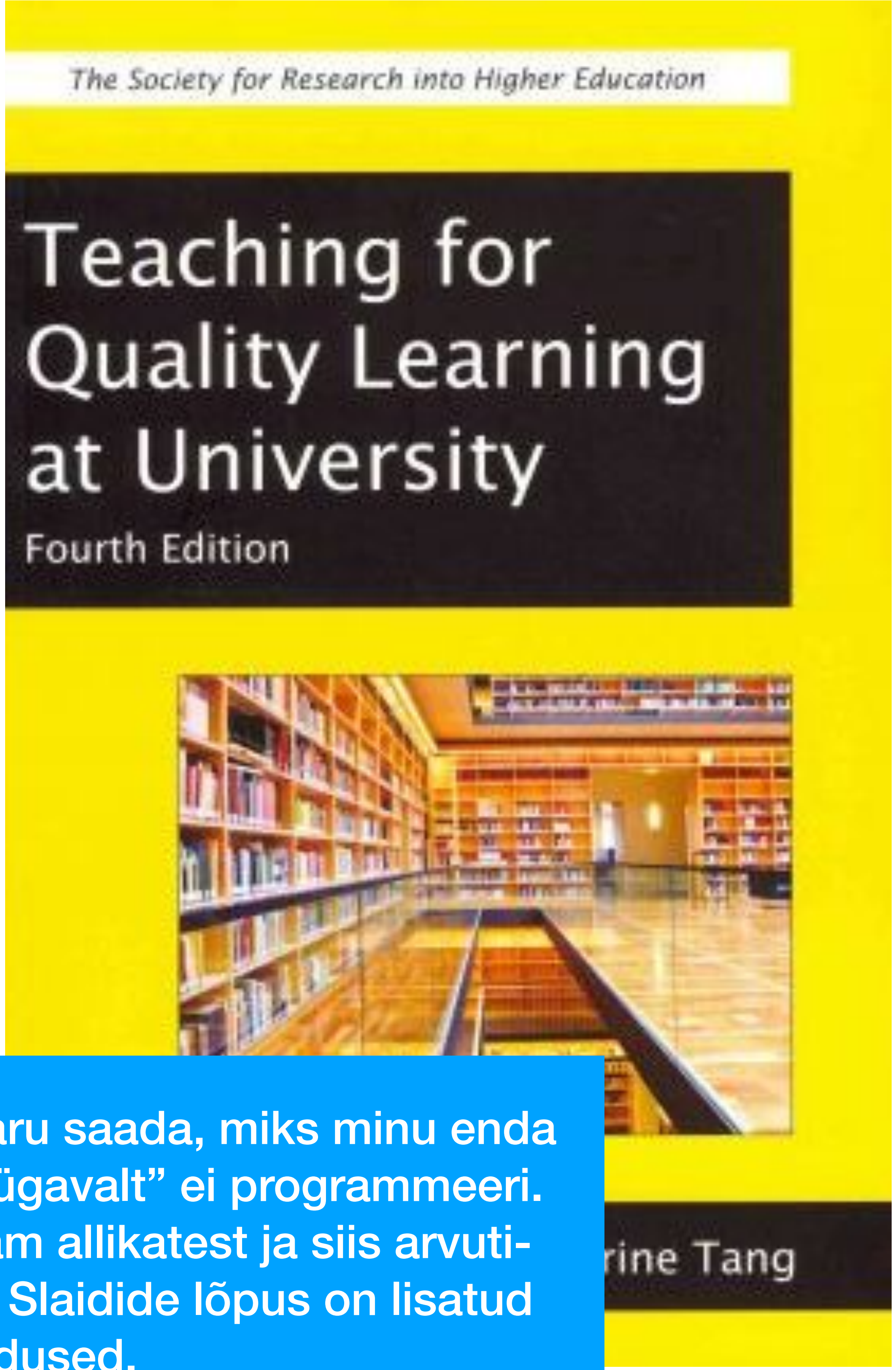
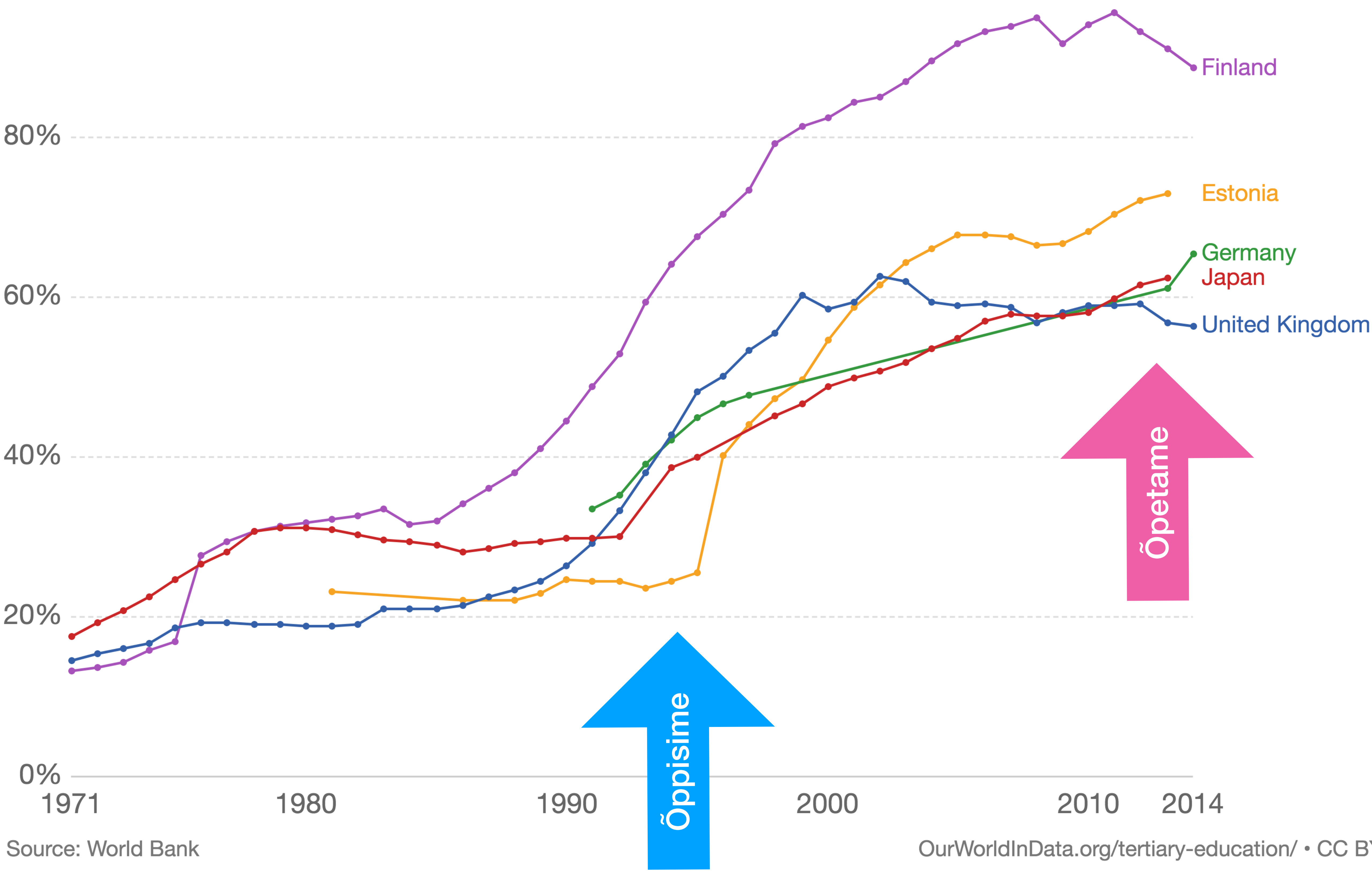
Programmeerist toetav õpetamine ülikoolis

