Computational Thinking and Digital Skills Workshop

Universiteit Leiden
The Netherlands

TU Delft
Delft University of Technology

Erasmus University Rotterdam

some history …

90’s Research Questions

- **Developing Expertise in CT**: Syntax vs. Conceptual Knowledge Development in Visual Programming

- **Developing Expertise in CT:** Example Based Programming and Personalised Examples

- **Feedback and Learning**: Design of Feedback in Online Learning, formative feedback, stacked feedback, freedom of exploration

- **Technical and Pedagogical Integration** of Evaluation and Tutoring Services with Web-Based Textbooks

- **Technical and Pedagogical Design** and effects adaptive navigation support and personalised recommender systems

Syntax vs. Conceptual Knowledge

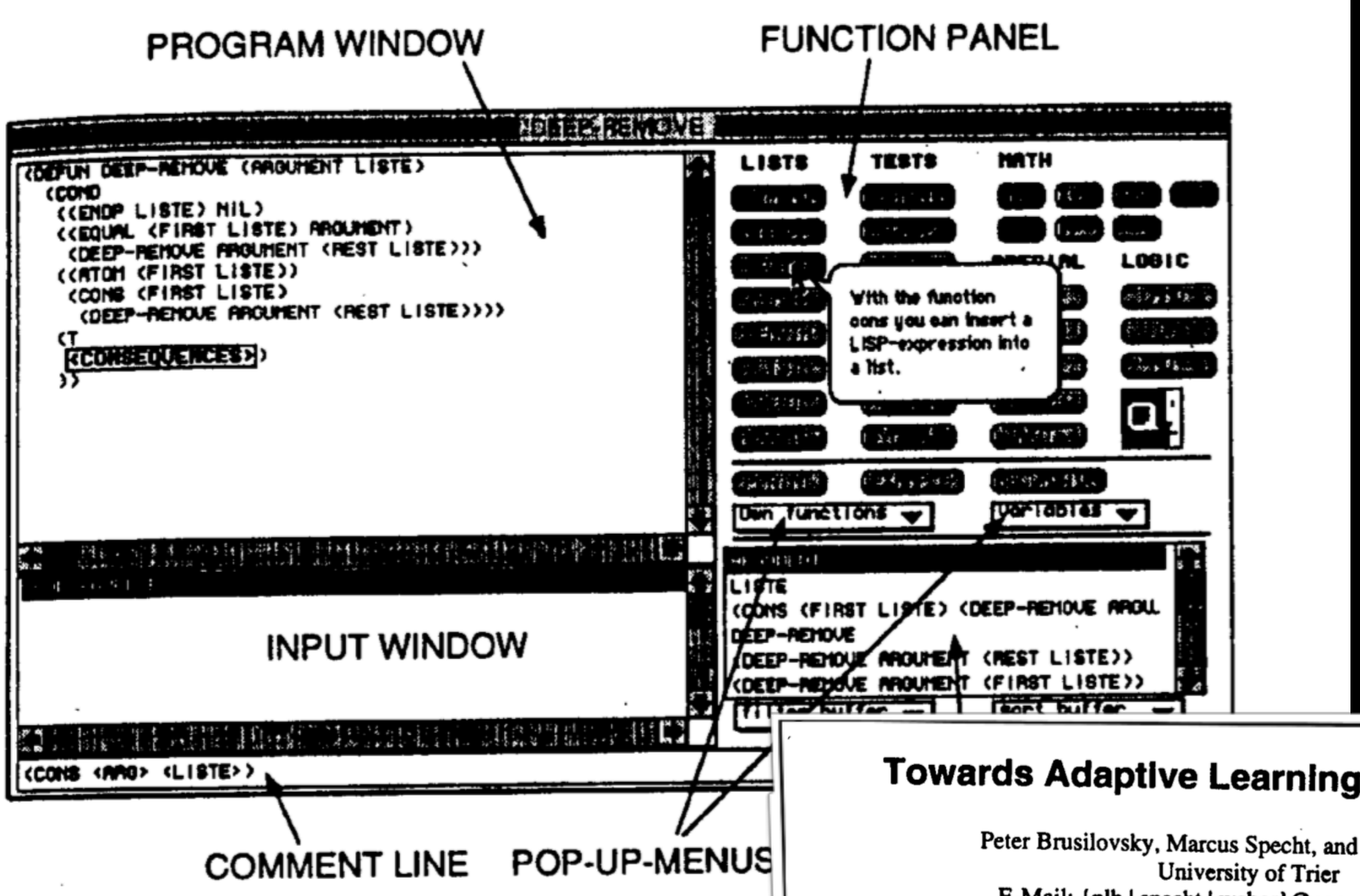


Figure 1: The syntax-driven structure edi

Towards Adaptive Learning Environments

Peter Brusilovsky, Marcus Specht, and Gerhard Weber
University of Trier
E-Mail: {plb | specht | weber}@cogpsy.uni-trier.de

