# PRINCIPLED ASSESSMENT OF COMPUTATIONAL THINKING

**Supporting Computer Science Teaching, Learning & Adoption Through Evidence-Centered Assessment**

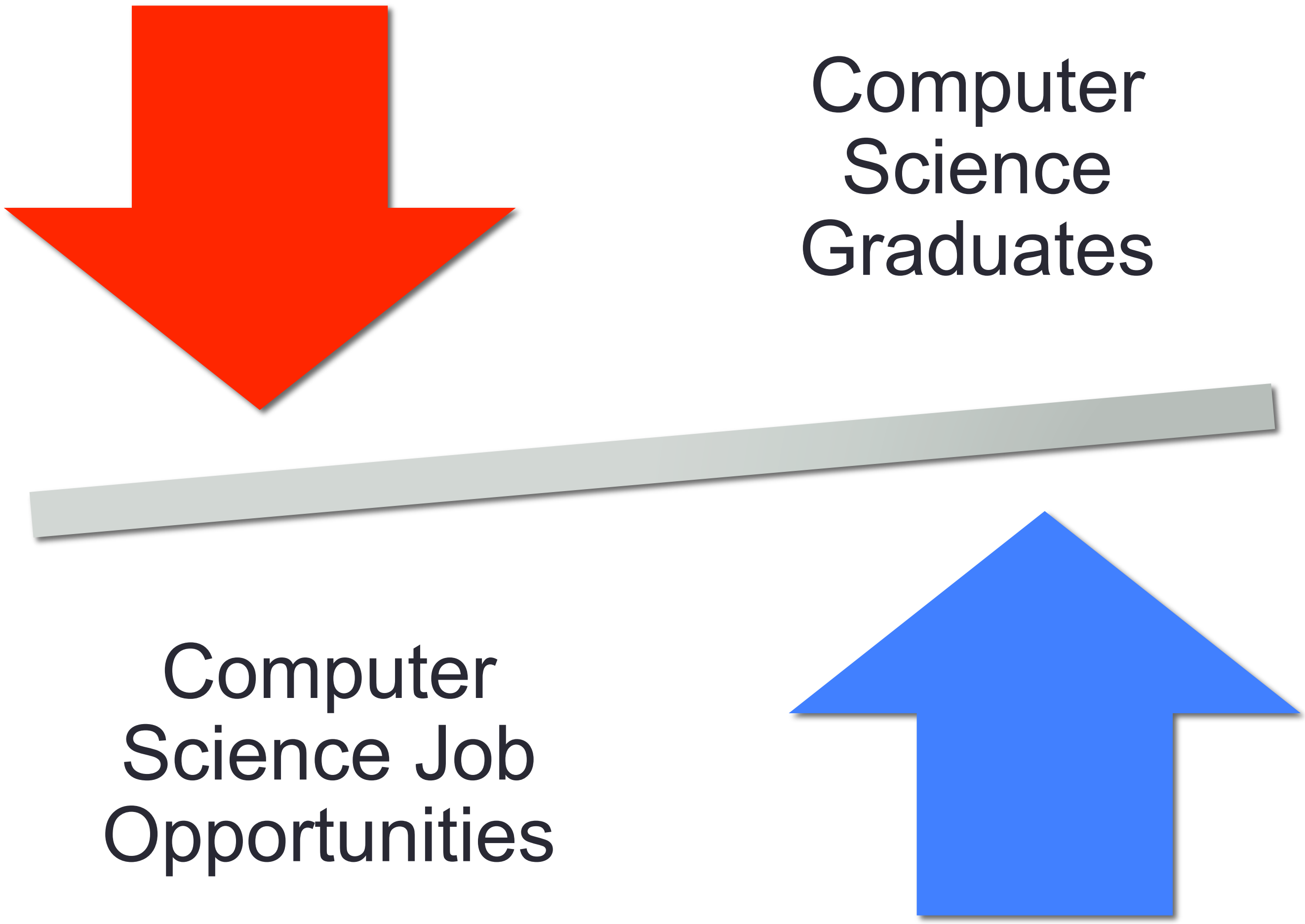
July 19th, 2013
University of California at Berkeley

Eric Snow
Center for Technology in Learning
SRI International

# Overview

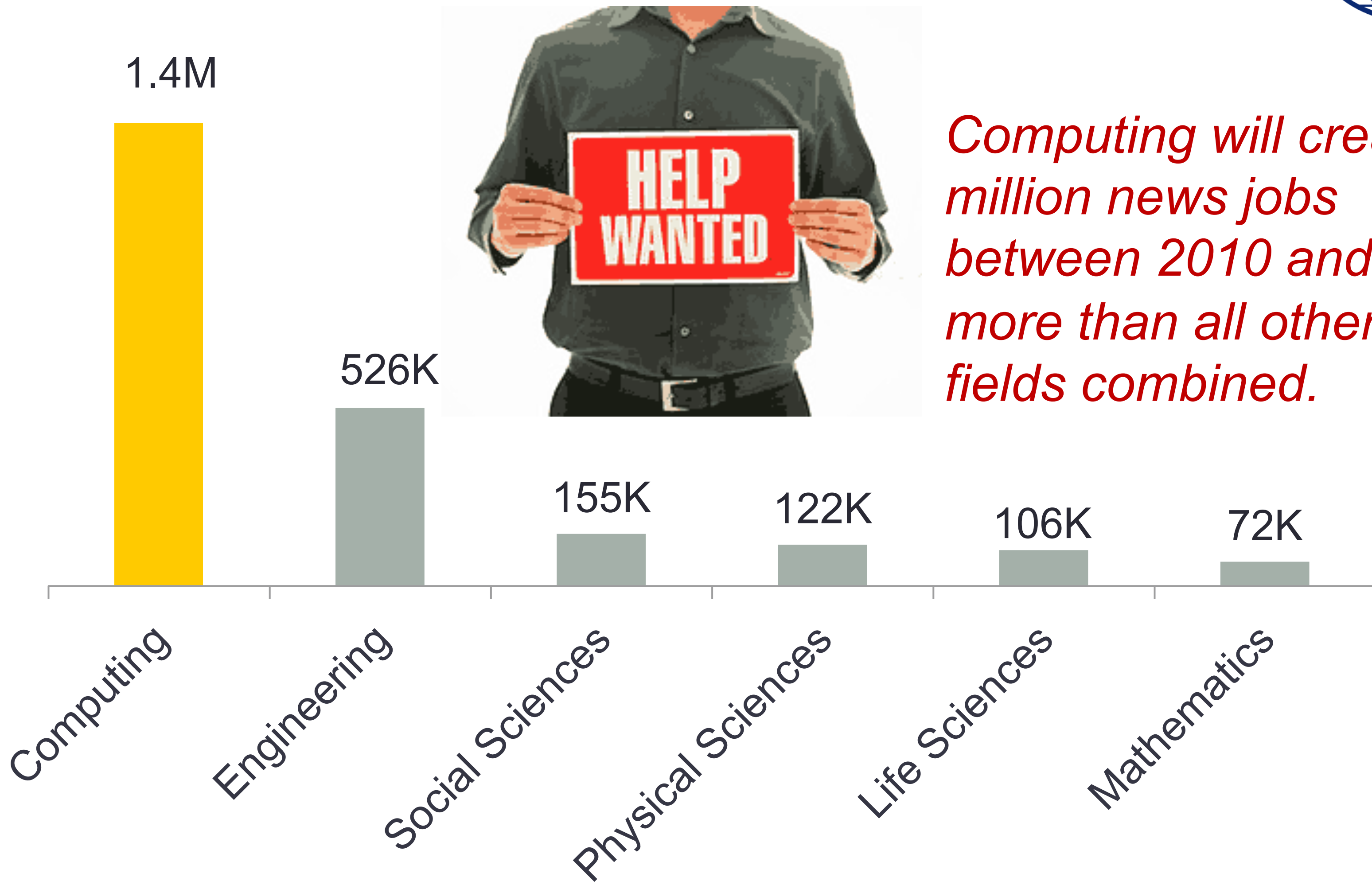
- Computer Science Education
  - United States & California
- Computational Thinking Practices
- Measuring Computational Thinking Practices
  - Principled Assessment of Computational Thinking (PACT)
  - Evidence-Centered Assessment Design (ECD)
  - Assessment Design Patterns

