Discussion questions and points

DUNE framework taskforce and developer workshop
June 2024

Nature of data products: what are the features of their types?

A *data product* is an object that can be passed by the framework to an algorithm (in any supported language). The framework will do some amount of bookkeeping for data products.

1. Data types and language interoperability
    1. Should a data product in Python look “pythonic”?
    2. Or is it OK for Python algorithms to work with data types that are implemented in C++, and wrapped for Python? E.g, can a data product type contain member function templates?
    3. What happens if and when DUNE wants to add another language to the list? (Julia, Rust, ...)
2. Inventing new data product types
    1. Should one be able to define a new data product type without writing C++ code?
    2. Must all data product types support reading and writing from permanent storage (IO)?
    3. Must all data product types be representable in all languages?
3. Data product types and IO support
    1. Can you read *part* of a data product from a file, without reading the *entire* data product?
    2. If so, how fine-grained is the reading access?

Persistency

- Differences between tabular and object-oriented data
  1. art uses ROOT storage in an object-oriented way (row-wise vs. columnar)
  2. art can give you all the vertices of a single event
  3. art cannot give you all the z coordinates of the vertices in the input file at one time
  4. Do you require the ability to do 3 in the framework program or directly on the ROOT file with no extra framework or library support?
- Do you have use cases for HDF5 other than reading DAQ files?
- When do you hope to use Python in a framework context?
- What kind of dependencies are allowable for a Python algorithm?

Checkpointing, failures and errors

- A ***failure*** is an event from which the framework cannot recover (e.g. segmentation violation, receiving a kill signal, cut power cord).

- We understand ***checkpointing*** to mean:

  A process used in computing to save the state of a running application or system at a specific point in time, enabling it to be resumed from that point in case of later failure.

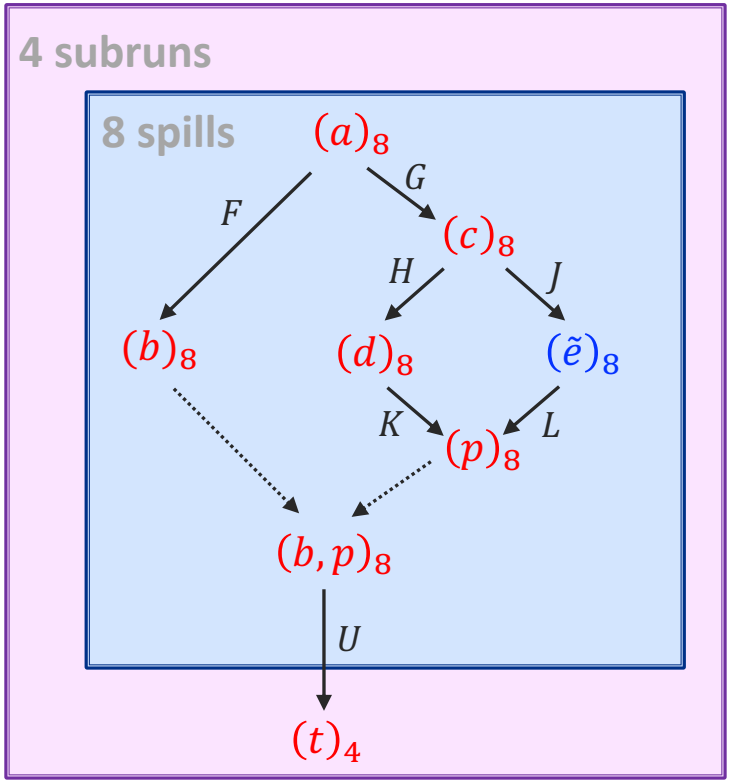
  Is this what you mean?

- An ***error*** is an undesired event upon which the program can take mitigating action.
  - Errors that cannot be mitigated locally are signaled via an exception.

Checkpoints in a framework

- Checkpoints are identified in the workflow and work performed thus far is saved.
  - In event-centric frameworks, all event data are written at the end of each event.
    - Run and subrun products that are accumulations of event data are problematic if a failure occurs.
  - In a multi-threaded environment, multiple algorithms can execute concurrently, and the granularity of the data processed by the algorithms can vary substantially.
    - For a system that supports FPUs, FPU data could be written after each algorithm executes.
- What are the consistency guarantees required of the file being written?
- It's an issue of making efficient use of computing resources. Where and what are you trying to optimize?