Vesal Vojdani
PLAS Seminar

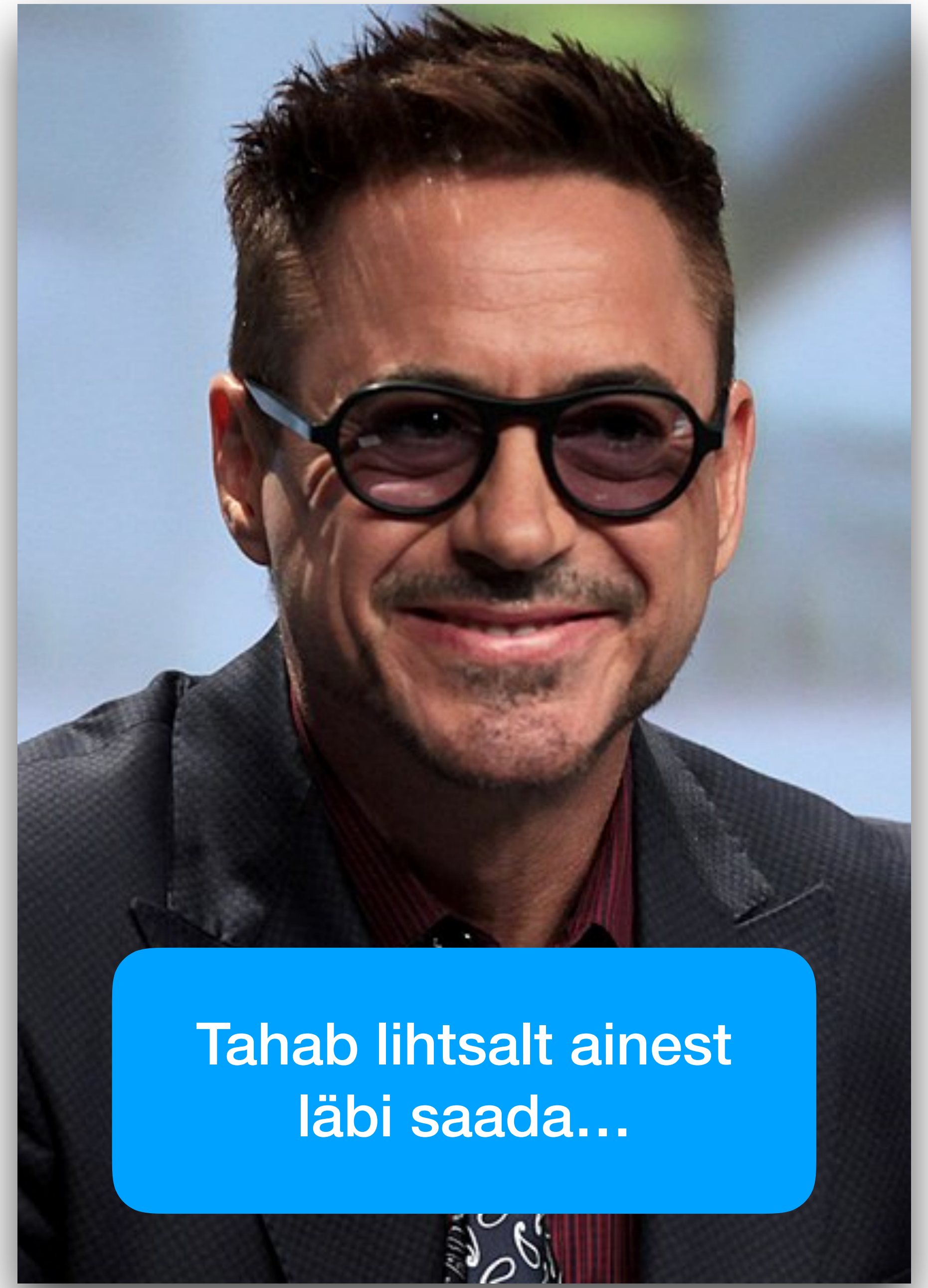


Tegemist on siin minu katsega aru saada, miks minu enda laps ja paljud minu õpilased “sügavalt” ei programmeeri. Kõigepealt alustasin mainstream allikatest ja siis arvutiteaduse spetsiifilist kirjandust. Slaidide lõpus on lisatud mõned järeldused.

Ülikoolis õppijate osakaal potentsiaalsetest minejatest (kõik ülikoolis õppijad / elanikkond vanusel 18-23)



Source: World Bank

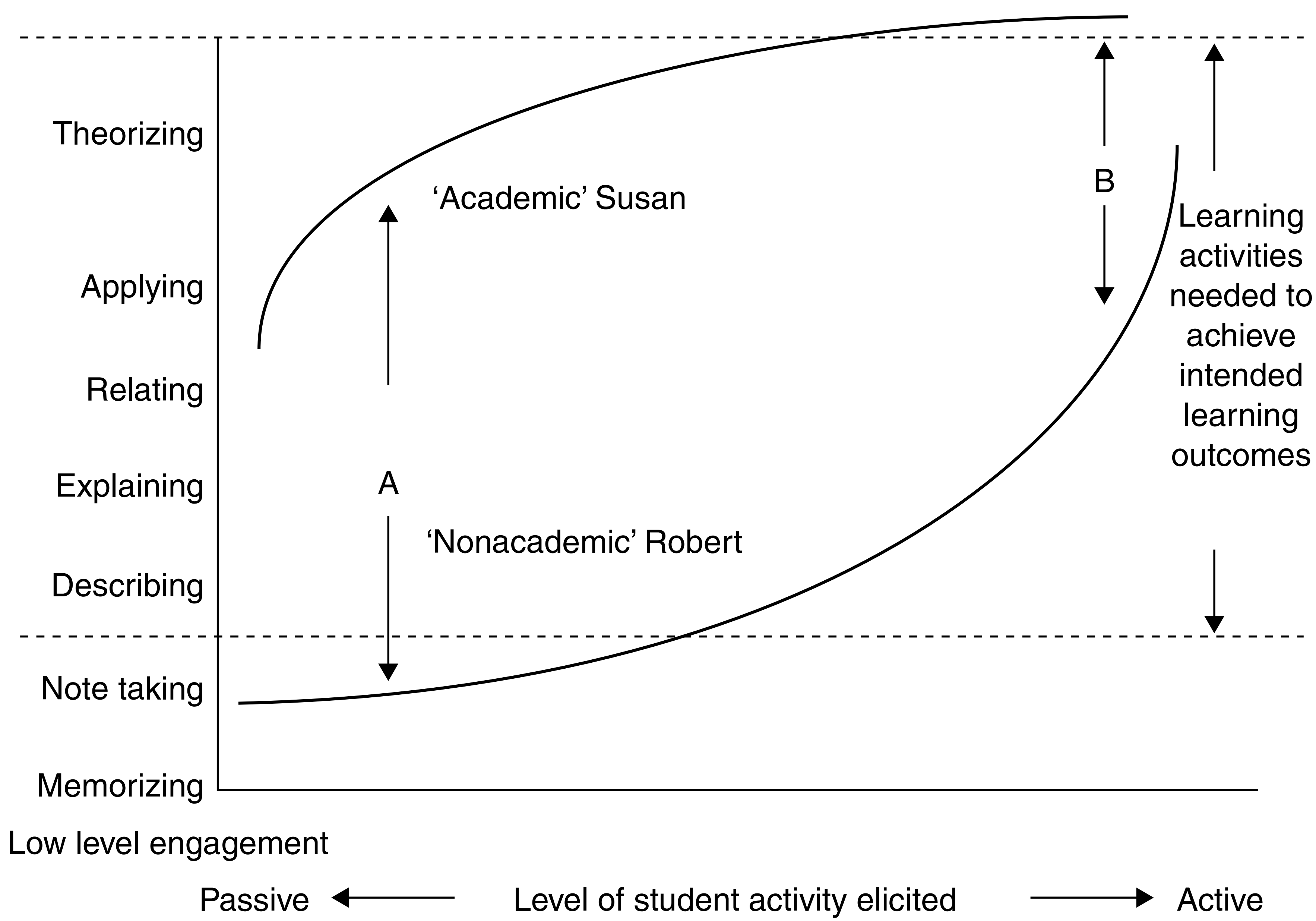


Robert Downey, Jr. (CC BY-SA 2.0) by Gage Skidmore

Susan Boyle (CC BY-SA 3.0) by Wasforgas

Tudengite Mitmekesisus

“The Robert and Susan Problem”

