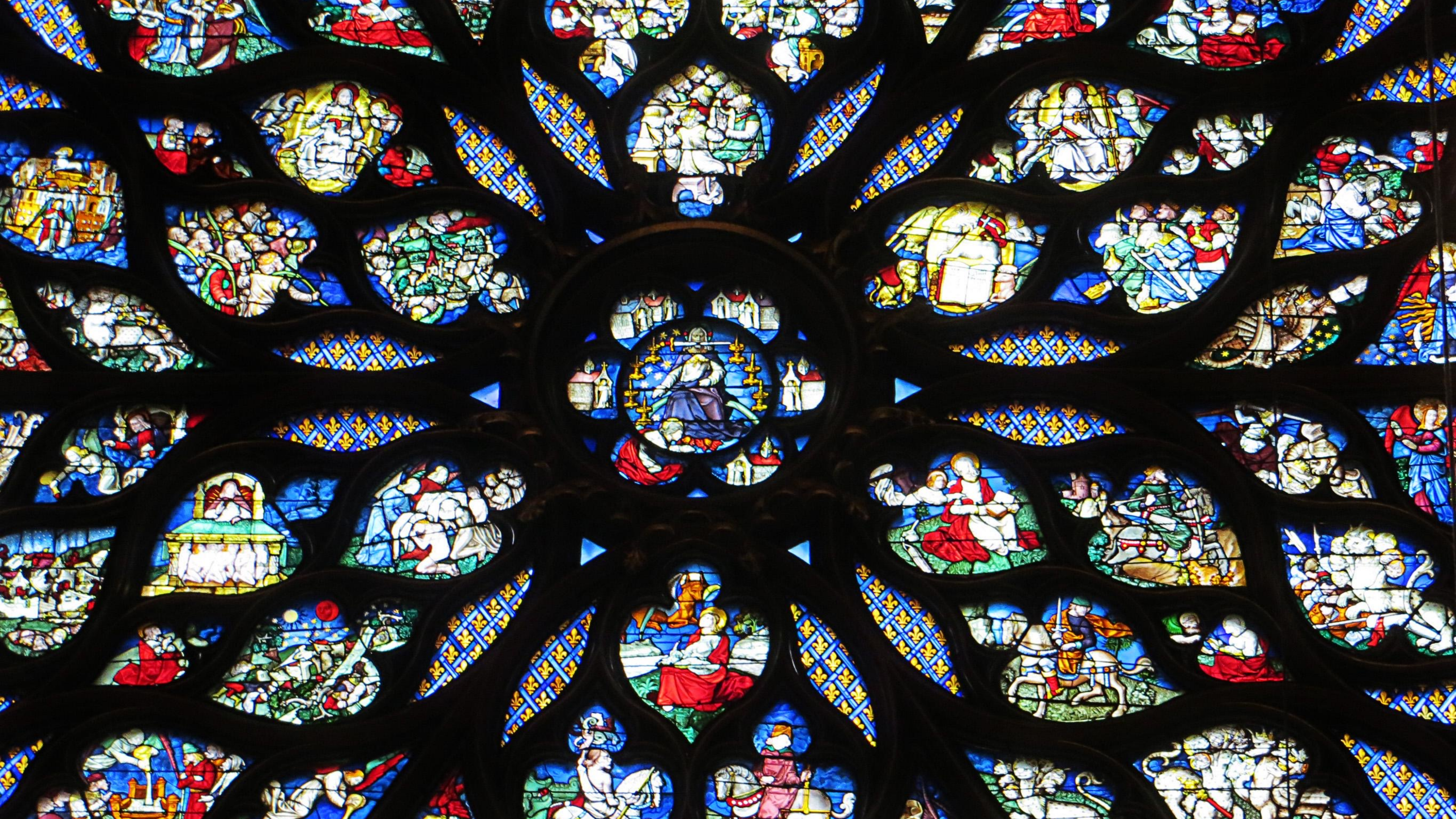
Software Design

Jeffrey Carver
University of Alabama
carver@cs.ua.edu



Design Principles

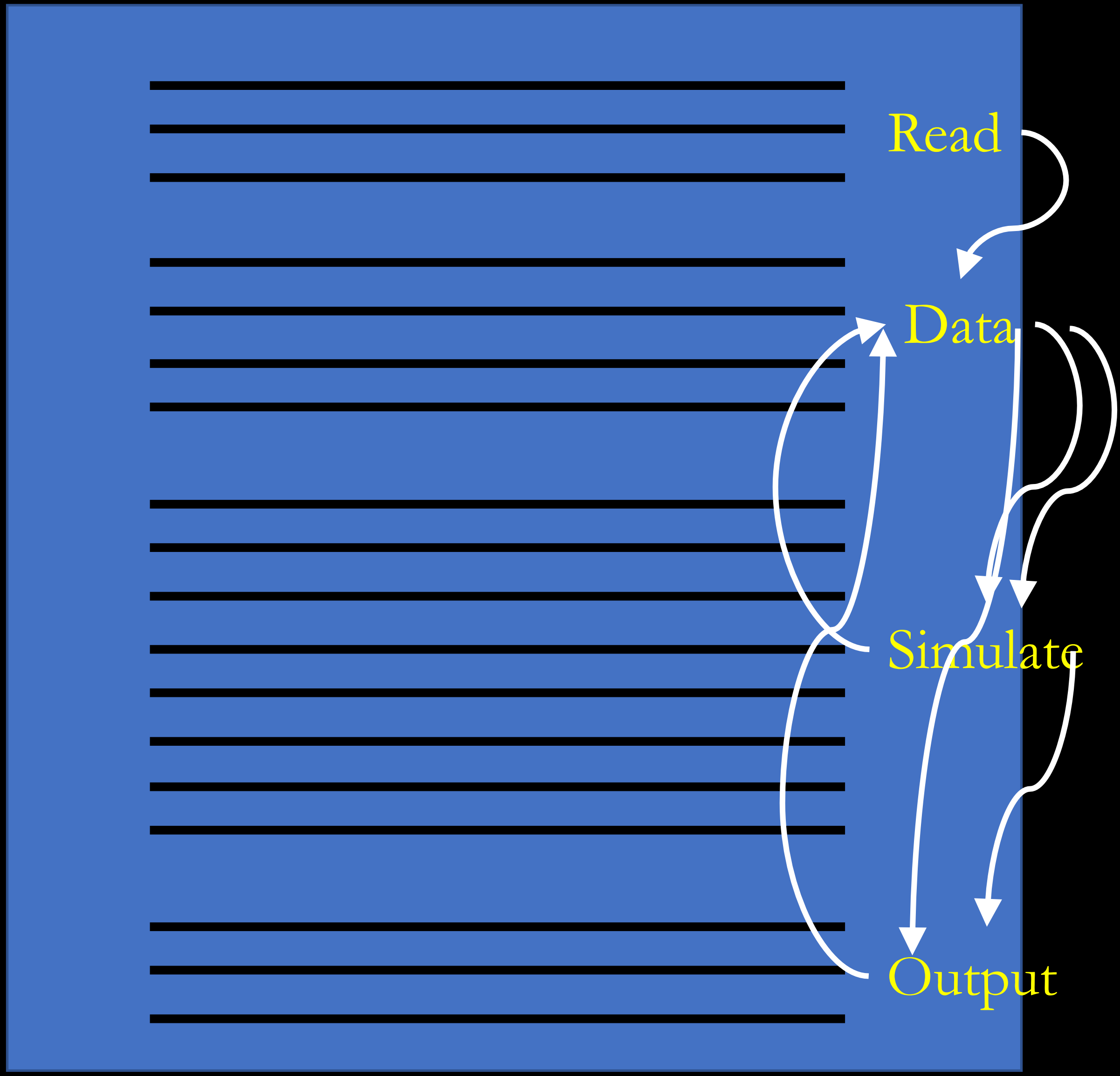
- Traceability
- Minimize intellectual distance
