olabe is professor of Christian Brothers University (CBU), in Memphis, USA. He is doctor engineer of telecommunications by the Polytechnic University of Madrid, and member of the Electrical and Computer Engineering Department at CBU. He has taught courses in Electromagnetic Field Theory, Data Communications, Data Structures, and Data Base Design. His research activities include the areas of: a) computer network design; b) digital design; and c) computational thinking and cognitive processes.

Xabier Basogain is professor of the University of the Basque Country - Euskal Herriko Unibertsitatea. He is doctor engineer of telecommunications by the Polytechnic University of Madrid, and member of the Department of Engineering Systems and Automatics of the School of Engineering of Bilbao, Spain. He has taught courses in digital systems, microprocessors, digital control, modeling and simulation of discrete events, machine learning, and collaborative tools in education. His research activities include the areas of: a) soft computing and cognitive sciences to STEM; b) learning and teaching technologies applied to online education and inclusive education; c) augmented and virtual reality with mobile technologies.

Miguel Ángel Olabe is professor of the University of the Basque Country - Euskal Herriko Unibertsitatea. He is doctor industrial by the University of the Basque Country, and member of the Department of Engineering Communications of the School of Engineering of Bilbao, Spain. He has taught courses in computer structure and architecture, programming, modeling and simulation of networks, multimedia services, and creation and distribution of multimedia content in education. His research activities include the areas of: a) soft computing and cognitive sciences to STEM; and b) learning and teaching technologies applied to online education and inclusive education.