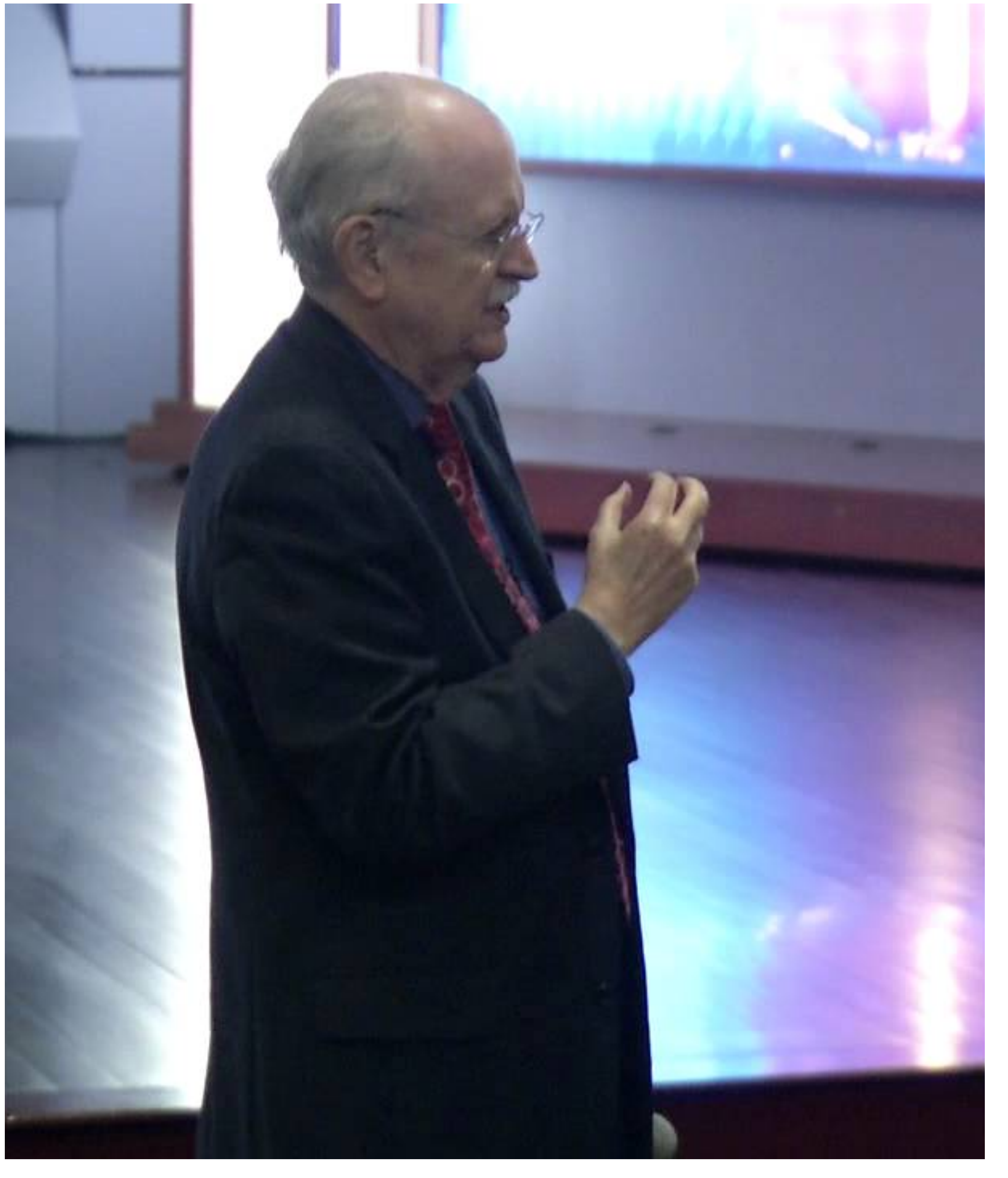


Põhiküsimus:

Kuidas õpetada nii, et Robert õpiks nagu Susan?

Ken Baini küsimus

- Millal te viimati õppisite midagi väga sügavalt?
- (Kas programmeerimisega seoses midagi meenub?)
- Nüüd see küsimus: mis olid need **tingimused/eeldused**, mis viisid selleni?



AKTs meeldib paljudele tudengitele just lõplike automaatide osa!

$$\text{Olulisus} \times P(\text{edukus})$$

Motivatsiooni valem

P(edukus) on subjektiivne usk, et saab hakkama!

Isegi motiveeritud õppija…

Ronja jälgib väga hästi juhendit, aga ei õpi sügavalt!

(Üks vihje on andestatav: tädi võib peale ülesanne täitmist edasi liikuda, aga seda on ka varem siin keskkonnas olnud.)

Ei ole asjast aru saanud, kui…

Esimese tsükli teeb kohe ära, aga siis kulub >30min ja 75 vihjet, et esimest kõrvitsat ei jätaks vahele

Miks ei õpita sügavalt?

- Ken Bain süüdistab koolisüsteemi, mis tingib neid pealiskaudselt õppima.
- Neljaaastased on uudishimulikud ja naudivad õppimist. Neil on sisemine motivatsioon.
- Koolis aga õpetatakse hinnete ja testide järgi õppima.
- Hindamine peab soosima sügavam õppimine.

Paper Session: Curriculum Issues #1 SIGCSE'18, February 21-24, 2018, Baltimore, MD, USA

A Systematic Review of the Use of Bloom's Taxonomy in Computer Science Education

Susana Masapanta-Carrión
Pontificia Universidad Católica del Ecuador
Quito, 17012184, Ecuador
smmasapanta@puce.edu.ec

J. Ángel Velázquez-Iturbide
Universidad Rey Juan Carlos
28933 Móstoles, Madrid, Spain
angel.velazquez@urjc.es

Õpiväljund/hindamine

Pean aru saada, mis see täpselt on,
mida ma tahan (siiamaani edutult) õpetada.

Remember Understand Apply Analyze Evaluate Create

Problem Statement for EggCartons

Problem Statement

There are two types of egg cartons. One type contains 6 eggs and the other type contains 8 eggs. John wants to buy exactly n eggs. Return the minimal number of egg cartons he must buy. If it's impossible to buy exactly n eggs, return -1.

Definition

Class: EggCartons
Method: minCartons
Parameters: int
Returns: int
Method signature: int minCartons(int n)
(be sure your method is public)

Constraints

- n will be between 1 and 100, inclusive.

Examples

0)
20
Returns: 3
He should buy 2 cartons containing 6 eggs a

1)
24
Returns: 3
There are two ways to buy 24 eggs: buy 4 ca
cartons.

2)
15
Returns: -1
He can't buy an odd number of eggs.

3)
4
Returns: -1


```java
public static int minCartons(int n) {
    for (int big = n/4; big >= 0; big--) {
        int small = (n - big*4) / 3;
        if (big * 4 + small * 3 == n) {
            return small + big;
        }
    }
    return -1;
}
```

Kuidas hinnata sügavust?

Sul on kahte liiki pakendid (6 muna ja 8 muna)
Kuidas osta n muna võimalikult väheste pakenditega?

specific learning outcome or test item depends on its context. A task that challenges the analysis and synthesis skills of a beginner becomes routine application of knowledge for a more advanced learner. Similarly, a student who has been taught how to solve a problem that is extremely similar to the test item will demonstrate skills lower in the taxonomic order than one who is solving it from first principles. This is a generic problem but computer science-specific difficulties also manifest themselves.

Praktikumis lahendasime:

Sul on kahte liiki pakendid (3 muna ja 4 muna)
Kuidas osta n muna võimalikult väheste pakenditega?