- Don’t reinvent the wheel
- Accommodate change
- Degrade gracefully

Fundamental Concepts

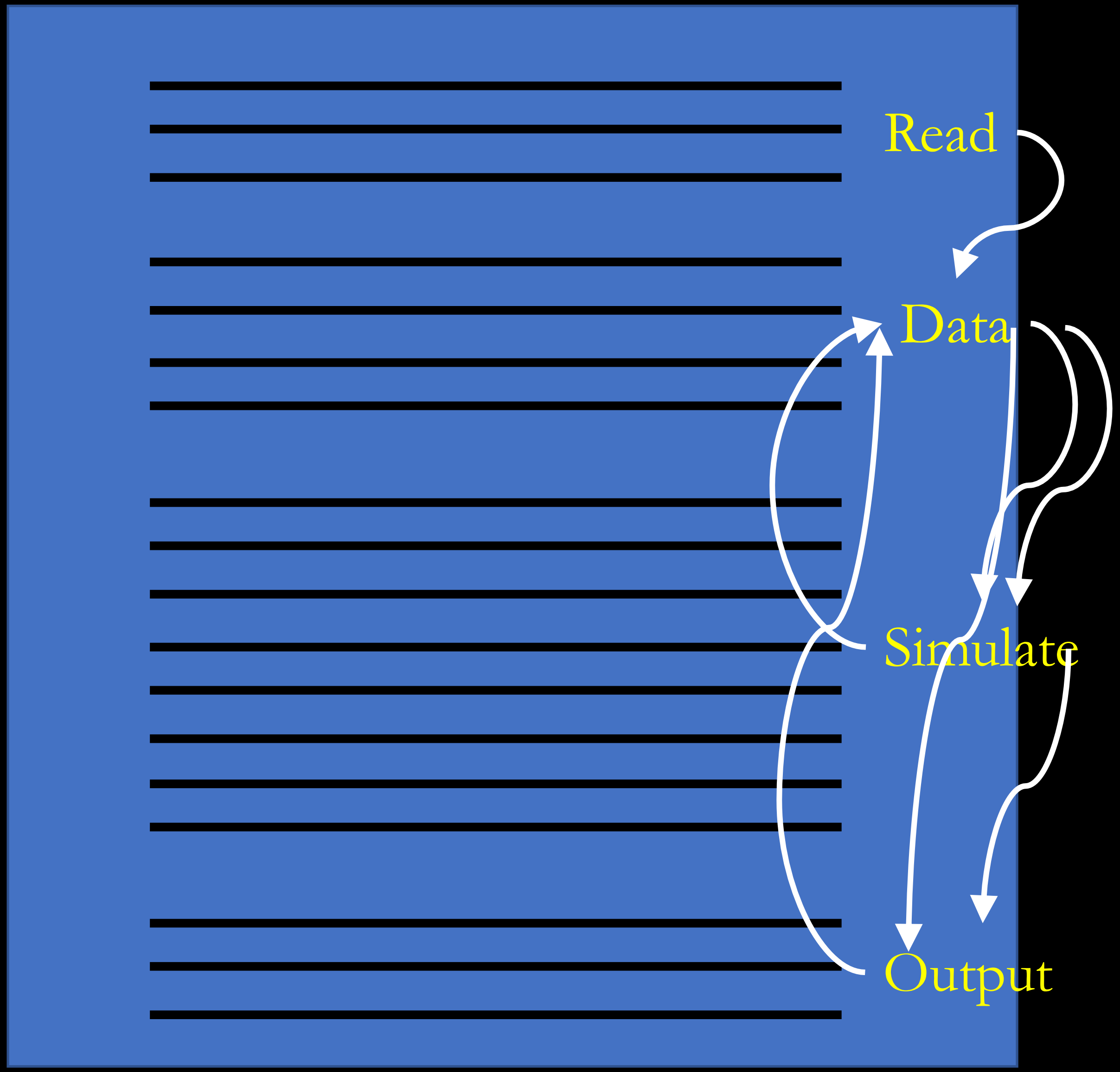
- Abstraction
- Patterns
- Modularity
- Hiding
- Functional independence