Checkpointing and concurrency



Checkpointing needs to be carefully thought out when executing a graph of algorithms in parallel.

On the left:

- Sets of data products are aggregated into data families (subruns and spills)
- Data product sequences are formed within a family—e.g. $(a)_8$ corresponds to the 8 data products labeled a from each of 8 spills.
- Data products need not by persisted to a file—e.g. $(\tilde{e})_8$, where the tilde means in-memory only.
- User-defined algorithms are applied to each data product in a sequence and are represented by capital letters (F, G, H, etc.).

Backwards compatibility wrt. art

- ProtoDUNE I and II data and algorithms are based on art and LArSoft, but DUNE ND and FD will be based on the new framework and LArSoft.
- What type of backward compatibility is required?
  1. Must DUNE be able to read data from existing art/ROOT files in the new framework?
  2. Must DUNE be able to read current ND files with the new framework?
  3. Must DUNE be able to build existing art modules for use in the new framework?
  4. Must DUNE be able to build existing “art-dependent” algorithms for use in the new framework?
  5. Must DUNE be able to use existing FHiCL files in the new framework?
  6. Is it acceptable to have a translation program that can ingest an art/ROOT file and emit an output file in the format expected by the new framework?
- Our expectations:
  1. Some changes will be required for art-using code to enable the above.
  2. Services will not be an integral part of the new framework.
  3. art will not be able to read new framework files.

Back-up slides

Definitions

Data product

An object of data the framework can provide as an input to a user-defined algorithm, or that can be produced as an output of an algorithm.

Data (product) set

A mathematical set of data products that is identifiable by the framework and used to determine which data products serve as inputs to an algorithm.

Data family

A category of collection of data sets (e.g.):

- *examples in art:* run, event, and subrun
- *other examples:* calibration interval, geometry/alignment interval, APA, trigger primitive, beam spill

Data family hierarchy

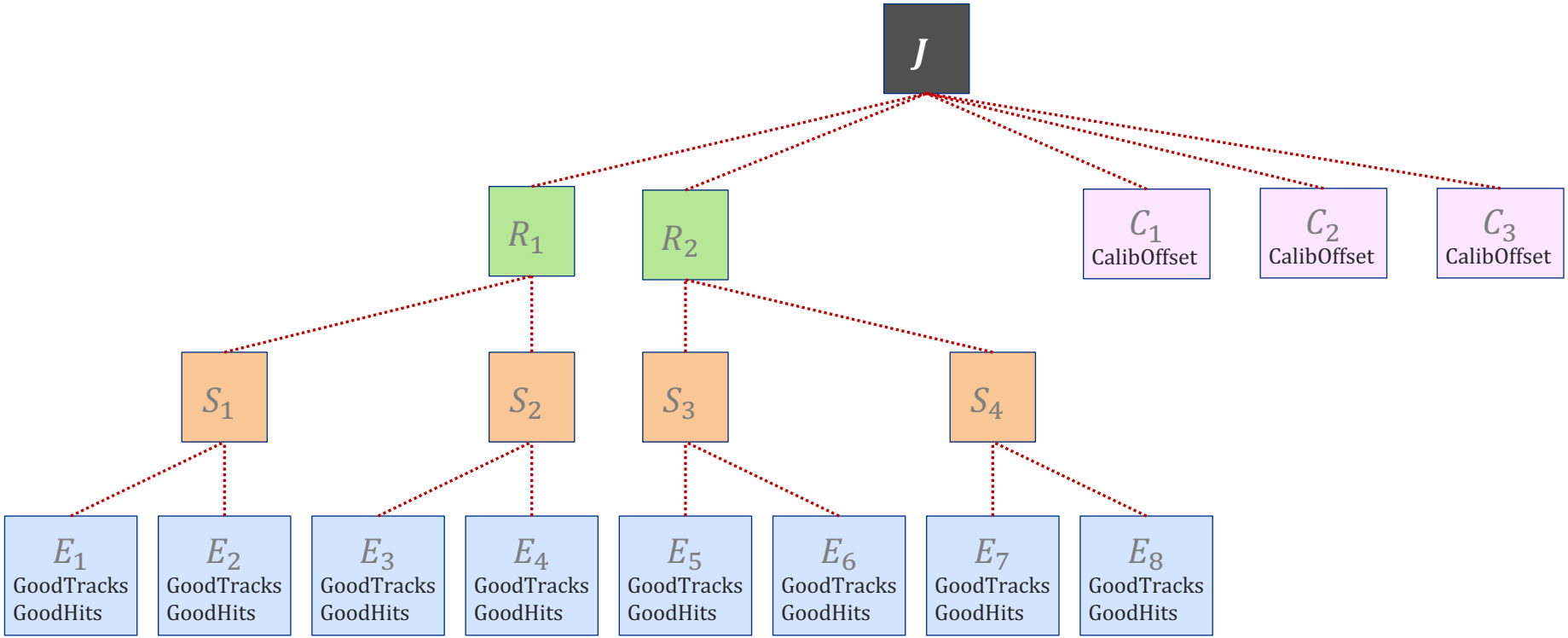
A hierarchy of data families (e.g. run ⊃ subrun ⊃ event)

