Developing Computational Thinking with WeDo 2.0 Projects

In this chapter, you will discover how you can use WeDo 2.0 to develop computational thinking skills in a science context.

Develop Computational Thinking with LEGO® Education WeDo 2.0 Projects

LEGO® Education is pleased to present these projects, which have been specifically designed for use in elementary school classrooms to develop students' computational thinking skills.

Computational thinking is a set of skills that everybody can use to solve everyday life problems. In WeDo 2.0, these skills are developed throughout each phase of every project. Development opportunities have been identified for you in each of the projects, it is up to you to focus on the ones that are most relevant to you and your students.

Every project in WeDo 2.0 combines the use of the LEGO bricks with an iconic programming language, enabling your students to find solutions to problems while being introduced to programming principles.

WeDo 2.0 develops computational thinking through coding activities, which bring students' creations to life, generating smiles and the desire to discover more.

Computer Science, Computational Thinking, Coding

While the science and engineering fields originated in the early ages of humankind, computer science has a much younger history. Nevertheless, this young discipline has influenced not only the way we approach science and engineering, but also the way we live our lives.

Computer Science is a STEM discipline, sharing attributes with science, technology, engineering, and mathematics.

All STEM disciplines present opportunities to develop a mindset and a lifelong set of practices. Among these practices, we find the ability to ask questions, to design solutions, and to communicate results.

Computational thinking is another one of these practices. It is a way in which we think and it is a way in which everybody can solve problems.

Computational thinking can be described as a group of skills, one of these skills being algorithmic thinking. “Code” or “coding” can be used to describe the action of creating an algorithm.

Coding is therefore one vehicle by which to develop computational thinking within a STEM context.

STEM Disciplines

Science, Technology, Engineering, Mathematics, Computer Science

Develop a mindset and life long set of practices

1. Ask questions and solve problems.
2. Use models.
3. Design prototypes.
4. Investigate.
5. Analyze and interpret data.
6. Use computational thinking.
    a. Decompose
    b. Abstract
    c. Think algorythmically (code)
    d. Evaluate
    e. Generalise
7. Engage in argument from evidence.
8. Obtain, evaluate, and communicate information.

What is Computational thinking ?

The expression “computational thinking” was first used by Seymour Papert, but Professor Jeannette Wing is known to have popularized the idea. She defined computational thinking 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.” (Wing, 2011)

