CT-Nord TeaEdu

Scaffolding Computational Thinking –

Enhancing Nordic Teacher Education

Module 1

Challenges faced in Computational Thinking content

Authors: Valentina Dagienė and Eglė Jasutė

Contributors: Vaida Masiulionytė-Dagienė

Reviewers:

Design (icons): Vaidotas Kinčius

Module *outline* is based on the work within the project "Scaffolding Computational Thinking - Enhancing Nordic Teacher Education" 2019 (CT-NordTeaEdu). Coordinator: Vilnius University (Lithuania). Partners: KTH Royal Institute of Technology(Swede),, Tallinn University (Estonia), University of Turku (Finland).

The project "Scaffolding Computational Thinking - Enhancing Nordic Teacher Education" (CT-NordTeaEdu) has received co-funding by the Nordplus Horizontal no. NPHE-2019/10157.

Activity 1.1. Introducing Computational Thinking

Warm-up discussion: 15 min
1.1.1 Discussion on Computational Thinking components: 45 min
Reading task: 30 min
Pair discussion task: 15 min
1.1.2 Characterizing Computational Thinking elements: 45 min
Pair discussion task: 15 min
Conclusion – general Discussion Task: 15 min

Total: 3 hours

Activity 1.2. Introducing Digital Competence Framework

Warm-up discussion: 15 min
1.2.1 Presentation of *DigCompEdu*: 60 min
Pair discussion: 15 min
Group work: 30 min
1.2.2 Self-evaluation of digital competencies: 30 min
1.2.3 Discussion on self-assessment: 30 min

Total: 3 hours

Activity 1.3. Learning activities

1.3.1 Google Ngrams to Computational Thinking and warm-up discussion: 15 min
Group work: 30 min
Discussion: 15 min
1.3.2 Introduction to Bebras challenge: 15 min
Bebras tasks as learning activities: 60 min
1.3.3 Pathfinding in unplugged: 30 min
1.3.4 Pathfinding in Scratch: 45
1.3.5 Instructional principles for CT: 60 min
Discussion: 30 min

Total: 5 hours

Contribution to the learning outcomes

Learning outcomes	Assessment methods
Identifies digital competencies to be developed	Digital competencies weal (https://digital-competence.eu/)
Working in pairs and groups, discusses and analyses areas of digital competencies	Sharing of opinions and thoughts on areas of digital competencies
Working in a group, prepares a model of how teachers' digital competencies can be improved	Presentations of the models for improvement of teacher digital competencies, developed in groups
Develops a lesson plan aimed at development of chosen digital competencies	Presentation of properly structured lesson plan for development of 2-3 digital competences
Critically self-evaluates one's own digital competencies using SELFIE model	Self-reflections on Effective Learning by Fostering the use of Innovative Educational Technologies (SELFIE for teachers)
Be able to describe Computational Thinking as a connecting mechanism between a subject area and Information and Comunication Technologies (ICT)	
Recognize elements of Computational Thinking in scenarios for students' activities	

Activity 1.1: Introduction to Computational Thinking

In this unit you will explore the concept of Computational Thinking and compare various ways to characterize it in terms of problem-solving activities.

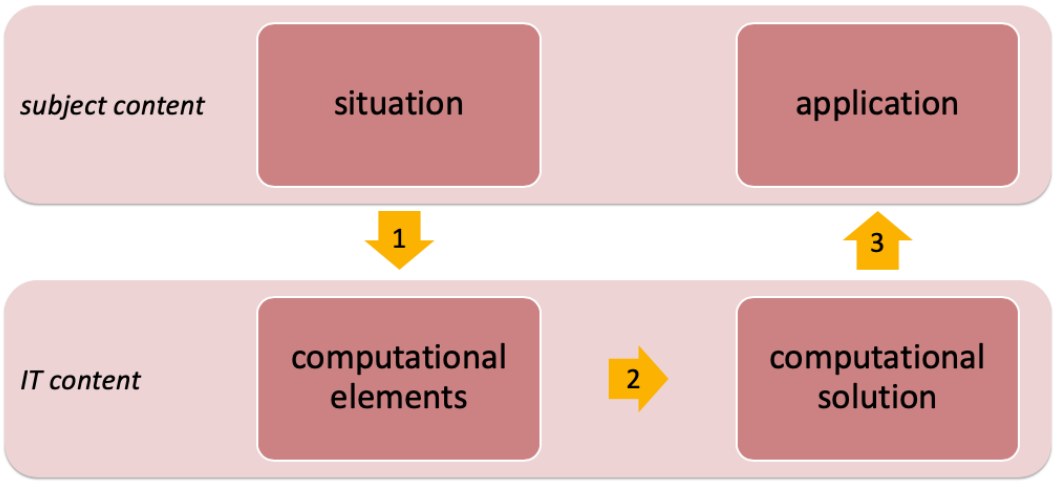
Warm up discussion

To put the full potential of computers to use in a variety of subjects and activities, more digital skills are required than being able to operate a program like text processor or maintain a social media account. Establishing such a connection between subject matter and information technology requires specific problem-solving skills. Those skills are in the domain of Computational Thinking

1.1.1 Discussion on Computational Thinking components

Reading Task

Read the text „A brief introduction to computational thinking“ in Appendix 1. This text distinguishes three main steps in Computational Thinking: the arrows (1), (2) and (3) in the diagram.



Pair Discussion Task

Discuss the two classroom scenarios below, taken from Yadav et al. (2018, p. 381). Which activities would you consider as computational thinking? How do they relate to the steps (1), (2) and (3) in the text in Appendix 1?

Scenario 1:

Westwood Elementary school will start the next school year with a 1:1 iPad initiative. Mr. Nowak has decided to have his 2nd grade students use their iPads to predict weather (temperature, precipitation, and wind) for a week. Each student draws a picture of what they think the weather will look like. Sara, a student, also wanted to keep track of the temperatures that everyone predicted. Mr. Nowak started a Google spreadsheet where each student entered their predicted temperatures. The next day, they recorded the actual weather by using Accuweather App on their iPads and entering the information in the Google sheet. Olivia also wanted to record the actual temperature in Sara's spreadsheet so that they could compare how their predictions compared to what the weather actually was. After a week, they projected the Google spreadsheet on the smartboard and subtracted the differences between the observed and predicted temperatures. Mr. Nowak demonstrated how to make a bar graph of those differences.

Scenario 2:

All the second-grade classes are taking a field trip! The school cafeteria packed PB&J lunches for everyone in identical paper bags, except for Sara and Olivia who have are allergic to peanuts. The lunch paper bags are labelled with all the student names and divided them up into 10 boxes with 10 lunches per box. The lunches were placed in boxes in alphabetical order by last name. Mr. Nowak wants to check to be sure that Sara and Olivia receive peanut-free lunches. They help him search through the boxes. Olivia Velazquez knows that her lunch will probably be near the end, so she looks at the first lunch in each box until she finds one starting with a letter close to the end of the alphabet. When she finds the box that begins with Jemal Summer's lunch, she then looks at the last lunch in that box. It is Billy Wagner's so she knows she must be close! She looks at the lunch right next to Billy's, and it is hers. Happily, she sees that the cafeteria remembered to pack her a cheese sandwich and carrots.