# Computer Science Education




*Source: Slides adapted from Dan Lewis, Santa Clara University*

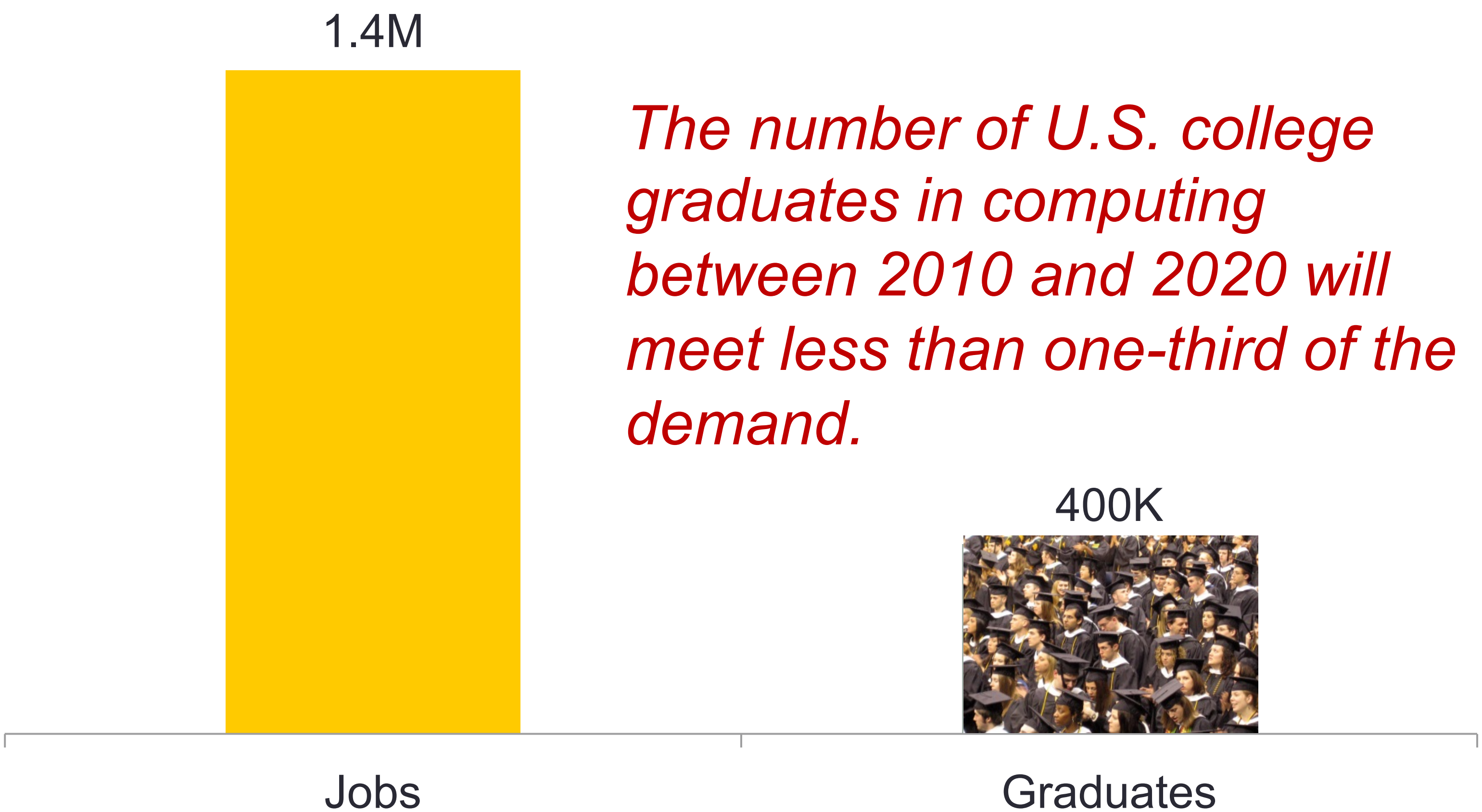
# Employment Projections




*Computing will create 1.4 million news jobs between 2010 and 2020 - more than all other STEM fields combined.*