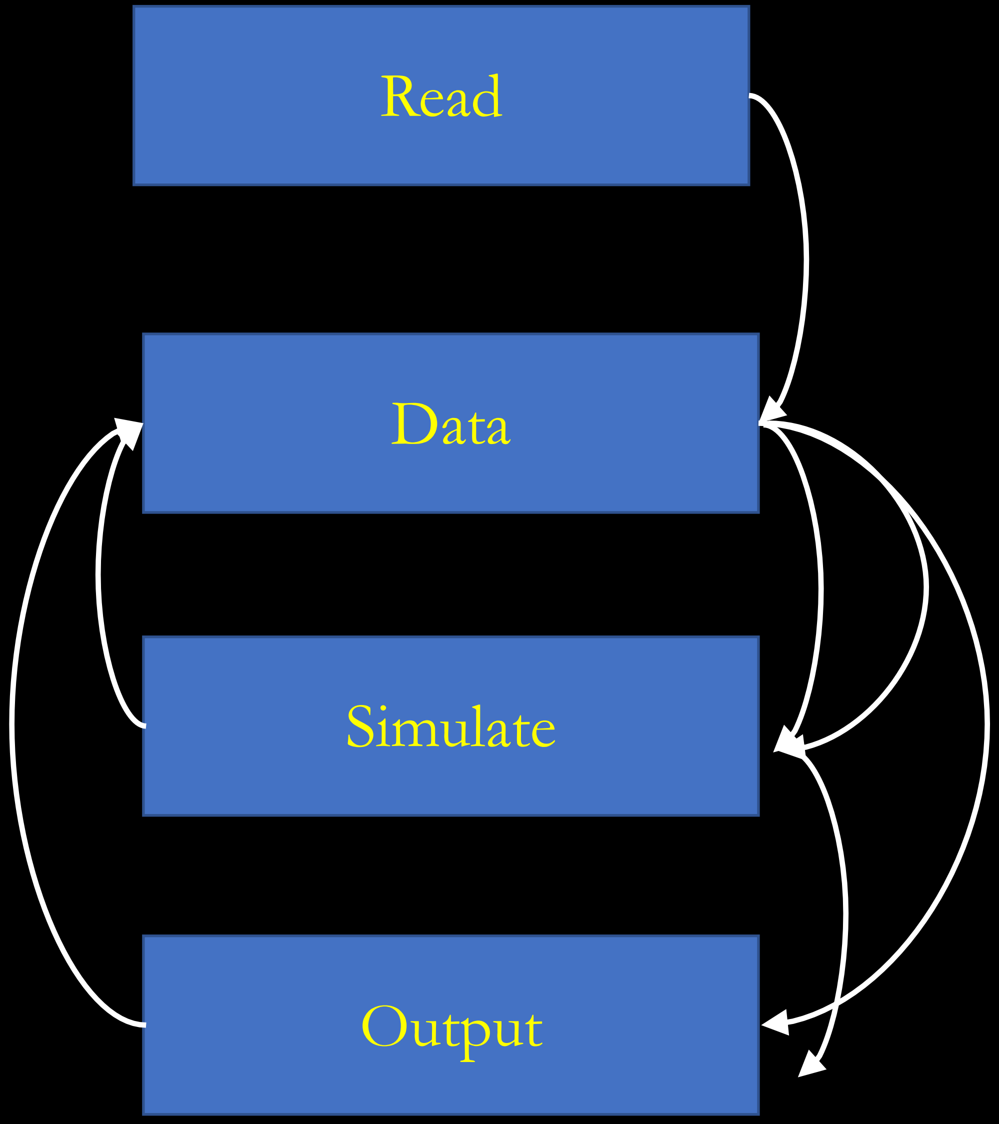
Example

- Simulate a simple water flow
  - Read in data
  - Data
    - Prepare
    - Manipulate Data
    - Convert Units
  - Perform simulation
    - Simulation specific calculations
    - Generic solvers
  - Output the results