1.1.2. Characterizing Computational Thinking elements

The steps in computational problem solving can be described globally as follows.

1. Decontextualization: translating the problem or question in a subject matter domain ('context') into computational terms;
2. Computational problem solving: constructing an executable solution;
3. (Re)contextualization: translating the solution back to the subject domain.

Definitions of Computational Thinking vary in the way they emphasize the activities carried out in the steps (1), (2) and (3). Selby and Woollard (2013, p. 5), for example, describe computational thinking as "an activity, often product oriented, associated with, but not limited to, problem solving. It is a cognitive or thought process that reflects

- the ability to think in abstractions,
- the ability to think in terms of decomposition,
- the ability to think algorithmically,
- the ability to think in terms of evaluations,
- the ability to think in generalizations."

Although there is some overlap, we can globally associate the above elements to the steps in our model. The first two elements mainly have to do with Step (1): analyzing patterns within a problem or situation (abstraction), and breaking up a problem into smaller subproblems (decomposition). Algorithmic thinking is used mostly within Step (2), whereas evaluating a solution and investigating how it can be generalized connects the computational solution to the subject matter domain, which takes place in Step (3). We can summarize this in a table:

	(1)	**(2)**	**(3)**
Selby & Woollard (2013)	Abstraction decomposition	algorithmic thinking	Evaluation generalization

Pair Discussion Task

You will find three other well-known operationalizations below. Extend the above table with three more rows. Categorize the elements you find in the three columns. Do you recognize any elements in the classroom scenarios 1 and 2?

Definitions of computational thinking

Selby & Woollard (2013, p. 5)

Computational thinking is an activity, often product oriented, associated with, but not limited to, problem solving. It is a cognitive or thought process that reflects

- the ability to think in abstractions,
- the ability to think in terms of decomposition,
- the ability to think algorithmically,
- the ability to think in terms of evaluations,
- the ability to think in generalizations.

Atmatzidou & Demetriadis (2016, p. 664)

CT skills	Description	Student skills (The student should be able to…)
Abstraction	Abstraction is the process of creating something simple from something complicated, by leaving out the irrelevant details, finding the relevant patterns, and separating ideas from tangible details [52]. Wing [2] argues that the essence of CT is abstraction.	1. Separate the important from the redundant information. 2. Analyse and specify common behaviours or programming structures between different scripts. 3. Identify abstractions between different programming environments.
Generalisation	Generalisation is transferring a problem-solving process to a wide variety of problems [38].	Expand an existing solution in a given problem to cover more possibilities/cases.
Algorithm	Algorithm is a practice of writing step-by-step specific and explicit instructions for carrying out a process. Kazimoglu et al. [37] argue that selection of appropriate algorithmic techniques is a crucial part of CT.	1. Explicitly state the algorithm steps. 2. Identify different effective algorithms for a given problem. 3. Find the most efficient algorithm.
Modularity	Modularity is the development of autonomous processes that encapsulate a set of often used commands performing a specific function and might be used in the same or different problems [38].	Develop autonomous code sections for use in the same or different problems.
Decomposition	Decomposition is the process of breaking down problems into smaller parts that may be more easily solved. Wing [2] argues that CT is using decomposition when attacking or designing a large complex task.	Break down a problem into smaller/simpler parts that are easier to manage.

CSTA (2011, p. 7—9)

CT is a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them
- Logically organizing and analyzing data
- Representing data through abstractions such as models and simulations
- Automating solutions through algorithmic thinking (a series of ordered steps)
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem-solving process to a wide variety of problems

CT vocabulary:

- data collection
- data analysis
- data representation
- problem decomposition
- abstraction
- algorithms & procedures
- automation
- simulation
- parallelization

These skills are supported and enhanced by a number of dispositions or attitudes that are essential dimensions of CT. These dispositions or attitudes include:

- Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open-ended problems
- The ability to communicate and work with others to achieve a common goal or solution

Grover & Pea (2018, p. 23)

- CT-concepts
  - logic and logical thinking
  - algorithms and algorithmic thinking

  - patterns and pattern recognition
  - abstraction and generalization
  - evaluation
  - automation
- CT-practices
  - decomposition
  - making digital artifacts
  - testing and debugging
  - stepwise refinement (incremental development)
  - collaboration and creativity (connection to 21st century skills)

P. J. Denning and M. Tedre (2019)

- Compuational thinking is the mental skills and practices for
  - designing computations that get computers to do jobs for us, and
  - explaining and interpreting the world as a complex of information processes.

The design aspect reflects the engineering tradition of computing in which people build methods and machines to help other people. The explanation aspect reflects the science tradition of computing in which people seek to understand how computation works and how it shows up in the world. Design features immersion in the community being helped, explanation features being a dispassionate external observer.

The development of computational thinking opened six important dimessions that are characteristic of CT today.

- ***Methods***. Mathematicians and engineerings developed methods for computing and reasoning that non-experts could put to work simply by following directions.
- ***Machines***. Inventors looked for machines to automate computational procedures for the purpose of greater speed of calculation and reduction of human errors in carrying out computations.
- ***Computing Education***. University educators formed computer science to study and codify computation and its ways of thinking and practicing for institutions, business, science, and engineering.
- ***Software Engineering***. Software developers formed software engineering to overcome rampant problems with errors and unreliability in software, especially large software systems such as major applications and operating systems.
- ***Design***. Designers bring sensibilities and responsiveness to concerns, interests, practices, and history in user communities.
- ***Computational Science***. Scientists formed computational science to bring computing into science, not only to support the traditions of theory and experiment, but also to offer revolutionary new ways of interpreting natural processes and conducting scientific investigations.

Conclusion – general Discussion Task

Compare your findings with fellow students in class. Together, construct a short working definition of Computational Thinking in terms of steps and activities.

Learning resources

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. Robotics and Autonomous Systems, 75, 661–670.

Christenson, J.A., (1984). Gemeinschaft and gesellschaft: Testing the spatial and communal hypotheses. Social Forces, 63(1), pp.160-168.

CSTA. (2011). Computational thinking teacher resources, second edition. https://id.iste.org/docs/ct-documents/ct-teacher-resources_2ed-pdf.pdf

Dagienė, V., & Sentance, S. (2016). It's Computational Thinking! Bebras tasks in the curriculum. In International conference on informatics in schools: Situation, evolution, and perspectives (pp. 28–39).

Denning, P. J., Tedre, M (2019). Computational Thinking. The MIT press, Cambridge.

Grover, S., & Pea, R. (2018). Computational Thinking: A competency whose time has come. In S. Sentance, E. Barendsen and C. Schulte (Eds.), Computer science education: Perspectives on teaching and learning in school, Bloomsbury (pp. 19-38).

