SAND2023-01369PE

In Pursuit of Reproducibility: Declarative and Automated Post-Processing

Chad Hovey

2023-03-08

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Outline:

Reproducibility: Motivate, define, explore.

Declarative: Code style, link to reproducibility and automation.

Case Study: One attempt at a solution.

Q&A: What are your pain points and bottlenecks?

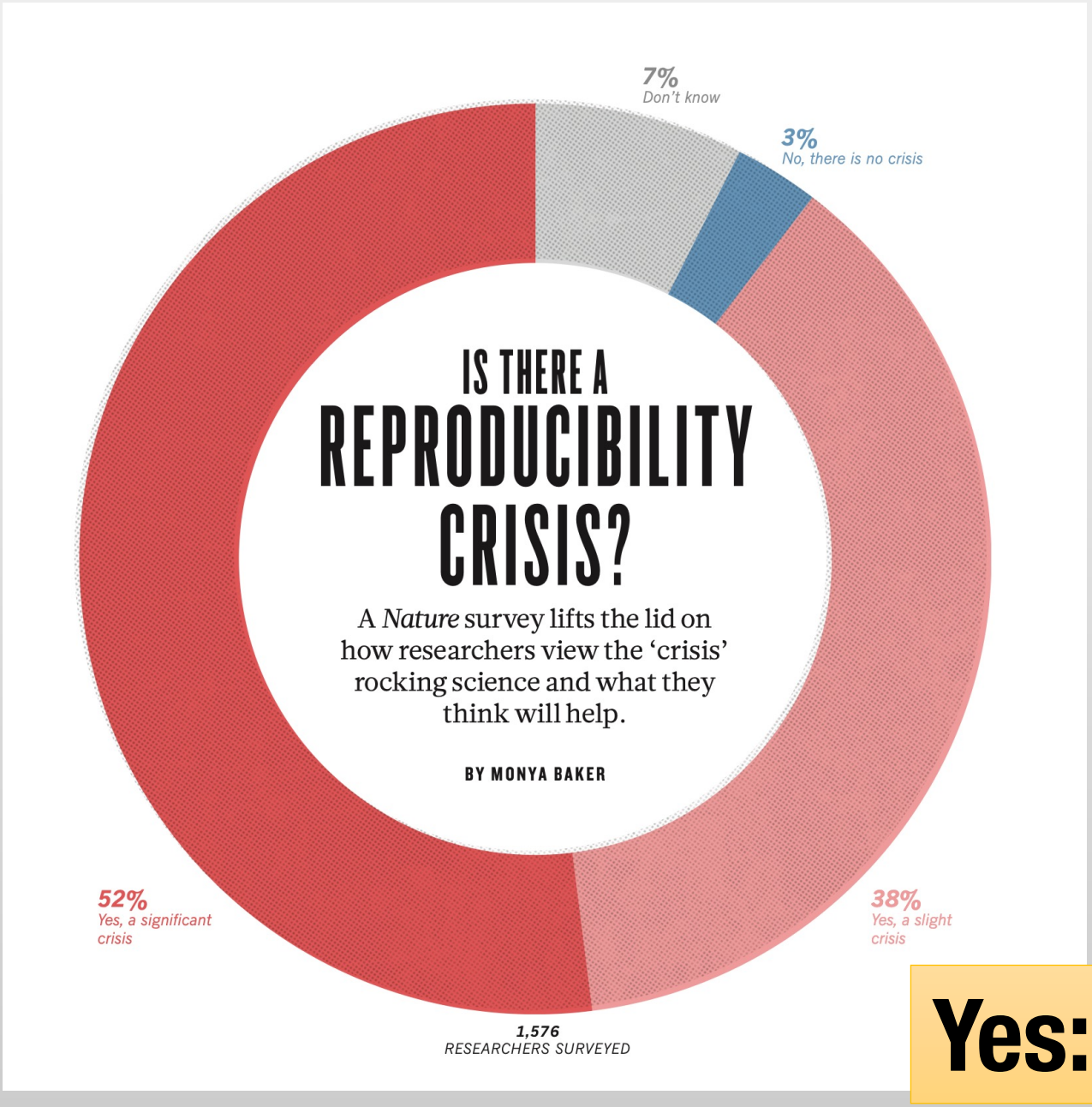
Outline:

Reproducibility: Motivate, define, explore.