Developing a Computer Science-specific Learning Taxonomy

Ursula Fuller
Computing Laboratory
University of Kent
Canterbury CT2 7NF
United Kingdom
U.D.Fuller@kent.ac.uk

Colin G. Johnson
Computing Laboratory
University of Kent
Canterbury CT2 7NF
United Kingdom
C.G.Johnson@kent.ac.uk

Tuukka Ahoniemi
Institute of Software Systems
Tampere University of Technology
Tampere, Finland
tuukka.ahoniemi@tut.fi

Diana Cukierman
School of Computing Science
Simon Fraser University
Burnaby, British Columbia
Canada
diana@cs.sfu.ca

Isidoro Hernán-Losada
Lenguajes y Sistemas Informáticos
Universidad Rey Juan Carlos
Madrid
Spain
Isidoro.hernan@urjc.es

Jana Jackova
Faculty of Management Science and Informatics
University of Zilina /Slovak University of Technology
Zilina, Slovak Republic
Jana.Jackova@fri.uniza.sk

Essi Lahtinen
Institute of Software Systems
Tampere University of Technology
Tampere
Finland
essi.lahtinen@tut.fi

Tracy L. Lewis
Information Technology
Radford University
Radford, VA 24142
USA
Tlewis32@radford.edu

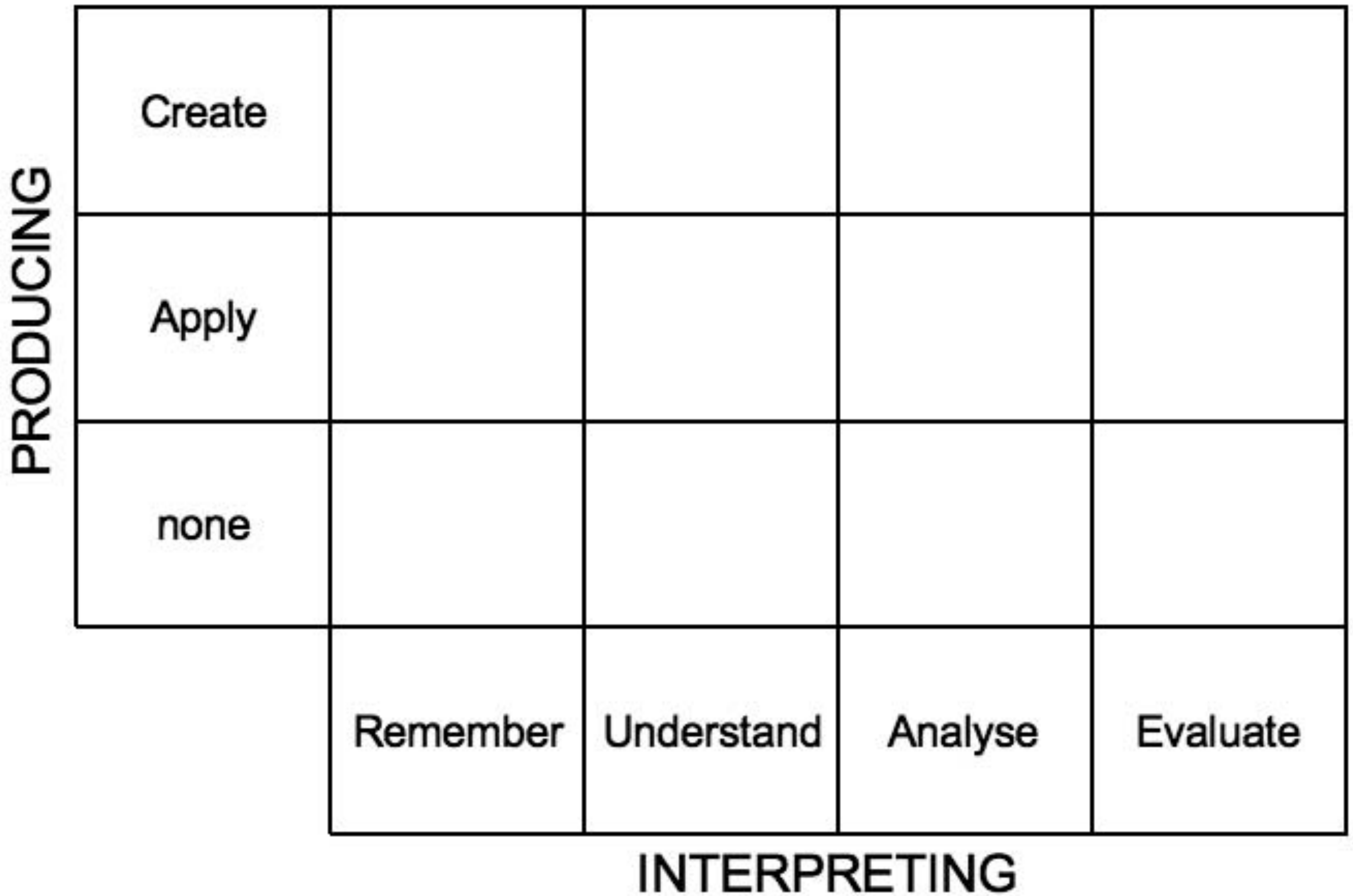
Donna McGee Thompson
Student Learning Commons
Simon Fraser University
Burnaby, British Columbia
Canada
dmcthomp@sfu.ca

Charles Riedesel
Computer Science & Engineering
University of Nebraska Lincoln
259 Avery Hall
Lincoln, Nebraska 68588-0115
USA
riedesel@cse.unl.edu

Errol Thompson
Massey University
Wellington
New Zealand
kiwiet@computer.org

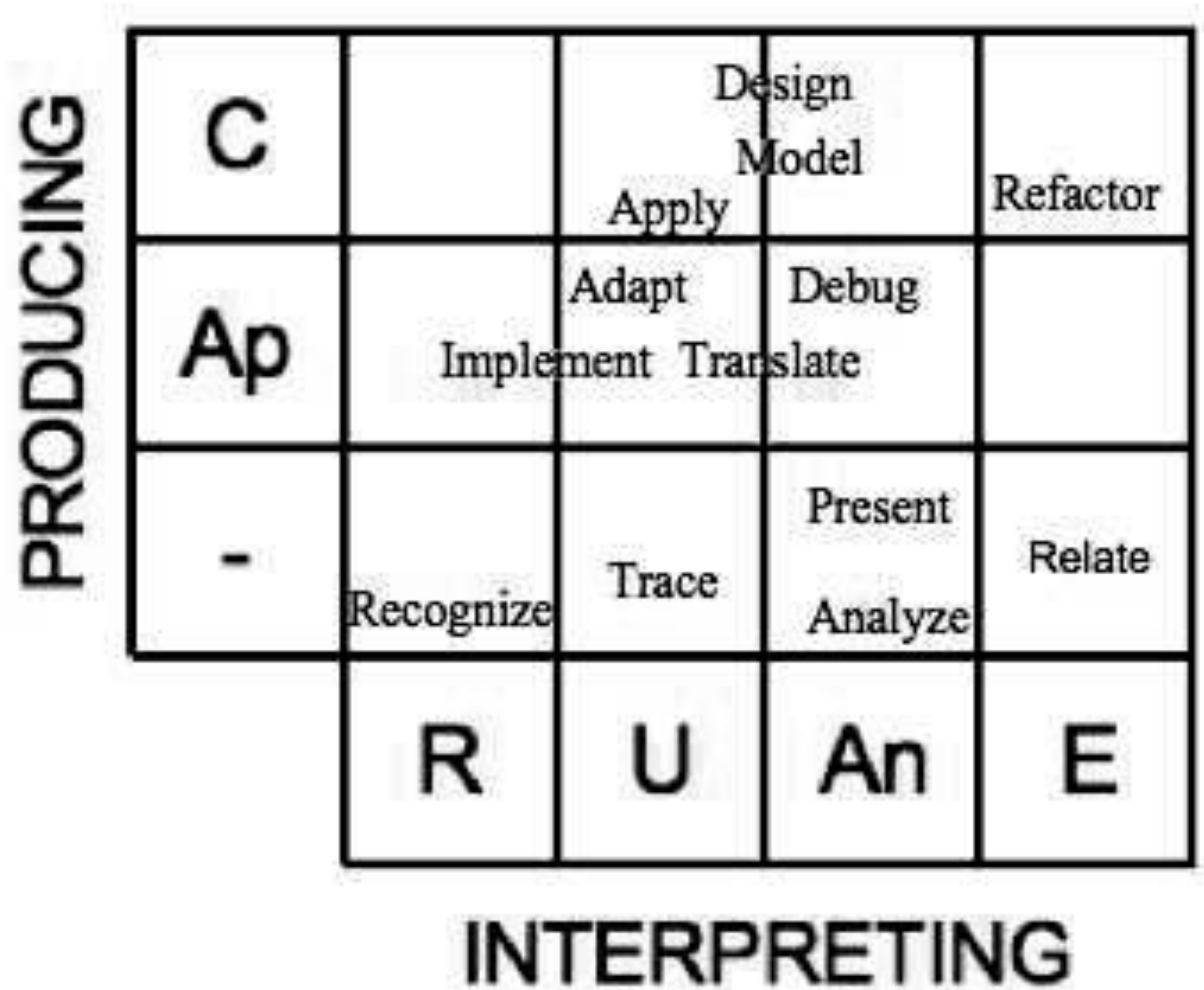
Arvutiteaduse mõtlemistasandid

ITiCSE töörühm üritasid välja töötada taksonoomia, millega aru saada programmeerija arengut