Scratch …

Scenarios of using Scratch

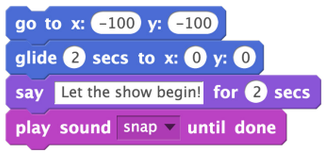
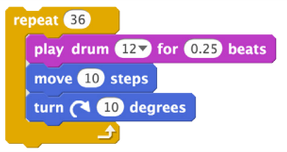
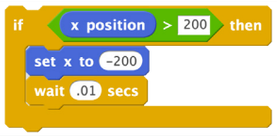
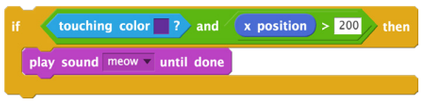
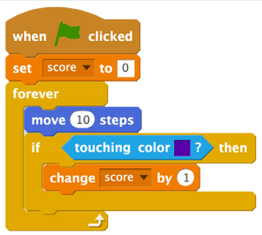
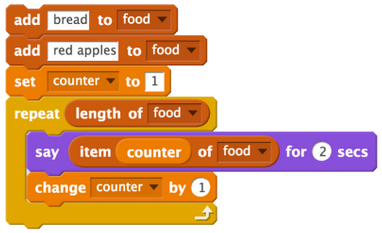
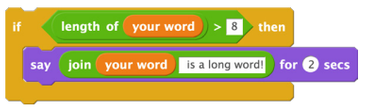
- Storytelling: from simple animations to real stories
- Games development: from mathematical games to interactive visual games
- Arts and Creativity: geometry, mandalas, complex mathematical visualisations
- Music and Interactivity
- …

CS concepts in Scratch

- Sequence, Iteration (loops)
- Random, Boolean logic
- Variables, Lists (arrays)
- Events, Threads, Synchronisations
- Procedures, Parameters
- Cloning, Physical Sensing

COMPUTATIONAL CONCEPTS SUPPORTED IN SCRATCH

In the process of creating projects with Scratch, young people develop as computational thinkers. They learn concepts, engage in practices, and develop perspectives they can use to express their ideas with code. This list features fundamental computational concepts that are supported in Scratch.

Concept	Explanation	Example
sequence	To create a program in Scratch, you need to think systematically about the order of steps.	
iteration (looping)	*forever* and *repeat* can be used for iteration (repeating a series of instructions)	
random	*pick random* selects random integers within a given range.	
conditional statements	*if* and *if else* check for a condition.	
boolean logic	*and*, *or*, *not* are examples of boolean logic	
variables	The variable blocks allow you to create variables and use them in a program. Variables can store numbers or strings. Scratch supports both global and object-specific variables.	
lists (arrays)	The list blocks allow for storing and accessing a list of numbers and strings. This kind of data structure can be considered a “dynamic array.”	
string manipulation	You can change or get information about strings of letters using *length of*, *letter of*, and *join*.	