Declarative: Code style, link to reproducibility and automation.

Case Study: One attempt at a solution.

Q&A: What are your pain points and bottlenecks?

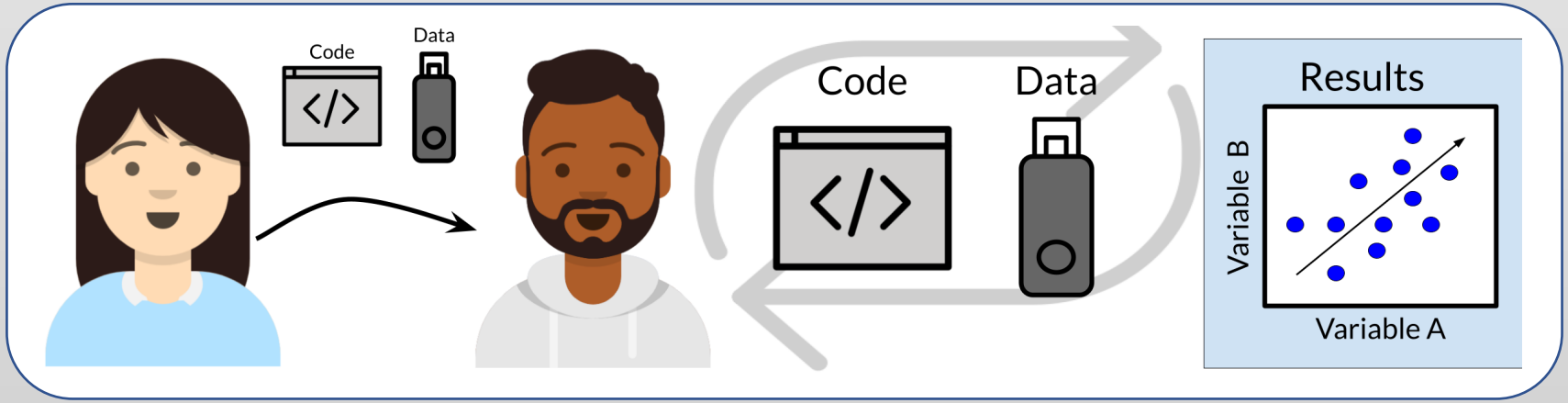


Yes: 90%

Reference: Baker M. 1,500 scientists lift the lid on reproducibility. *Nature*. 2016;533(7604). https://www.nature.com/articles/533452a and ~/sibl-dev/refs/Baker_2016_reproducibility_crisis.pdf

What?

Reproducibility: a different analyst re-performs the analysis with the same code and the same data and obtains the same results.



