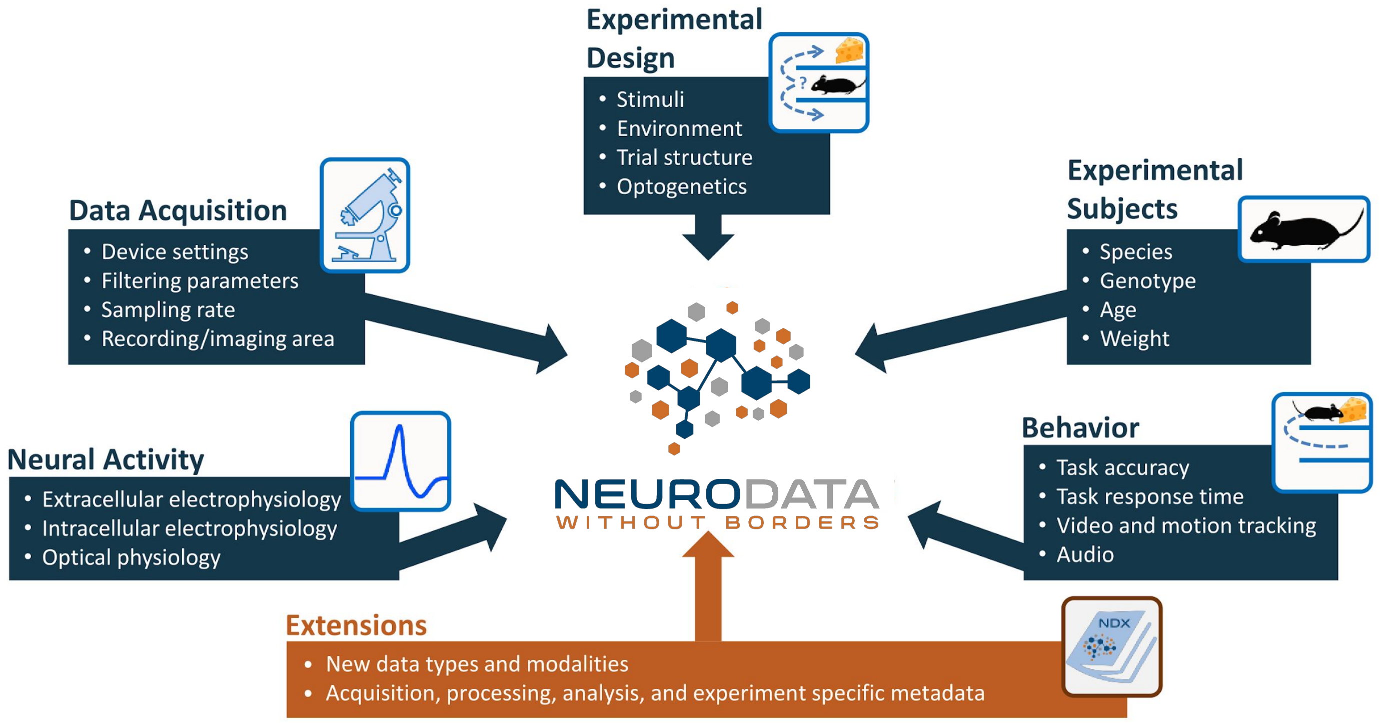
NeuroConv: A Python package for automatic conversions to NWB

July 25, 2022-
July 27, 2022

Cody Baker
Szonja Weigl
Heberto Mayorquin
Alessio Buccino
Ben Dichter

Neurodata Without Borders (NWB) packages all data needed for re-analysis



CATALYST NEURO

PSA: Rebranding

The alpha version of this project developed over the last couple of years was known as “NWB Conversion Tools”

As we approach an official v1.0.0 release, this has been renamed to “NeuroConv”, the original project proposed by DANDI that had an equivalent goal.

nwb-conversion-tools==0.11.38 >>> neuroconv 0.1.0

But future fixes and new features will only be found in NeuroConv

Converting data to NWB - Manual

The classical way of designing an NWB conversion is to utilize PyNWB or MatNWB to…

- Manually create file & add basic hard-coded metadata
- Figure out how to read from your source data within that programming language
    i. Can sometimes be very tricky
    ii. If there is an external package designed for this, it may have bugs you don't discover until you attempt this process
    iii. If designing the data access yourself, it can be challenging to safely parse every single detail of the original design
- Relies on somewhat detailed knowledge of the NWB standard (User Workshops certainly help with this!)
- Despite a library of tutorials for how to use PyNWB and MatNWB, it is impossible to cover every type of data (*i.e.*, extensions)
- Advanced I/O can be challenging (learn more about that tomorrow!)
    - Data is often too large to fit into memory
    - How to quickly and efficiently convert data on the TB scale
    - How to use the best lossless compression to reduce data size for both local and cloud storage

For maximal customization and control, this is certainly still the way to go…
But for conversions from common proprietary formats to standard NWB data types, there is an easier way!

