Relationships: computational thinking, pedagogy of programming, and Bloom's Taxonomy

Cynthia C. Selby
University of Southampton
Highfield
Southampton UK
C.Selby@Southampton.ac.uk

Objective

- Find a relationship between
  - Computational thinking
  - The teaching of programming
  - Educational frameworks
- For what reason
  - **More effective teaching of computational thinking**
  - More effective teaching of programming
  - **More effective assessing of computational thinking**
  - More effective assessing of programming

Computational thinking skills don’t “just emerge”.

Caitlin (Yesterday)

Literature review

- A working definition of computational thinking

- A thought process
- Decomposition
- Abstraction
- Evaluation
- Algorithmic thinking
- Generalisation

> "Computational thinking means many things to different people." Simon PJ (yesterday)

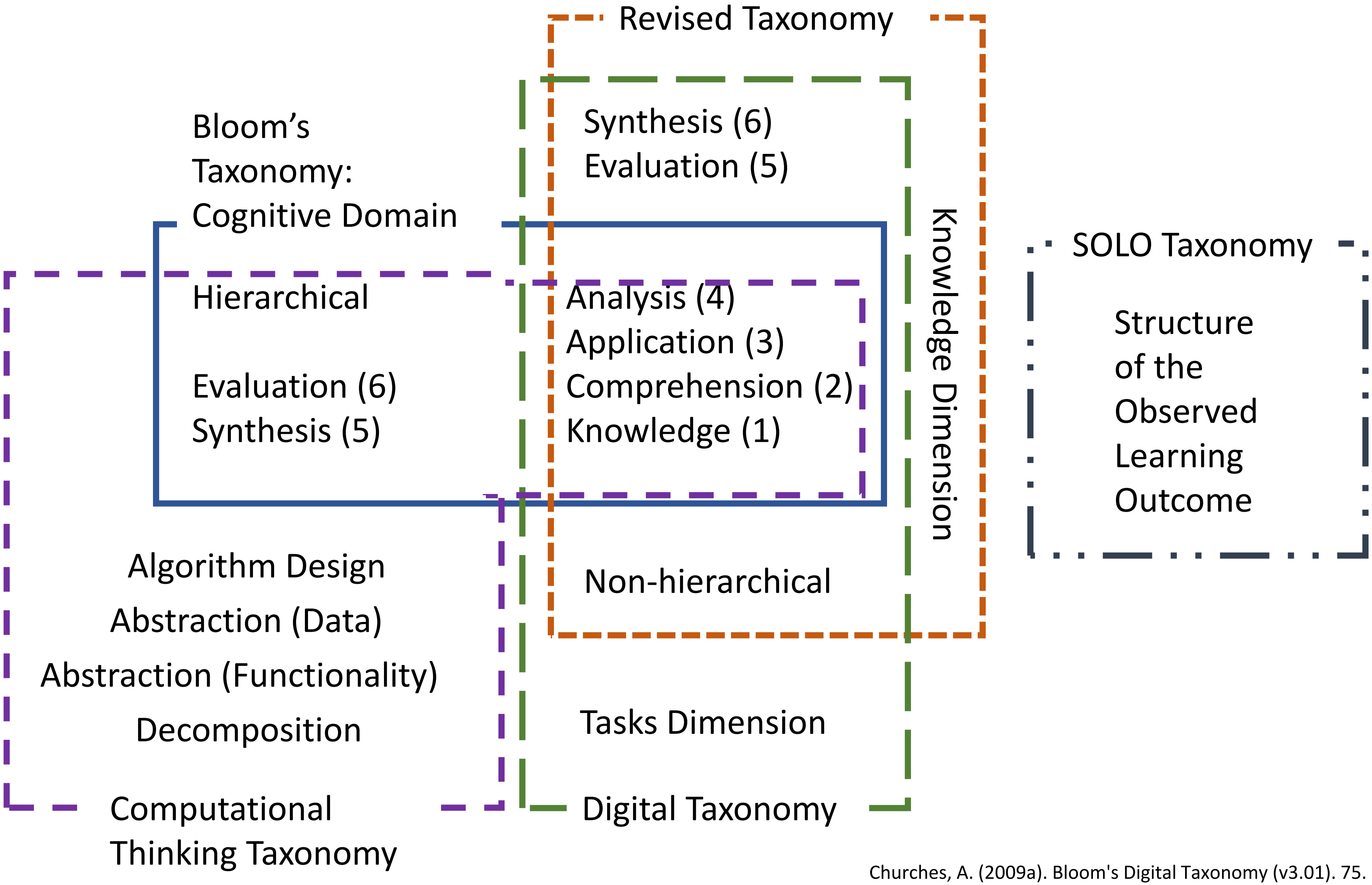
Papert, S. (1996). An Exploration in the Space of Mathematics Educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95-123.