Data model

A set of mechanisms enabling the definition, creation, identification, and organization of data products, as well as the relationships among them. The data model also specifies the mechanism for reading and writing persistable data products.

Definitions (in pictures)

From services discussion on April 26



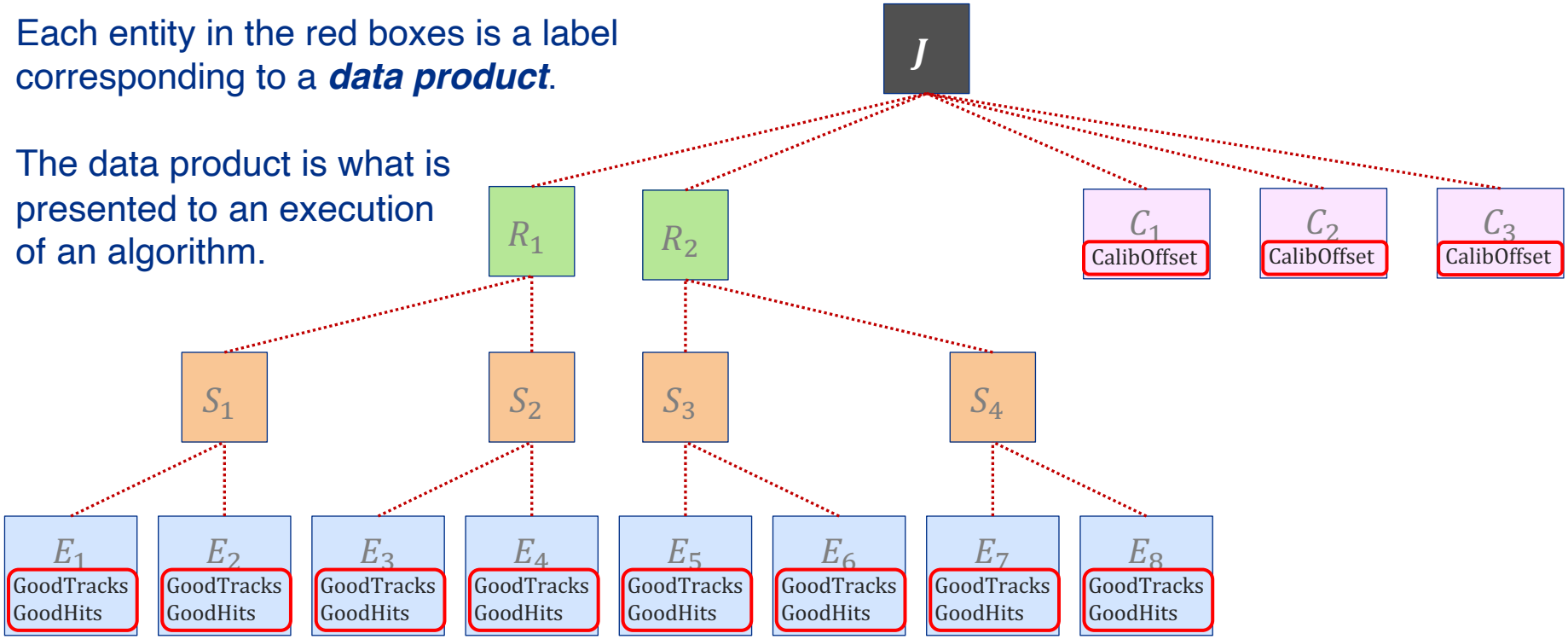
Fermilab

Definitions (in pictures)

From services discussion on April 26

Each entity in the red boxes is a label corresponding to a ***data product***.