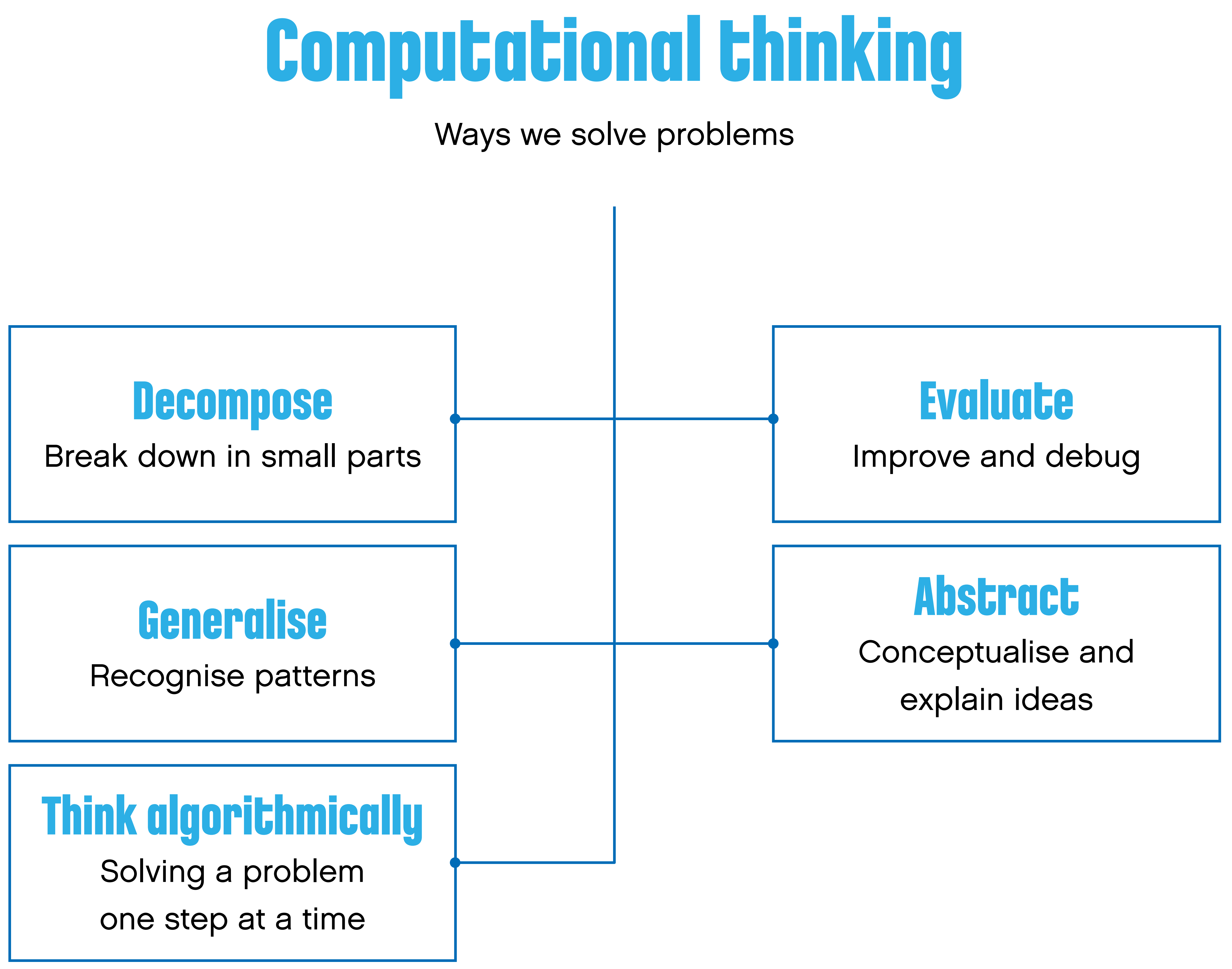
Computational thinking is used in various fields and situations, and we use it in our daily lives. Computational thinking skills are present in science, engineering, and mathematics. These skills can be defined as the following:

Decomposition
Decomposition is the ability to simplify a problem into smaller parts in order to ease the process of finding a solution. By doing so, the problem becomes easier to explain to another person, or to separate into tasks. Decomposition frequently leads to Generalization.

Example: When going on vacation, the preparation (or project) can be separated into subtasks: booking the airfare, reserving a hotel, packing a suitcase, etc.

Generalization (Pattern Recognition)
Generalization is the ability to recognize the parts of a task that are known, or have been seen somewhere else. This frequently leads to easier ways of designing algorithms.

Example: Traffic lights work by repeating the same series of actions forever.



What is computational thinking ?

Algorithmic Thinking

Algorithmic Thinking is the ability to create an ordered series of steps with the purpose of solving a problem.

Example one: when we cook from a recipe, we are following a series of steps in order to prepare a meal.
Example two: when playing with computers, we can code a sequence of actions that tell the computer what to do.

Evaluating or Debugging

This is the ability to verify whether or not a prototype works as intended, and if not, the ability to identify what needs to be improved. It is also the process a computer programmer goes through in order to find and correct mistakes within a program.

Example one: when we're cooking, we will periodically taste the dish to check whether or not it is seasoned correctly.
Example two: when we look for spelling mistakes and missing punctuation in our written work, we are debugging it so that it can be read correctly.

Abstraction

Abstraction is the ability to explain a problem or a solution by removing unimportant details. In other words, being able to conceptualize an idea.