Nende taksonoomia

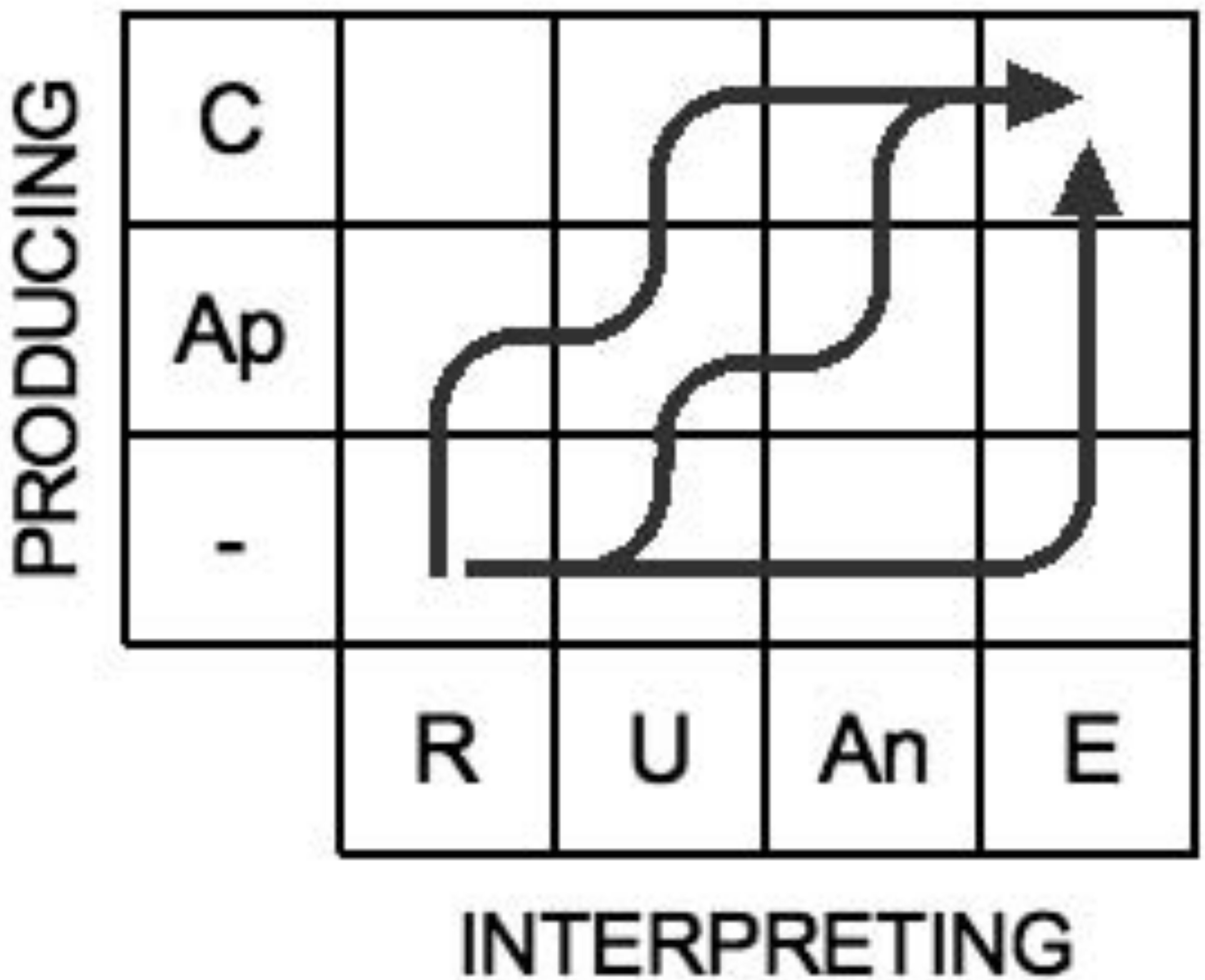
Loomine ja arusaamine olevat “semi-independent”



Verbide liigitus

Relate: merge sort versus quicksort…

Refaktoriseerimisel põhinevad hajutused võiks küll olla!



Õpilaste arenguteed

1) Ainult arusaamine (oskab juhendi järgi, palju abi vaja)
2) Ainult loomine (katse-eksitus, testid lähevad läbi…)

Using Bloom's Taxonomy To Code Verbal Protocols of Students Solving a Data Structure Problem

Jennifer Parham
School of Computing
Clemson University
Clemson, SC 29634
+1 864 245 4177
jparham@g.clemson.edu

Donald Chinn
Computing and Software Systems
University of Washington, Tacoma
Tacoma, WA 98402-3100
+1 253 692 4660
dchinn@u.washington.edu

D. E. Stevenson
School of Computing
Clemson University
Clemson, SC 29634
+1 864 656 5880
steve@cs.clemson.edu

Huvitab taksonoomia kasutus

Mis tasandil toimub mõtlemine A&A ülesanne lahendamisel?

“A *stack* is a standard data structure for which the following operations are defined for it:

push(x): places object x on the stack

pop(): takes the topmost element off the stack and returns it (if the stack is empty, then an error or exception is raised)

A stack works like a pile of dishes where the only things you can do to the pile is to put a dish on the top of the pile or take a dish off the top of the pile.

A stack is typically implemented using either an array or a linked list. When a stack is implemented using a linked list, both the push and pop operations can be performed in O(1) time per operation. That is, the time it takes to execute a push or pop operation is a fixed amount of time that is independent of how many objects are in the stack when the operation is performed.

For simplicity, let us assume that the elements in our stack consist of integers. We wish to create a data structure that not only supports the standard stack operations (push and pop), but also the operation **findMinimum**(), where findMinimum() returns the smallest element in the stack. It is easy to modify an implementation of a standard stack so that we can perform push and pop in O(1) time and findMinimum in O(n) time. However, we can do better.

1. Describe how you would implement this new data structure so that *all* the operations (including findMinimum) execute in O(1) time.
2. Explain what happens in each operation of your implementation and why each operation runs in O(1) time."