Literature review

- What do we already know about those things that learners find difficult?

- Notional machine (model of the machine)
- Programming (reading, tracing, writing)
- Problem-solving skills

Conceptual frameworks



Revised Taxonomy

Bloom's Taxonomy: Cognitive Domain

Synthesis (6)
Evaluation (5)

Hierarchical

Evaluation (6)
Synthesis (5)

Analysis (4)
Application (3)
Comprehension (2)
Knowledge (1)

Knowledge Dimension

SOLO Taxonomy

Structure of the Observed Learning Outcome

Algorithm Design
Abstraction (Data)
Abstraction (Functionality)
Decomposition

Non-hierarchical

Tasks Dimension

Computational Thinking Taxonomy

Digital Taxonomy

Selby, C. (2014). *How can the teaching of programming be used to enhance computational thinking?* Doctoral, University of Southampton. Retrieved from http://eprints.soton.ac.uk/366256/

Churches, A. (2009a). Bloom's Digital Taxonomy (v3.01). 75. Retrieved from educational origami website: http://edorigami.wikispaces.com/Bloom's+Digital+Taxonomy

Justification: the teachers

- Bloom’s Taxonomy: cognitive domain
- Revised taxonomy
- SOLO taxonomy
- Digital Bloom’s Taxonomy

All

Some

None

The performance of a beginner in a task may be attributed to the analysis or synthesis levels, while performance in the same task may only evidence application for more advanced learners. Fuller, et al. (2007)

FULLER, U., JOHNSON, C. G., AHONIEMI, T., CUKIERMAN, D., HERNÁN-LOSADA, JACKOVA, J., LAHTINEN, E., LEWIS, T. L., THOMPSON, D. M., RIEDESEL, C. & THOMPSON, E. 2007. Developing a computer science-specific learning taxonomy. *SIGCSE Bull.,* 39, 152-170.

Participants

- Teachers:
  - 39 at post-graduate level
  - 43 at higher education level (age 18+)
  - 28 at post-16 level
  - 126 at combined secondary and post-16 level
  - 42 at secondary only
  - 19 at primary
- Others:
  - Non-teaching academics
  - Professional bodies
  - Industry
  - Awarding organisations