Example: When describing a bicycle, we use only some details to describe it. We might mention its type and color, and add more details for someone who has a real interest in bikes.

A Process for Developing Computational Thinking Skills

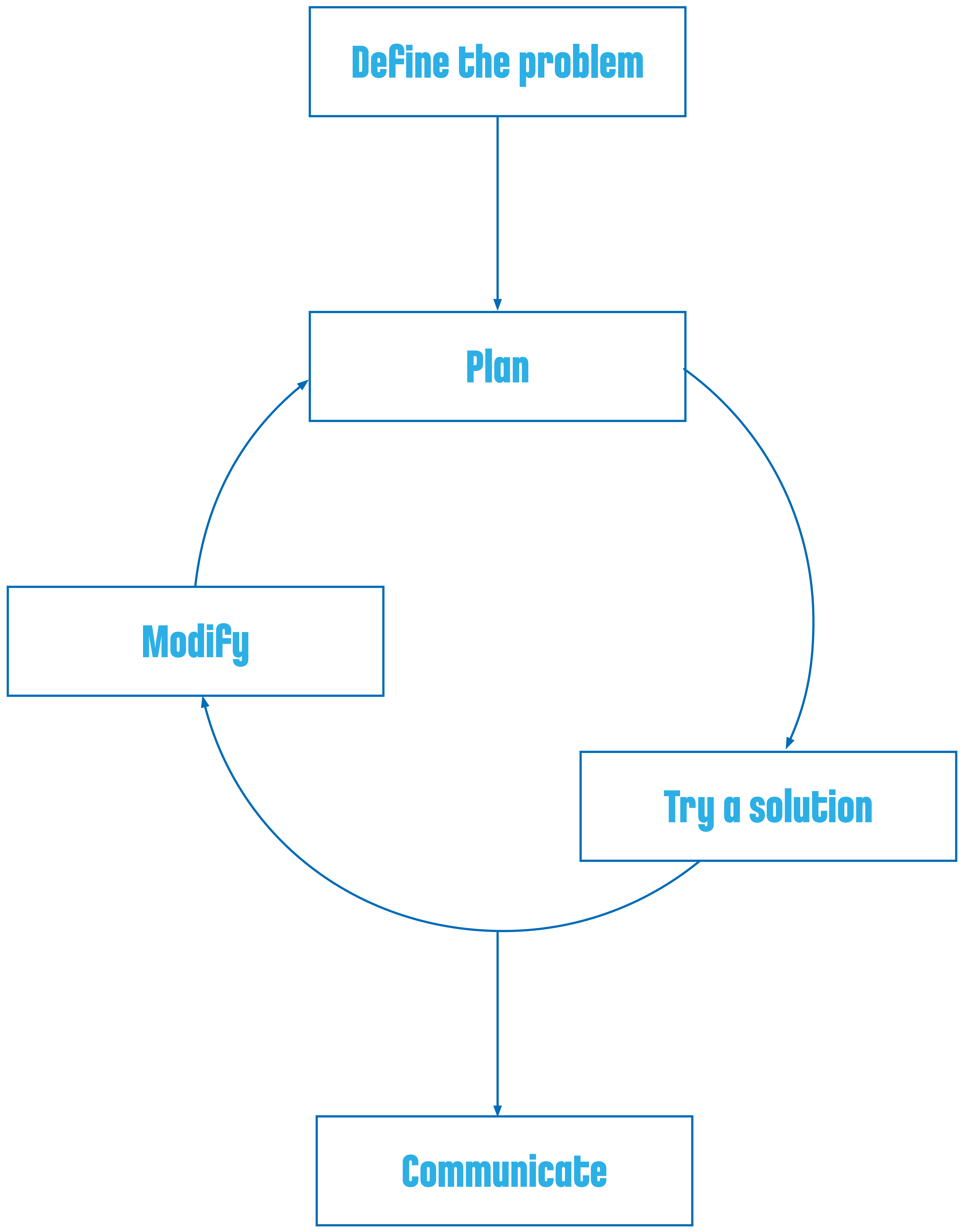
Using an Engineering Design Process
When looking for solutions to a problem, engineers use a design process. They go through a series of phases that guide them toward a solution. During each of these phases, some of their skills are used or developed. It is those skills that we we refer to as "computational thinking skills."