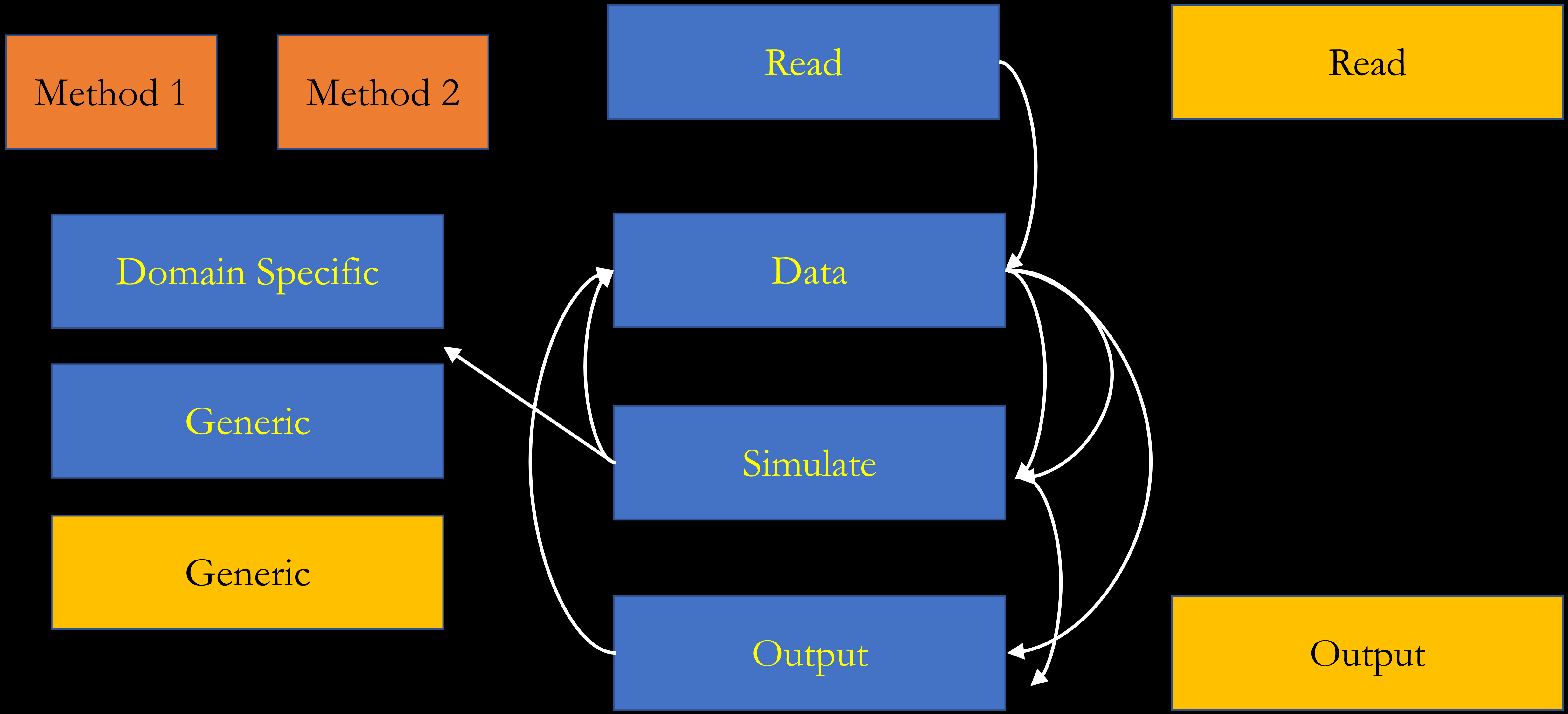


Example





Example



Method 1

Method 2

Read

Read

Domain Specific

Data

Generic

Simulate

Generic

Output

Output

CIROH
Cooperative Institute for
Research to Operations
in Hydrology

Software Design

Jeffrey Carver
University of Alabama
carver@cs.ua.edu

CIROH
Cooperative Institute for
Research to Operations
in Hydrology

Software Testing

Jeffrey Carver
University of Alabama
carver@cs.ua.edu

Testing Principles

- Identify Goals
- Define Levels
- Validation vs. Verification
- Planning
- QA Activities
- Test-Driven Development (TDD)