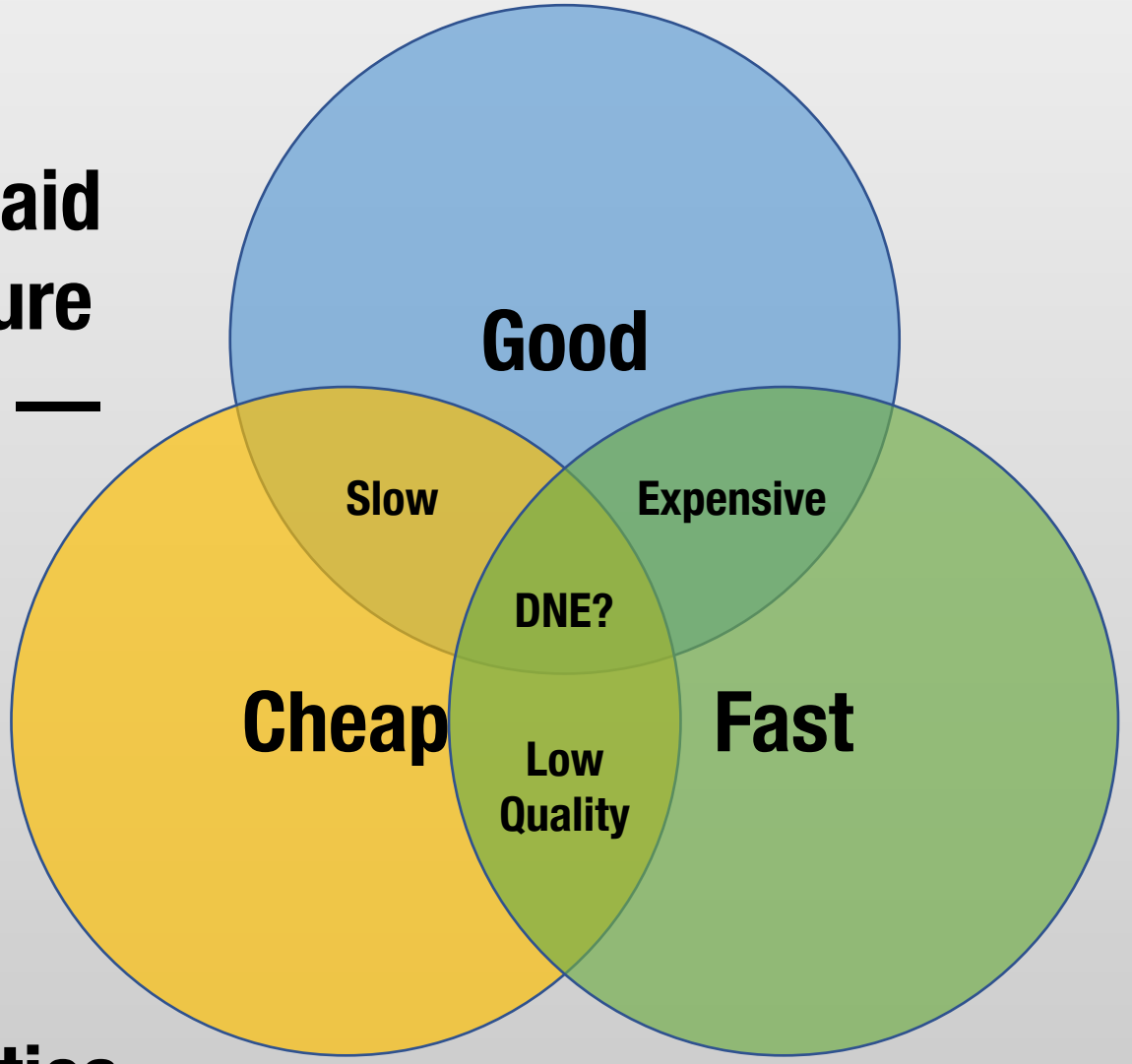
Repeatability: is reproducibility but with same analyst.

Reference: Patil P, Peng RD, Leek JT. A statistical definition for reproducibility and replicability. BioRxiv. 2016 Jul 29:066803. https://www.biorxiv.org/content/10.1101/066803v1 and https://jhudatascience.org/Reproducibility_in_Cancer_Informatics/defining-reproducibility.html .

Why?

"More than 60% of respondents said that each of two factors — pressure to publish and selective reporting — always or often contributed."

Competing priorities of good, cheap, and fast (pick any two).



Leaders and performers set priorities.

Reference: Baker M. *Op. cit.*

Outline:

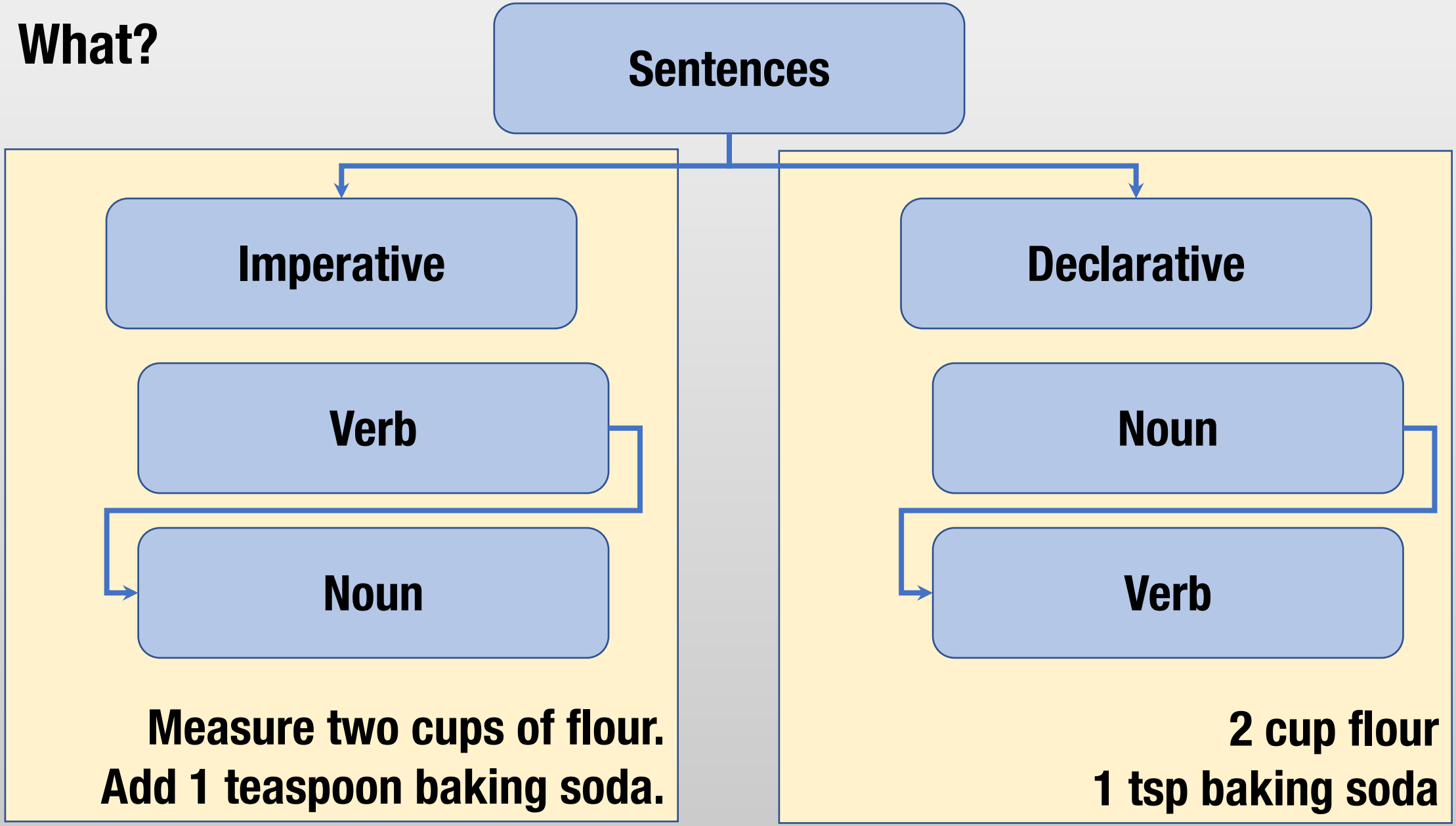
Reproducibility: Motivate, define, explore.

Declarative: Code style, link to reproducibility and automation.

Case Study: One attempt at a solution.

Q&A: What are your pain points and bottlenecks?

What?


Sentences
Imperative
Verb
Noun
Declarative
Noun
Verb
Measure two cups of flour.
Add 1 teaspoon baking soda.
2 cup flour
1 tsp baking soda

Noun → **Verb**

INGREDIENTS

Yield: 4 to 6 servings

- 3 tablespoons vegetable oil
- 1 medium yellow onion, diced
- 3 cloves garlic, finely chopped
- 3 tablespoons grated ginger
- 1 tablespoon garam masala
- 1 6-ounce can tomato paste
- ¾ teaspoon kosher salt
- 3 pounds boneless, skinless chicken thighs or breasts, cut into 2-inch pieces