Converting data to NWB - Automatic

neuroconv makes this task easier by joining with SpikeInterface, Neo, and ROIExtractors; ecosystems that support read/write for a large variety of established formats from Intracellular, Extracellular, and Optical electrophysiology (25+ Recording, 27+ Sorting, 8+ Segmentation/Imaging)

- Many of these formats are tested using continuous integration on example datasets available at git-annex and G-Node GIN (ecephys+icephys and ophys)
- In each of these ecosystems, we've built the data engineering tools for the NWB write methods so that users only need to call a few lines of code

```
from datetime import datetime
from dateutil import tz
from neuroconv import TiffImagingInterface

file_path = … / "imaging_datasets" / "Tif" / "demoMovie.tif"
nwbfile_path = ".../my_nwbfile.nwb"
session_start_time=datetime(2020, 1, 1, 12, 30, 0, tzinfo=tz.gettz("US/Pacific"))

interface = TiffImagingInterface(file_path=file_path, sampling_frequency=15.0)
metadata = interface.get_metadata()
metadata["NWBFile"].update(session_start_time=session_start_time)
interface.run_conversion(nwbfile_path=nwbfile_path, metadata=metadata)
```

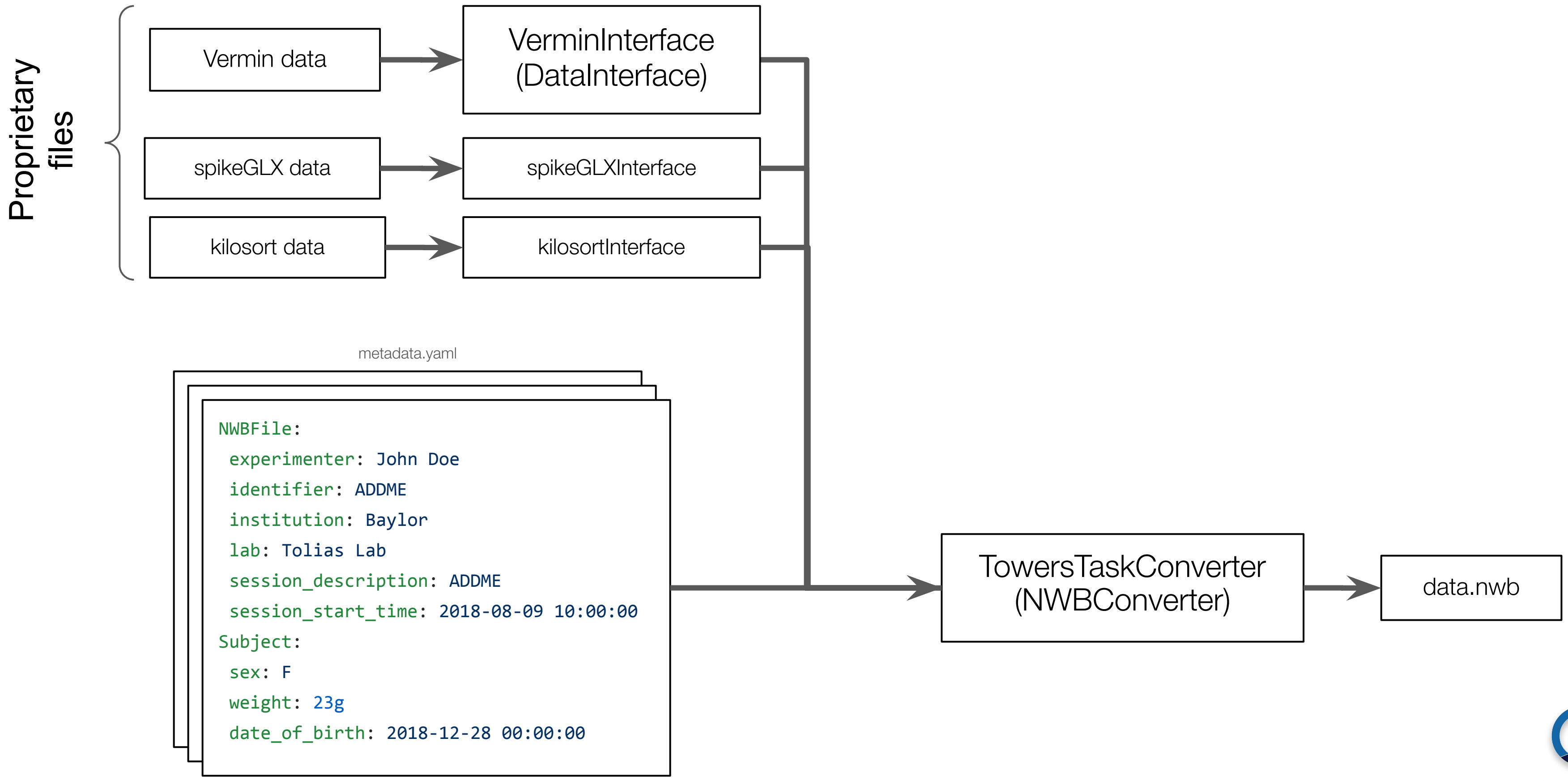
CATALYST NEURO

Converting data to NWB - Automatic

Maintaining these NWB write functions for external packages isn't the *only* aspect of the **neuroconv**…

- The primary task is to reduce complexity when integrating multiple data sources
  - A very common task is to pair neurophysiology recordings with timed behavior detection, such as events or position tracking
- We simplify this process via two classes: `NWBConverter` and `DataInterface`
- The `NWBConverter` handles the NWBFile I/O, optionally including file creation. It contains `DataInterfaces`; one for each data stream
- A `DataInterface` defines a standard workflow for initializing I/O from a data source, how to extract and structure text-based metadata, and defines how to write that data into an NWBFile (often using those functions from external packages)

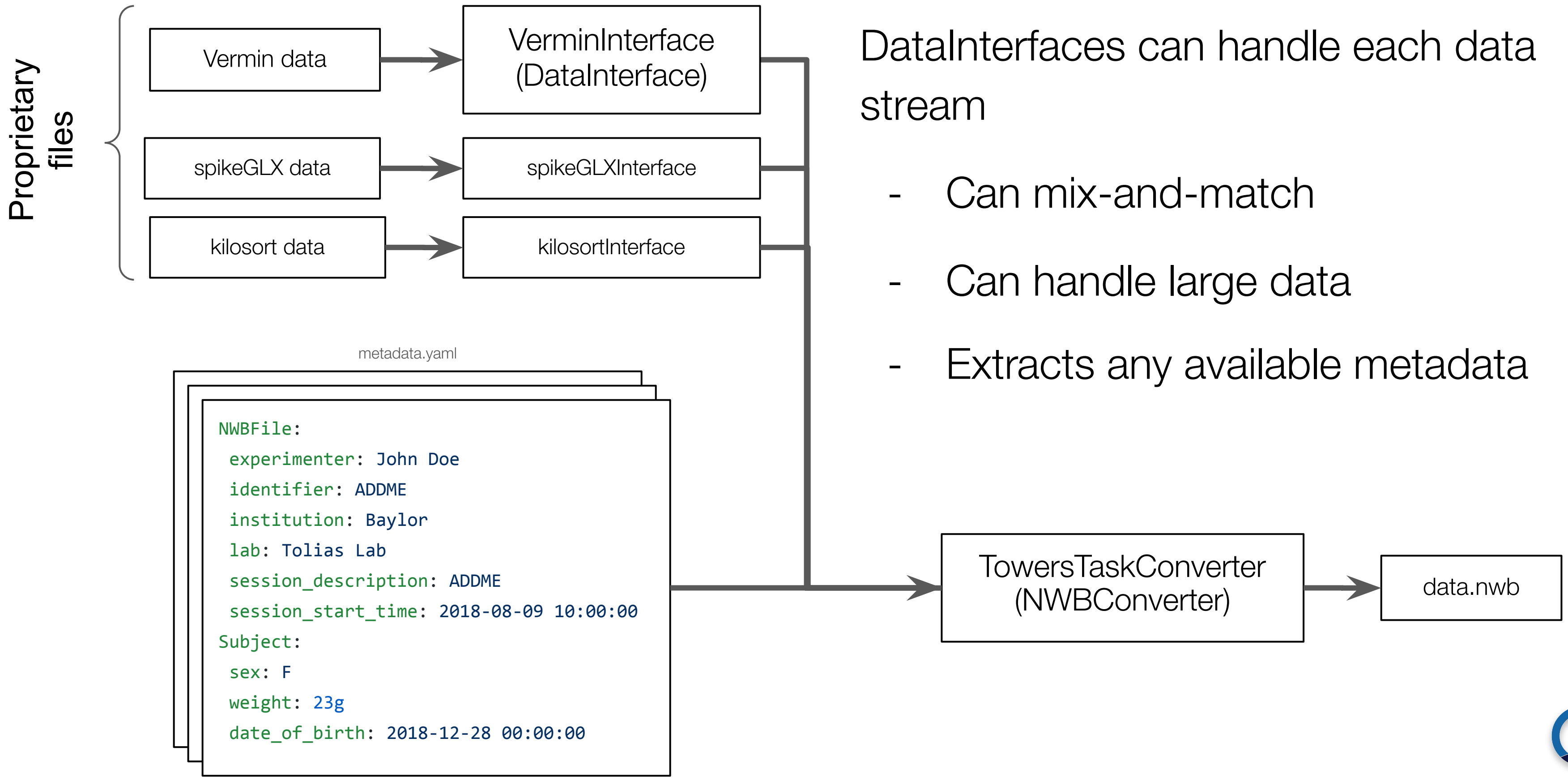
Strategy: Modularize by data stream



```yaml
NWBFile:
 experimenter: John Doe
 identifier: ADDME
 institution: Baylor
 lab: Tolias Lab
 session_description: ADDME
 session_start_time: 2018-08-09 10:00:00
Subject:
 sex: F
 weight: 23g
 date_of_birth: 2018-12-28 00:00:00
```