The data product is what is presented to an execution of an algorithm.

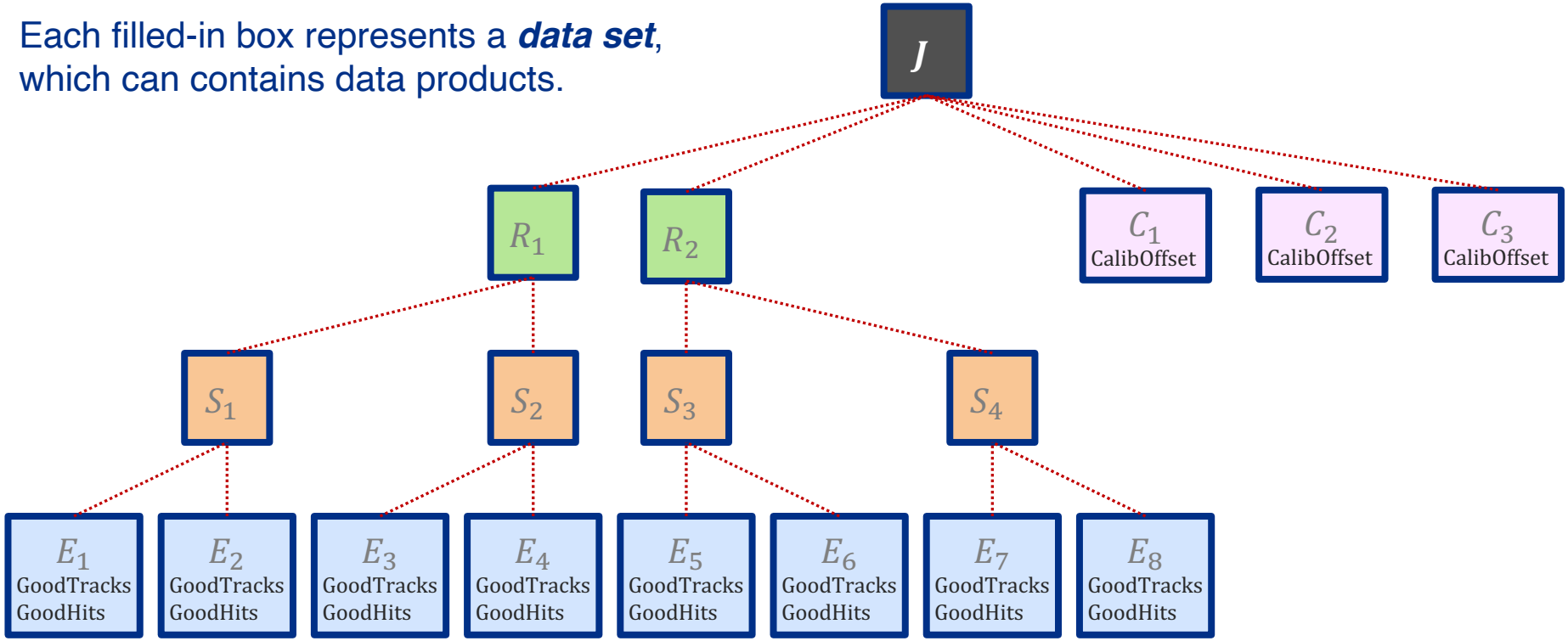


Fermilab

Definitions (in pictures)

From services discussion on April 26

Each filled-in box represents a ***data set***, which can contains data products.