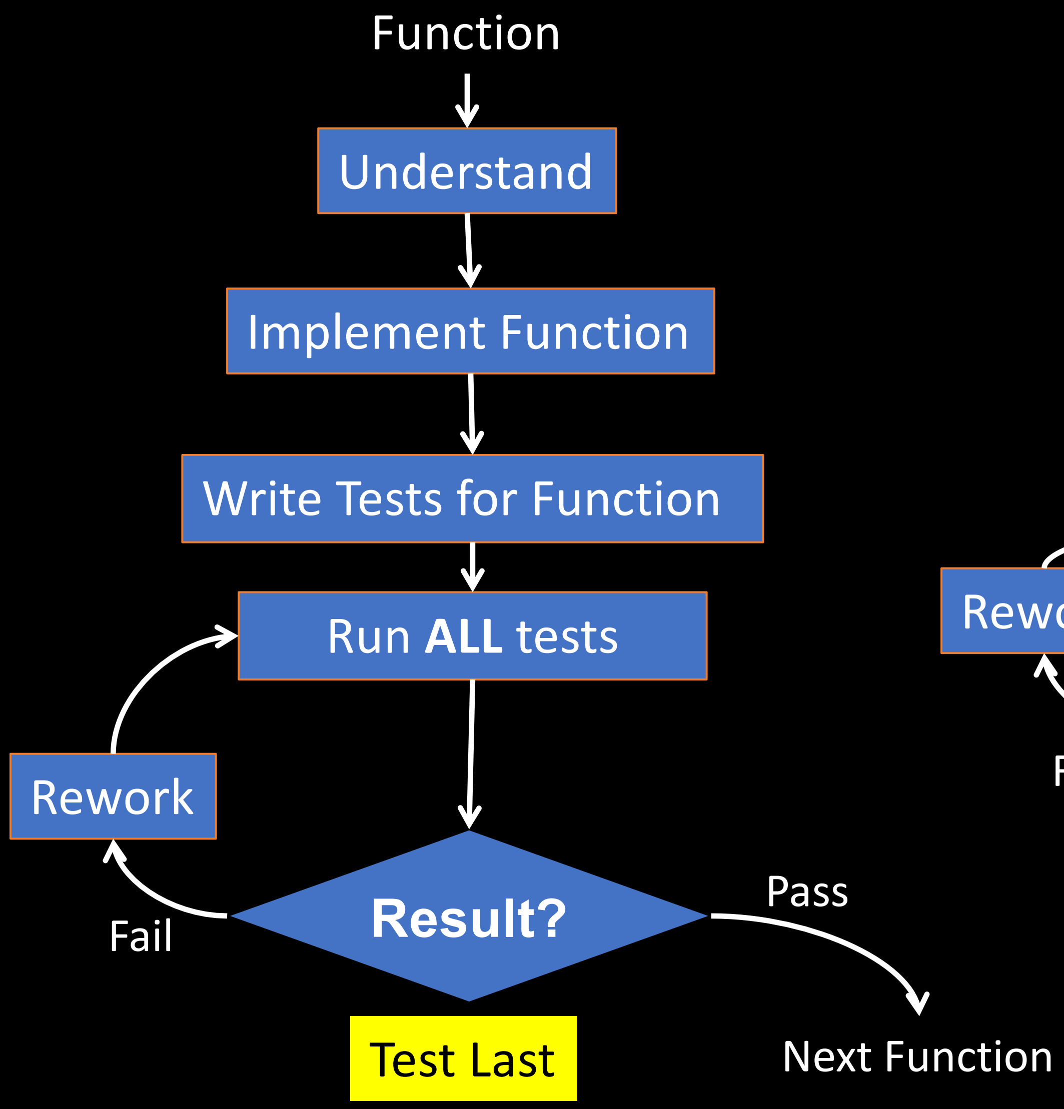
Test-Driven Development (TDD)

- Basic Idea
  - Write automated tests
  - Prior to developing functional code
  - Rapid iterations

- Focus on Unit Tests
  - Traditionally written after code is complete
  - In TDD tests are written before code

- Leads to analysis, design, and programming decisions

Test-Driven Development (TDD)