*Source: Bureau of Labor Statistics, Employment Projections 2010-2020*

# Jobs vs. Graduates

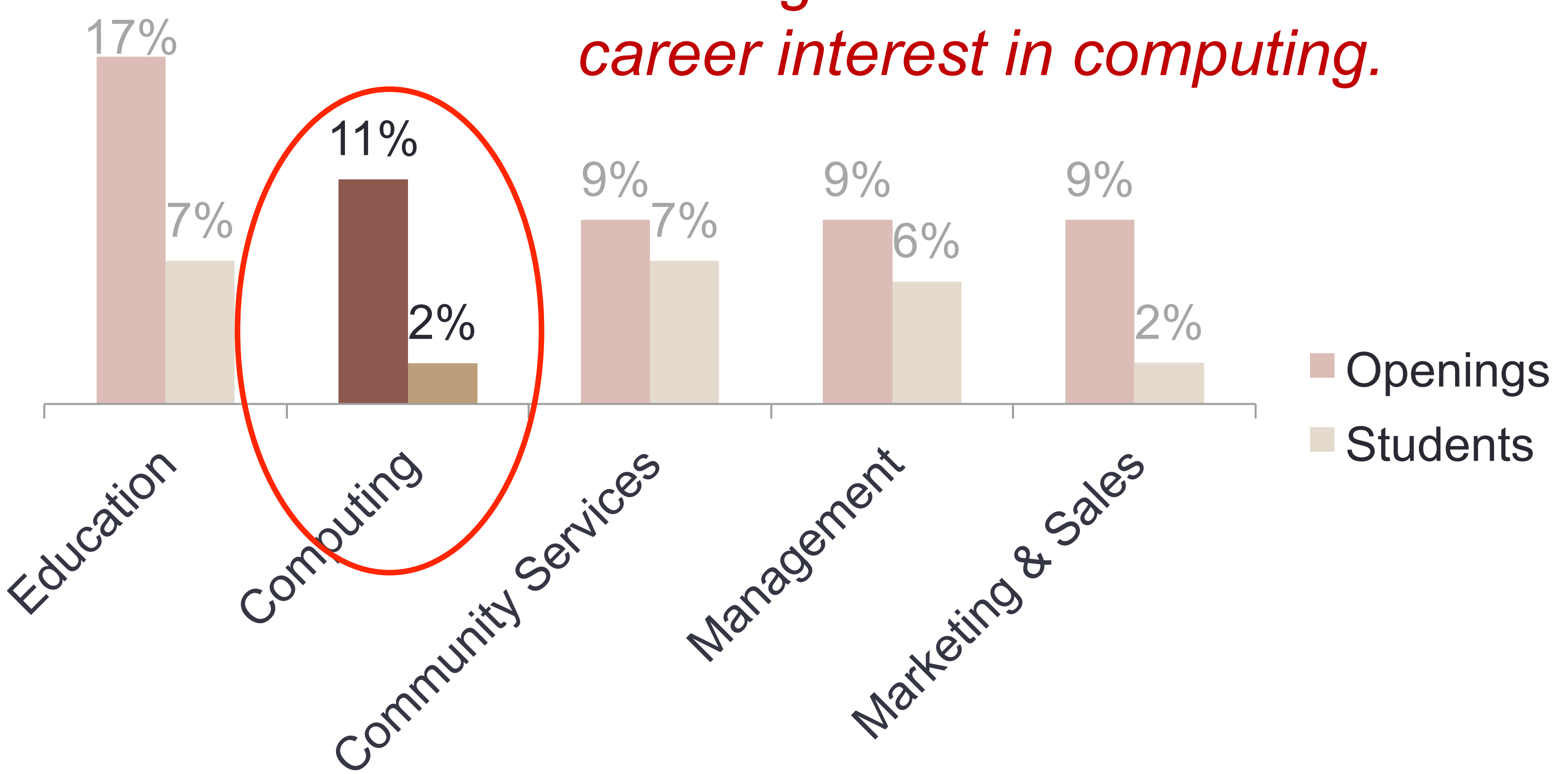



*The number of U.S. college graduates in computing between 2010 and 2020 will meet less than one-third of the demand.*

*Source: NCWIT, By the Numbers*

# Fastest-Growing Careers vs. Student Career Interests

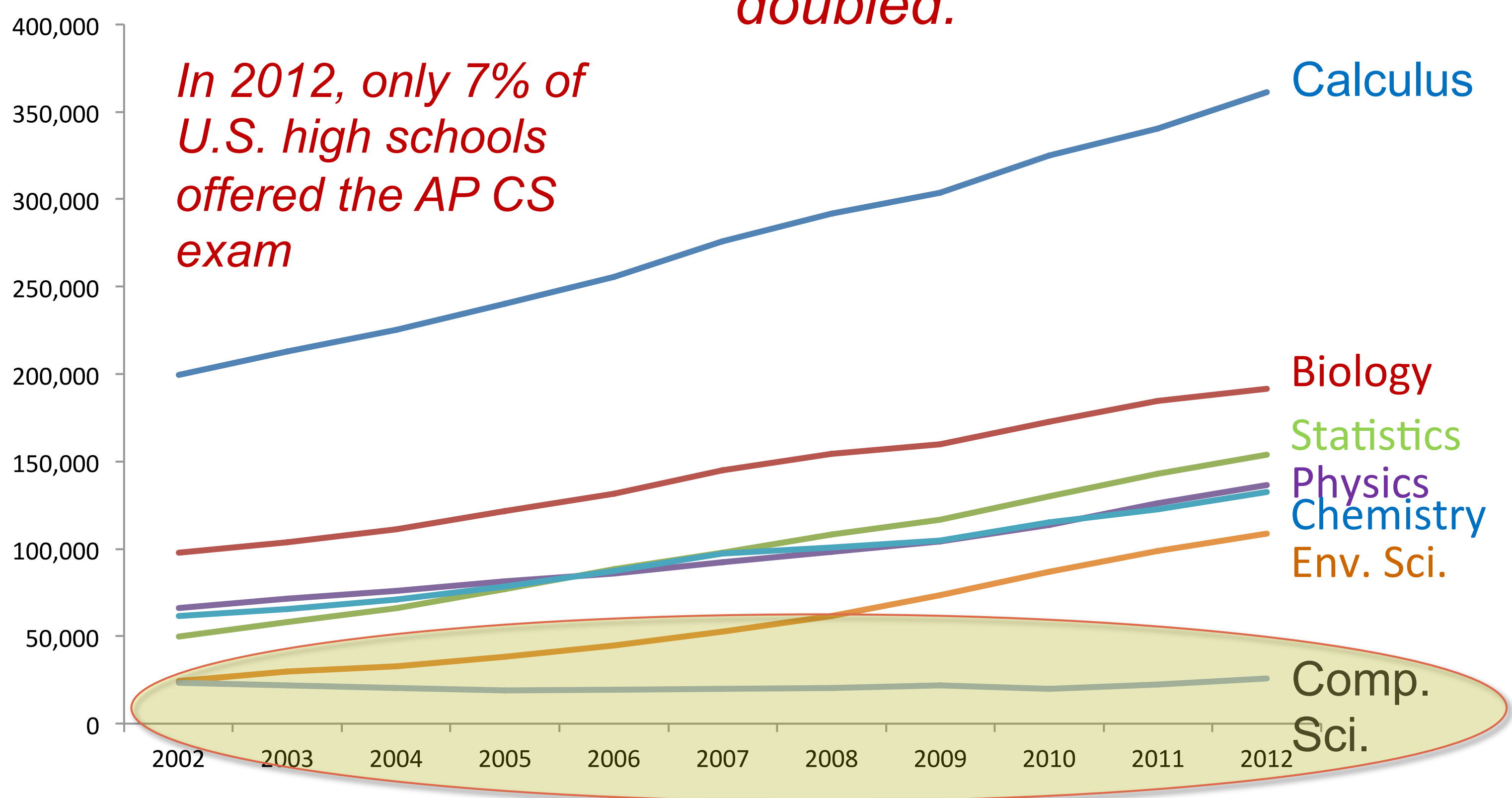
*Only 2% of ACT-tested high school graduates indicated a career interest in computing.*