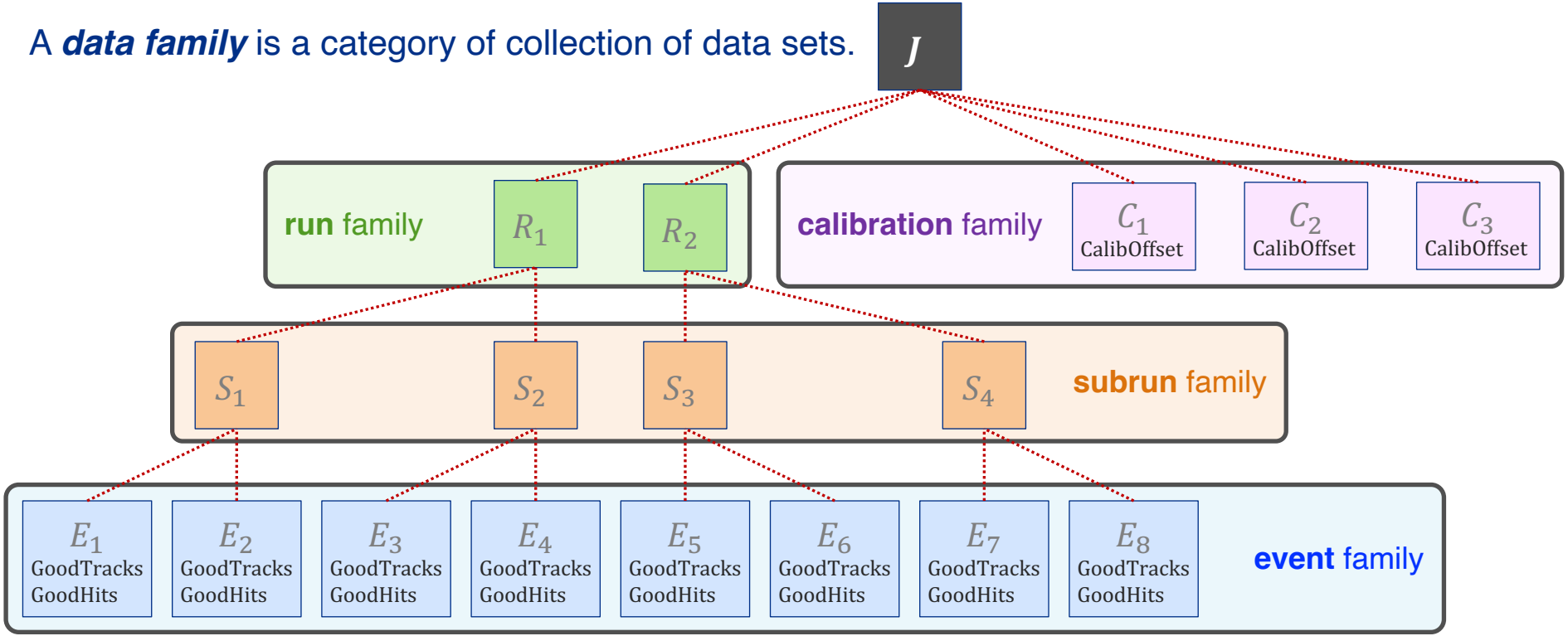


Fermilab

Definitions (in pictures)

From services discussion on April 26

A ***data family*** is a category of collection of data sets.