To learn more computational thinking, see scratched.gse.harvard.edu/ct/

WeDo Extension Blocks

LEGO WeDo Extension Blocks

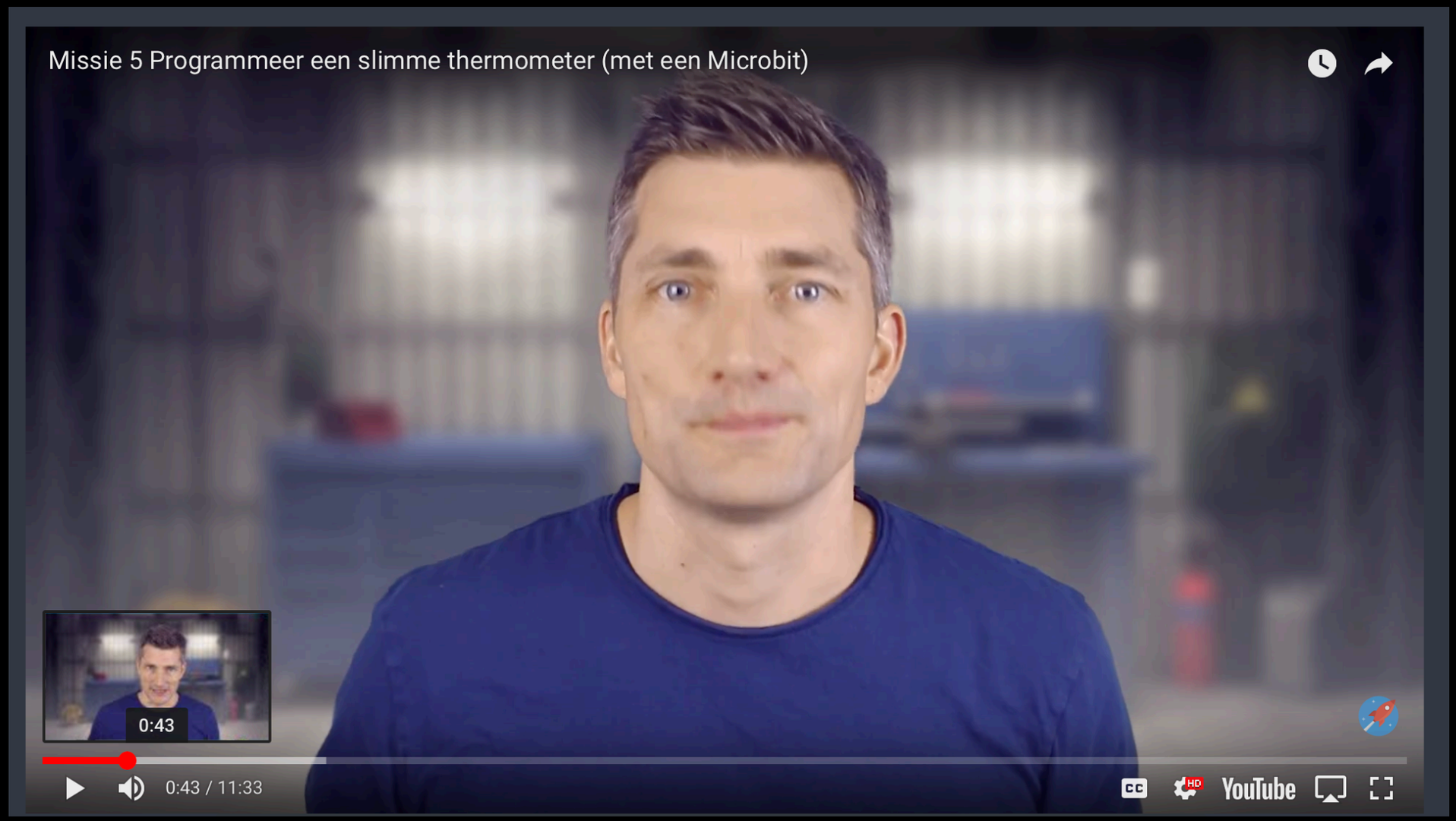
turn motor on for ○ secs	Turns a specific motor or the lights on for a certain amount of time. There are five options for the block, listed as "motor", "motor A", "motor B", "light" and "everything".
turn motor on	Turn a specific motor or the lights on indefinitely.
turn motor off	Turn a specific motor or the lights off.
set motor power ○	This block sets the power of a specific motor or the lights, controlling the speed at which the motor is spinning or the brightness of the lights.
set motor direction ○	This block sets the direction that a specific motor should turn with. There are three options for the direction, listed as "this way", "that way", and "reverse". The first two are equivalent to clockwise and counter-clockwise. Reverse switches the direction.

when distance < ○	This hat block runs a script when the distance becomes less (or greater) than a specified value.
when tilt = ○	This hat block runs a script when the tilt value becomes equal (or not equal) to a specified value. The tilt sensor returns 0-4, with 0 indicating not tilted, 1 tilted down, 2 tilted right, 3 tilted up, and 4 tilted to the left.
distance	It reports the distance sensor value.
tilt	It reports the tilt sensor value.

SRA Programming EV3

- Higher analytical skill when applying more SRA programming
- Tendency for higher mathematical skill when applying more SRA loops

#3 How to raise awareness and link to families and everyday life?