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.

Selby, C. & Woollard, J. (2013) Computational thinking: the developing definition, available via internet: http://eprints.soton.ac.uk/356481, accessed: 10 February 2021

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: measuring teacher understanding of computational ideas for teaching science. Computer Science Education, 28(4), 371–400.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.

Activity 1.2. Introducing the Digital Competence Framework

Aims of this activity: to introduce to digital competencies and analyse the DigCompEdu framework, to help students to get acquainted with the Self-reflection on Effective Learning by Fostering the use of Innovative Educational Technologies (SELFIE)

Warm up discussion

Lecturer uses prepared slides with video record, discussion question and possible overview of discussion.

Video introduction: 4th industrial revolution

https://www.youtube.com/watch?v=uvP4DnH1URg (CC?)

Ask students to discuss in groups of 3–4 about their understandings of digital competence: What are digital competences?

Students should provide concrete examples and discuss features.

Theoretical background: 4th industrial revolution and digital literacy. Definitions of important concepts in the European Digital Competence Framework for Educators (DigCompEdu). Areas of digital competencies. Ways of improving digital competencies. Assessment and self-assessment (using SELFIE - Self-reflection on Effective Learning by Fostering the use of Innovative Educational Technologies).

1.2.1. Presentation of DigCompEdu framework

Lecturer's presentation of the *DigCompEdu* framework is combined with discussion.

Overview of presentation

Digital competence involves the confident and critical use of electronic media for work, leisure, and communication. These competencies are related:

- to logical and critical thinking,
- high-level information management skills, and
- Well-developed communication skills.

Fig. 1. Digital competencies

As the teaching professions face rapidly changing demands, educators require an increasingly broad and more sophisticated set of competencies than before. In particular the ubiquity of digital devices

and the duty to help students become digitally competent requires educators to develop their own digital competence. On international and national level a number of frameworks, self-assessment tools and training programmes have been developed to describe the facets of digital competence for educators and to help them assess their competence, identify their training needs and offer targeted training.

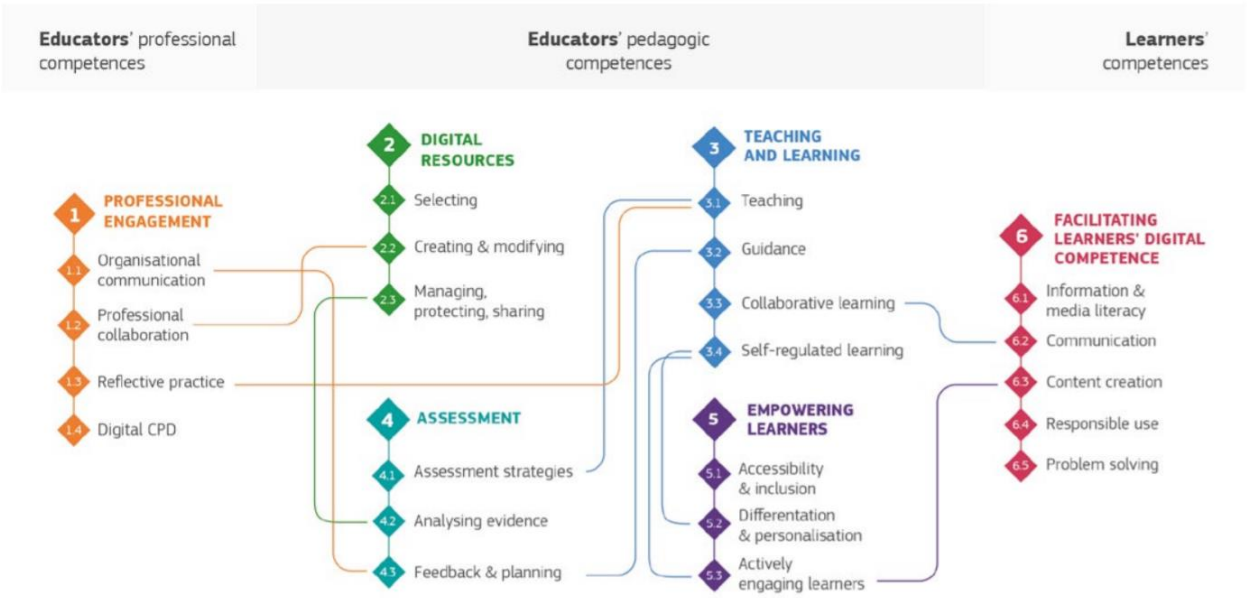


Fig. 2. Areas of digital competencies for educators

Each area with competencies, progression model, proficiency levels have to be described and presented in slides. The description can be taken from DigCompEdu document: https://ec.europa.eu/jrc/en/search/site/digcompedu?page=2

Pair Discussion

In what ways could you improve your digital competencies?

Group work

Each group gets worksheet with one area of digital competencies and have to think what competencies teacher improves directly in everyday life and which have to be strengthened. Students must think how these competencies can be developed. The presentation of group work have to be prepared and presented to all participants

1.2.2. Self-evaluation of digital competencies

There is a wide variety of self-assessment tools and curricula, both internationally and nationally, to describe various aspects of educators' digital competence, as well as to help them assess their competences, identify training needs and offer targeted training. This publication presents the Common European Digital Literacy System for Educators (DigCompEdu), which is based on the analysis and clustering of these tools.

DigCompEdu is a science-based system that assists in policy making and can be directly applied to regional and national measures and training programs. It also offers a common language and approach to facilitate dialogue and exchange of good practice across borders.

Video about SELFIE rating system

https://www.youtube.com/watch?v=8_6hVoYXCAI

SELFIE (Self-reflection on Effective Learning by Fostering the use of Innovative Educational Technologies) is a tool designed to help schools embed digital technologies into teaching, learning and student assessment. It can highlight what's working well, where improvement is needed and what the priorities should be. The tool is currently available in the 24 official languages of the European Union with more languages to be added over time.

SELFIE gathers – anonymously – the views of students, teachers and school leaders on how technology is used in their school. This is done using short statements and questions and a simple 1-5 agreement scale. The statements cover areas such as leadership, infrastructure, teacher training and students' digital competence.

The assessment takes around 30 minutes. Questions are tailored to each group. For example, students get questions relating to their learning experience, teachers reflect on training and teaching practices and school leaders address planning and overall strategy.

Based on this input, the tool generates a report – a snapshot ('SELFIE' :-)) of a school's strengths and weaknesses in their use of digital technologies for teaching and learning. The more people in the school taking part, the more accurate the SELFIE of their school will be.

The results and insights from the SELFIE exercise are for your school only and are not shared unless you choose to do so.

The findings can help you see where you are at and, from there, start a conversation on technology use and develop an action plan for your school. SELFIE can then be used at a later stage to gauge progress and adapt the action plan.