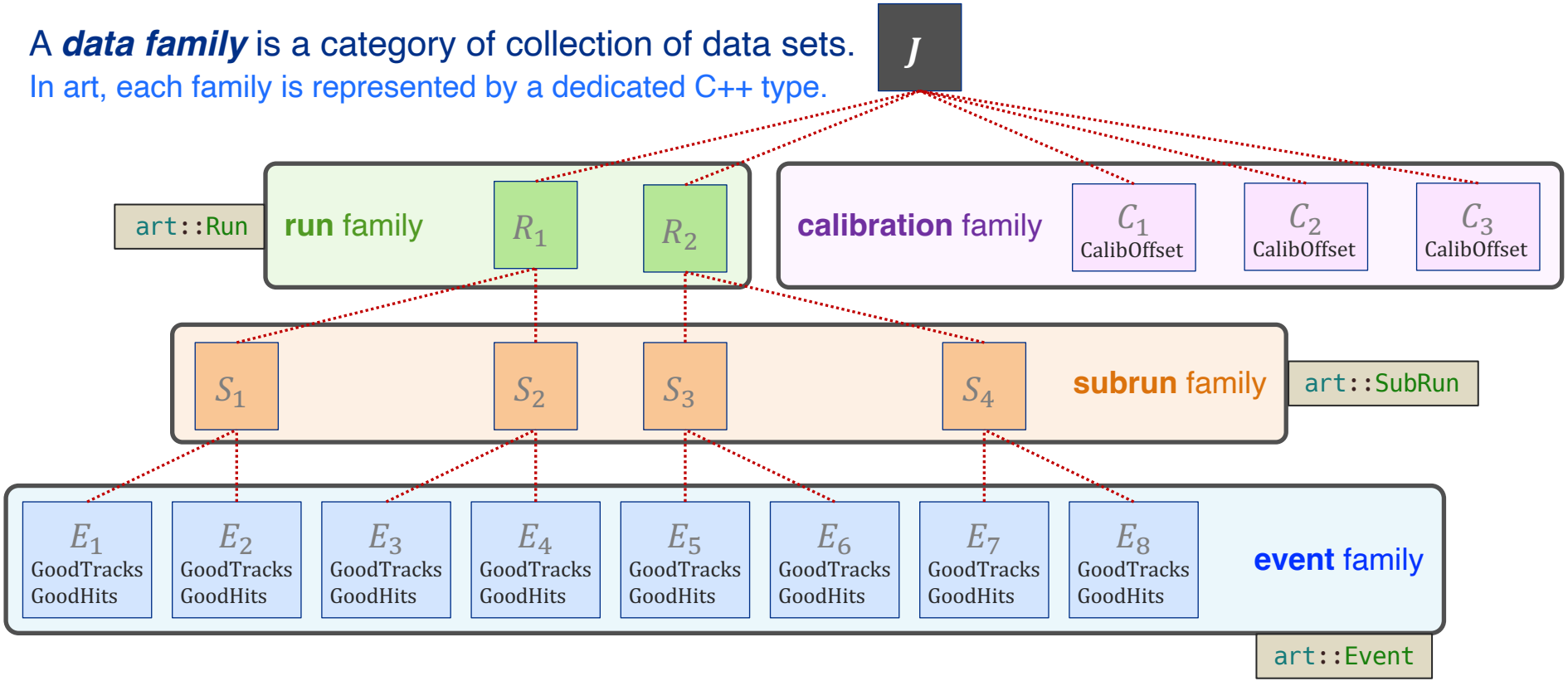
Fermilab

Definitions (in pictures)

From services discussion on April 26

A ***data family*** is a category of collection of data sets.

In art, each family is represented by a dedicated C++ type.