Function

Understand

Implement Function

Write Tests for Function

Run **ALL** tests

Rework

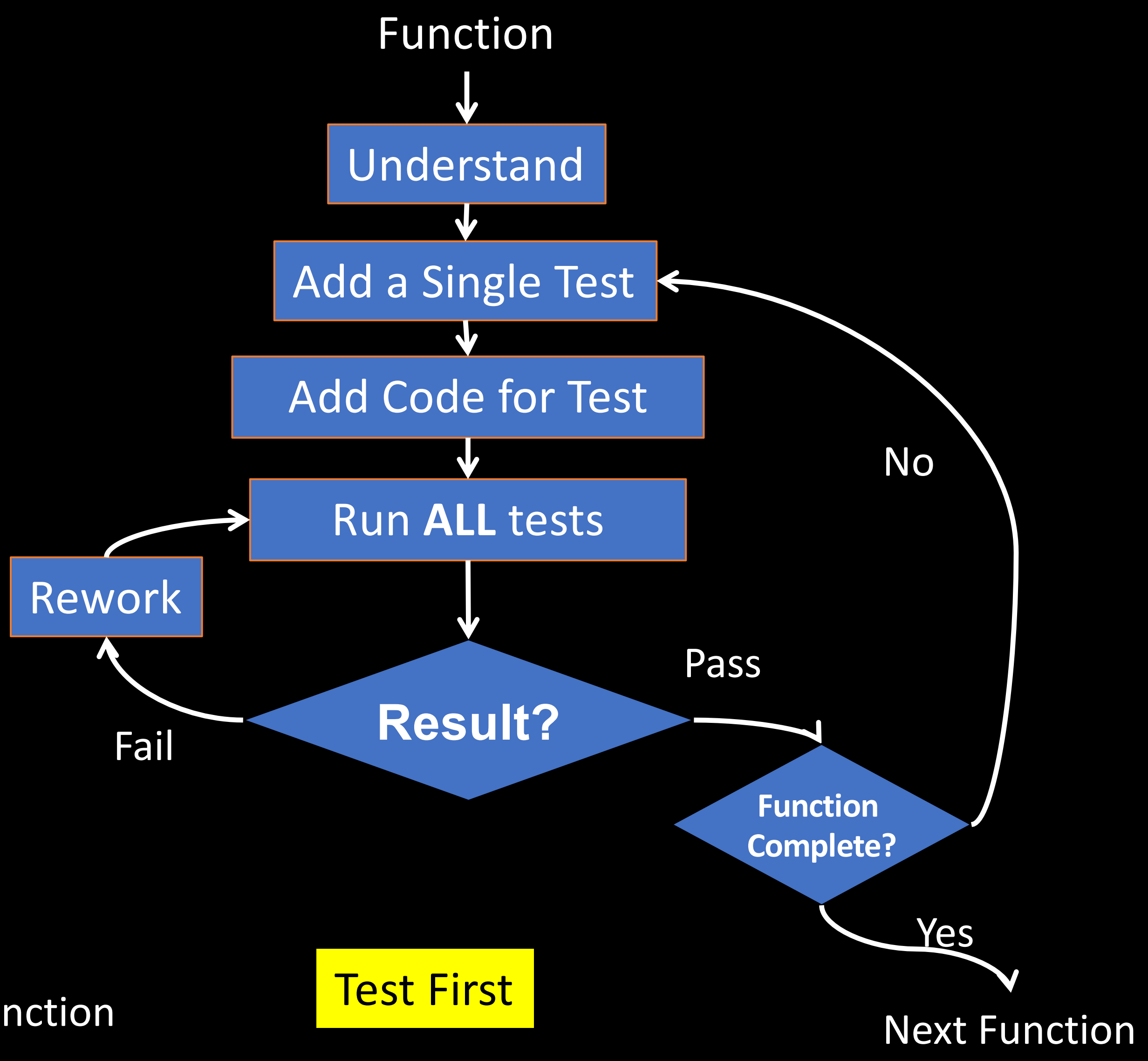
Result?

Fail

Pass

Next Function

Test Last



Function

Understand

Add a Single Test

Add Code for Test

Run **ALL** tests

Rework

Result?

Fail

Pass

Function Complete?