Benefits:

- SELFIE involves the whole school community – school leaders, teachers and students – in a 360-degree process covering many areas of school practice.
- Because every school is unique, the tool can be customised. Your school can select and add questions and statements to suit your needs.
- SELFIE allows all participants to answer questions that match their experience, as students, teachers or school leaders.
- SELFIE is free of charge. Answers are anonymised and data is secure.
- You can take the assessment from a computer, tablet or smartphone.
- On completing SELFIE, each school receives a tailor-made, interactive report which provides both in-depth data and quick insights into strengths and weaknesses.

1.2.3. Discussion on self-assessment

In the discussion, give your opinion on how meaningful the self-assessment tools for personal development of a teacher are? How much do they benefit the body? Specify the positive and negative aspects specifically. Suggest a way for the teacher to evaluate his / her digital competences in order to benefit and encourage them to develop.

Advisor to the lecturer: If possible, the learners are divided into groups of 4 and answers the discussion questions. Provides his insights in a general discussion. At the end, the lecturer summarizes the discussion, providing summaries of the learners. Learners could use any of the concept map technology mentioned above.

Learning resources

Digital Competence Framework for Educators (DigCompEdu).
https://ec.europa.eu/jrc/en/publication/eur-scientific-and-technical-research-reports/european-framework-digital-competence-educators-digcompedu

This report presents a common European Framework for the Digital Competence of Educators (DigCompEdu). DigCompEdu is a scientifically sound background framework which helps to guide policy and can be directly adapted to implement regional and national tools and training programmes.

Falloon G. (2020). From digital literacy to digital competence: the teacher digital competency (TDC) framework. Educational Technology Research and Development. https://link.springer.com/article/10.1007/s11423-020-09767-4

Oberländer M., Beinicke A., Bipp T. (2020). Digital competencies: A review of the literature and applications in the workplace. Computers & Education, Volume 146.

https://ec.europa.eu/education/schools-go-digital/how-selfie-works_en
SELFIE https://ec.europa.eu/education/schools-go-digital_en

Activity 1.3. Learning activities

The aim of this activity is to introduce some learning activities to improve students computational thinking. There are presenteted Ngram, unplugged activities and Bebras tasks.

1.3.1 Google Ngrams to improve Computational Thinking

The Google Ngram viewer is a graphing-tool that depicts word frequencies from a large corpus of books, visualising a term's or a phrase's use over time. The tool uses literature sources printed between 1500 and 2019 in American English, British English, French, German, Italian, Spanish, Russian, Hebrew and Chinese, and thereby, it can be employed to investigate changes in language over the years. Fluctuations in language use can depict social-cultural changes and thus, provide an understanding of how different social-cultural transformations evolve through history. In this activity, we are going to use Google Ngrams to examine cultural and social changes through time as they are reflected by the use of specific terms in books.

Warm-up discussion

Before you start working in this activity, click on the following link to access the Google Ngram viewer (https://books.google.com/ngrams). You will notice that the tool already suggests an example that is depicted in the following figure. In this example, we see the results for three phrases, "information technology", "computer science", and "informatics". The tool identifies the frequency of these phrases as they appear in books between 1919-2019. As you can see, all terms have been reported in the literature since the 1960s. The "information technology" was mentioned more times in 2000s comparing with „computer science" and „informatics"

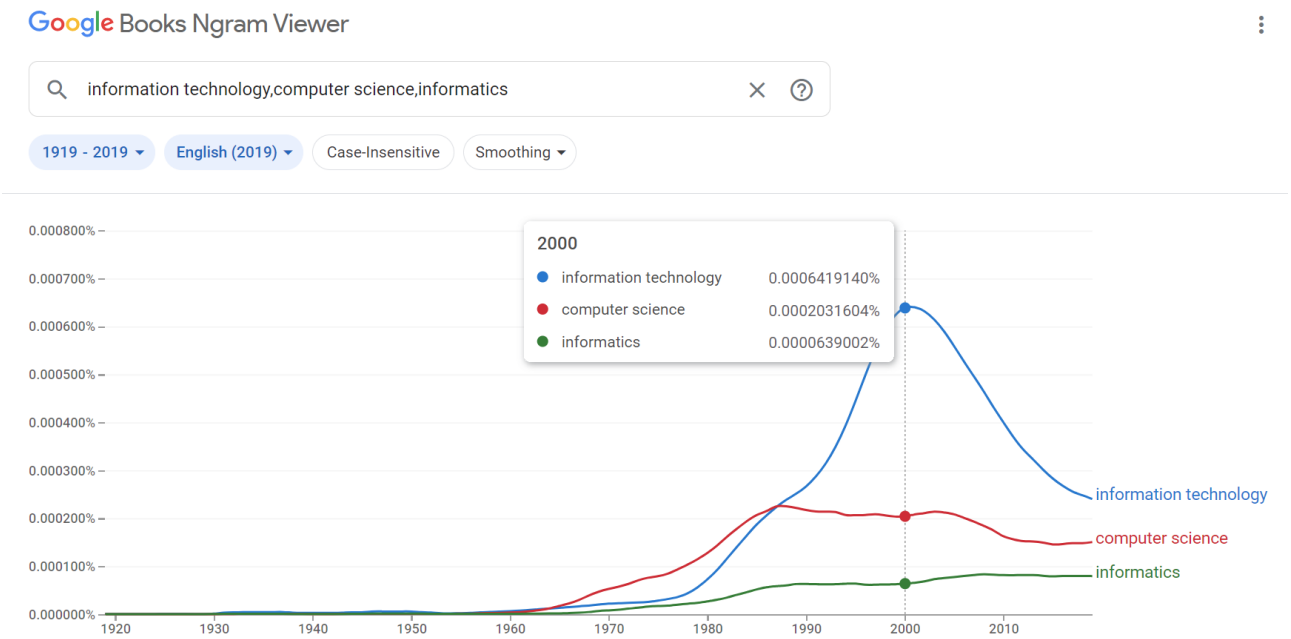


Fig. 3. Google Ngram Example for the terms: "information technology", "computer science", and "informatics".

What other trends do you notice regarding the frequency of the terms in figure 1?

Group Task

Click on the following link to access Google Ngrams (https://books.google.com/ngrams).

1. Use the search field (erase any search terms already in the field) and type the following two words separated by a comma: Duty, Decision.
2. From the drop-down menu «from the corpus» select American English (2012) and then click the button «search lots of books».
3. What do you notice about the frequency of these two terms over time?
4. In the same way, explore the following frequency of the following words (select at least two):
    1. Give and Get,
    2. Benevolence and Acquisition,
    3. Act and Feel,
    4. Deed and Emotion
5. How do the frequencies of these words fluctuate?

Discussion

Compare your findings in class. Which aspects did you find easy, which ones were difficult? What did you learn?