PREPARATION

Step 1

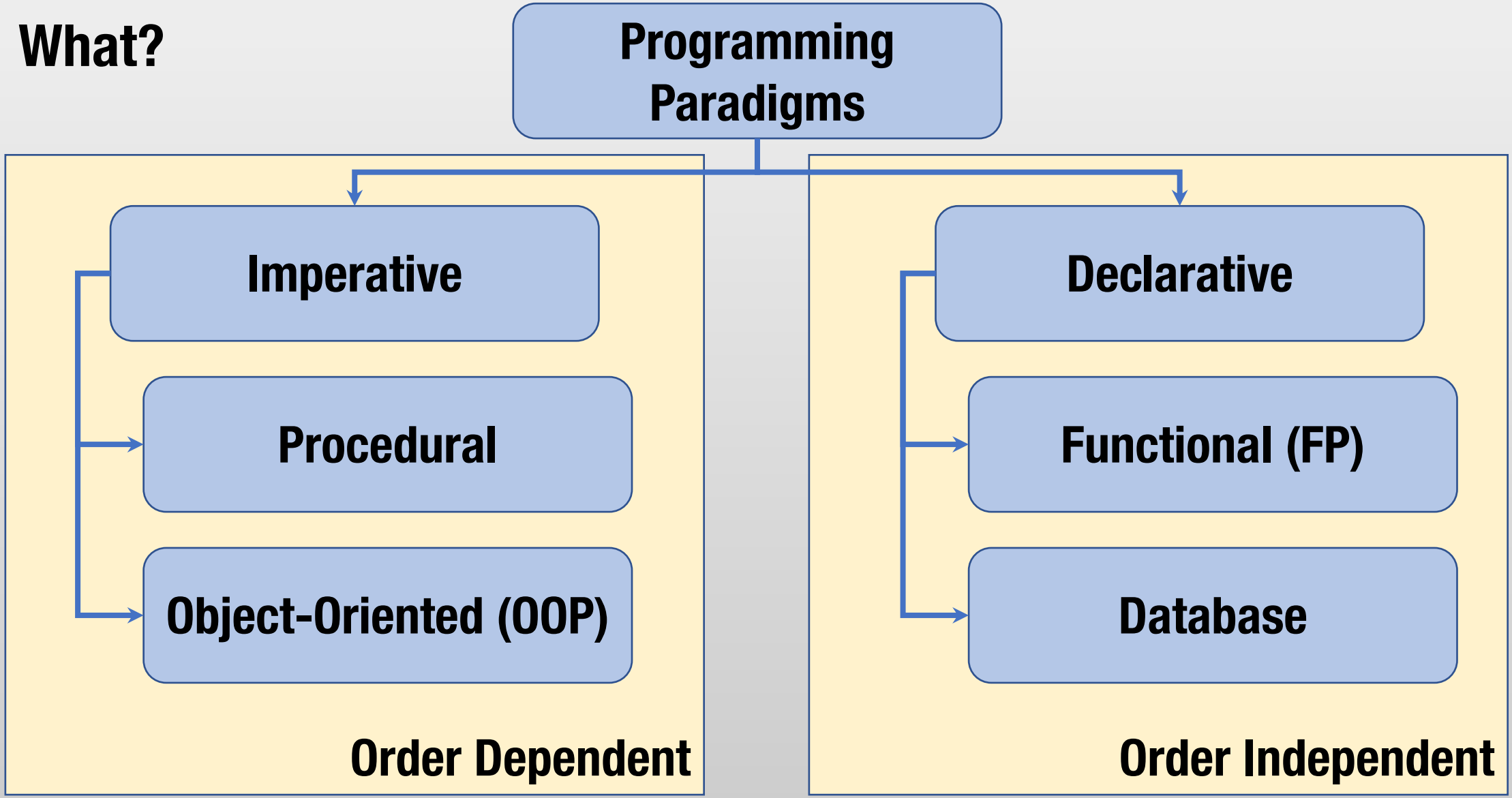
In medium skillet, heat oil over medium-high heat. Add onions to skillet, and cook until softened, about 3 minutes. Reduce heat to medium, add garlic and ginger, and cook another 2 minutes. Add garam masala, tomato paste and salt; cook and stir 2 minutes.

Step 2

Place chicken pieces in a slow cooker, then add tomato paste mixture, lime zest and juice, coconut milk and chicken stock. Stir everything together, cover and cook on low heat setting for 4½ to 5 hours, until the chicken is cooked through. (You may let it cook up to 7 hours if necessary, but the chicken may be very soft and shred.) Garnish with cilantro and serve with basmati or jasmine rice, and naan if you have some.

