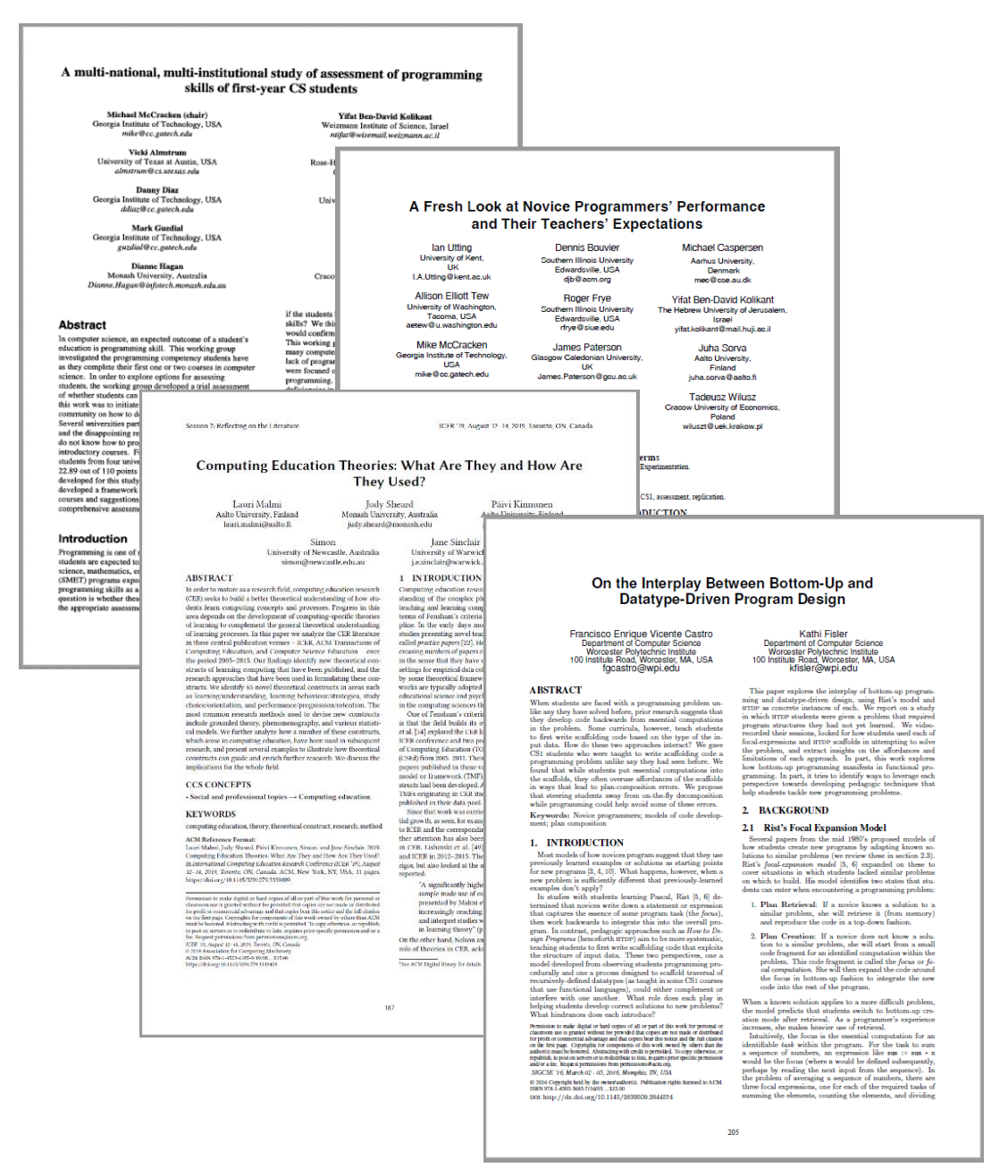
Coding the Coder: Developing Models of Novice Programmer Cognition