Resources

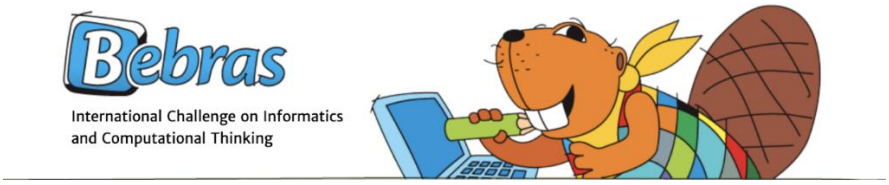
Christenson, J.A., (1984). Gemeinschaft and gesellschaft: Testing the spatial and communal hypotheses. *Social Forces*, 63(1), pp.160-168.

Greenfield, P.M., (2013). The changing psychology of culture from 1800 through 2000. *Psychological Science*, 24(9), pp.1722-1731.

1.3.1. Introduction to Bebras challenge

Bebras is an online challenge an informatics and Computational Thonking.

The elements of the challenge are so-called Bebras tasks, each covering one or more computational concepts. In this activity you will explore Bebras tasks both as units of the contest and as ‘unplugged’ learning activities for computational thinking.



Pair Task

- Go to the local Bebras website www.bebras.org and select up to five example Bebras tasks (try to include some variation).
- Carry out the tasks individually and then compare and discuss them with your partner.
- Can you classify the tasks in terms of the computational thinking steps and activities you learned in this module?

1.3.2. Bebras Tasks as learning activities

Bebras tasks are categorized in terms of the computational concepts being covered, each task carries an explanation of the connection between the task and computer science.

Valentina Dagienė (the 'mother of Bebras') and Sue Sentance investigated the use of Bebras tasks as CT learning activities:

Pair Task

• Read the article.

• Review the concepts and activities you encountered in units 2 and 3. For each of these modules, find an example of a computational concept and a matching Bebras task.

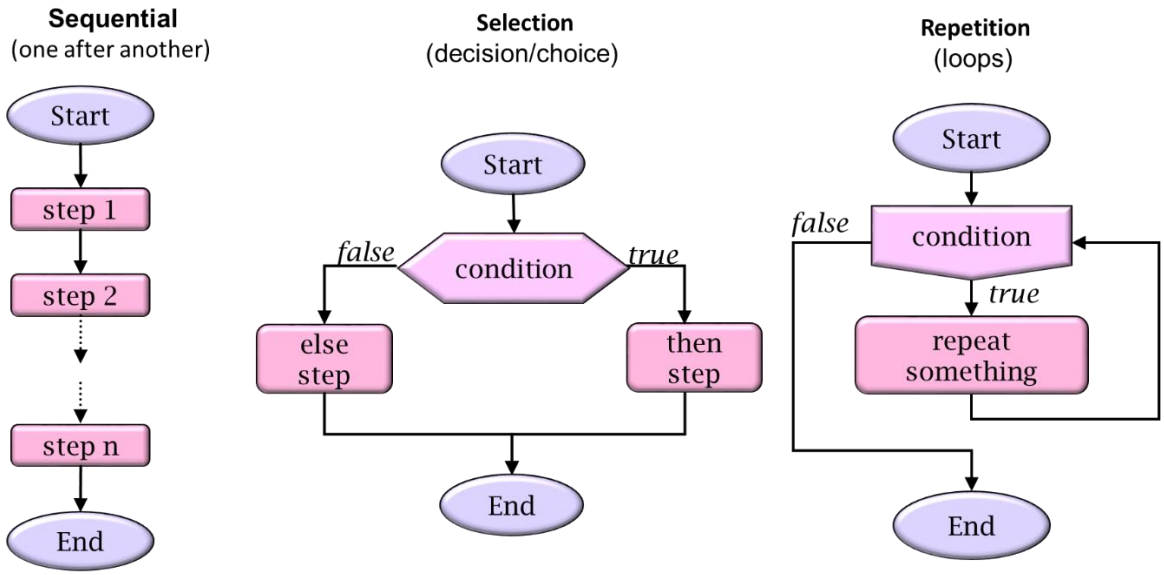
Pathfinding: paths in a maze

Maze solving is the process of finding a path through the maze from the start to finish. Some maze solving methods are designed to be used inside the maze by a traveller with no prior knowledge of the maze, whereas others are designed to be used by a person (or computer program) that can see the whole maze at once. Maze solving is a variant of a more practical class of problems that is known under the name of pathfinding. Given a number of locations that may be interconnected (for example, a collection of cities that are connected by roads), a pathfinding algorithm searches for a route that connects these locations with the intention of determining the best route (e.g. the shortest or cheapest route).

Our intention is to use these pathfinding problems to gain first insight into problem analysis and algorithm design without detailed knowledge of computer programming and programming languages. However, in order to be able to formulate possible solutions with sufficient precision, we will first use flow charts again.

Flowchart elements

Below you see the three basic building blocks with which flowchart can be composed.



Each flow chart is built up using the following three basic elements

1. **Sequence**: an ordered series of instructions that are executed one after the other.
2. **Selection**: it is determined on the basis of a conditional expression whether the then-branch (the branch with the label true) or the else-branch (the branch with the label false) should be executed.
3. **Repetition**: the body of the element (the branch with label true) is repeated as long as the condition evaluates to true. Then the program will continue by following the branch labeled false.

Travelling salesman problem

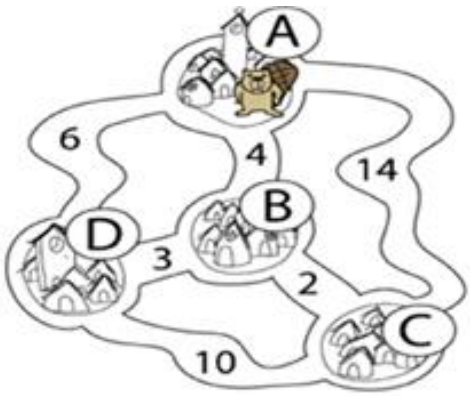
The so-called traveling salesman problem is a classic example of a task in which an optimal solution is sought.

"Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?"

We are now going to look at a solution to this problem that, while not always finding the best route, is intuitive and easy to understand, namely the *nearest neighbor algorithm*. The nearest neighbour algorithm (NNA) lets the salesman choose the nearest unvisited city as his next move.

Pair Task

Beavers swim from city A, they go up visit at city B, C and D and then come back in A. They want to find the shortest route.



1. Which route do they find if they would use NNA?
2. Does NNA always give the right answer (i.e. for any list of cities and connections between those)? If not, give a concrete counter-example

Although the NA is intuitively clear, it can be difficult to formulate this algorithm precisely. We try to express this algorithm in a flowchart where we first have to determine which primitive instructions we can use for this. These primitives must be powerful enough on the one hand, but also sufficiently abstract on the other, so that we do not get caught in all kinds of details.

Pair Task

Specify the NNA in a flowchart.

Hint: Think first about the basic instructions/primitives that you are allowed to use.

