Designing a Multi-Faceted SOLO Taxonomy to Track Program Design Skills Through an Entire Course

Francis Castro

Kathi Fisler

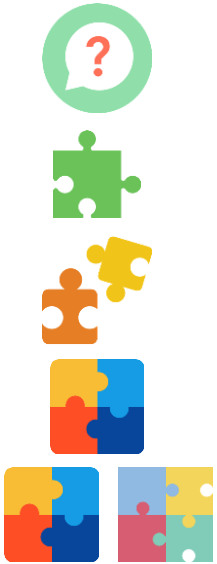
WPI

BROWN

How to **track** the development of students' programming **skills**?

How to **categorize** different "**levels**" of skill?

Structure of Observed Learning Outcomes (**SOLO**) captures different levels of complexity of learning outcomes in 5 levels:

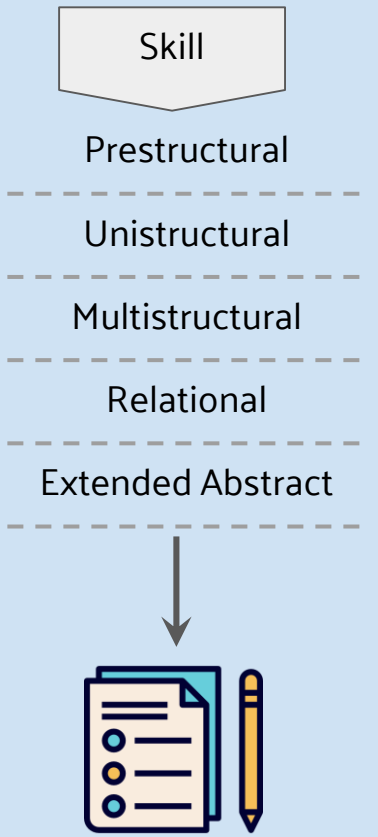


Pre-structural	No understanding
Uni-structural	Understand a single aspect
Multi-structural	Understand several aspects independently
Relational	Inter-operation of several aspects
Extended Abstract	Generalize to a new domain

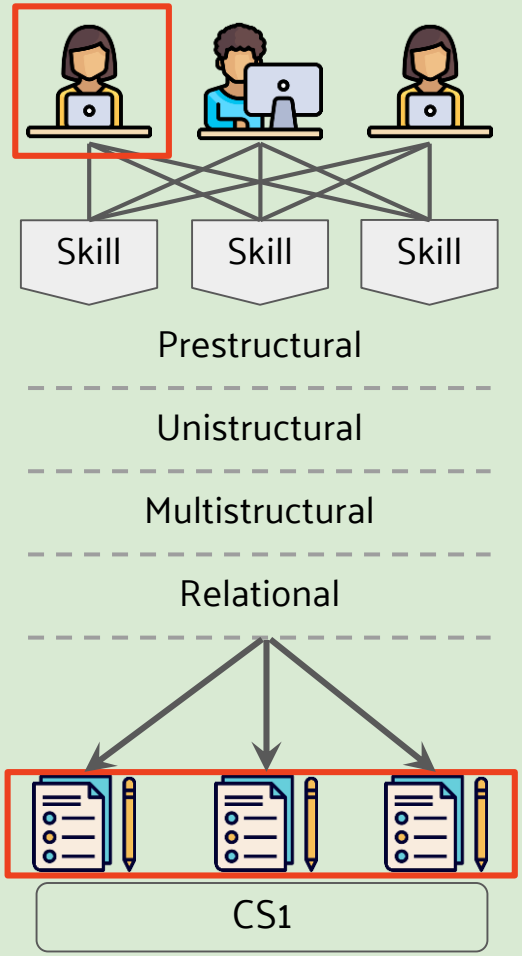
Identify a skill to assess

Define a SOLO taxonomy for the skill

Apply the taxonomy to student work on a single assessment



Build taxonomy from student data



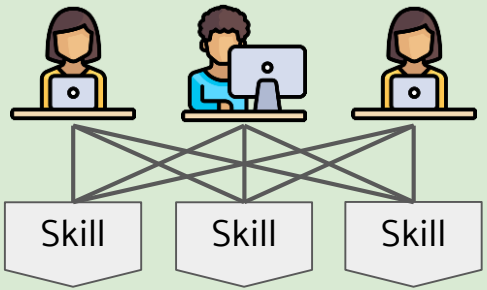
Multi-dimension taxonomy to capture multiple skills