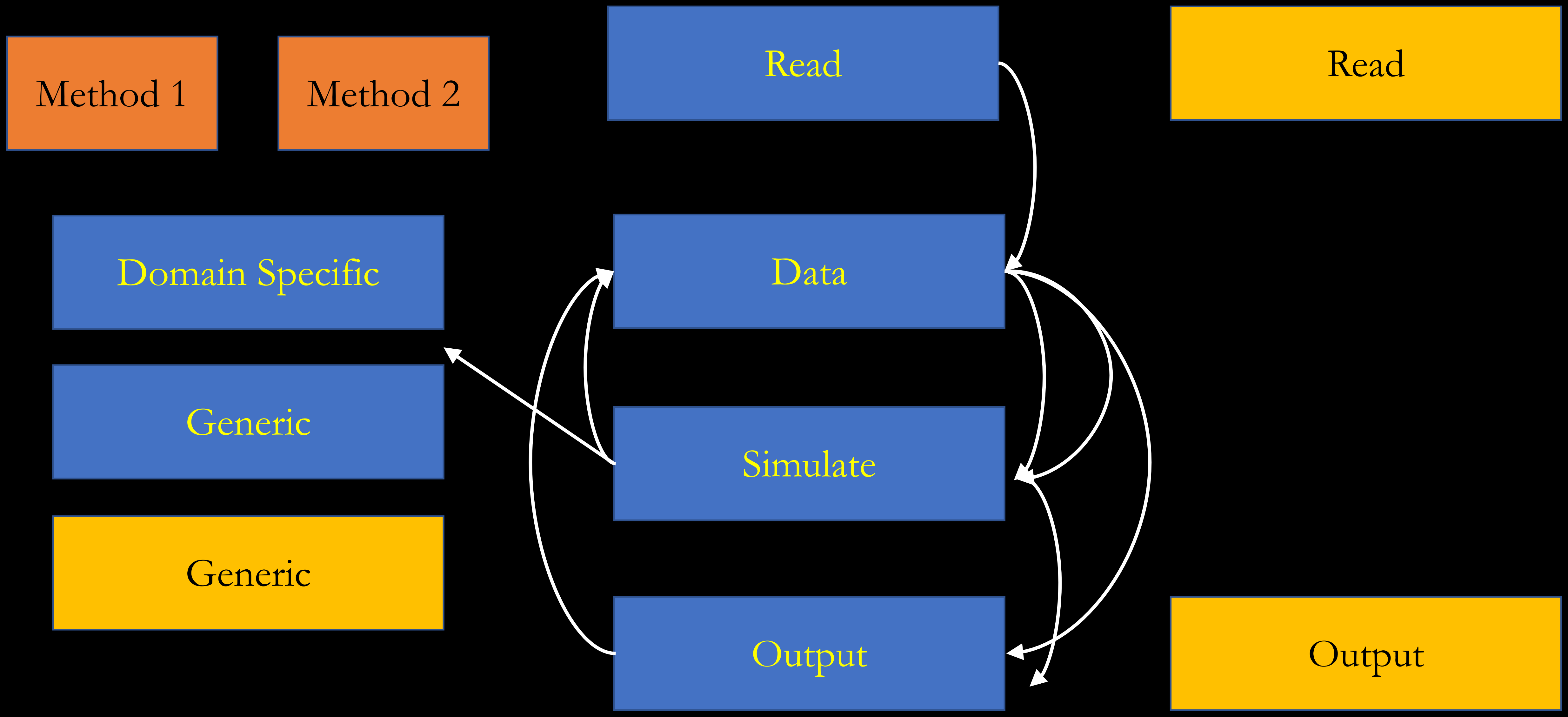
No

Yes

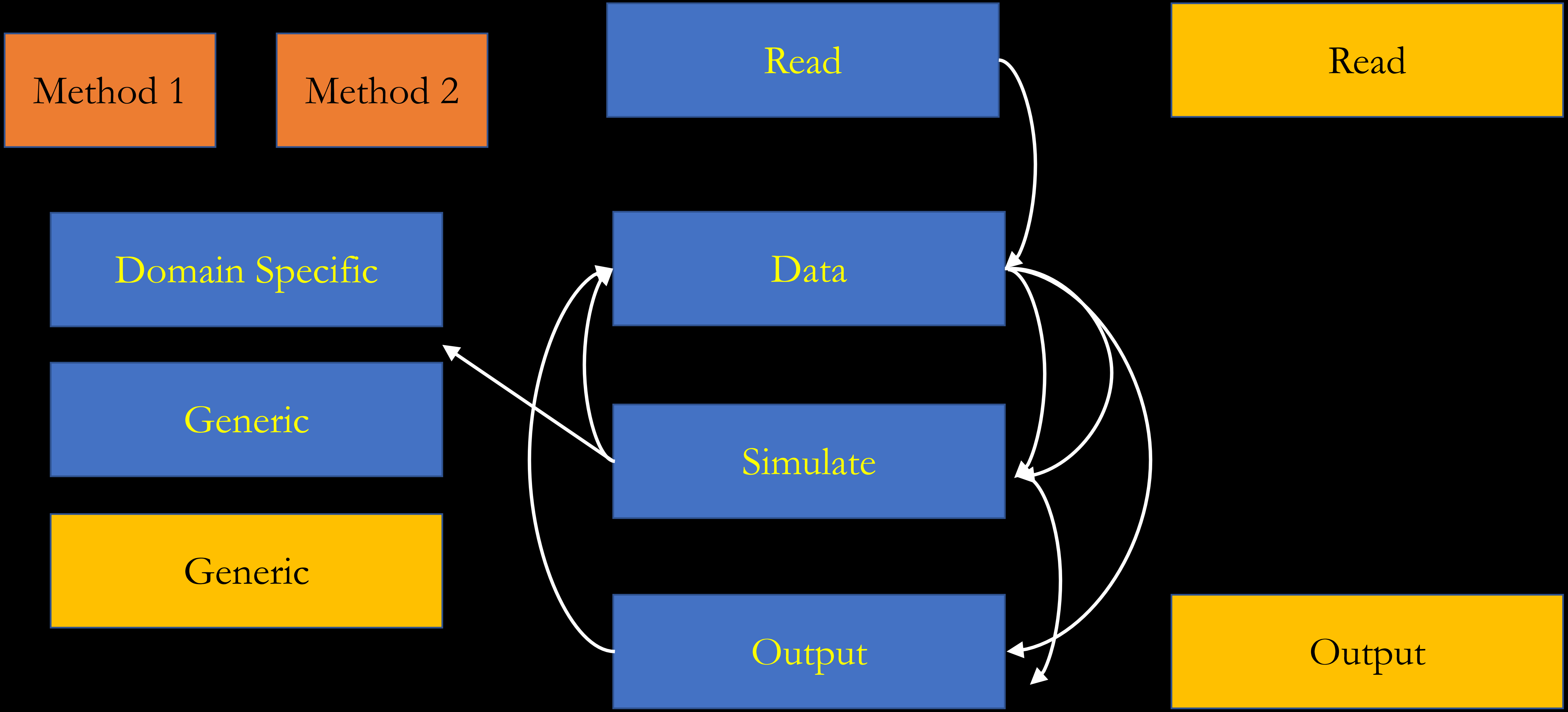
Next Function

Test First

Example


Method 1
Method 2
Read
Read
Domain Specific
Data
Generic
Simulate
Generic
Output
Output