*Source: ACT Research, The Condition of College & Career Readiness, 2012*

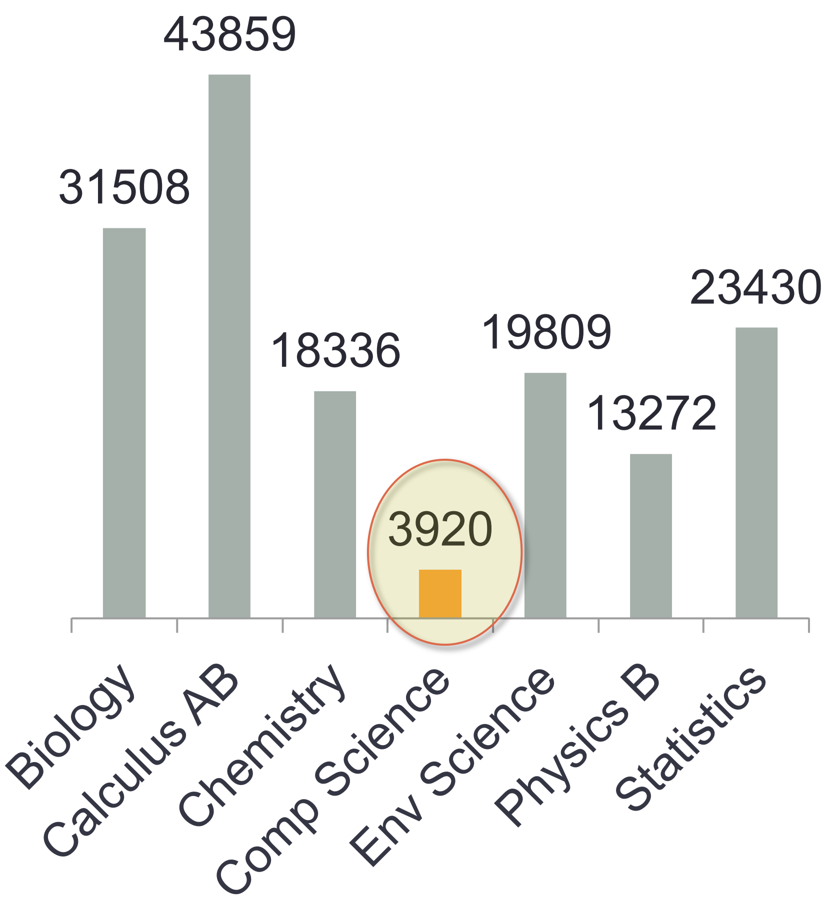
# AP Test Takers

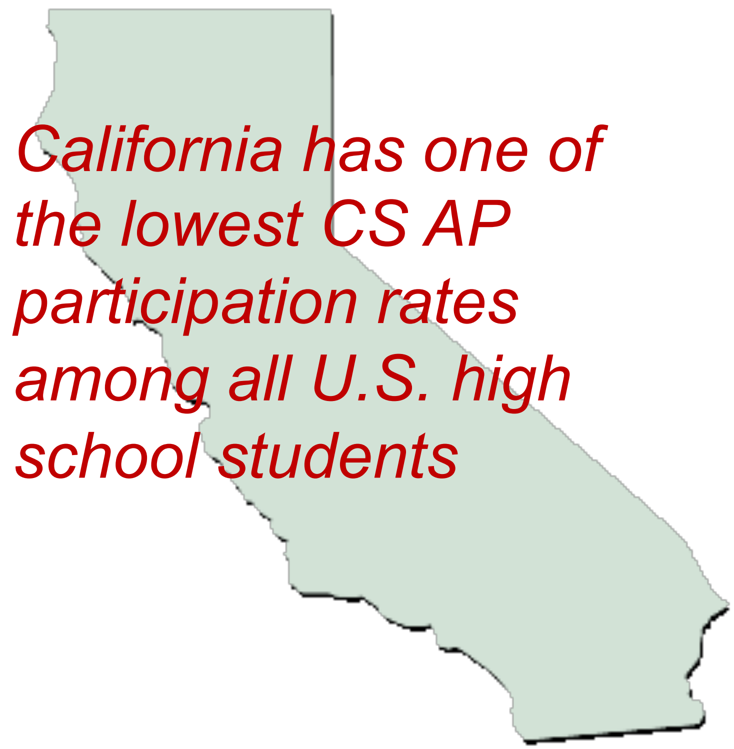
*The number of AP CS test takers has remained relatively unchanged while the total of all STEM fields more than doubled.*




*Source: College Board, AP Examination Volume Changes, 2002-2012*

# California AP Test Takers

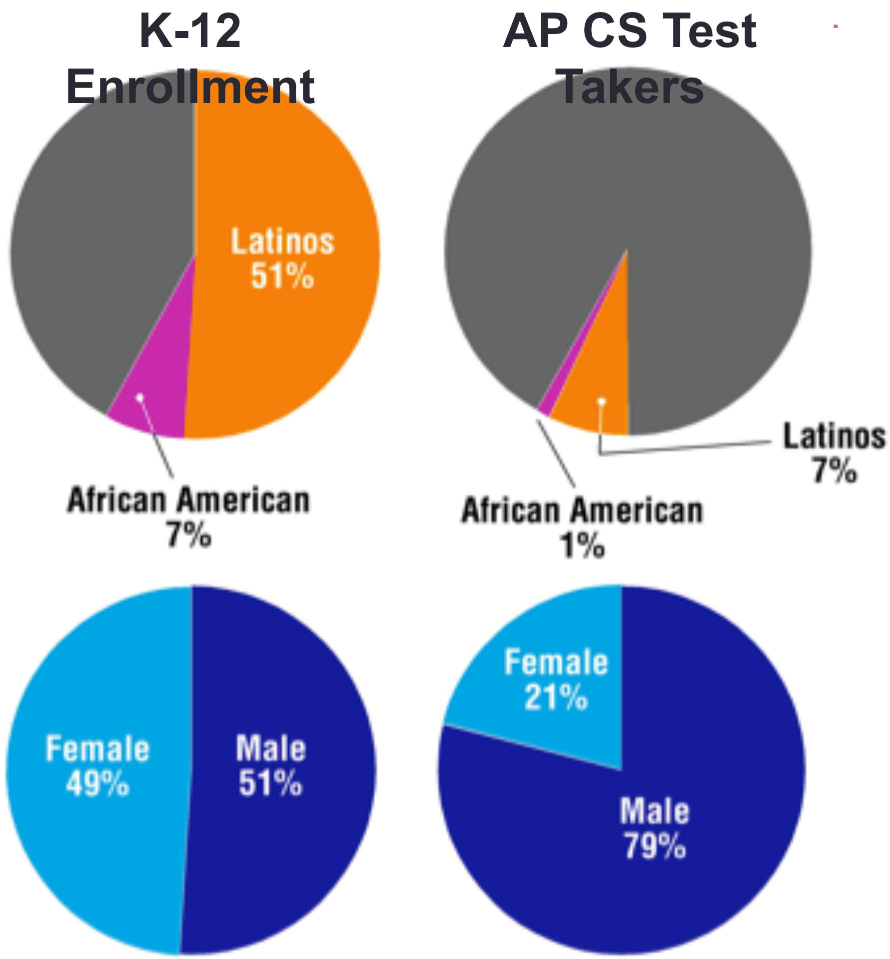


*California has one of the lowest CS AP participation rates among all U.S. high school students*

