**Overview**

To maximize the return-on-investment into creation of neuroscience data sets and enhance reusability, it is critical to have data through standardized and extensible data model and management solutions. In addition to standardizing metadata and data, support for best practices and data interoperability, fault tolerance, and an advanced software architecture at facilitating the use, extension, integration, and maintenance of NWB is critically needed. NWB-N project aims to ensure that the software foundation of NWB adheres to these principles and enables efficient management of an ever-growing collection of neuroscience data sets. Here, we apply software engineering principles to create an advanced software architecture and define abstractions to enable separation of the NWB specification language, format data, storage, and data API.

**Specification Language:**

How to formally define neuroscience data standards?

To support the formal and verifiable specification of neuroscience data files, NWB defines and uses the NWB specification language. To organize complex data, the specification language uses easy-to-use primitives, e.g., groups, datasets, Attributes, and links. For NWB 2.0, we have simplified and extended the specification language to ease readability, interpretability and expressiveness. Specifically, we:

- Simplified the use of several primitives, e.g., inheritance and inclusion of types
- Added support for object- and region-referent data types and improved specification of links, e.g., for compound data types
- Added support for compound data types, enabling the specification of tables and complex data types
- Added support for default names and values
- Replaced scalar encoded in keys with explicit key-value pairs to avoid collision of human keys and ease human interpretation
- Added easy-to-use mechanisms for reading and writing data

**Format Specification:**

How to organize complex collections of neuroscience data?

The NWB specification language is specifically the NWB specification language and formally defines the organization of neuroscience data via the NWB file format. For NWB 2.0, we:

- Introduced the concept of NWBContainer and NWBDataSets as common base neurodata types for information modeling and to improve clarity of organization of processed data in NWB
- Recognized unique neurodata types / name / object for all objects for identification
- Improved and expanded meta-data library and object access, search, organization, and interpretation via a combination of:
  - A central ElectrodeTable for electro-metabas and
  - A compliant ElectrodeGroup containers for collective electrode metadata
- Replaced implicit links and data-structures with explicit links and relationship models to facilitate unique direct human and programmable data interpretation
- Assigned unique neurodata types / name / object for all objects for identification
- Improved consistency and completeness of metadata and object names
- Improved organization of the metadata schema via new format release and documentation
- Improved metadata implementation for improved and improved clarity of the organization of data

**Data Storage:**

How to store large collections of neuroscience data?

A central ElectrodeTable for electro-metabas and a compliant ElectrodeGroup containers for collective electrode metadata. For NWB 2.0, we:

- Introduced the concept of NWBContainer and NWBDataSets as common base neurodata types for information modeling and to improve clarity of organization of processed data in NWB
- Recognized unique neurodata types / name / object for all objects for identification
- Improved consistency and completeness of metadata and object names
- Improved organization of the metadata schema via new format release and documentation
- Improved metadata implementation for improved and improved clarity of the organization of data

We continue to see a vibrant community forming around NWB-N. Below we highlight several collaborative efforts with and contributions to the NWB-N community related to the development of NWB 2.0 beta.

**Using Modern Software Processes for PyNWB**

To enable a collaborative development process we have developed a new modular architecture for open-source development and project management processes using GitHub. The Architecture was designed to ensure software quality and accessibility and we have developed:

- An advanced and extensible software architecture for PyNWB
- A continuous integration test suite
- A quick start guide for using PyNWB

**NWB-N Governance**

To guide the next phase of growth, and to coordinate the various efforts, NWB-Neurophysiology has created a governance structure. This will allow NWB-N to grow in a bottom-up, but coordinated manner. Participation in the effort is open to any interested neuroscientist, with overall planning coordinated by an Executive Board (EB) with seven members, each appointed for a term of 2-3 years. The current membership of the Executive Board is K. Bouchard (UCB), D. Gennady (HHMI Janelia), K. Svoboda, U. Lowell (HHMI), (LBNL) Kiggins (Kitware) and countless emails and phone calls.

**NWB-N Data and Storage**

Data are stored in a variety of formats as part of the NWB-N infrastructure. NWB-N includes:

- NWB Core components
- NWB Data models
- NWB File format
- NWB Data API
- NWB Validation

**NWB-N HPC**

With PyNWB we have developed a new modular architecture and API in the last data read/write and high-performance, parallel data analysis is critical to enable labs to keep up with ever growing data volumes. The Neurophysiology Without Borders (NWB-N) project uses towards generating a unified data format for cellular neuroscience data for a multi-user, multi-backend environment and an advanced software architecture at facilitating the use, extension, integration, and maintenance of NWB-N is critically needed. NWB-N-HPC project aims to ensure that the software foundation of NWB adheres to these principles and enables efficient management of an ever-growing collection of neuroscience data sets. Here, we apply software engineering principles to create an advanced software architecture and define abstractions to enable separation of the NWB specification language, format data, storage, and data API.