Verzamel de materialen:

1. Microbit
2. battery pack + 2 batterijen
3. laptop of computer
4. usb kabel
5. schaar
6. lijm
7. sjabloon geprint op karton
8. spul om te customizen

https://youtu.be/YkJq9MJ21C8

How can we APPLY this In Higher Education?

Why to learn coding ?

Computational thinking (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

Why to learn coding ?

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

Integration in Curriculum

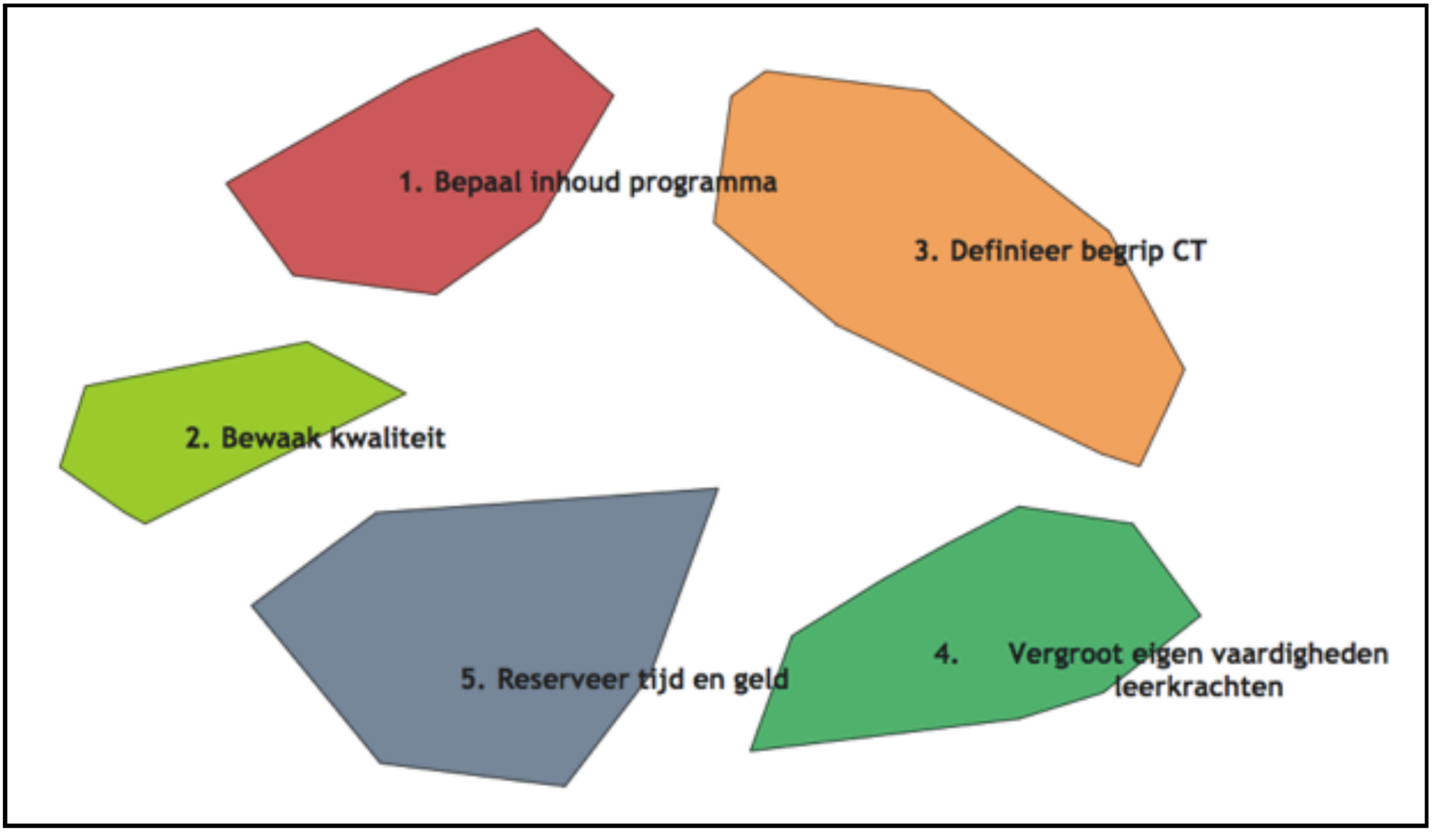
CT Concept, Capability	CS	Math	Science	Social Studies	Language Arts
Data collection	Find a data source for a problem area	Find a data source for a problem area, for example, flipping coins or throwing dice	Collect data from an experiment	Study battle statistics or population data	Do linguistic analysis of sentences
Data analysis	write a program to do basic statistical calculations on a set of data	count occurrences of flips, dice throws and analyzing results	analyze data from an experiment	identify trends in data from statistics	identify patterns for different sentence types
Data representation	use data structures such as array, linked list, stack, queue, graph, hash table, etc	use histogram, pie chart, bar chart to represent data; use sets, lists, graphs, etc. to contain data	summarize data from an experiment	summarize and represent trends	represent patterns of different sentence types
Problem decomposition	define objects and methods; define main and functions	apply order of operations in an expression	do a species classification		write an outline
Abstraction	use procedures to encapsulate a set of often repeated commands that perform a function; use conditionals, loops, recursion, etc.	use variables in Algebra; identify essential facts in a word problem; study functions in algebra compared to functions in programming; use iteration to solve word problems	build a model of a physical entity	summarize facts; deduce conclusions from facts	use of simile and metaphor; write a story with branches
Algorithms & procedures	study classic algorithms;	do long division, factoring; do	do an experimental procedure		write instructions