Use the taxonomy to assess student progress across multiple assessments in a full course

Context of the data: CS1 course

Building the taxonomy

Lessons learned in applying the taxonomy to assess student progress

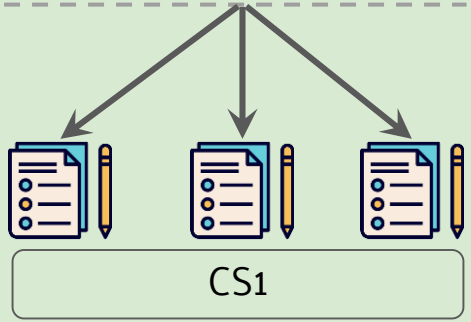


Prestructural

Unistructural

Multistructural

Relational



Build taxonomy from student data

Multi-dimension taxonomy to capture multiple skills

Use the taxonomy to assess student progress across multiple assessments in a full course

Course context A 7-week **CS1** course, based on **How to Design Programs**, with programming in Racket (Scheme variant)

Design recipe

Example:
Write a function to sum a list of numbers

*Note:
Semicolon (;) used for comments

Describe the shape of input

```
; A list-of-number is
; - empty or
; - (cons number list-of-number)

(define even-nums (cons 2 (cons 4 (cons 6 empty))))
```

Describe the function behavior

```
; sum-nums : list-of-numbers -> number
; Produces the sum of all numbers in the list
```

Function examples

```
(check-expect (sum-nums even-nums) 12)
```

Function template based on input type

```
; List Template
; (define (list-fxn list-input)
;   (cond [(empty? list-input) ... ]
;         [(cons? list-input) ... (first list-input)
;                                 (list-fxn (rest list-input)) ... ]))
```

Function details

```
(define (sum-nums nums-list)
  (cond [(empty? nums-list) 0 ]
        [(cons? nums-list) (+ (first nums-list)
                              (sum-nums (rest nums-list)))]))
```

Participants: **13 student volunteers**, distributed across first exam grades (A=6, B=3, C=4)

Study session **every 2 weeks** (starting after the first exam), **3 sessions** per student

Session 1	**Activity**	Interview* on homework problem
	Topic	List of tuples/structures (sum cost of ads for a political candidate)
Session 2	**Activity**	Interview* on homework problem
	Topic	n-ary trees (check oxygen levels in a river system)
Session 3	**Activity**	Think-aloud and post-interview*
	Topic	Rainfall (average non-negative numbers from a list until sentinel)

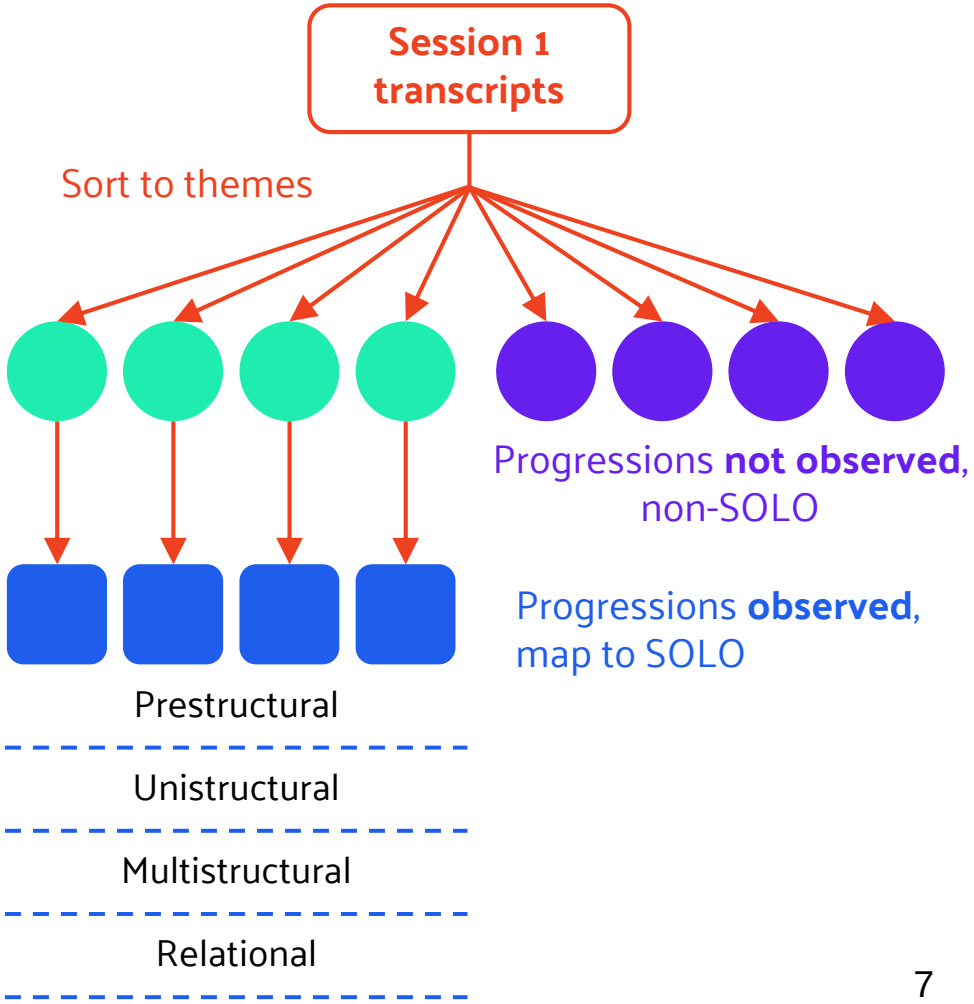
*Interview questions asked students to describe how they approached problems and their use of the design recipe