Source: College Board, AP Program Participation and Performance Data 2012

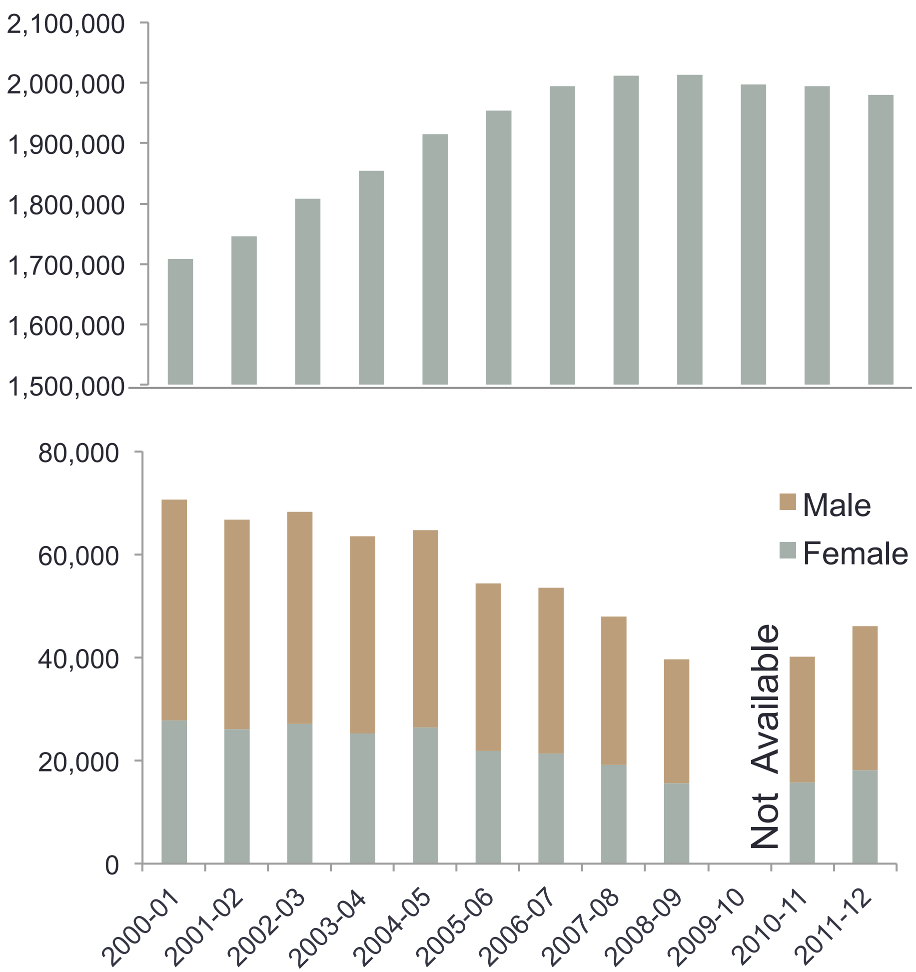
# California AP Demographics




*Computer Science has failed to attract representative numbers of Latino, African American and female students.*

*Source: www.exploringcs.org*

# California Enrollment




*From 2000 to 2012, total California high school enrollment increased 12% from 1.7 to 1.9 million, …*

*while enrollment in computer science and programming courses fell 35% from 71 to 46 thousand.*

*Source: Dataquest, California Department of Education*

# Computer Science Education


**Opportunities to learn computer science and engage in computational thinking practices, particularly in high school?**

# Key Curriculum Developments

## Exploring Computer Science (ECS)

An entry-level pre-AP introduction to the breadth of CS and on **computational thinking practices**. Prereq: Algebra I

http://www.exploringcs.org/

## Computer Science Principles (CSP)

A new AP course emphasizing **computational thinking practices** nested within seven big ideas in computing; fits between ECS and the existing CS AP course.

http://www.csprinciples.org/

# Computational Thinking

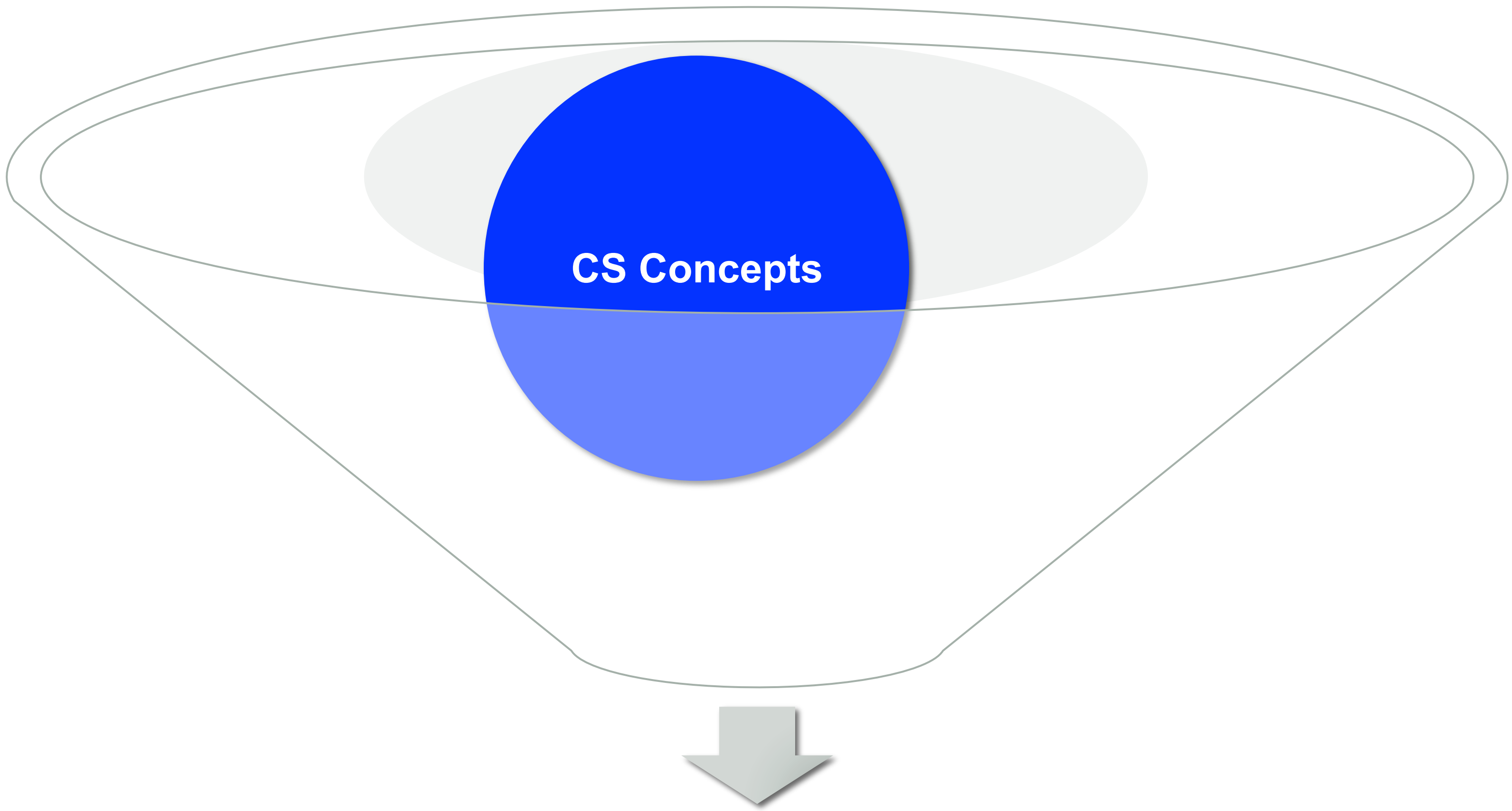
Popularized by Jeanette Wing in her 2006 Association for Computing Machinery, *Viewpoints* article:

> **“Universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.”**

Characteristics of computational thinking:

- Conceptualizing, not programming
- Ideas, not artifacts
- A way that humans, not computers, think
- Integrates computer science concepts, inquiry skills and, increasingly, communication and collaboration skills
- Complements and combines mathematical and engineering thinking