Reference: Steinhauer J. Slow-Cooker Butter Chicken, *New York Times Cooking*, https://cooking.nytimes.com/recipes/1018803-slow-cooker-butter-chicken

What?


Programming Paradigms
Imperative
Procedural
Object-Oriented (OOP)
Order Dependent
Declarative
Functional (FP)
Database
Order Independent

What?

Summation Example

```
# imperative
total = 0
for item in items:
    total += item.value
```

```
# declarative
total = sum([item.value for item in items])
```

```
# FP
total = map(sum, items)
```

Outline:

Reproducibility: Motivate, define, explore.

Declarative: Code style, link to reproducibility and automation.

Case Study: One attempt at a solution.

Q&A: What are your pain points and bottlenecks?

Problem: Given large data sets, e.g., finite element analysis results, how can we

- reduce the amount of analyst time required to post-process the data, and
- for parameter studies, how can we achieve consistency in the post-processing of our results across datasets and across time such that
  - all analysts' results are uniform,
  - all results are forever reproducible and retain a pedigree that links the post-processing output to the source database FEA output?

Solution: Created a Python package called xyfigure, which maintains state data in a .json file (since then, I recommend using .yml, a superset of .json, because .yml is easier to read, write, and debug).

This file-based approach is the declarative "noun" that is acted upon by "verb" programs (functions) that re-represent but do not mutate the underlying source data.

Description: The xyfigure Python package is open source due to the Office of Naval Research mandate for collaboration with our university and industry partners. The package created uniform, reproducible, and pedigreed output as publication-ready high-resolution LaTeX figures. Outputs were used directly in our *Military Medicine* publication.

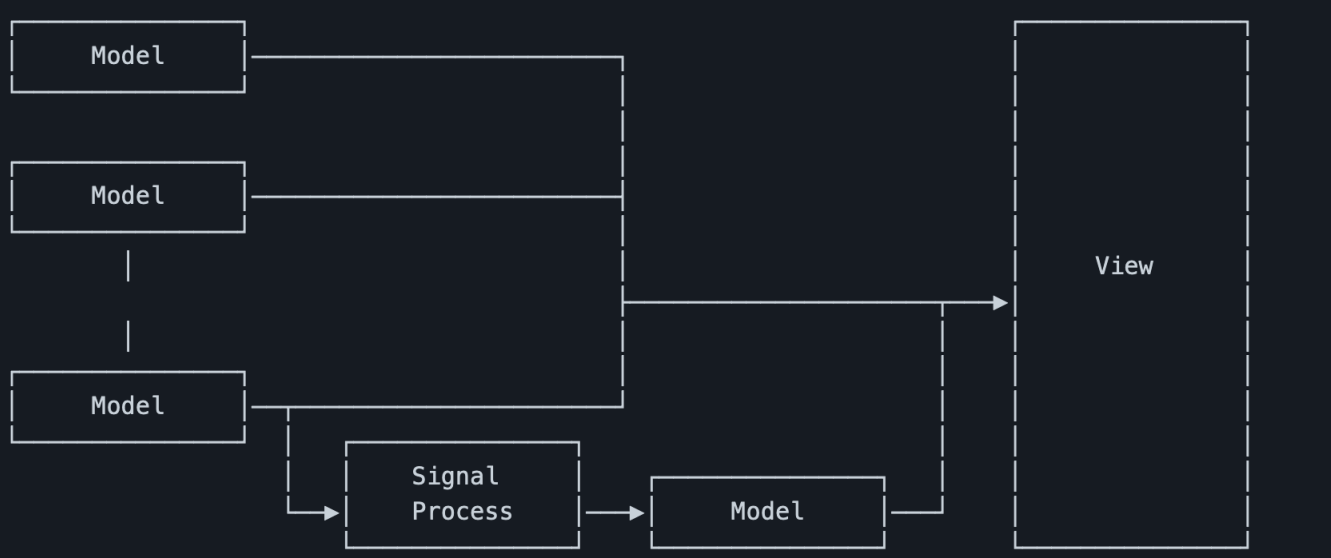
See https://github.com/sandialabs/sibl/tree/master/cli/src/xyfigure

Functions on Data: Signal processing – Butterworth filter, differentiation, integration, cross-correlation, three-points angular velocity.