Methods

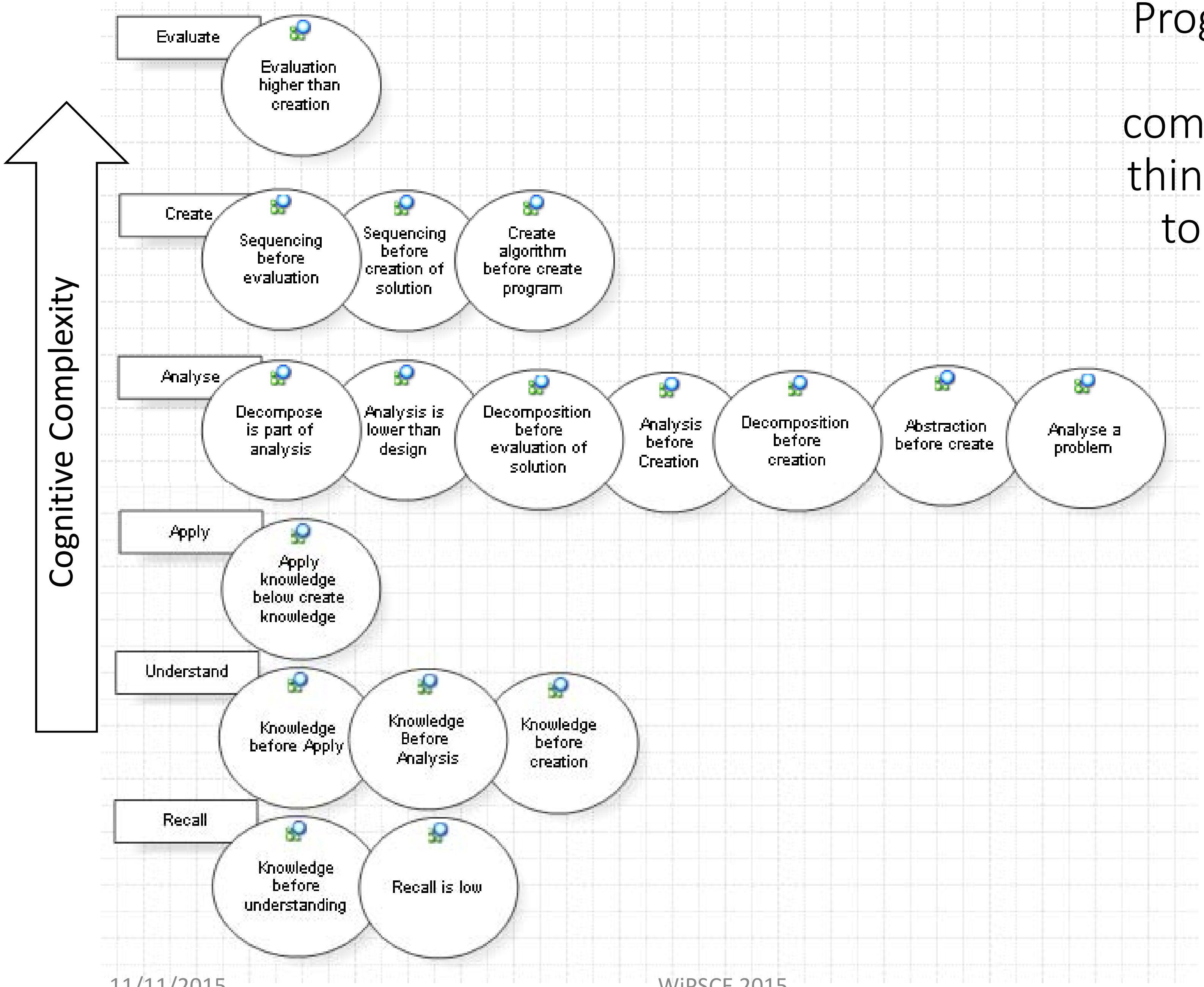
- Online questionnaire
  - Demographics
  - Programming
  - Computational thinking
  - Problem solving
- Community of practice online forum threads
  - All threads with subject lines indicating topics
- Interviews (teachers only)
  - Meaning of terms
  - How each appear in classrooms
  - Relationships
  - Comparisons