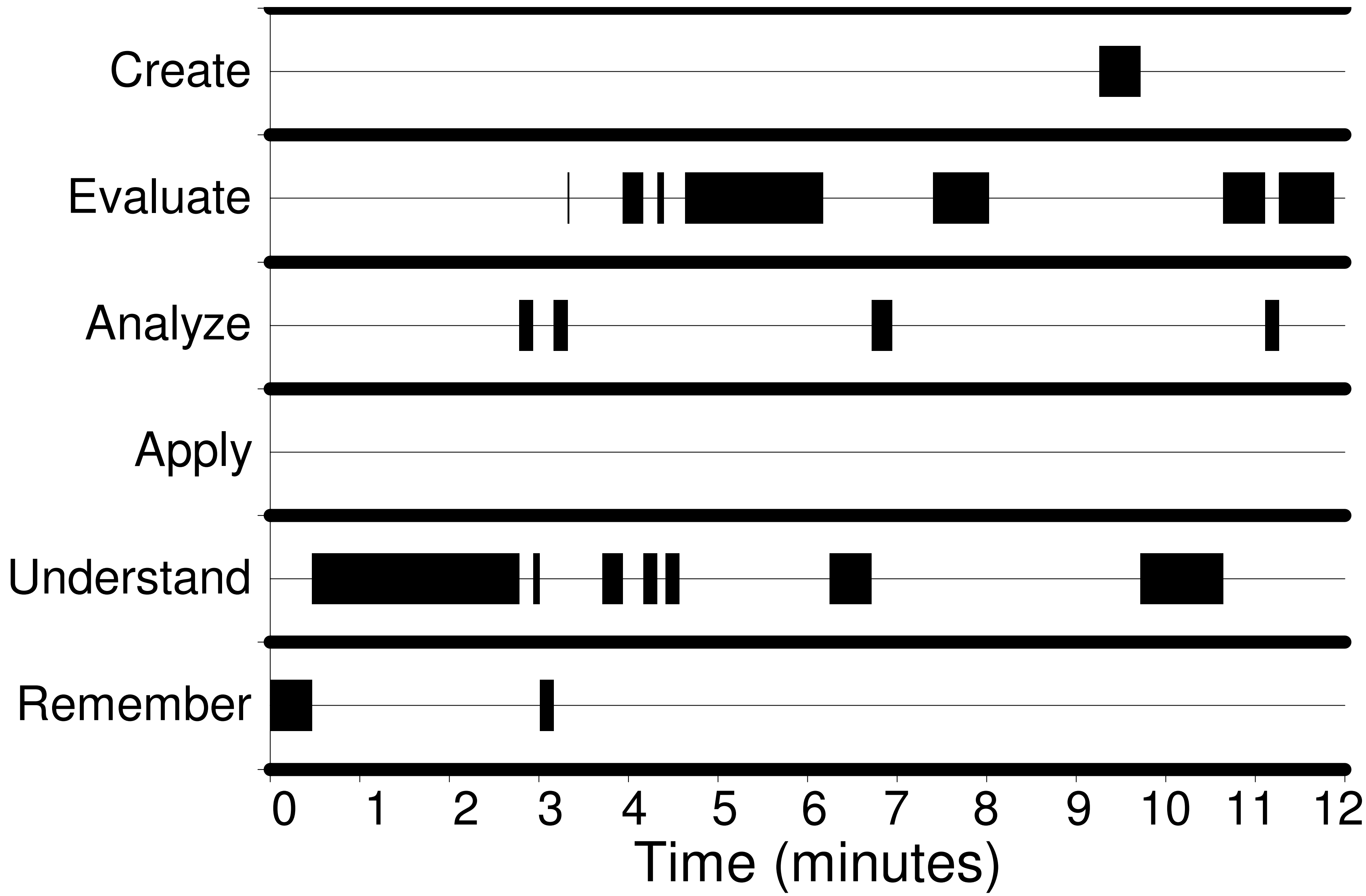
Minu enda mõttekäik

- Proovime lihtsalt miinimum ühes muutujas meelde jätta…
- OK, pop() puhul oleks eelmist vaja.
- Aga, hmm, äkki jätaks vanu meelde, et neil on siis ka oma magasin (või viited esialgsesse, ah, teeme lihtsalt praegu)
- Kas siis tõesti kehtib see, et kui eemaldan element mõlemast stack'ist, siis ta ongi ülejäänud stack'i vähim?
- JAAAAA, ahaaaaa, nad ongi ju lisamise järjekorra mõttes kooskõlas, see ongi see invariant, jne…

Analyze	Break material into its constituent parts and determine how the parts relate	"Is there some kind of data structure that I should be using other than this, other than the linked list?"
		"… because my next field would be already taken."
Evaluate	Make judgments based on criteria and standards	"The point was to keep the data structure, the data organized."
		"That would take up too much space."
Create	Put elements together; reorganize elements into a new pattern or structure	"Now I am trying a different kind of one with the elements before it was smaller, so as I enter a 5…"
		"I need to insert these in the right order, because it should be going to the front of the list not the back."

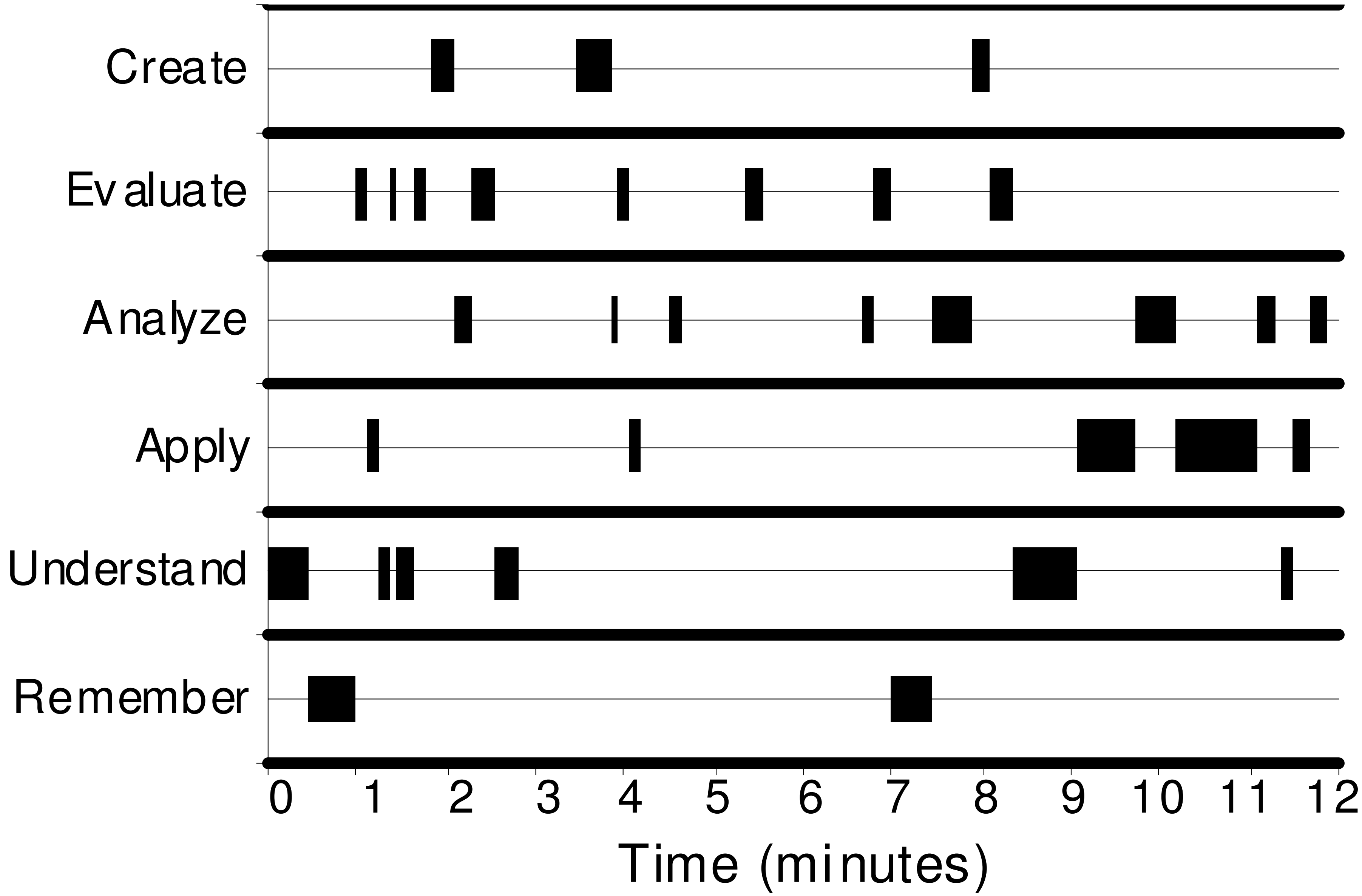
Tudengite mõtteid liigitati

See on nüüd ainult 7 tudengi põhjal…



Kõige kehvem lahendaja

Väga hilja üritab midagi luua!