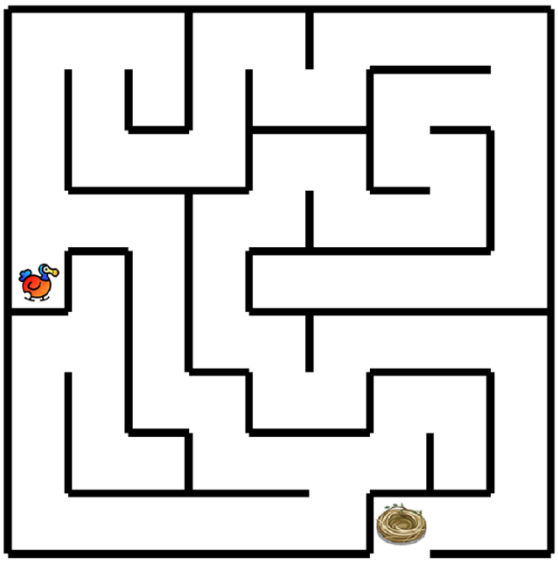
Finding a path in a maze

Finding a path in a maze is another example of a task that is easy to understand, but also challenging enough to solve. There are several variants of this problem: find a path out of a maze, find a path through a maze or find a path to a specific location inside a maze. First, we can note that these variants can be generalized to: find a path from position A to position B.

It is also important what we know about the maze while we search for the goal position B. In this assignment, we assume that we know nothing about the size and structure of the maze. We can walk through the corridors of the maze, looking one step ahead at a time. We therefore assume that at any point in the maze we can check whether we can walk straight (and therefore not stand in front of a wall) and possibly adjust our direction of movement.

The best-known rule for traversing a maze is the wall follower (also known as either the left-hand rule or the right-hand rule). If all walls of a maze are connected together then by keeping one hand in contact with one wall of the maze the traveler is guaranteed not to get lost and will reach the exit.

To make the problem more concrete, we use the following scenario where the maze traveler is represented by a dodo who is somewhere in the maze trying to find his nest hidden elsewhere.



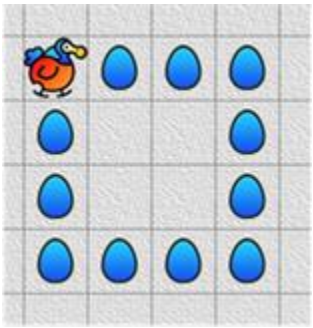
Before we can specify a search algorithm, we must again indicate with which primitive instructions we can control the dodo.

We distinguish between *commands* (to have the dodo take a step forward or change the direction of movement) and *queries* (with which we can have the dodo provide local information about its surroundings).

- Commands:
  - *turn left*: turn 90 degrees counterclockwise
  - *turn right*: turn 90 degrees clockwise
  - *move*: move one square ahead
  - *lay egg*: lay an egg (at the current location)
- Queries/tests:
  - *can move?* can you take a step forward?
  - *nest found?* have you found the nest?

Pair Task

Use a flowchart to show an algorithm with which the dodo produces the following egg pattern. Hint: use repetitions.



Pair Task

Specify the follow-the-left-wall strategy in a flowchart.

1.3.3. Pathfinding unplugged

Preparation

For example, using painter's tape, stake out a maze on the floor. Make sure the corridors are wide enough for a person to walk through easily. The maze may be a bit simpler than the maze in the picture. Exchange the flowcharts from the previous assignment. Designate a place in the maze as the final destination and one of you will be somewhere in the maze far enough away from this final destination.

Pair Task

One of you will be standing somewhere in the maze far enough away from the final destination.

- The other person reads the instructions from the flowchart and the one in the maze follows these instructions meticulously.
- Is the final destination reached? If so, also try out whether this applies to other starting locations in the maze. If not, what is wrong with the algorithm? Improve the algorithm and check whether the improvement has the desired effect.
- Return the improved version of the flowchart to their owners.

1.3.4. Pathfinding in Scratch

Finally, you will implement the algorithm from the previous assignment in Scratch. We have created an initial scenario that you can use for this. Open the start scenario PathThroughMaze.sb3.

A brief explanation now follows. The primitive dodo instructions have been added to the scenario as separate blocks. The *layEgg* instruction is missing because it is not needed for this task. All other commands work as previously indicated. The queries have been realized in a special way, also because self-defined blocks in Scratch cannot yield any results. The test to check whether the dodo has found the nest can easily be realized with a predefined block from the sensing palette. We introduced a new block for the *canMove* test. To return a result, we use a variable that we, like the block itself, called *canMove* as well. So after you execute the *canMove* block you can inspect the value of the corresponding variable to see if the dodo can take a step or not. The variable *canMove* contains either the value 0 or 1. The value 0 indicates that no step can be taken while value 1 indicates that it is possible.

1.3.5. Instructional principles for Computatinal Thinking

In a review article, Lye and Koh (2014) analyze 27 empirical studies on Computational Thinking, in particular studies on classroom activities involving programming. Among other things, Lye and Koh identify instructional approaches fostering computational thinking.

Lye and Koh grouped the approaches into four categories, which often appear in combinations: reinforcement of computational concepts, reflection, information processing, and constructing programs with scaffold.

Pair Task

Discuss your experiences in the Pathfinding activity:

- What are the essential differences between the ‘unplugged’ and the ‘plugged’ variant?
- What challenges for your future students do you expect for each of these?

Pair Task

1. Read the summary on the four categories of instructional strategies.
2. Analyze the computational thinking activities you carried out yourself in this module. Which of the strategies do you recognize? Discuss.

Pair Task

1. Read the summary on assessment of computational thinking.
2. Discuss which would be suitable assessment approaches for the activities you carried out in Unit 2 or Unit 3. Select one possible approach for each activity. How would you use these approaches to evaluate students’ computational thinking skills?

Discussion

Compare your findings in class. Which aspects did you find easy, which ones were difficult? What did you learn?