Grounded Theory

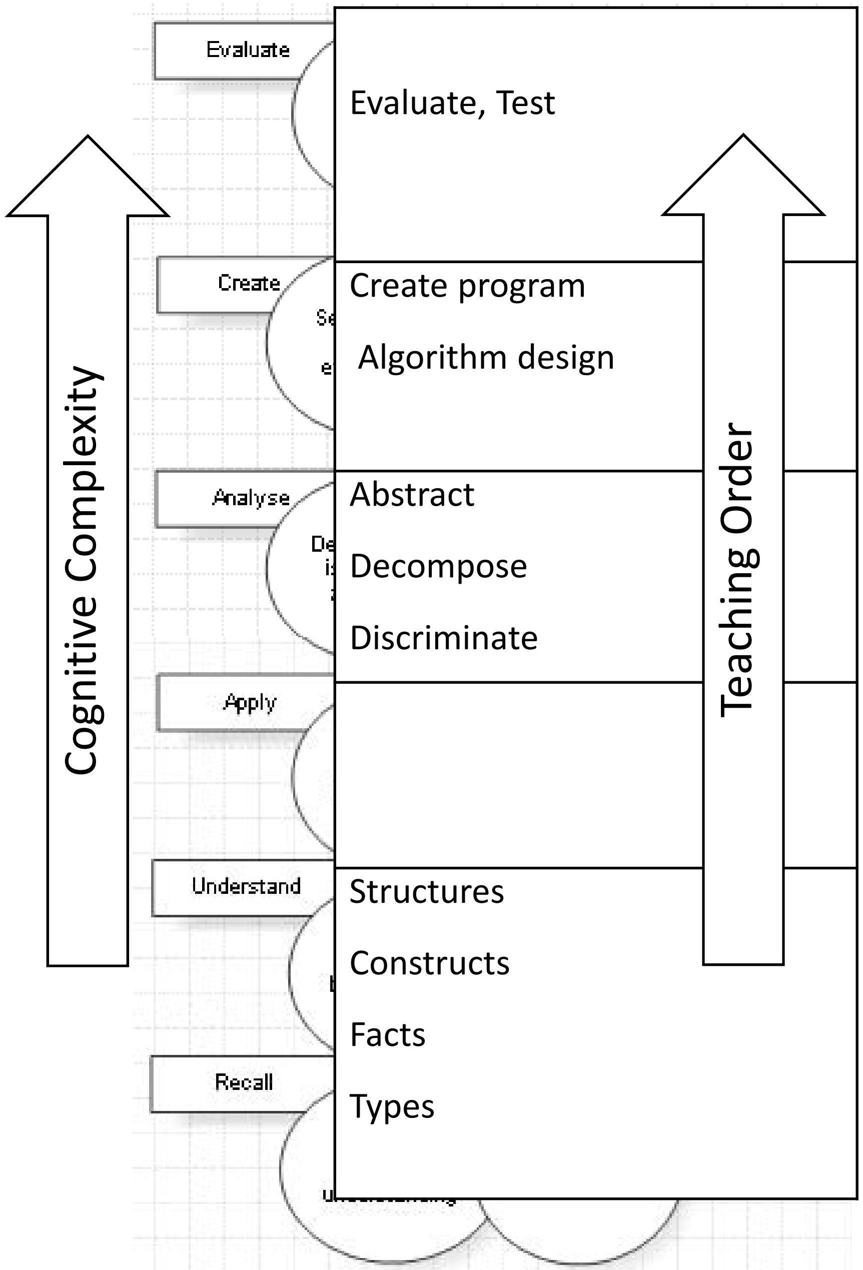
- Data collected simultaneously via all instruments.
- Qualitative (NVivo) analysis of data in line with Strauss and Corbin (1998) in cycles with data collection
- The model was amended to reflect newly collected data
- From first questionnaire to model took 16 months during 2012 and 2013

STRAUSS, A. & CORBIN, J. 1998. *Basics of qualitative research: Techniques and procedures for developing grounded theory,* London, Sage Publications Ltd.

Programming and computational thinking maps to Bloom's


Cognitive Complexity
Evaluate
Evaluation higher than creation
Create
Sequencing before evaluation
Sequencing before creation of solution
Create algorithm before create program
Analyse
Decompose is part of analysis
Analysis is lower than design
Decomposition before evaluation of solution
Analysis before Creation
Decomposition before creation
Abstraction before create
Analyse a problem
Apply
Apply knowledge below create knowledge
Understand
Knowledge before Apply
Knowledge Before Analysis
Knowledge before creation
Recall
Knowledge before understanding
Recall is low