Effects of Learning Programming

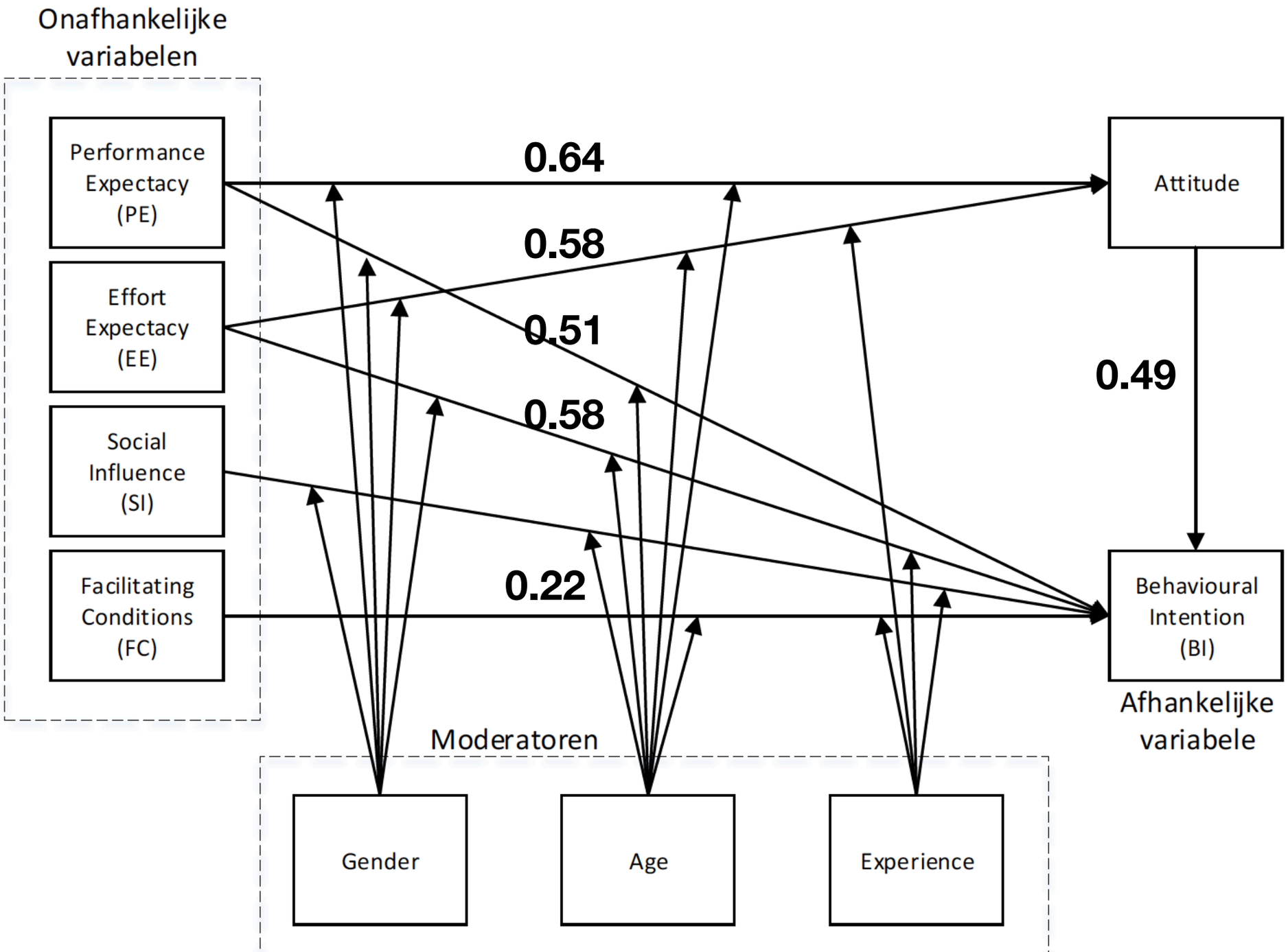
- general competences: Problem solving, creativity, reflection, metacognitive skills, mathematical thinking
- computer science related skills: CS concepts, analysis, planning, control flow, debugging, abstraction