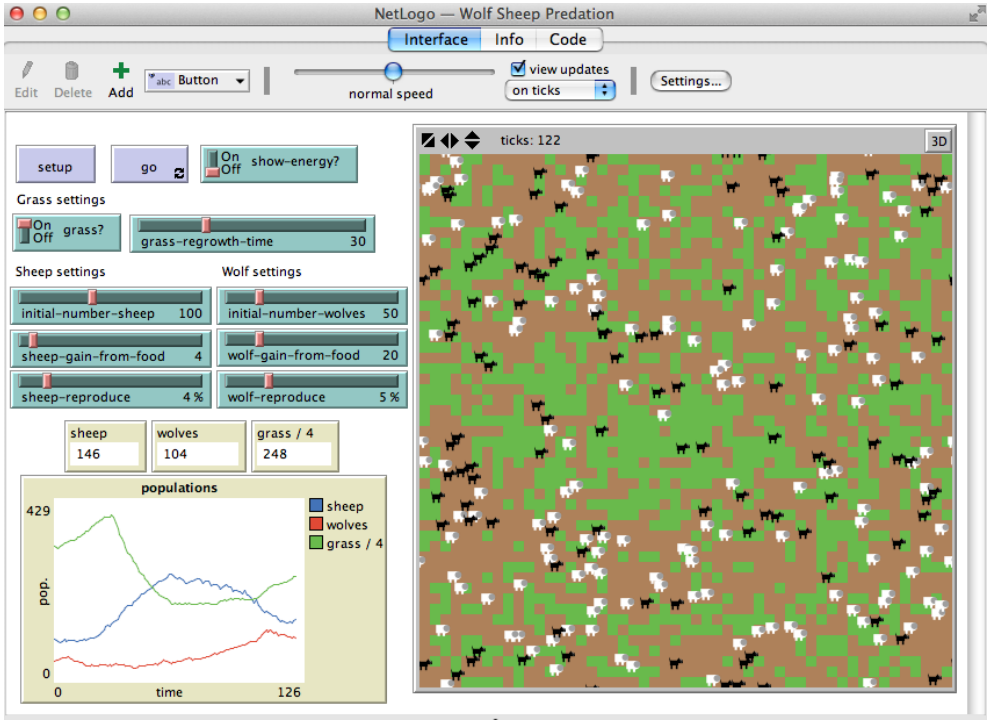
Appendix 1.

A brief introduction to computational thinking

Many researchers can't do their work without the help of computer models. Think of climate researchers making predictions about sea level rise or scientists trying to unravel the mechanisms behind cancer. Also in other professional practices the computer has become indispensable. Computers have fundamentally changed our way of working and learning.

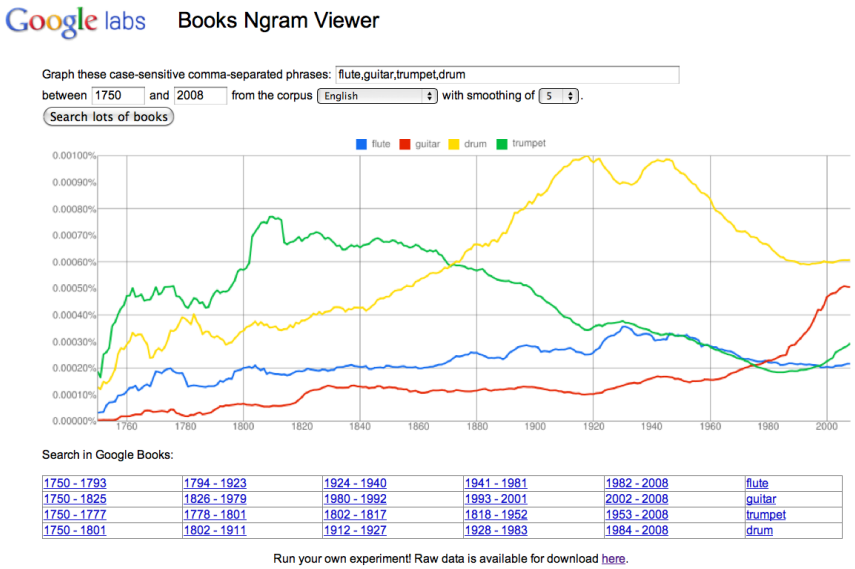
However, we do not always put the potential of the computer to full use. Indeed, the skills required for this go beyond using a program such as Word or managing a Facebook page. On the other hand, you won't have to become an IT specialist. However, some basic concepts of computer science will be useful to make the necessary connection between IT and another field. And in establishing this connection, so-called *computational thinking* plays a prominent role.

Take, for example, a simulation of an ecosystem with sheep and wolves. This allows us to make predictions about populations and gain insight into biological mechanisms. The image below shows a screenshot of NetLogo, a program in which such a simulation can be realised. In order to construct the simulation, some programming skills are needed. Furthermore, one needs to know, or find out, what important aspects should be included in the model.

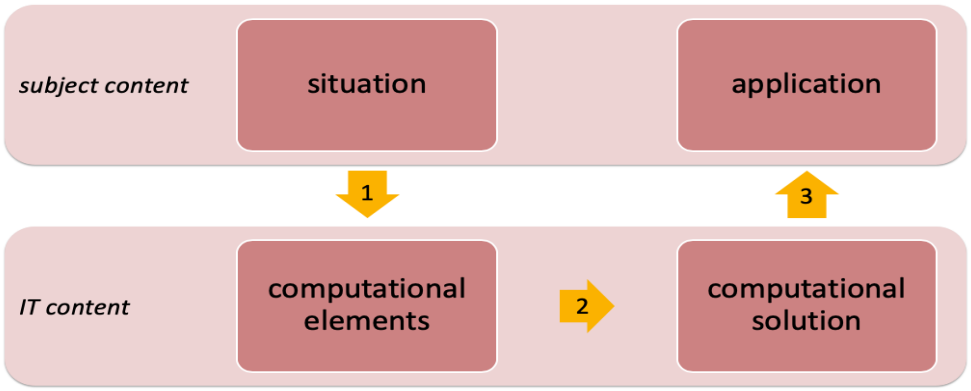


Simulations have proved to be useful in all kinds of domains: to predict traffic jams, to analyze consumer behavior, to run profit forecasts, etc.

But programming is not always necessary. An example of using standard digital tools is the use of analysis tools such as Google Ngram viewer. With this online application, we can analyze the occurrence of words in written sources, which has great potential in literature or history classes.



These examples involve *computational thinking*, a skill in which knowledge of specific subject matter and knowledge from computer science come together. Computational thinking can be seen as a form of *problem solving* using concepts stemming from computer science. The three arrows in the diagram below illustrate this.



Computational thinking involves (1) translating a problem or question into terms that prepare for finding a computational solution, such as information or data, and processes or algorithms. Then one can (2) construct a solution with the help of existing applications or with self-made algorithms and programs, and then (3) translate the solution back into the specific subject area.
As said before, the way in which the computer is used in the process can vary: from using existing apps to constructing (or adjusting) a computer program.

Sometimes it is easy to identify the data to work on (in Step 1), but it is difficult to extract get meaningful information from those. Suppose a journalist would like to know how often certain words have been used in news items. The digital data sources are available, but it is impossible to manually analyze them. After obtaining computational thinking skills, this journalist will see the potential of the data and know how to automatically generate the information she needs. She knows how to apply the right applications or is able to write her own program to carry out the task.