Programming and computational thinking maps to Bloom's



Evaluate, Test

Create program

Algorithm design

Abstract

Decompose

Discriminate

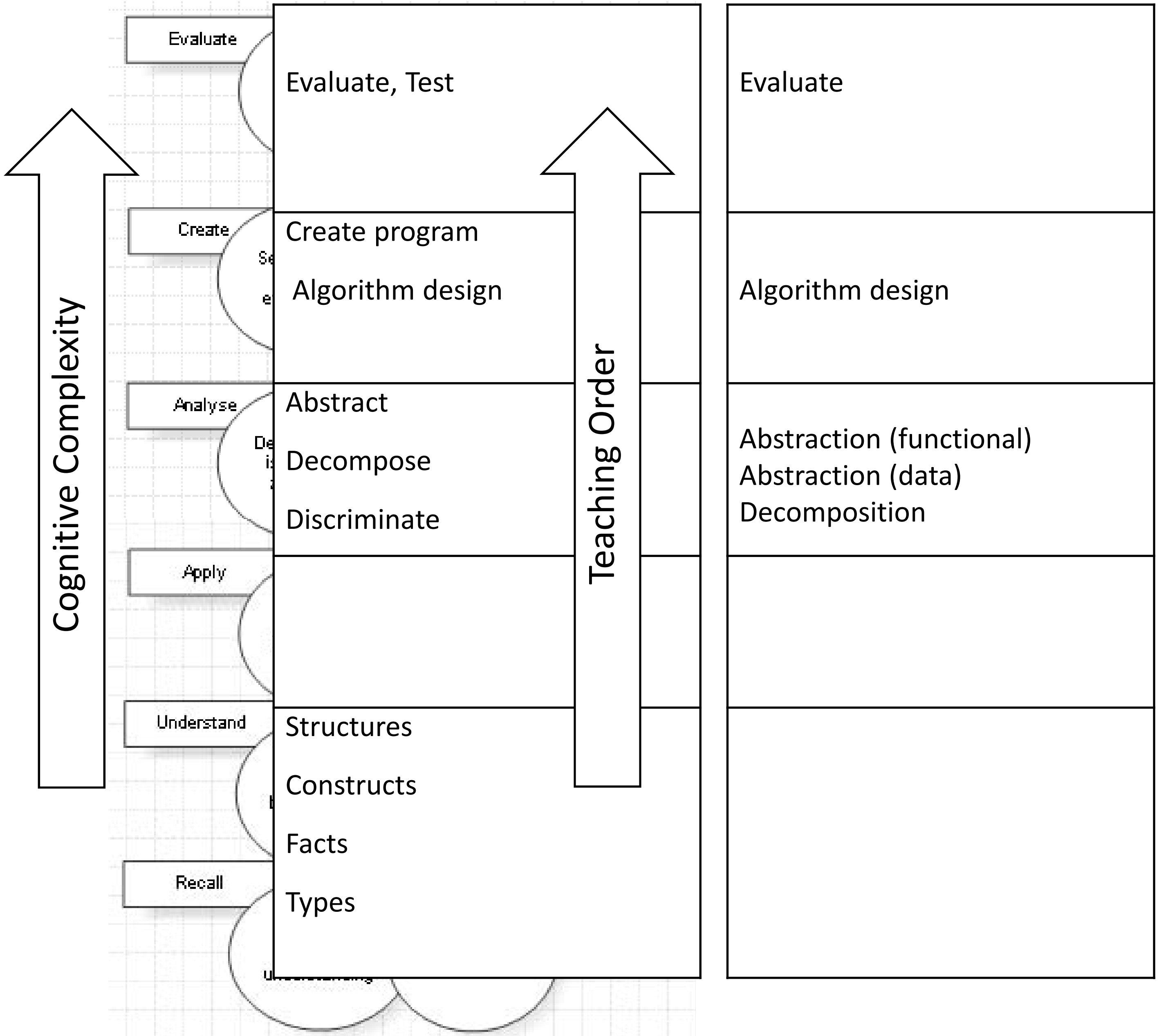
Structures

Constructs

Facts

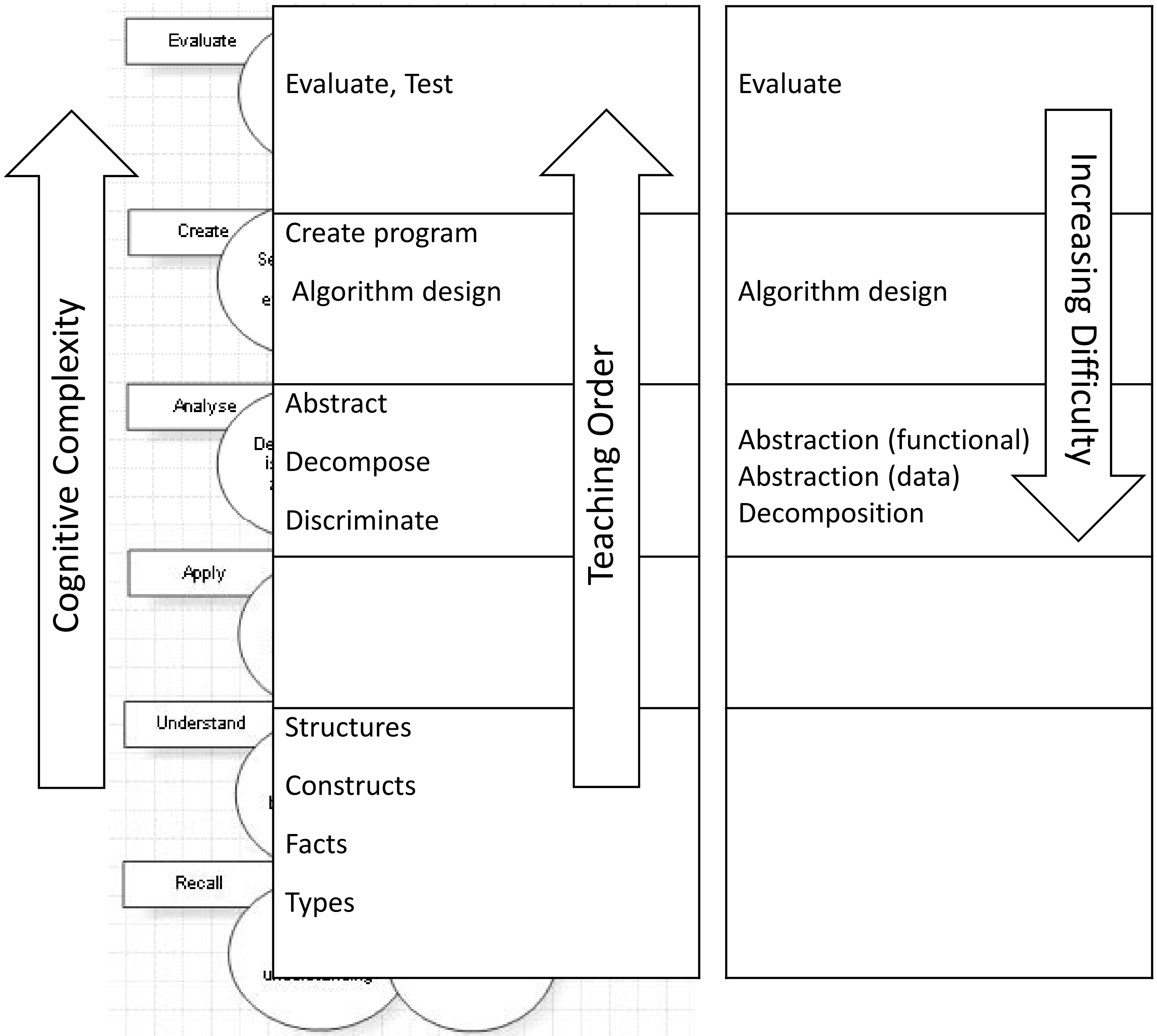
Types

Programming and computational thinking maps to Bloom’s

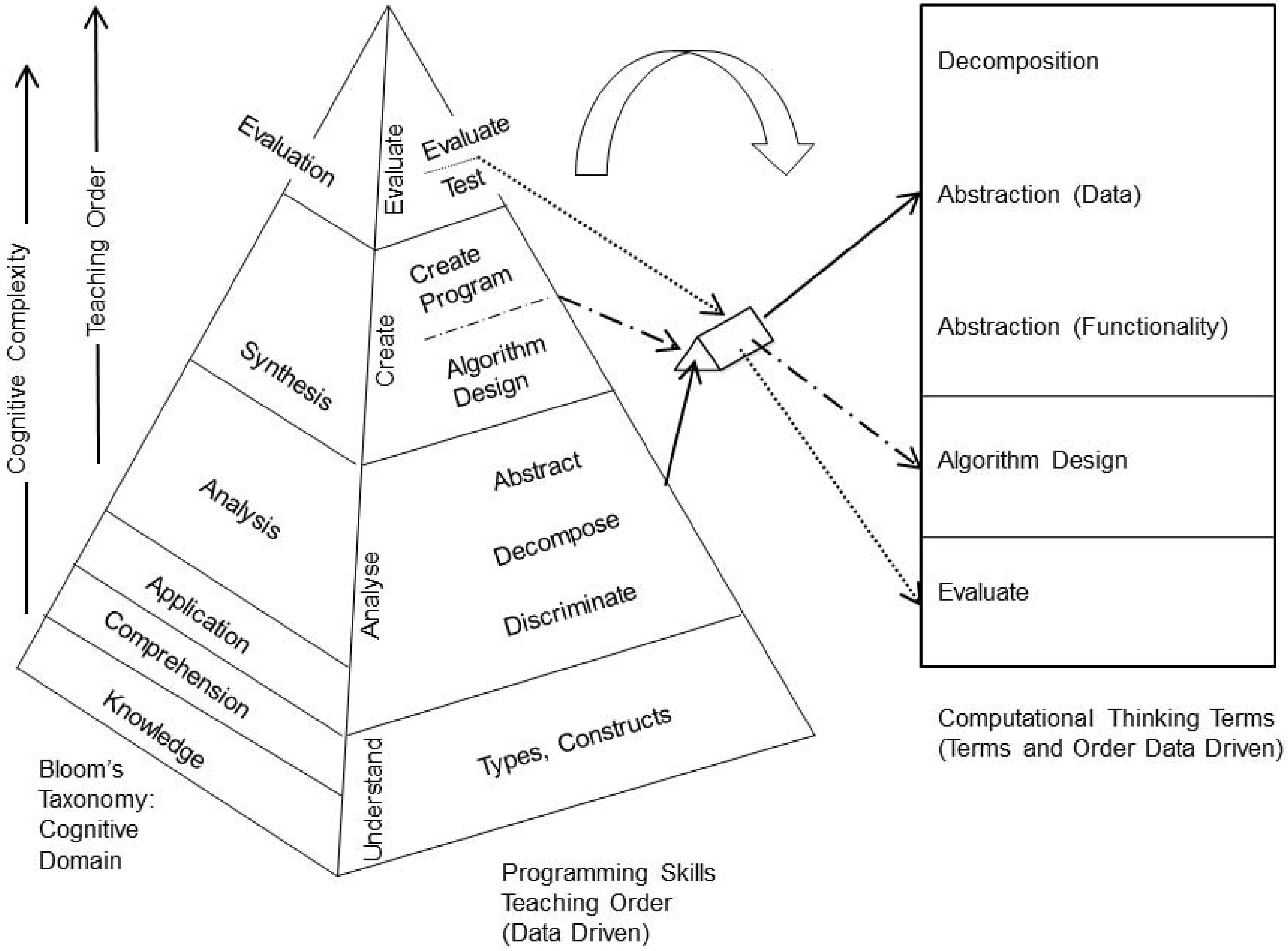


Evaluate, Test	Evaluate
Create program Algorithm design	Algorithm design
Abstract Decompose Discriminate	Abstraction (functional) Abstraction (data) Decomposition
Structures Constructs Facts Types	

Programming and computational thinking maps to Bloom's



Evaluate	Evaluate, Test	Evaluate
Create	Create program Algorithm design	Algorithm design
Analyse	Abstract Decompose Discriminate	Abstraction (functional) Abstraction (data) Decomposition
Apply		
Understand Recall	Structures Constructs Facts Types	



Bloom's Taxonomy: Cognitive Domain

Programming Skills Teaching Order (Data Driven)

Computational Thinking Terms (Terms and Order Data Driven)

Our classrooms

- Devote more time to developing the skill of decomposition before moving onto the other programming skills
  - Practice with both known and unknown problems
- Devote more time to developing the skill of abstraction, as applied to functionality and data

Thank you

Why is decomposition difficult?

- Some reasons suggested by the participants:

- A lack of experience
- Incomplete understanding of the problem to solve
- The order of teaching programming

Other interesting observations

- Debugging is consistently placed on the Analysis level, even though it may require some application and/or evaluation
- Where’s generalisation?
- Coding is generally taught before flowcharting. Can that right?
- Translation happens best in the order of brain → flowchart → pseudocode → code