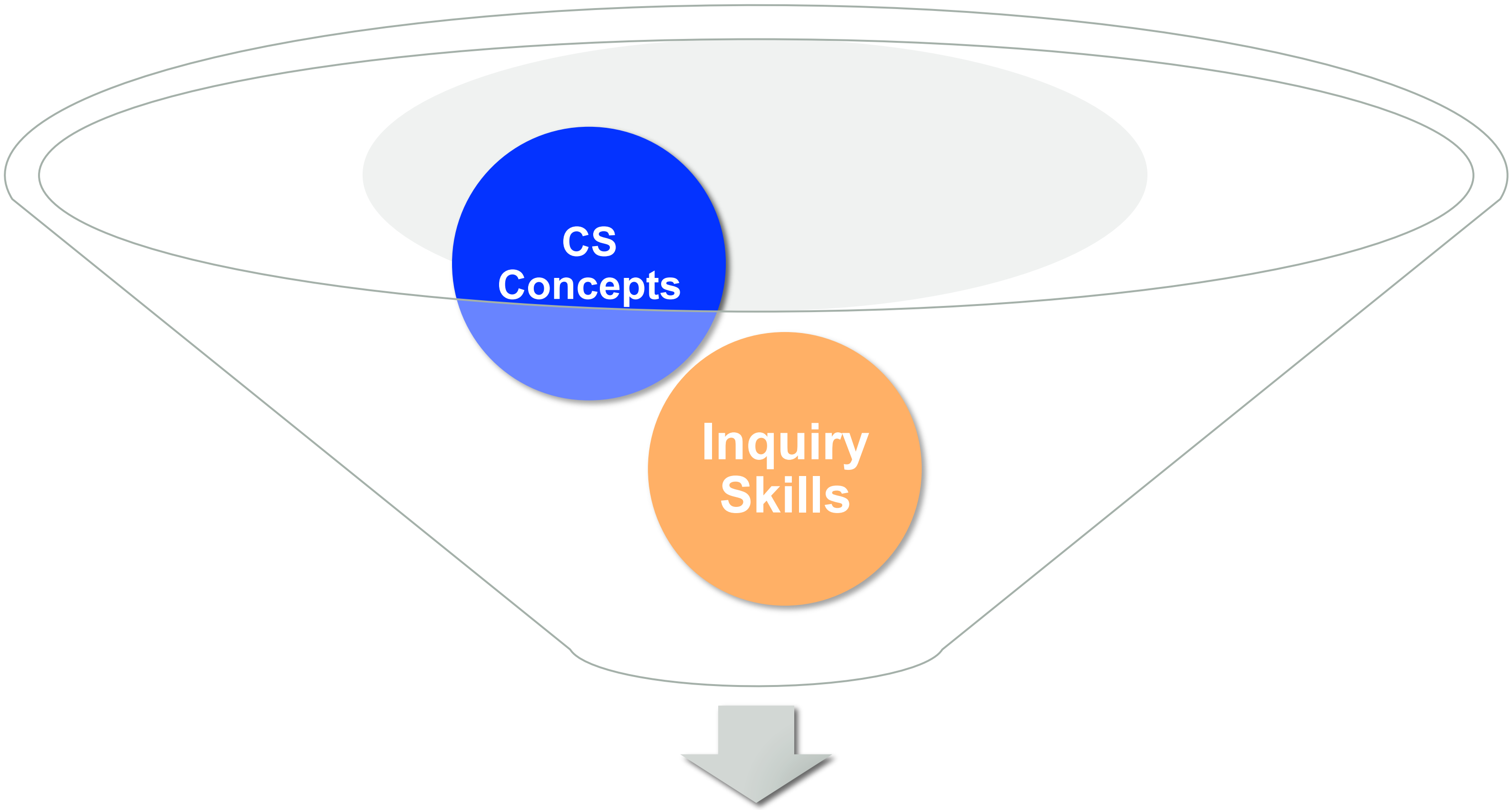
# Computational Thinking



# Computational Thinking

| CS Concepts | Inquiry Skills | Communication & Collaboration Skills |
|---|---|---|
| **Algorithms** | Evaluate | Publish |
| **Programming** | Explore | Present |
| **Recursion** | Investigate | Build Consensus |
| **Abstraction** | Explain | Discuss |
| **Debugging / Testing** | Elaborate | Distribute Work |
| **Variables** | Modeling | Lead/Manage Teams |

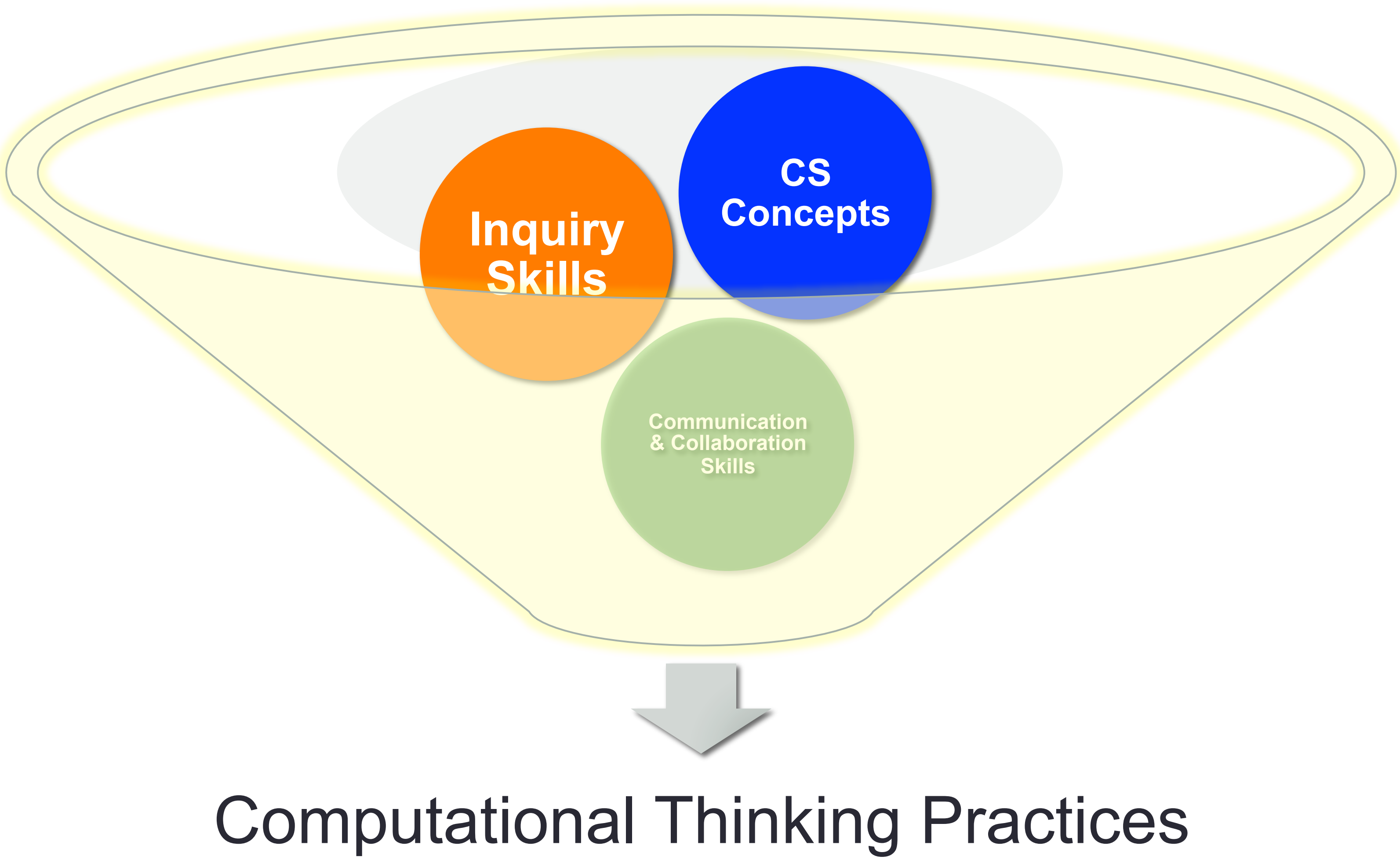
# Computational Thinking



# Computational Thinking

| CS Concepts | Inquiry Skills | Communication & Collaboration Skills |
|---|---|---|
| Algorithms | Evaluate | Publish |
| Programming | Explore | Present |
| Recursion | Analyze | Build Consensus |
| Abstraction | Explain | Discuss |
| Debugging / Testing | Elaborate | Distribute Work |
| Variables | Model | Lead/Manage Teams |