Francisco Castro
Worcester Polytechnic Institute

PROBLEM

Novices continue to find learning to program very challenging

> "We don't really know what makes programming so hard, and we don't yet have enough theory to explain why things work when we get it right."
>
> - Mark Guzdial, Computing education researcher, Communications of the ACM, Jan. 2018



- Multi-national, multi-institutional studies continue to show that novices find learning to program **challenging**
- The current theoretical infrastructure on the learning and teaching of programming is **young** and **limited**
- This **limits** our understanding of **how to build effective tools and instruction** that enhance programming education

Why should we teach programming well?

Programming is a skill used in various disciplines

Understanding code provides people with resources to **understand and question the design decisions used on technologies people use**

Programming is a skill with **economic value**:
Access to this economically-valuable skill is a **social justice issue**

If we want to develop tools and interventions for enhancing programming education, these must be built around viable models of how learning and thinking around programming actually works

METHODS

Field studies with college students learning programming

WPI · Northeastern University · BROWN

I used methods from social science, cognitive science, and learning science to explore and understand learners' programming processes and experiences

Think-aloud studies

Learners verbalized their thought-processes as they wrote programs

Interviews

Interviewed learners about their experience of learning programming

Ethnographic studies

Observed learners during programming classes, labs, and instructor office hours

Video Analysis

Analyzed recordings of learners' programming in programming environments (IDEs)

TAKEAWAYS

Analysis of programmer data led to the development of a model of programmer cognition that captures the different levels of thinking involved in programming and the interactions between these elements of thinking

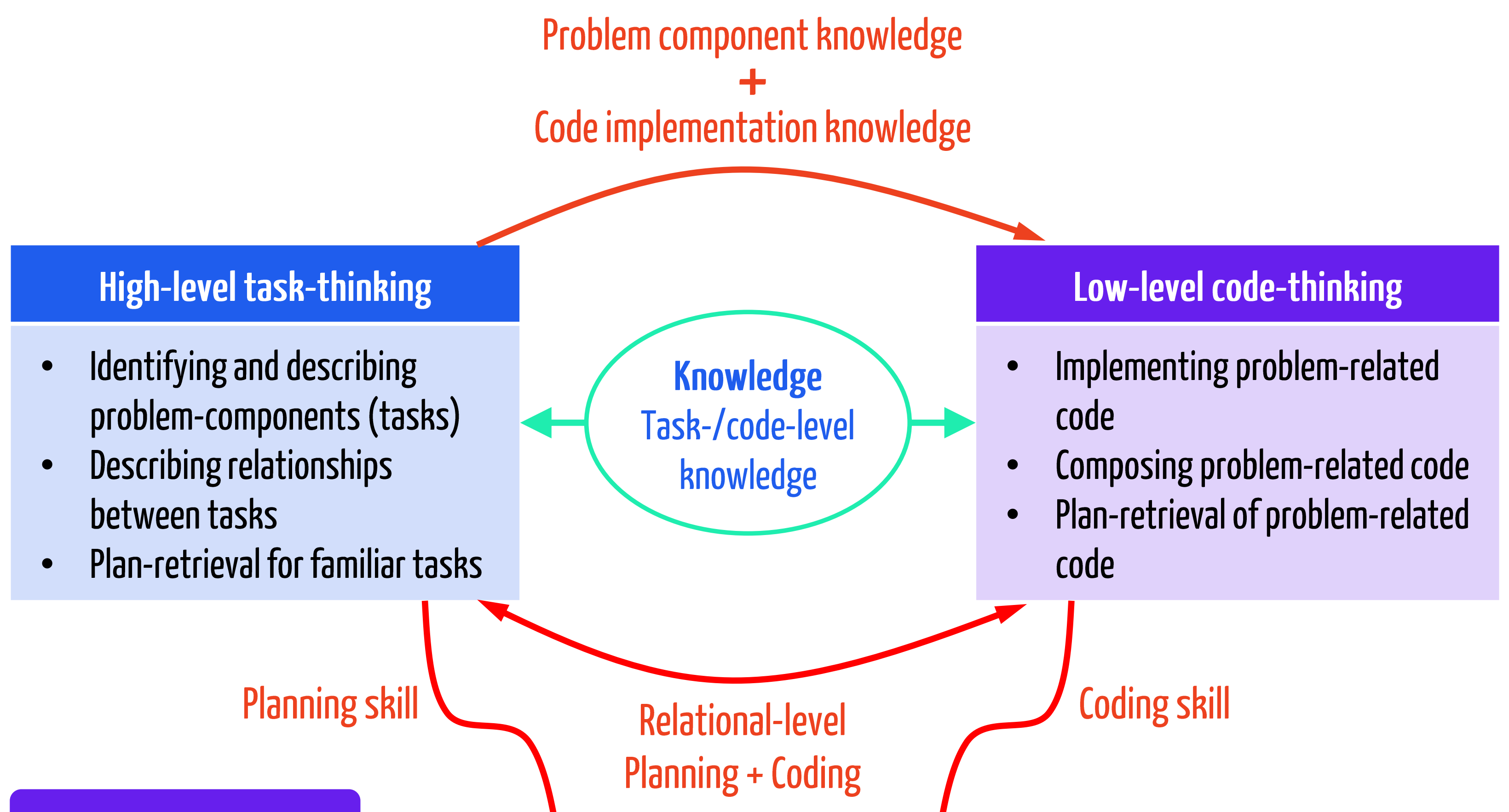
- Learners move between abstract problem representations (task-level) & concrete representations (code-level)