Edukas lahendaja

Proovib juba varakult midagi välja mõelda

Minu enda mõttekäik

- Proovime kõigepealt lihtsalt seda meelde jätta…
- OK, pop() puhul oleks eelmist vaja.
- Aga, hmm, äkki jätaks vanu meelde, et neil on siis ka oma magasin (või viited esialgsesse, ah, teeme lihtsalt p
- Kas siis tõesti kehtib see ldan element mõlemast stack'ist, siis ud stack'i vähim?
- JAAAAA, ahaaaaa, nad on ju meil sünkroonis, see ongi see invariant, jne…

Create!

Evaluate/
Analyze

Mina seda kohe
algpunktis ei näeks!

Using Unstructured Practice plus Reflection to Develop Programming/Problem-Solving Fluency

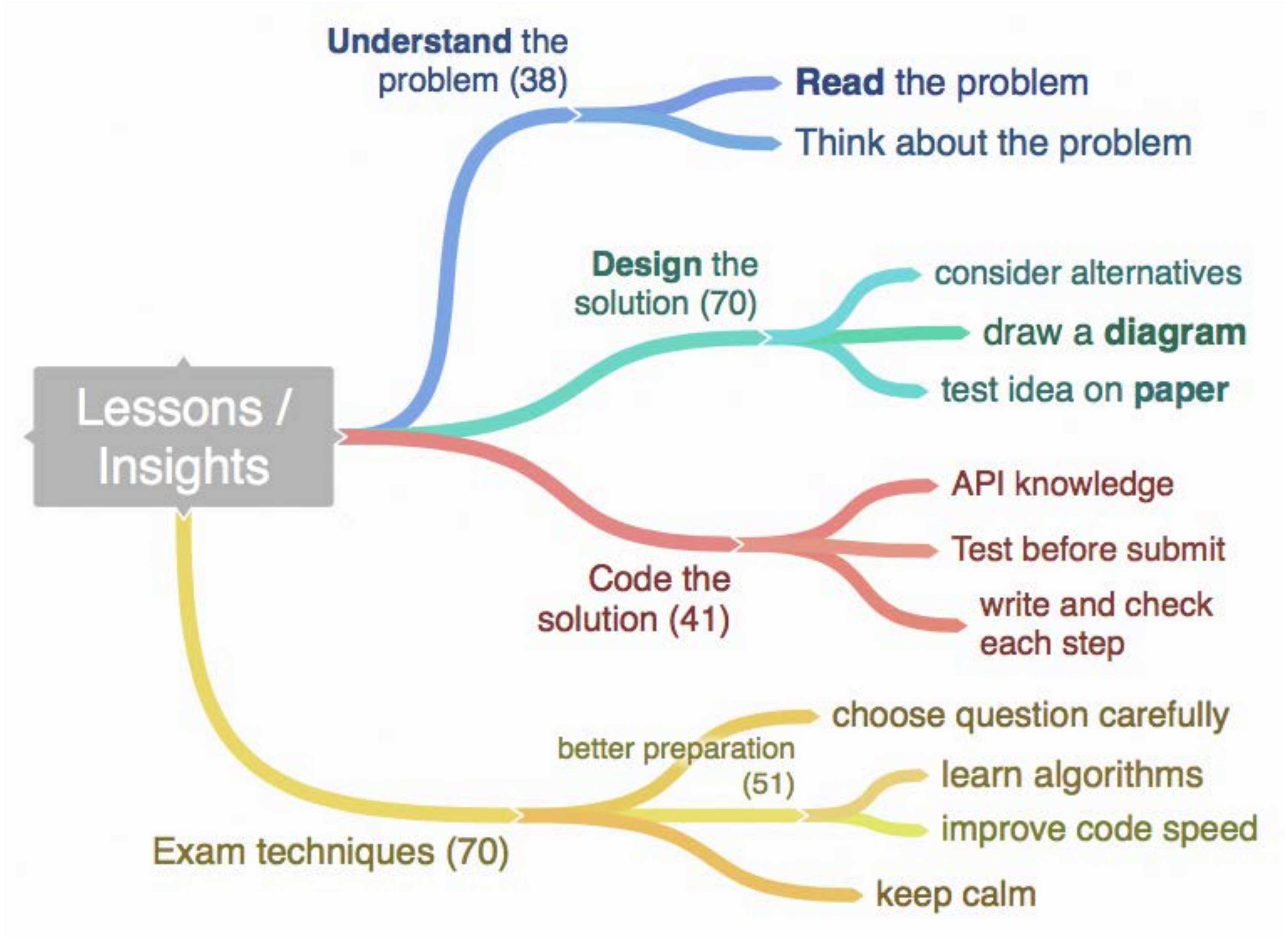
Cruz Izu
The University of Adelaide
Adelaide, Australia
cruz.izu@adelaide.edu.au

Brad Alexander
The University of Adelaide
Adelaide, Australia
bradley.alexander@adelaide.edu.au

Kuidas siis ikkagi õpetada?

Problem Coding Practice

- Kursus kestis 12 nädalat
- Kohustuslik bakalaureuses (>100 tudengit)
- Iga nädal valivad 3-5 ülesannet pakutud 8 ülesandest.
- Üks eeltest ja 3 eksamit (4, 8 ja 12 nädal)
- Eksamite järel on vaja täita eneseanalüüsi vorme.
- Aine on üsna detailselt kirjeldatud ja ülesannete nimed TopCoderi andmebaasis nädalate kaupa.



Mis oli kõige kasulikum eksamiks?

Describe some insights, shortcuts or algorithmic tricks that were most valuable to you in your exam or your practice before the exam.

Mis tegi päeva produktiivseks?

Think back to the most productive day or week that you have recently had.
Write down the things that you did (or didn't do) that made it productive.

Skill Category	Description	Cited
Design	make a plan or write an strategy, seek alternatives	44%
Algorithmic	identify problem type, recursion, graph algorithms	39%
Coding	faster coding, C++ fluency, better code structure	34%
Problem solving	generic skills to approach problems, self-learning	12%
Other SE	Testing, documentation, team work, communication	8%

8. Nädal: Mis on arenenud?

What software development skills do you think you have developed most as a result of this course and by how much?

Before	**After**	Count
Just start coding straight away	Design my algorithm before coding	37%
Mostly/only use brute force	Identify best algorithmic approach	37%
Stick to the first idea	Look problem from multiple perspectives	%11
Journal was a drag	Journal helps design by keeping track of my thoughts	7%
Not a fan of recursion	Confident to use recursion when needed	5%

Lõpus: Kas lähenemine on muutunud?

With the aid of two contrasting examples (before and after) briefly describe how your approach to algorithmic problem solving has changed during this course. Are these changes, if any, likely to persist?

Järeldused

Sügav Programmeerimine

- Sügav õppimine vajab mingil viisil sisemise motivatsiooni loomist ja selleks peab inimene ka uskuma sellesse, et saab hakkama! AKT aines on väga palju tööd vaja teha, et nad usuks endasse, aga…

- Üks asi on olla motiveeritud sügavalt õppida, aga teine asi on päriselt õppida sügavalt programmeerida. Isegi motiveeritud õppijad, kes panevad meeletu hulk aega ainesse, küsivad liiga palju näidislahendusi. Eksamil on meil ikkagi erinev ülesanne. Siis paluvad ka selle näidislahendusi…