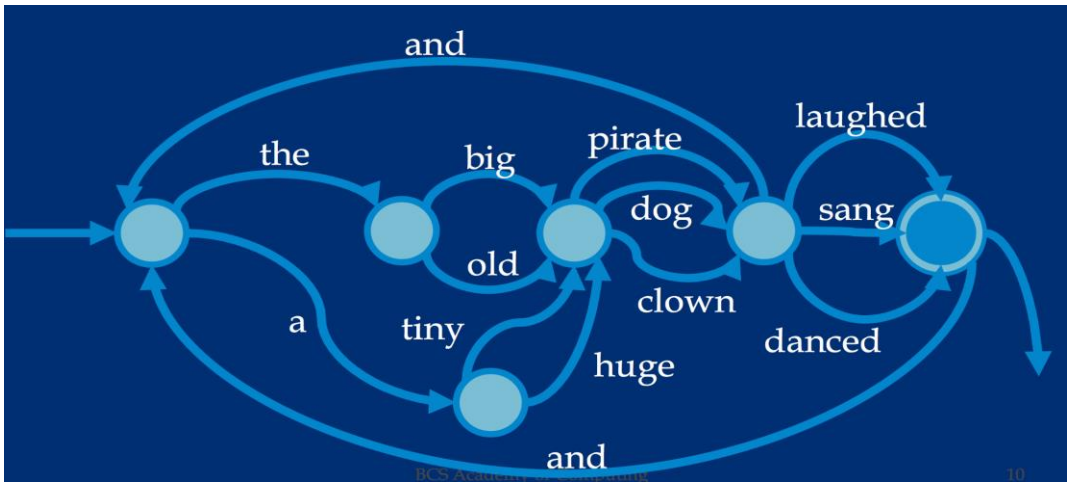
Jeannette Wing (2006) propagated the concept of computational thinking. She characterized it as a thinking process to formulate problems and describe solutions in such a way they can be carried out with the help of a computer. Wing argued that computational thinking will become important in every profession and subject.

Since then, the notion of computational thinking has been made more concrete by various people. The operationalizations brought up so far seem to vary greatly, but a closer look shows that they

roughly amount to the same concept, but emphasize different steps in the thinking process. The widely used definition by Shelby and Woollard (2013) for example, emphasizes the link between the subject content and IT-content (arrows 1 and 3), whereas the description by the American Association of computer teachers CSTA (2011) elaborates Arrow 2 in more detail.

An essential aspect of problem solving through computational thinking is that it leads to an operational, *executable* solution: it can be *carried out* by a computer (or a human). Such a solution may be, for example, an algorithm, a program or a simulation.

This operational aspect also adds a new perspective to the traditional, more analytical way of looking at learning content. See, for example, the figure below, taken from an English language class. Here we see a so-called finite automaton, a well-known model from computer science. By starting left and following the arrows until 'exiting' via the rightmost arrow, various English sentences can be made. In this case it is not the computer that generates sentences (although that could be done perfectly using this automaton), students can do this manually. nstead of a classical analytical approach (by parsing sentences), the concept of grammar is approached synthetically (by generating sentences). This creates new learning activities: students can experiment with language and grammar by adding and omitting arrows.



Generation of English sentences (thanks to Simon Peyton Jones).

These examples of 'computational thinking without the computer' are interesting by-products of translating subject matter into computational terms. In order to let students make more use of the potential of the computer, CT education is required.

Computational thinking pioneer Seymour Papert (1980) introduced programming lessons for children. The idea was that these children would develop novel problem-solving skills that they would apply in new situations. Although Papert's approach led to valuable innovations, the intended *transfer* turned out to be disappointing. The lesson we can learn is that it is not a good idea to put computational thinking entirely in a stand-alone 'subject' and expect that students spontaneously start using computers to solve problems in other subject areas. The embedding of computational thinking entirely in existing subjects is the other extreme; students will probably not learn the basic skills required for computational problem solving. It seems that a combined approach offers the best perspective: incorporating computational thinking within existing subjects, and providing separate training in computational skills (Arrow 2), which require a specific pedagogy.

This is a translation of the second part of the article (in Dutch):

Barendsen, E., & Bruggink, M. (2019). Het volle potentieel van de computer leren benutten: over informatica en computational thinking (*Learning to put the full potential of the computer to use: about computer science and computational thinking*). *Van Twaalf tot Achttien*, 29(10), 16–18.

Learning resources of the module

Presentations for educators

Each presentation can be adopted according to the lecturer's or students' group needs.
Activity 1.1 Introduction presentation (pptx)
Activity 1.2 DigCompEdu model presentation (pptx)
Activity 1.2 SELFIE presentation (pptx)
Activity 1.4 A - Brief introduction to CT.pdf; B - Definitions of CT.pdf
Activity 1.4 Scratch file: PathThroughMaze.sb3

Videos

Video introduction: 4th industrial revolution; https://www.youtube.com/watch?v=uvP4DnH1URg
Video about SELFIE rating system: https://www.youtube.com/watch?v=8_6hVoYXCAI

Readings for educators and students

How SELFIE works. https://ec.europa.eu/education/schools-go-digital/how-selfie-works_en
Digital Competence Framework for Educators (DigCompEdu). https://ec.europa.eu/jrc/en/publication/eur-scientific-and-technical-research-reports/european-framework-digital-competence-educators-digcompedu

Summary of instructional strategies for CT teaching and learning, based on
Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61.

Tool for students

Activity 1.2 Digital competencies weal https://digital-competence.eu/
Activity 1.2 SELFIE https://ec.europa.eu/education/schools-go-digital_en

References

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. Robotics and Autonomous Systems, 75, 661–670.

Bård Ketil Engen (2019). Understanding social and cultural aspects of teachers' digital competencias. Comunicar, n. 61, v. XXVII, 2019. file:///C:/Users/mokyk/Downloads/10.3916_C61-2019-01-english.pdf

Christenson, J.A., (1984). Gemeinschaft and gesellschaft: Testing the spatial and communal hypotheses. Social Forces, 63(1), pp.160-168.

CSTA. (2011). Computational thinking teacher resources, second edition. https://id.iste.org/docs/ct-documents/ct-teacher-resources_2ed-pdf.pdf

Dagienė, V., & Sentance, S. (2016). It's Computational Thinking! Bebras tasks in the curriculum. In International conference on informatics in schools: Situation, evolution, and perspectives (pp. 28–39).
Denning, P. J., Tedre, M (2019). Computational Thinking. The MIT press, Cambridge.
Falloon G. (2020). From digital literacy to digital competence: the teacher digital competency (TDC) framework. Educational Technology Research and Development. https://link.springer.com/article/10.1007/s11423-020-09767-4
Greenfield, P.M., (2013). The changing psychology of culture from 1800 through 2000. Psychological Science, 24(9), pp.1722-1731.
Grover, S., & Pea, R. (2018). Computational Thinking: A competency whose time has come. In S. Sentance, E. Barendsen and C. Schulte (Eds.), Computer science education: Perspectives on teaching and learning in school, Bloomsbury (pp. 19-38).
Oberländer M., Beinicke A., Bipp T. (2020). Digital competencies: A review of the literature and applications in the workplace. Computers & Education, Volume 146.
Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.
Selby, C. & Woollard, J. (2013) Computational thinking: the developing definition, available via internet: http://eprints.soton.ac.uk/356481, accessed: 10 February 2021
Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: measuring teacher understanding of computational ideas for teaching science. Computer Science Education, 28(4), 371–400.
Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.