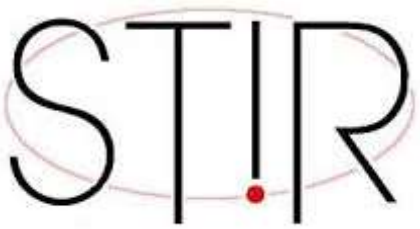
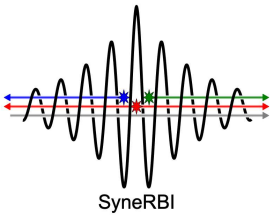




PET Raw Data Standardisation and the role of Open Source Software

Kris Thielemans
on behalf of the
PET Raw Data Standardisation Initiative

PET Acquired Data Standards Initiative

- Originated from discussions on *Sitek, A. et al. Artificial Intelligence in PET: An Industry Perspective. PET Clinics 16, 483–492 (2021)*
- Pushed forward by Arman Rahmim, led by Glenn Wells
- Large group of interested people
  - ~50 total
  - Academics + industry
  - Clinical and preclinical
  - First meeting Feb 2022
  - Informal and not currently associated to any society

UCL

PET Acquired Data Standards Initiative

Vision:

- To create an open standardized data format for raw emission-tomography acquisition data that will allow data storage, pre-reconstruction processing and evaluation, and reconstruction of clinically-acceptable and quantitative images.

Mission:

- Our short-term goal is to define a standardized (vendor-independent) data format and content for PET listmode and associated data; define the standard used for storage and transmission of that data; and develop software to access and manipulate the standardized data.
- Our long-term goal is to expand the standard's scope to include other emission modalities such as SPECT and planar nuclear imaging.

UCL

Sub-groups

- Data Elements (content)
  Kris Thielemans (UCL) and Adam Kesner (MSK)
  - *List mode data*
  - *Calibration data (normalisation etc)*
  - *Sinograms*
  - *Associated data (ECG, respiratory etc)*
- Data Storage Format (container)
  Michael Hansen (MS)
- Data Transmission
- Use cases

UCL

How?

- Description will be human *and* machine readable.
- As opposed to a “format”, we are working towards an API, automatically generated from the formal description.
- Vendor-format will be converted to standard (different output “container formats” will be supported)
  - converters should be written with vendor support
  - forward looking: older scanners might not be supported

UCL

How does this benefit OSS?

- No need for research agreements
- Vendor-independent processing
  - Much larger impact
  - Opens potential for multi-centre trials using (almost) identical processing => image standardization (more than harmonization)
  - Big data opportunities, future-proofed for yet-undeveloped reconstruction algorithms
- Removes need to write difficult and hard-to-maintain code to decode existing file formats (efficiently)

UCL

How does this initiative benefit from OSS?

- Practical expertise in the community
- “Description -> API” will be OSS
- OSS will form the basis for converters
- OSS Hackathons/projects to implement use-cases

UCL

More information on Description/API

- Led by Michael Hansen (MS) and implemented by John Stairs (MS)
- Will be used for MR data (replacing ISMRMRD)
- Working prototype, will be OSS on GitHub (end of 2022?)

UCL

Code generation tooling for streaming data formats

Michael Hansen & John Stairs

domain model (DSL)

Code Generator

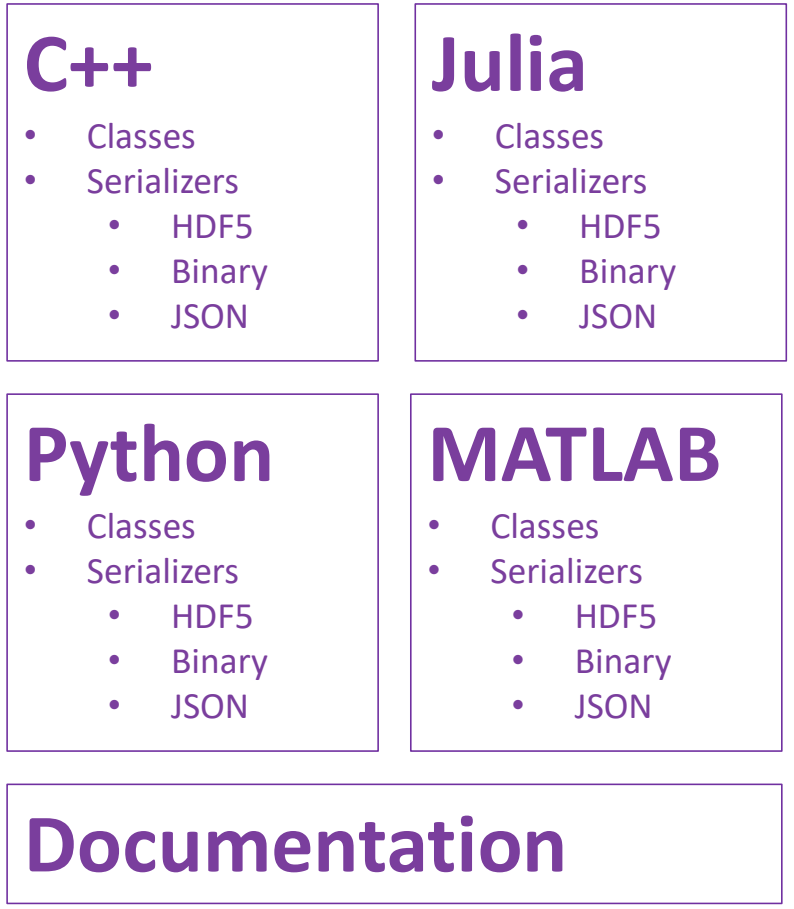
→

- Classes
- Serializers
  - HDF5
  - Binary
  - JSON

domain model (DSL)

Code Generator

→



```
Header: !record
  fields:
    subject: SubjectInformation?
    conditions: !vector
      items: ExperimentalCondition

Acquisition: !record
  fields:
    header: Header
    data: !array
      items: complexfloat32

RawData: !protocol
  sequence:
    header: Header
    data: !stream
      items: Acquisition
```

Code Generator

→

```
struct Header {
  // ...
};

struct Acquisition {
  // ...
};


class RawDataWriter {
  // ...
};

class RawDataReader {
  // ...
};
```

```
{
  Binary::RawDataWriter writer("test.bin");
  writer.WriteHeader({SubjectInformation{"John Doe"}, {ExperimentalCondition{42}}});
  Acquisition acquisition;
  for (int n = 0; n < 10; ++n) {
    acquisition.data = {{1.0 + 2.0i, 3.0 + 4.0i}};
    writer.WriteData(acquisition);
  }
  writer.EndData();
}
{
  Binary::RawDataReader reader("test.bin");
  Header header;
  reader.ReadHeader(header);
  std::cout << "Header: " << header.subject->name << std::endl;
  Acquisition acquisition;
  while (reader.ReadData(acquisition)) {
    std::cout << "Acquisition: " << acquisition.data << std::endl;
  }
}
```

Role of STIR

- Existing list mode code could be used to provide a single converter for data formats supported by STIR
- STIR data formats will need to be adapted to work with the new API
- Help realise the potential of PET