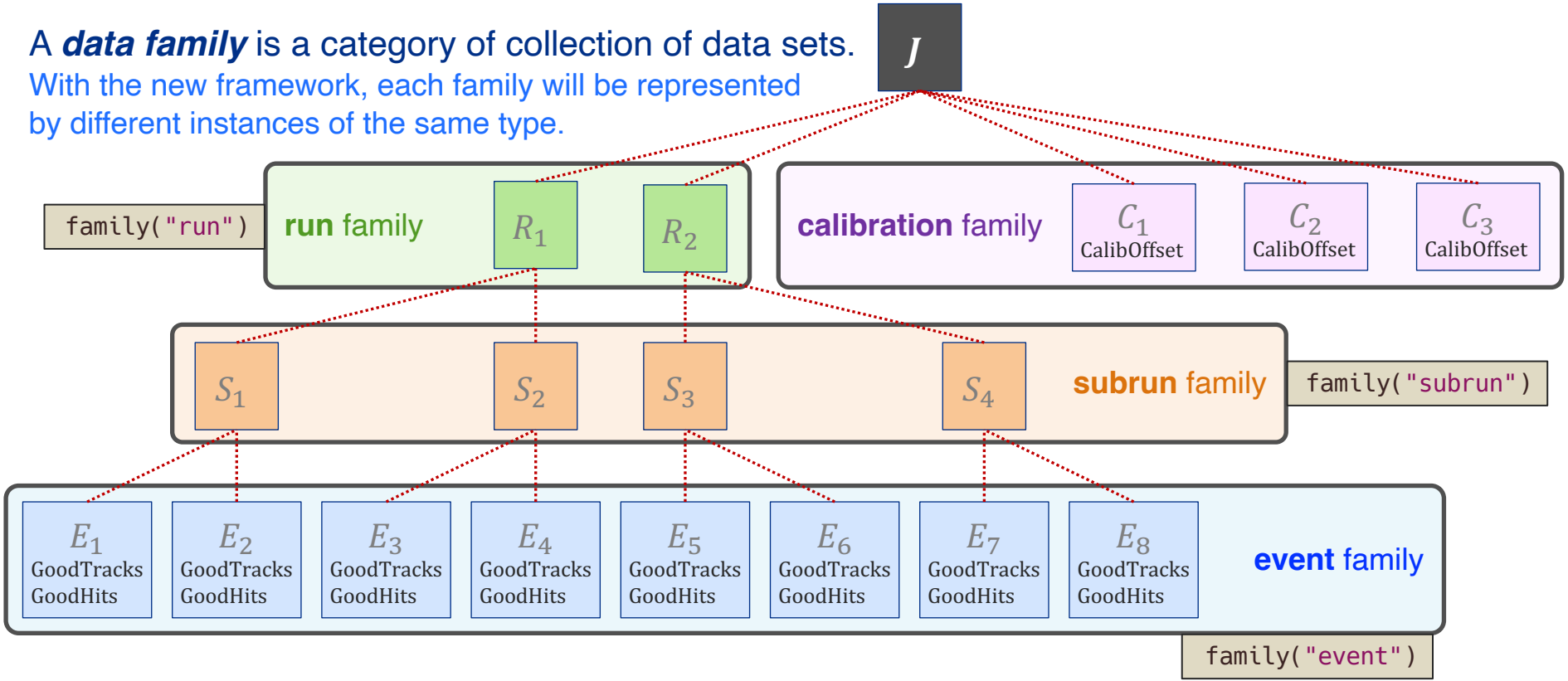
Fermilab

Definitions (in pictures)

From services discussion on April 26

A ***data family*** is a category of collection of data sets.

With the new framework, each family will be represented by different instances of the same type.