Declarative .json: **Users operate on creating a dictionary of key-value pairs ("nouns") that get operated on by functions ("verbs"), without the user needing to create, debug, and maintain code.**

See https://github.com/sandialabs/sibl/blob/master/cli/doc/documentation.md

Signal processing may be performed on one or more models, using the `signal_process` dictionary, to create a new model, which can also be used by the view. A conceptual flow diagram of multiple models, one with signal processing, and one view is shown below.



Input:

XYFigure Dictionary

Below are dictionary `"key": value` pairs, followed by a description, for each of the XYFigure dictionary constitutents.

Main XYFigure Dictionary

The XYFigure dictionary is the main dictionary.

- It is stored in an ordinary text file in `.json` format.
- It is composed of one or more `model` dictionaries, followed by one or more `view` dictionaries.

`"model_name":`	dict	The key `"model_name"` is a `string` that must be a globally unique identifier (`guid`) in the `.json` file. Contains the `model` dictionary. Non-singleton; supports `1..m` models.
`"view_name":`	dict	The key `"view_name"` is a `string` that must be a globally unique identifier (`guid`) in the `.json` file. The key Contains the `view` dictionary. Non-singleton, supports `1..n` views. **Note:** In general, this `"view_name"` key can be any unique string. However, when the `.json` input file is to be used with the unit tests, this `"view_name"` key string must be exactly set to `"figure"` for the unit tests to work properly.

Input:

Model Dictionary

The model dictionary contains items that describe how each `(x,y)` data set is constructed and shown on the view.

`"class":`	`"model"`	Specific string to generate the XYModel Python class. In the model dictionary, the `"class"` key must have a value of `"model"`.
~~`"folder":`~~	string	**DEPRECATED 2021-10-29** Value *relative to the current working directory* of the path and folder that contains the input data. For the current working directory, use `"."`.
`"folder":`	string	**NEW** Value of the absolute file path to the `file`. Supports `~` for user home constructs `~/my_project/input_files` as equilvalent to, for example, `/Users/chovey/my_project/input_files`.
`"file":`	string	Value of the comma separated value input file in `.csv` (comma separated value) format. The first column is the `x` values, the second column is the `y` values. The `.csv` file can use any number of header rows. Do not attempt to plot header rows; skip header rows with the `skip_rows` key.
`"skip_rows":`	integer	*optional* The number of header rows to skip at the *beginning* of the `.csv` file. Default value is `0`.
`"skip_rows_footer":`	integer	*optional* The number of footer rows to skip at the *end* of the `.csv` file. Default value is `0`.
`"xcolumn":`	integer	*optional* The *zero-based index* of the data column to plotted on the x-axis. Default is `0`, which is the **first column** of the `.csv` file.

Output:

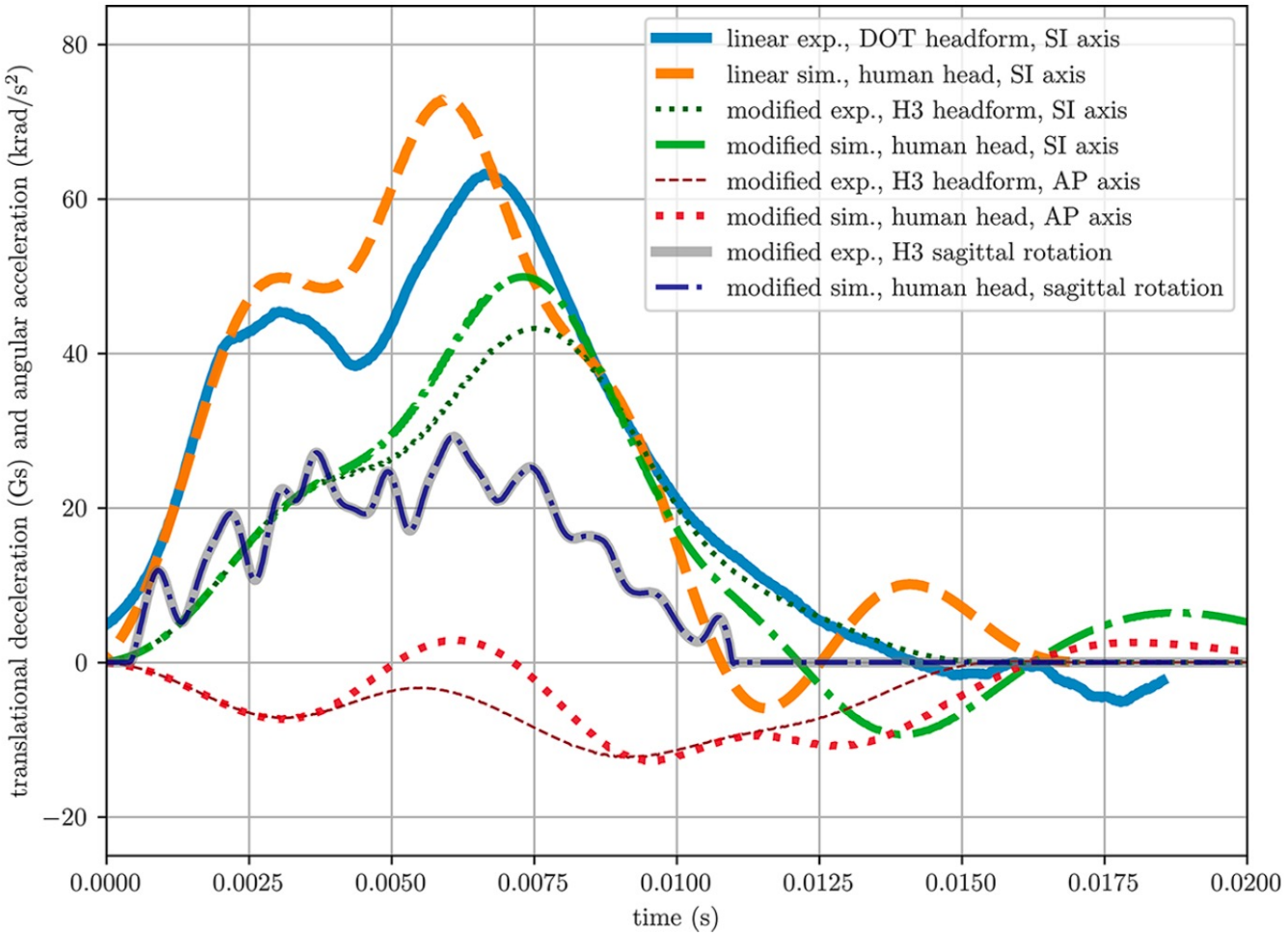


FIGURE 2. Translational deceleration time histories in the superior-inferior (SI) direction are shown by the bold solid curve for the linear constrained experiment (Department of Transportation headform) and by the bold dashed curve for the linear constrained simulation (human head). Translational decelerations in the anterior-posterior (AP) and superior-inferior (SI) direction are shown by the dot/dash-dot and dash/dot curves, respectively, for the modified experiment (Hybrid III or H3 headform) and simulation (human head). The angular head acceleration about the sagittal plane is shown to be the same for the modified experiment (solid curve) and simulation (dash-dot curve).

Reference: Terpsma R, Carlsen RW, Szalkowski R, Malave S, Fawzi AL, Franck C, Hovey C. Head impact modeling to support a rotational combat helmet drop test. Military medicine. 2021 Sep 11 and https://www.sandia.gov/app/uploads/sites/147/2022/01/Terpsma_2021_001.pdf.

Outline:

Reproducibility: Motivate, define, explore.

Declarative: Code style, link to reproducibility and automation.

Case Study: One attempt at a solution.

Q&A: What are your pain points and bottlenecks?

Additional Links:

map, filter, reduce

https://realpython.com/python-map-function/
https://realpython.com/python-filter-function/
https://realpython.com/python-reduce-function/

https://github.com/hovey/pyschool/blob/master/src/pyschool/pattern/functional_programming.py
https://github.com/hovey/pyschool/blob/master/src/pyschool/pattern/functional_programming_02.py
https://github.com/hovey/pyschool/blob/master/src/pyschool/pattern/functional_programming_03.py

snlcopyright

https://github.com/sandialabs/snlcopyright