In WeDo 2.0, students follow a similar process:

Defining the Problem
Students are presented with a topic that guides them to a problem or to a situation they wish to improve. Sometimes, a problem can have a lot of details. To make it easier to solve, the problem can be broken down into smaller parts.

By defining the problem in a simple way and by identifying some success criteria, students will develop a skill called "Decomposition."

In other words:
- *Is the student able to explain the problem by themselves?*
- *is the student able to describe how they will evaluate whether or not they were successful in solving the problem?*
- *Is the student able to break down the problem into smaller and more manageable parts?*



A Process For Developing Computational Thinking Skills

Planning
Students should spend some time imagining different solutions to the problem, and then make a detailed plan for executing one of their ideas. They will define the steps they will need to go through in order to reach the solution. By identifying the parts of the task they might have see before, they will develop a skill called “Generalization.”

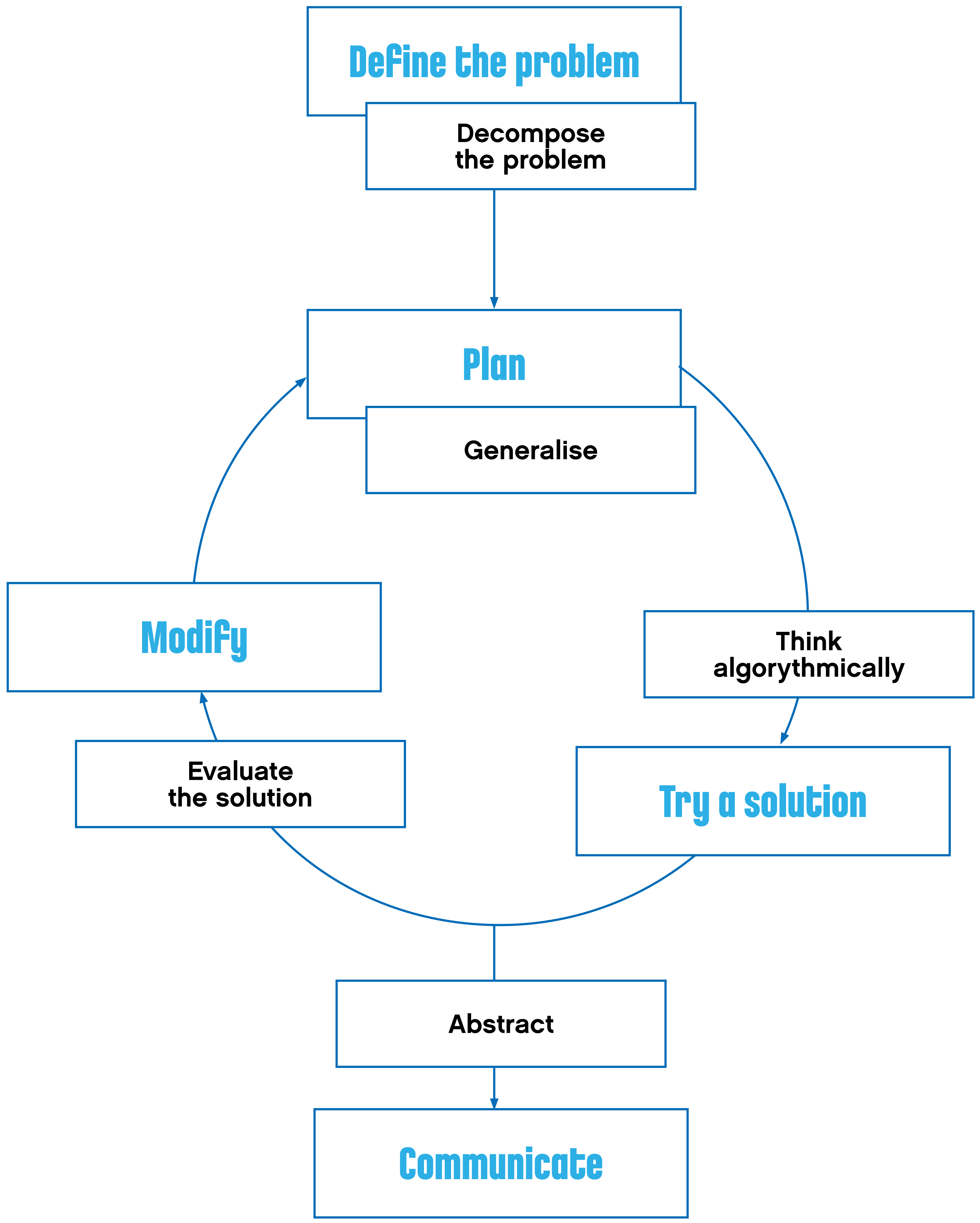
In other words:
- *Is the student able to make a list of actions to program?*
- *Is the student able to identify parts of programs that he or she could use?*
- *Is the student able to reuse parts of programs?*