Building the taxonomy

Open-coded **Session 1 data** from a random sample of **6 students** to identify themes (2 per grade bin)

Within some themes, comments suggest a **progression of conceptual complexity** akin to SOLO

In others, no observable core skill* (beyond scope)



The taxonomy

Rainfall: sum, count, divide

	Methodical Choice of Tests and Examples	Writing and Evaluating Function Bodies	Decomposing Tasks and Composing Solutions	Leverage Multiple Function Representations
P	Cannot write tests	Cannot write functions	Does not identify relevant tasks	Just dives into writing code
U	Can't explain purpose of written tests	Primitive operations on primitive types	Identifies tasks, but no logical separation	Mechanical use of design recipe
M	Multiple but unrelated tests	Complex expressions, no function semantics	Decomposed tasks, no semantic composition	Representations seen within problem context
R	Tests cover a problem space	Semantics of function calls & return contexts	Semantic decomposition & composition of tasks	Mechanisms of how representations relate

Syntactic

Semantic

The relational level establishes logical connections between schema/artifacts from prior levels

We applied the taxonomy to **all transcripts** --- 3 sessions x 13 students

Observations:

- We successfully categorized students' data using the taxonomy, even beyond the sample (validation)

student5

Sess. #	Tests	Fxns	Decmp	FxnRep
1	R	M	U	U
2	R	R	R	U
3	R	R	R	M

- Students show being at different levels for different skills at a given time

✓ Multidimensional taxonomy captures variances in ways skills develop

We applied the taxonomy to **all transcripts** --- 3 sessions x 13 students

Observations:

student1

Sess. #	Tests	Fxns	Decmp	FxnRep
1	U	M	U	U
2	R	R	M	U
3	U	M	M	U

- Some students show non-linear progression of skills through the sessions
  - Skills may not have been internalized well
  - Problems may push students towards particular levels
  - Drops may reflect the problem complexity at which students can apply skills

Takeaways

- Multidimensional taxonomy can capture (1) **different skills** and (2) **variances in ways students develop** in these skills

- 2 ways of using the taxonomy across a course:
  - ✓ Give a set of problems students attempt at multiple points in a course and use the taxonomy to gauge skill levels at each point
  - ✓ (This work) Give a sequence of increasingly difficult problems, apply the taxonomy, and look at whether (1) students can scale skills or (2) skills break down at certain levels of complexity

- We need to be aware of **subtleties in the design of problems** when drawing conclusions about students' progress using the taxonomy

Looking ahead

- Does a student's level in one skill indicate or depend on progress in other skills?

- Validation: does the taxonomy differentiate students in similar ways as instructors would?

Questions, comments, and
touristy recommendations in Finland :)

fgcastro@cs.wpi.edu

@_franciscastro_