CATALYST NEURO

Strategy: Modularize by data stream

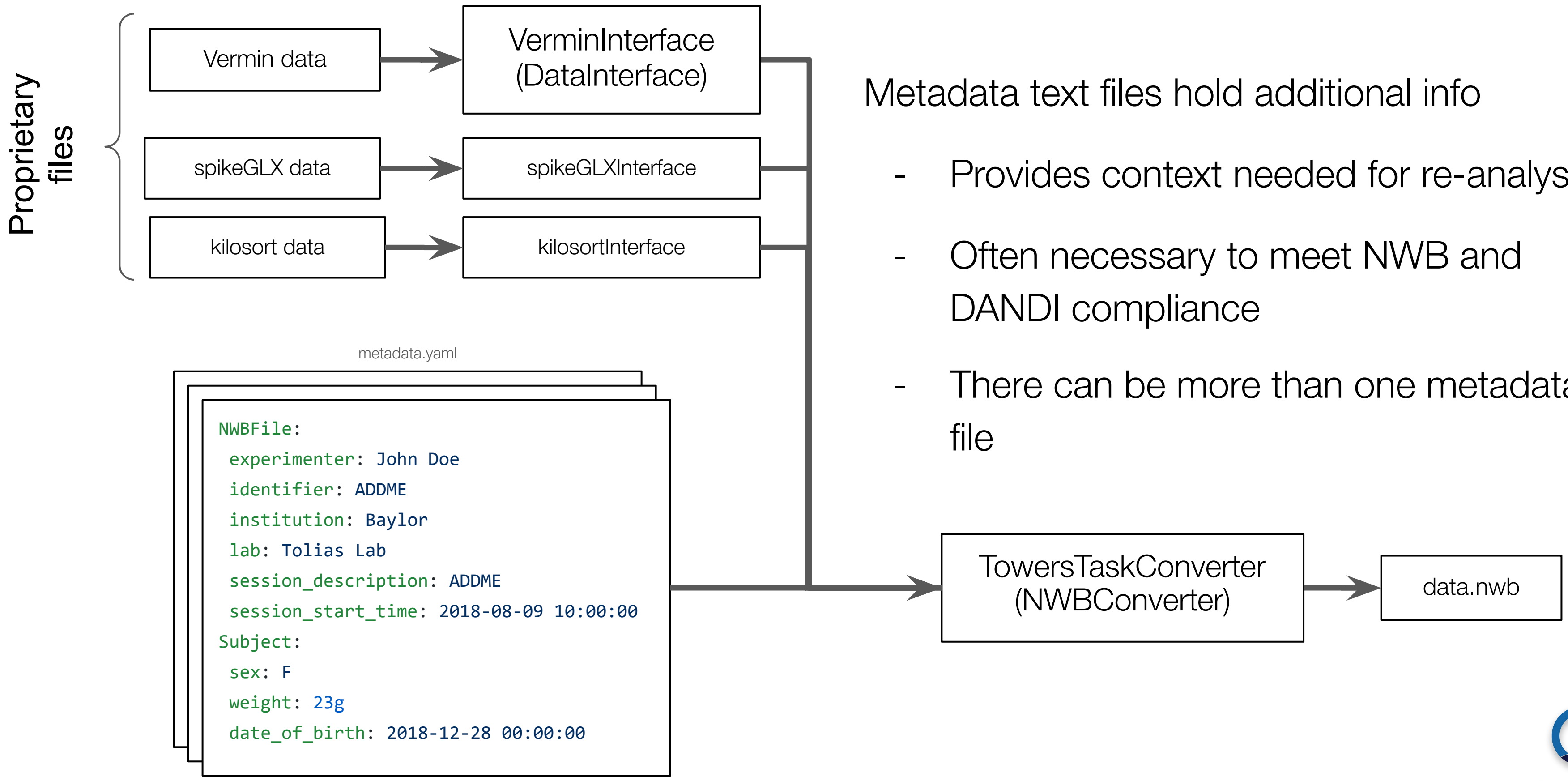


```yaml
NWBFile:
 experimenter: John Doe
 identifier: ADDME
 institution: Baylor
 lab: Tolias Lab
 session_description: ADDME
 session_start_time: 2018-08-09 10:00:00
Subject:
 sex: F
 weight: 23g
 date_of_birth: 2018-12-28 00:00:00
```