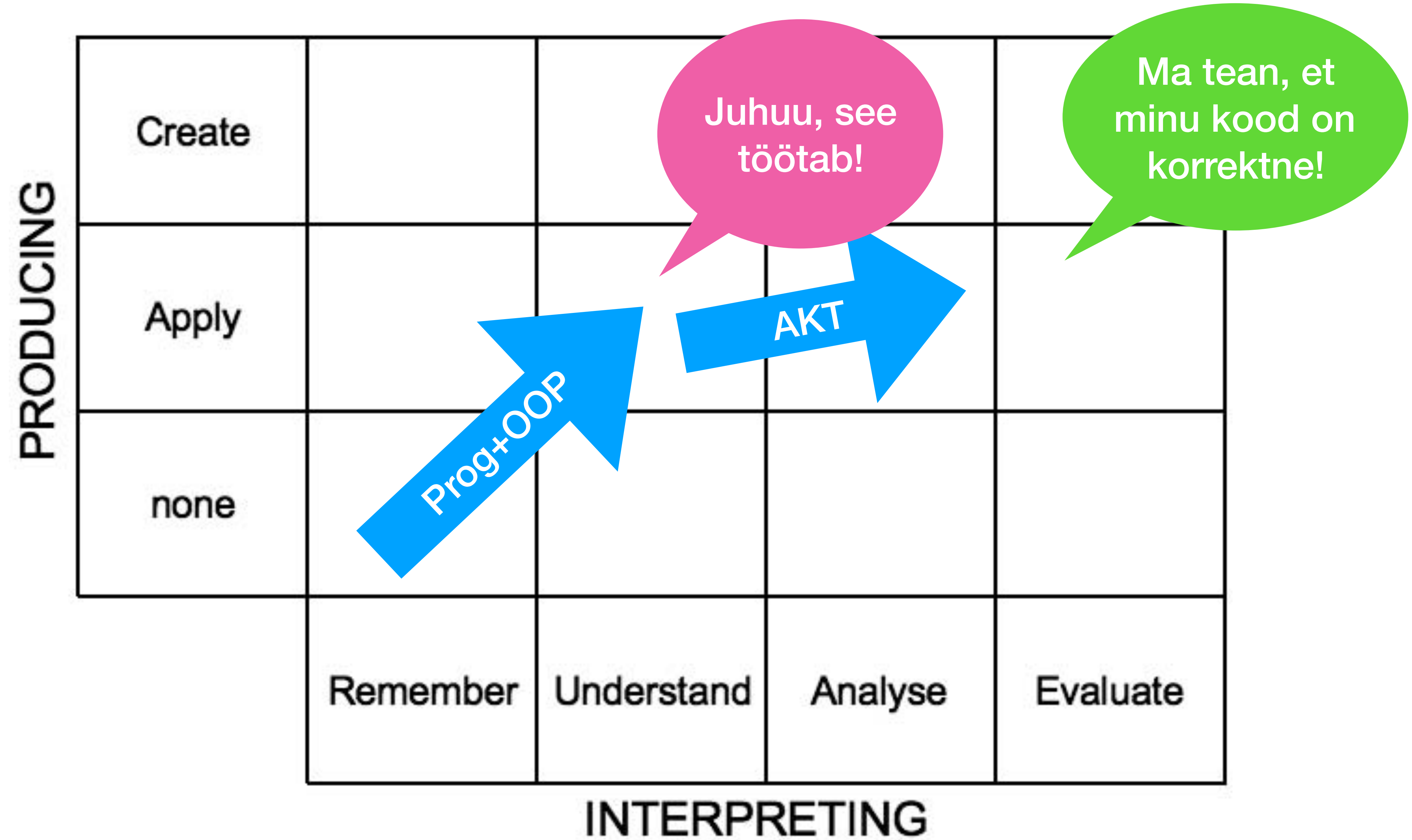
- Kuidas motiveerida neid näidete assimileerimise asemel põhiprintsiipe selgeks teha?

Järeldused artiklitest

- Meil võib kasuks olla natuke selgemalt mõelda, mis on iga aine roll programmeerimise mõttetasandite maatriksis. Mõned mõtted selle kohta järgmistel slaididel.

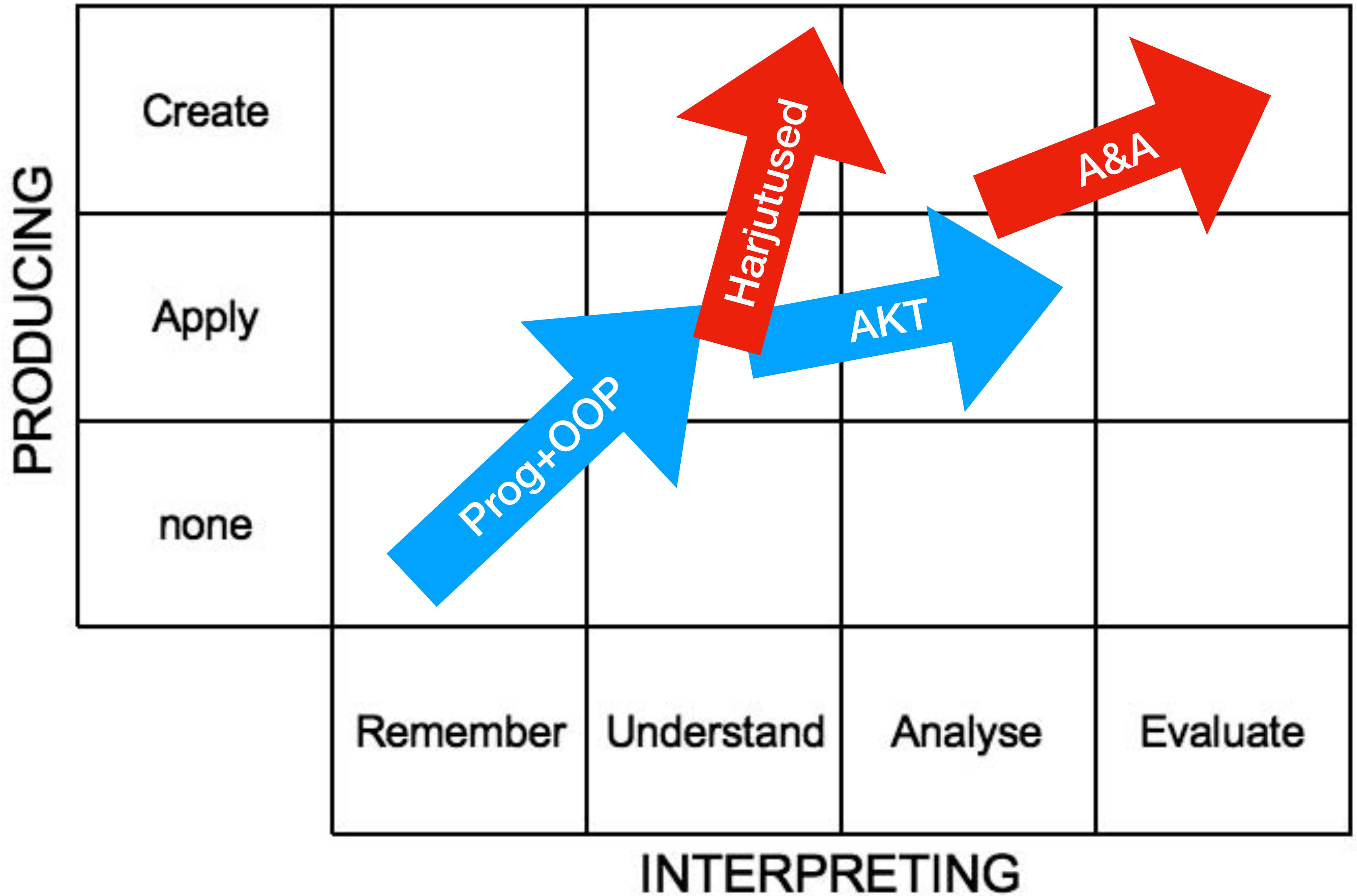
- Sügavam programmeerimise oskus on ikkagi väga raske õpiväljundina fikseerida. Head probleemilahendajad opereerivad ka oluliselt varem kõrgemal mõtlemistasandil.

- Me peame rohkem tähelepanu pöörama probleemi lahendamise oskuse õpetamise. Sealjuures on enesereflektsioon oluline. Kui eesmärk on mõjutada seda, kuidas nad probleemidele lähenevad, siis ei piisa ainult selles, et “las nad lahendavad rohkem ülesandeid”.



Mis on meie kursuste rollid?

Ma enne mõtlesin, et AKT liigub loomise teljes, aga tegelikult on ta pigem arusaamist arendav aine!



Suurem küsimus on algoritmika juures!

Kas näiteks harjutuste aines võtta peamine kohustus õpetada just probleemi lahendamise oskusi?

"Studying processes and problem solutions is very central to, if not the essence of, computer science. One could say that solving problems and producing an effective and efficient solution is the core goal of a computer science professional."

–Ursula Fuller et al.

Kui niimoodi mõelda ainete eesmärkide peale, siis peab muidugi rohkem resursse panustama Harjutuste ainesse.