# Computational Thinking Practices



# Computational Thinking Practices

| CS Concepts | Inquiry Skills | Communication & Collaboration Skills |
|---|---|---|
| Algorithms | Evaluate | Publish |
| Programming | Explore | Present |
| Recursion | Analyze | Build Consensus |
| Abstraction | Explain | Discuss |
| Debugging / Testing | Elaborate | Distribute Work |
| Variables | Model | Lead/Manage Teams |

# Computational Thinking Practices

| Exploring Computer Science | Computer Science Principles |
| --- | --- |
| Analyze the effects of developments in computing | Analyzing Problems and Artifacts |
| Design and implement creative solutions and artifacts | Developing Computational Artifacts |
| Apply abstractions and models | Abstracting |
| Analyze their computational work and the work of others | Analyzing Problems and Artifacts |
| Connect computation with other disciplines | Connecting Computing |
| Communicate computational thought processes, procedures and results to others | Communicating |
| Collaborate with peers on computing activities | Collaborating |

# Computational Thinking Practices

| Exploring Computer Science | Computer Science Principles |
|---|---|
| Analyze the effects of developments in computing | Analyzing Problems and Artifacts |
| Design and implement creative solutions and artifacts | Developing Computational Artifacts |
| Apply abstractions and models | Abstracting |
| Analyze their computational work and the work of others | Analyzing Problems and Artifacts |
| Connect computation with other disciplines | Connecting Computing |
| Communicate computational thought processes, procedures and results to others | Communicating |
| Collaborate with peers on computing activities | Collaborating |

# Computational Thinking Practices

| Exploring Computer Science | Computer Science Principles |
|---|---|
| Analyze the effects of developments in computing | Analyzing Problems and Artifacts |
| Design and implement creative solutions and artifacts | Developing Computational Artifacts |
| Apply abstractions and models | Abstracting |
| Analyze their computational work and the work of others | Analyzing Problems and Artifacts |
| Connect computation with other disciplines | Connecting Computing |
| Communicate computational thought processes, procedures and results to others | Communicating |
| Collaborate with peers on computing activities | Collaborating |

# Computational Thinking Practices

| Exploring Computer Science | Computer Science Principles |
| --- | --- |
| Analyze the effects of developments in computing | Analyzing Problems and Artifacts |
| Design and implement creative solutions and artifacts | Developing Computational Artifacts |
| Apply abstractions and models | Abstracting |
| Analyze their computational work and the work of others | Analyzing Problems and Artifacts |
| Connect computation with other disciplines | Connecting Computing |
| Communicate computational thought processes, procedures and results to others | Communicating |
| Collaborate with peers on computing activities | Collaborating |

# Computational Thinking Practices

| Exploring Computer Science | Computer Science Principles |
|---|---|
| Analyze the effects of developments in computing | Analyzing Problems and Artifacts |
| Design and implement creative solutions and artifacts | Developing Computational Artifacts |
| Apply abstractions and models | Abstracting |
| Analyze their computational work and the work of others | Analyzing Problems and Artifacts |
| Connect computation with other disciplines | Connecting Computing |
| Communicate computational thought processes, procedures and results to others | Communicating |
| Collaborate with peers on computing activities | Collaborating |

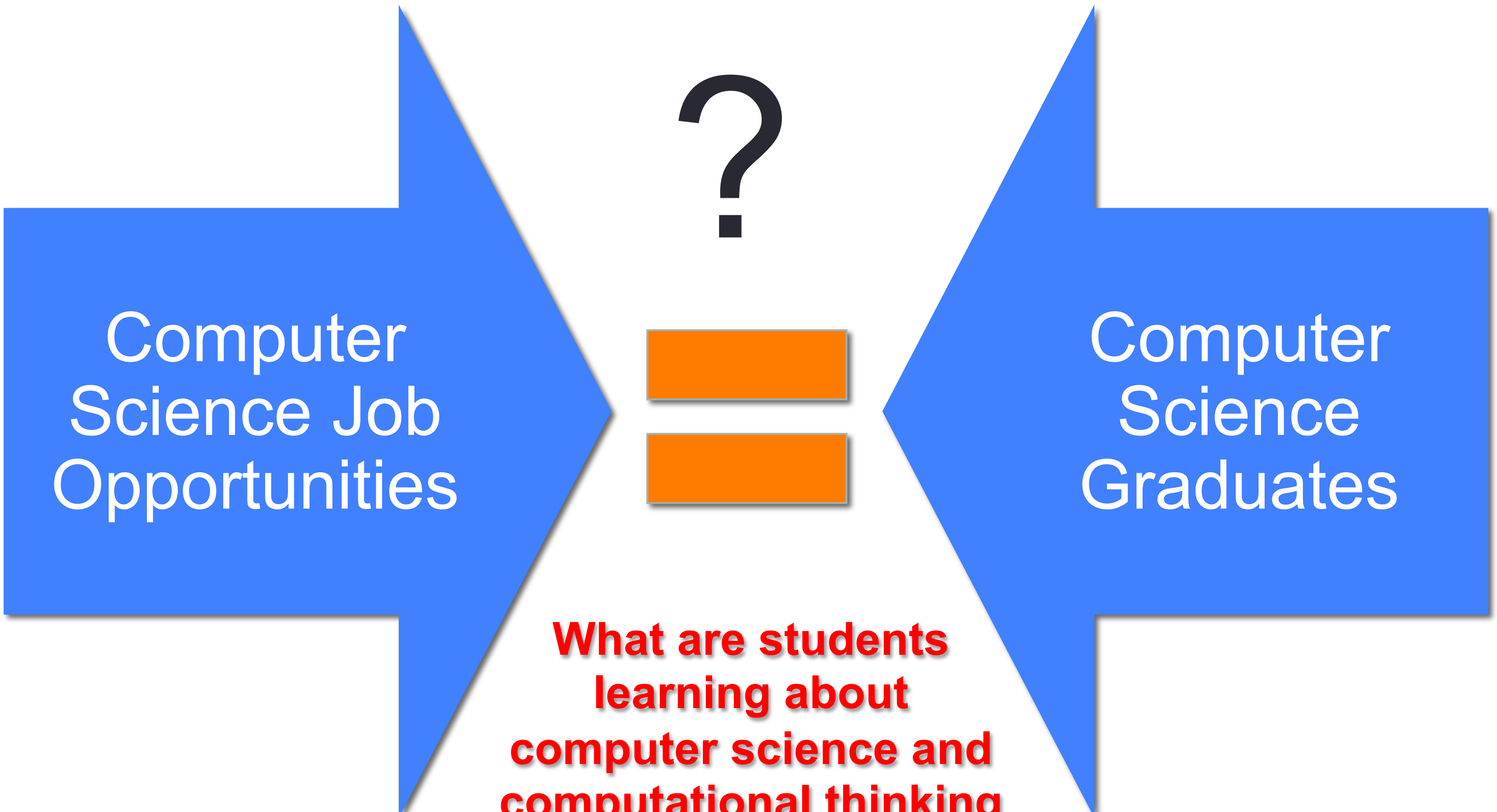
# Computational Thinking Practices

| Exploring Computer Science | Computer Science Principles |
|---|---|
| Analyze the effects of developments in computing | Analyzing Problems and Artifacts |
| Design and implement creative solutions and artifacts | Developing Computational Artifacts |
| Apply abstractions and models | Abstracting |
| Analyze their computational work and the work of others | Analyzing Problems and Artifacts |
| Connect computation with other disciplines | Connecting Computing |
| Communicate computational thought processes, procedures and results to others | Communicating |
| Collaborate with peers on computing activities | Collaborating |