DataInterfaces can handle each data stream

- Can mix-and-match
- Can handle large data
- Extracts any available metadata

Strategy: Modularize by data stream

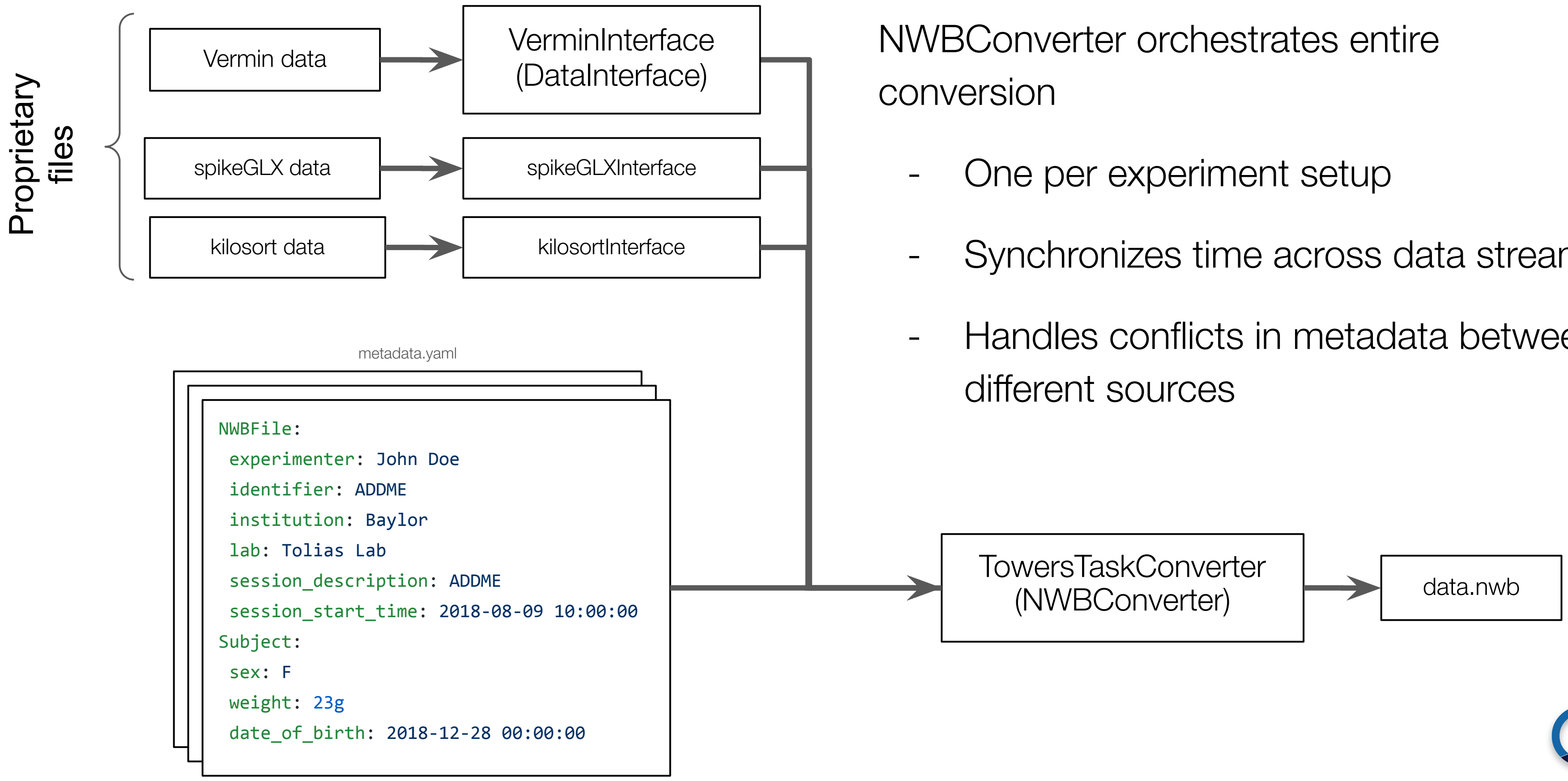


```yaml
NWBFile:
 experimenter: John Doe
 identifier: ADDME
 institution: Baylor
 lab: Tolias Lab
 session_description: ADDME
 session_start_time: 2018-08-09 10:00:00
Subject:
 sex: F
 weight: 23g
 date_of_birth: 2018-12-28 00:00:00
```