Some results ..

What is necessary ?



Are teachers ready ?



How to make it visible ?

Thema CT	Low level	Medium level	High level
Denken in stappen – Een probleem opdelen in kleinere deelproblemen of in deelvragen. – Een probleem zo formuleren dat het met behulp van een computer is op te lossen.	*"…Vaak kregen mensen gewoon taken, van hee, kan jij dit voor ons doen? Dan werd het aangeleverd, vervolgens als een persoon dat had aangeleverd, dan werd het in de groep gegooid. En dan ging iedereen kijken of hij nog zelf kon uitbreiden of verbeteren. Dus zelf code toevoegen aan het gemaakte deel…dat geleverd werd." (ST108)*	*"…Die tussenstap, daar waren we te laat achter gekomen. En daarna hebben we vanaf het herkansingmoment de draad weer opgepakt. Want we hadden de database al opgezet en vervolgens gingen we toen echt tutorials zoeken, een connectie maken met de database. En vervolgens dat je de gegevens uit de database filtert om het in een grafiek weer te geven" (ST111)*	*"…Dat zijn alle stappen en moet ik dus stap voor stap gaan kijken hoe ik dat moet doen. En dan in code om gaan zetten. Of opdelen van problemen. Als je een groot probleem hebt, krijg je het nooit opgelost". (ST69)*
Abstraheren – De essentie verduidelijken zonder zich in details te verliezen. – Schematiseren/modelleren door gebruik te maken van schetsen, tabellen, grafieken of modellen.	*"…De belangrijkste onderdelen, op de manier van hoe we het uiteindelijk gedaan hebben? Oké. Voornamelijk de voorkennis die mensen al hadden in de projectgroep en Internet."(ST52)*	*"…De code is in die zin belangrijk, dat je iets moet hebben om te kunnen laten zien. Maar het hoeft niet ingewikkeld te zijn, als je het maar mooi kan presenteren. Dus die code is wel degelijk belangrijk, anders heb je niks. Maar buiten dat, als je het af hebt dan zorg je ervoor dat de randzaken ook allemaal in orde zijn". (ST68)*	*"…Uiteindelijk kwam ik dus met een tabelletje met alle requirements. In die requirements stonden bepaalde termen, termen die dus niet uitgelegd waren…Die moeten goed gedefinieerd worden voordat we kunnen beginnen aan het project en dat is dus ook wat we gedaan hebben." (ST32)*
Algoritmisch denken – Stap-voor-stap specifieke en expliciete instructies maken om een proces uit te voeren. – Logische volgordelijkheid toepassen.	*"…Nee, ja, maar dat was gewoon tussendoor. Zo van: hee, let je daar nog effe op? Maar dat werd niet behandeld in zo'n bestand, bijvoorbeeld op OneDrive. Dat was meer hoe ben je, doe je goed mee. Maar we gingen niet echt inhoudelijk in op de code …"(ST68)*	*"…Gewoon zoveel mogelijk opties, gekke dingen die je maar kan bedenken, altijd proberen. En zodra ik het uit zou willen geven wat ik heb gemaakt, dan ga ik gewoon aan kennissen en vrienden vragen van joh, kan je het testen? Om de een of andere reden krijgen ze het altijd voor elkaar om het alsnog te breken!" (ST52)*	*"…Door overal comments neer te zetten. En als het eenmaal werkt, dus ook wat netter te maken, dus dingen in functies zetten. Het zijn vaak heel lange codes en het wordt echt spaghetticode als we ook nog code to gaan gebruiken." (ST69)*

Current pilots at TUD ..