# Computational Thinking Practices

| Exploring Computer Science | Computer Science Principles |
|---|---|
| Analyze the effects of developments in computing | Analyzing Problems and Artifacts |
| Design and implement creative solutions and artifacts | Developing Computational Artifacts |
| Apply abstractions and models | Abstracting |
| Analyze their computational work and the work of others | Analyzing Problems and Artifacts |
| Connect computation with other disciplines | Connecting Computing |
| Communicate computational thought processes, procedures and results to others | Communicating |
| Collaborate with peers on computing activities | Collaborating |

# Computer Science Education




**What are students learning about computer science and computational thinking practices, particularly in high school?**

# Measuring Computational Thinking Practices

- Computational thinking remains underspecified, particularly as a measurement domain.
- There are few clearly articulated and comprehensive domain models for guiding the development of assessment tasks for measuring computational thinking outcomes, particularly in high school computer science.
- Efforts are currently underway to develop assessments for the Exploring Computer Science and Computer Science Principles curricula.

# Principled Assessment of Computational Thinking

SRI International®

## How can we improve computer science teaching, learning, and adoption through evidence-centered assessment?

**NSF-CE21 Planning Grant (9/2011 – 3/2013)**

**NSF-CE21 Special Project (9/2012 – 8/2014)**

**http://pact.sri.com**

Principled Assessment of Computational Thinking

SRI International

## Long Term Goals

- High quality assessments of computational thinking
- Help with the adoption of high school computer science courses through assessments that stakeholders recognize as useful for making decisions about student learning

## Short Term Objectives

- Analyze the computational thinking domain, particularly how ECS CT practices and learning objectives are related to relevant national and state standards (e.g., CSTA, Common Core)
- Create **design patterns** for major computational thinking concepts that can be used to develop new assessments as curriculum evolves
- Create, for Exploring Computer Science, **4 unit assessments, a summative assessment, and field test** them by April 2014

Principled Assessment of Computational Thinking

SRI International

## Summer - Fall 2012

- Analyze computational thinking domain, particularly as it is represented in *Exploring Computer Science*

## Fall 2012 – Summer 2013

- Use Evidence-Centered Design to model the computational thinking domain in a set of assessment design templates for *Exploring Computer Science*

## Summer – Fall 2013

- Use design templates to guide the development of unit and summative assessments for *Exploring Computer Science*

## Fall 2013 – Spring 2014

- Pilot *Exploring Computer Science* assessments

# Evidence-Centered Assessment Design

Framework organized around three guiding questions:

1. What knowledge, skills, or other attributes (constructs) should be assessed?
2. What behaviors or performances should reveal those constructs?
3. What tasks or situations should elicit those behaviors?

ECD views **assessments as providing evidence for supporting arguments** about what a student knows and can do.

ECD is a particularly **useful framework for dealing with novel and/or hard-to-assess constructs**, like computational thinking practices.

# ECD Design Patterns

**Design Patterns** are tools for organizing assessment arguments at a narrative level.

The often serve as a type of **advanced organizer** for multi-disciplinary teams prior to working on the details of specific assessment tasks.

**Focal KSAs** are the core of design patterns.

- Specify the **K**nowledge, **S**kills and **A**ttributes one wants to measure.
- Often represent a blend of relevant standards and curriculum objectives.

**Potential Observations** specify the behaviors that count as evidence of the Focal KSAs.

**Potential Work Products** specify the types of tasks that can elicit the desired behaviors.

# ECS Unit 1: Human-Computer Interaction

## Topics

- Computers and the Internet
- Models of Intelligent Behavior
- Societal Impacts of Computing

## Example Objectives

- Analyze the characteristics of hardware components to determine the applications for which they can be used
- Evaluate the results of web searches and the reliability of information found on the Internet
- Communicate legal and ethical concerns raised by computing innovation
- Explain the implications of communication as data exchange

# Design Patterns

## ECS Unit 1: Human-Computer Interaction

| Focal KSA | Potential Observations | Potential Work Products |
|---|---|---|
| Ability to compare the components of two or more computers for the suitability to a particular application. | Appropriateness of the comparison of the components of two or more computers for a particular application.<br><br>*Did the student correctly identify differences in the components of the computers that relate to the application?*<br><br>*Did the student correctly explain how the differences in the components affect the suitability for the application?* | Comparison of computers for the suitability to a particular application. |

# Design Patterns
## ECS Unit 1: Human-Computer Interaction

### Additional Features

Characteristic:
- information about more than one computer
- information about the application

Variable:
- type of application
- level of details specified about the application and the computers
- knowledge of particular computer components required

# ECS Unit 2: Problem Solving

## Topics

- Algorithms and abstraction
- Connections between Mathematics and Computer Science
- Societal impacts of computing

## Example Objectives

- Determine if a given algorithm successfully solves a stated problem
- Create algorithms that meet specified objectives
- Compare the tradeoffs between different algorithms for solving the same problem

# Design Patterns

## ECS Unit 2: Problem Solving

| Focal KSA | Potential Observations | Potential Work Products |
| --- | --- | --- |
| Ability to explain the inputs of an algorithm, how it operates on that input, and what the outputs are. | Accuracy of the explanation of the parts of an algorithm.<br><br>*Did the student correctly identify the inputs of an algorithm?*<br><br>*Did the student correctly and completely describe the operations performed on those inputs?*<br><br>*Did the student correctly describe the outputs?* | The explanation of the algorithm. |

# Design Patterns
## ECS Unit 2: Problem Solving

### Additional Features

Characteristic:

- an algorithm must be provided

Variable:

- algorithm provided
- inputs to the algorithm
- outputs of the algorithm
- complexity of the algorithm

# Design Patterns
## ECS Unit 3: Web Design

### Topics

- Web page design and development
- Computers and the Internet
- Algorithms and abstraction
- Societal impacts of computing

### Example Objectives

- Create web pages with a practical, personal, and/or societal purpose
- Select appropriate techniques when creating web pages
- Use abstraction to separate style from content in web page design and development

# Design Patterns

## ECS Unit 4: Introduction to Programming

### Topics

- Programming algorithms and abstractions
- Connections between mathematics and computer science
- Societal impacts of computing

### Example Objectives

- Select appropriate programming structures
- Locate and correct errors in a program
- Explain how a particular program functions
- Justify the correctness of a program
- Create programs with practical, personal, and/or societal intent

# Closing Comments

- There are not enough skilled graduates to fill the current and projected future demands for computer science jobs
- There is a need for computer science coursework with robust assessment programs earlier in students' high school education experiences
- **Exploring Computer Science** and **Computer Science Principles** are addressing this need

# Closing Comments

- ECS and CSP target **computational thinking practices**, which include knowledge of CS concepts and inquiry, collaboration and communication skills
- **Evidence-Centered Design (ECD)** provides a **robust framework** for supporting the creation of assessments for this hard-to-measure domain
- ECD **Design Patterns** can serve as a type of **advanced organizer** for multi-disciplinary teams prior to working on the details of specific assessment tasks

# Thank You!

Eric Snow
Center for Technology in Learning
SRI International
eric.snow@sri.com