Example



Method 1

Method 2

Read

Read

Domain Specific

Data

Generic

Simulate

Generic

Output

Output

CIROH
Cooperative Institute for
Research to Operations
in Hydrology

Software Testing

Jeffrey Carver
University of Alabama
carver@cs.ua.edu

CIROH
Cooperative Institute for
Research to Operations
in Hydrology

Software Documentation

Jeffrey Carver
University of Alabama
carver@cs.ua.edu

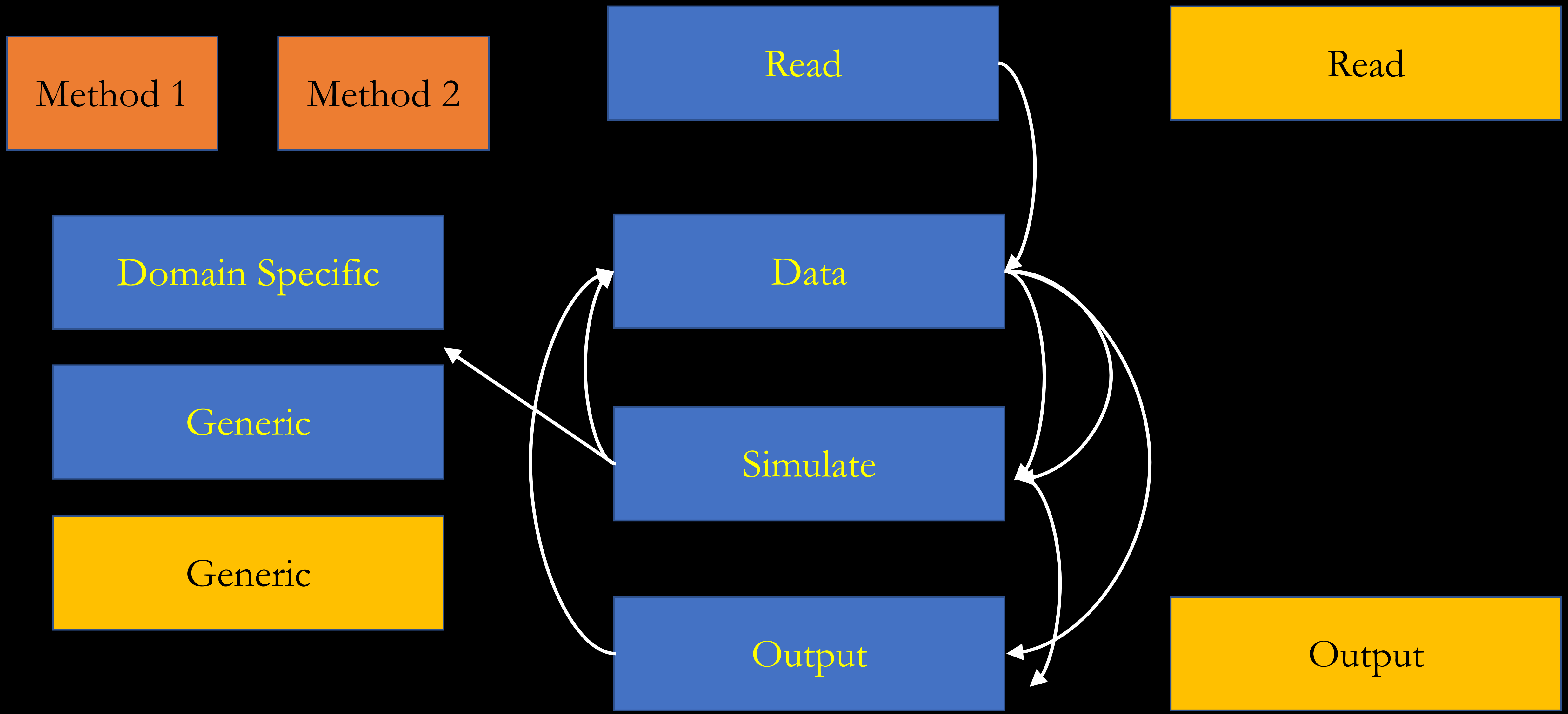
Types of Documentation

- Publications
- README files
- Method/Function/Class Headers
- Comments within methods
- Variable names
- Tests

README Files

- Name of software package
- Short description of software package
- High-level description of the features
- Prerequisites (e.g. other packages required)
- Installation/deployment instructions
- Basic Usage
- Credits/License

Example



Method 1

Method 2

Read

Read

Domain Specific

Data

Generic

Simulate

Generic

Output

Output

Reminders

- Think about documentation before you start
- Think about your motivation for documentation
- “What are best practices for research software documentation” blog post has a nice flowchart to help with the decision process

Credits

- “Preparing Software for Reuse and Release”- https://carpentries-incubator.github.io/python-intermediate-development/42-software-reuse/index.html

- “What are best practices for research software documentation?” - https://www.software.ac.uk/blog/2019-06-21-what-are-best-practices-research-software-documentation

CIROH
Cooperative Institute for
Research to Operations
in Hydrology

Software Documentation

Jeffrey Carver
University of Alabama
carver@cs.ua.edu