Metadata text files hold additional info

- Provides context needed for re-analysis
- Often necessary to meet NWB and DANDI compliance
- There can be more than one metadata file

Strategy: Modularize by data stream



```yaml
NWBFile:
 experimenter: John Doe
 identifier: ADDME
 institution: Baylor
 lab: Tolias Lab
 session_description: ADDME
 session_start_time: 2018-08-09 10:00:00
Subject:
 sex: F
 weight: 23g
 date_of_birth: 2018-12-28 00:00:00
```

NWBConverter orchestrates entire conversion

- One per experiment setup
- Synchronizes time across data streams
- Handles conflicts in metadata between different sources

CATALYST NEURO

Recent additions: YAML Conversion Specification

For users less familiar with coding in Python, we have developed a text-only YAML language for specifying a full NWB conversion

- Which is basically just file/folder paths and metadata not contained anywhere else
- Example: GIN_conversion_specification.yml
- Callable from the command line! (also the primary way of uploading to DANDI)

Demonstration of Hands-on Tutorial

- Slides and notebooks+ are located on the GitHub for the workshop projects:

  https://github.com/NeurodataWithoutBorders/nwb_hackathons/tree/main/HCK13_2022_Janelia/projects

- Instructions for setup are in the README.md
  - Everything is already setup to run from the DANDIHub platform
  - Running locally might require some downloading of example data

Showcase of Customized Conversions

- You can build custom DataInterfaces that use PyNWB to support a wide range of different and complex experiment types

- Check out our catalogue of labs that have used past versions of the project (mostly on the previous name of “NWB Conversion Tools”)

  - Showcase:
    - Custom data interfaces for signal retrieval from acquisition system
    - Synchronization with behavioral systems (TTL signals)
    - Continuously recorded signals that span multiple sessions/files

Data Access and Transfer Tools

Even if you don't plan on using NeuroConv for performing your entire conversion, feel free to check out our collection of data management tools including…

- Constant updates to large-array buffering/iteration (many features of this may eventually find their way to HDMF)
- Automated data transfer submission for Figshare, Globus
  - Rclone support coming soon
- Some convenience helper functions for accessing and creating NWBFiles and their contents
- Handy utility functions for interacting with JSON and YAML files, path globbing, typing, and more!

https://github.com/catalystneuro/neuroconv/tree/main/src/neuroconv/tools

and

https://github.com/catalystneuro/neuroconv/tree/main/src/neuroconv/utils

Future Features

- We want to hear about new feature requests, additional data formats (we will request example data, though!) or other enhancements: simply open a ticket at

  https://github.com/catalystneuro/neuroconv/issues

- Some items we have planned in the long-term include…
  - A user-friendly drag-and-drop interface web interface + graphical form for populating metadata
  - AWS-deployable resources for super-fast parallel conversion
  - Specifiable iteration patterns for folder-based YAML-defined conversion