- Learners engaged in each level of thinking at different sub-levels of complexity

- When programming solutions:

  Successful learners tend to engage in both high- and low-level thinking at the Relational level

  Less successful learners tend to focus at the code-level and have difficulty reaching the Relational level



Problem component knowledge
+
Code implementation knowledge

High-level task-thinking

- Identifying and describing problem-components (tasks)
- Describing relationships between tasks
- Plan-retrieval for familiar tasks

Knowledge
Task-/code-level knowledge

Low-level code-thinking

- Implementing problem-related code
- Composing problem-related code
- Plan-retrieval of problem-related code

Planning skill

Relational-level
Planning + Coding

Coding skill

Skill Levels

Prestructural	Does not identify relevant tasks	Cannot define functions
Unistructural	Identifies tasks, but no logical separation	Call primitive operations on primitive types
Multistructural	Decomposed tasks, no semantic composition	Define code structures, no semantic composition
Relational	Semantic decomposition & composition of tasks	Define code structures and semantic composition

Some open questions

How might programming language design influence the model? (e.g. blocks-based)

How can we leverage insights from the model to design learner-centered programming tools and curricula?

How well does the model account for learners from different contexts (e.g. K-12, professional training)?

Who benefits?

Educators, Policy-makers
Ground curricular design, learning standards, and assessments on a viable model of how learning (among novices) work

Computing researchers
Provides a theory that helps explain findings and extend current research about CS learners, tools, and curricula

Educational Technologists, Programming tool designers
Design experience-enhancing tools that are learner-centered

Programming learners!

About Me

I am a PhD candidate at the WPI Department of Computer Science and a Visiting Researcher at the Brown University Department of Computer Science.

My research and interests revolve around understanding human cognition and behavior in people's use of computational tools and technologies.

I explore how people develop mental models about computational tools, how these mental models shape their interactions and workflows with their tools, and how people's environments and contexts influence these interactions.

Contact details

Website	franciscastro.github.io
Email	fgcastro@cs.wpi.edu
Twitter	@_franciscastro_
LinkedIn	linkedin.com/in/fevgcastro

Acknowledgments
This work is supported by US National Science Foundation grant no.s 1116539 and 1500039.