Trying
Each student is then tasked with creating the final version of their solution. In this phase of the process, they use iconic programing language to activate their LEGO® models. As the students code their ideas, they develop their Algorithmic Thinking skills.

In other words:
- *Is the student able to program a solution to a program?*
- *Is the student able to use sequence, loops, conditional statements, etc.?*

Modifying
Students will evaluate their solution according to whether or not their program and model meet the success criteria. Using their Evaluation skills, they will determine whether they need to change, fix, debug, or improve some part of their program.

In other words:
- *Is the student making iterations of their program?*
- *Is the student fixing problems in their program ?*
- *Is the student able to judge if the solution is linked to the problem ?*



A Process for Developing Computational Thinking Skills

Communicating
Students will present the final version of their solution to the class, explaining how their solution meets the success criteria. By explaining their solution with the right level of detail, they will develop their Abstraction and communication skills.

In other words:
- Is the *student explaining the most important part of their solution?*
- *Is the student giving enough detail to enhance comprehension?*
- *Is the student making sure to explain how their solution meets the success criteria*?

Developing Computational Thinking through Coding

In order to develop their Algorithmic Thinking, students will be introduced to some programming principles. As they develop their solutions, they will organize a series of actions and structures that will bring their models to life.

The most common WeDo 2.0 programming principles students will use are:

1. Output
Output is something that is controlled by the program students are writing. Examples of outputs for WeDo 2.0 are sounds, lights, display, and turning motors on and off.

2. Input
Input is information that a computer or device receives. It can be entered through the use of sensors in the form of a numeric or text value. For example, a sensor that detects or measures something (such as distance) converts that value into a digital input signal so it can be used in a program.

3. Events (Wait for)
Students can tell their program to wait for something to happen before continuing to execute the sequence of actions. Programs can wait for a specific amount of time, or wait for something to be detected by a sensor.

4. Loop
Students can program actions to be repeated either forever, or for a specific length of time.

5. Functions
Functions are a group of actions that are to be used together in specific situations. For example, the group of blocks that could be used to make a light blink would together be called, “the blink function”.

6. Conditions
Conditions are used by students in order to program actions that are to be executed only under certain circumstances. Creating conditions within a program means that some part of the program will never be executed if the condition is never met. For example, if the Tilt Sensor is tilted left, the motor will start, and if the sensor is tilted right, the motor will stop; if the Tilt Sensor never tilts left, the motor will never start and if it never tilts right, then the motor will never stop.