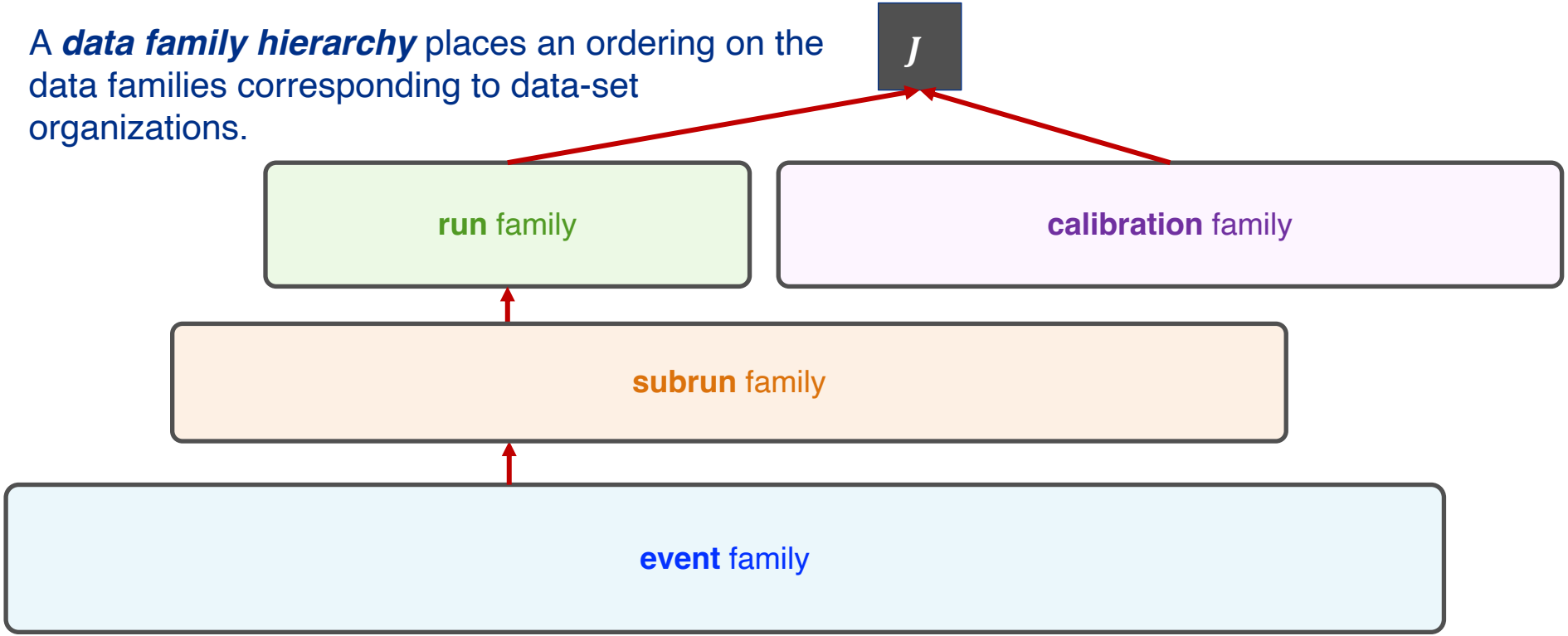


Fermilab

Definitions (in pictures)

From services discussion on April 26

A ***data family hierarchy*** places an ordering on the data families corresponding to data-set organizations.



Fermilab

When are families specified?

There are different time periods of relevance for a framework job:

When are families specified?

There are different time periods of relevance for a framework job:

1. Compile time

Period during which code is compiled into binary libraries, which can be loaded during the runtime of a framework program.

Family can be hard-coded **(art)**

When are families specified?

There are different time periods of relevance for a framework job:

1. **Compile time**

   Period during which code is compiled into binary libraries, which can be loaded during the runtime of a framework program.

   Family can be hard-coded **(art)**

2. **Configuration run time**

   Period during which a framework program ingests a user's configuration to determine what should be processed and how. The execution engine is initialized at this stage, but no physics data are processed.

   Family can be configuration parameter

When are families specified?

There are different time periods of relevance for a framework job:

1. **Compile time**

   Period during which code is compiled into binary libraries, which can be loaded during the runtime of a framework program.

   Family can be hard-coded **(art)**

2. **Configuration run time**

   Period during which a framework program ingests a user's configuration to determine what should be processed and how. The execution engine is initialized at this stage, but no physics data are processed.

   Family can be configuration parameter

3. **Processing run time**

   Period during which physics data are processed.

   Family, in principle, can be calculated from input data

When are families specified?

There are different time periods of relevance for a framework job:

1. **Compile time**

   Period during which code is compiled into binary libraries, which can be loaded during the runtime of a framework program.

Family can be hard-coded **(art)**

2. **Configuration run time**

   Period during which a framework program ingests a user's configuration to determine what should be processed and how. The execution engine is initialized at this stage, but no physics data are processed.

Family can be configuration parameter

3. **Processing run time**

   Period during which physics data are processed.

We cannot think of use cases that require such flexibility. Are you aware of any that do?

Family, in principle, can be calculated from input data

When can data sets within a family be specified?

Although we expect the specification of data families to be static once the configuration step is complete, we expect that the introduction to the framework of data sets within a family are dynamic throughout the job.

art example:

> art's execution engine reacts to whatever it receives from the input source. Although art allows only run, subrun, and event data sets, the number and contents of each data set is defined by the input source.
>
> art allows only the input source to specify when a data set is ready for processing.

When can data sets within a family be specified?

Although we expect the specification of data families to be static once the configuration step is complete, we expect that the introduction to the framework of data sets within a family are dynamic throughout the job.

art example:

> art's execution engine reacts to whatever it receives from the input source. Although art allows only run, subrun, and event data sets, the number and contents of each data set is defined by the input source.
>
> art allows only the input source to specify when a data set is ready for processing.

We expect that DUNE must be able to specify data sets not only by an input source **but also when processing data with regular user-defined algorithms** (e.g. splitting/unfolding data for more granular processing).

> To the best of our knowledge, no existing HEP data-processing framework has this ability.

Example of dealing with adjacent data

```
SomeOtherData process_both(SomeData old_data, SomeData data) { ... }
```

Example of dealing with adjacent data

```
SomeOtherData process_both(SomeData old_data, SomeData data) { ... }
```

art-like style

```
class MyProducer : public art::EDProducer {
public:
  void produce(art::Event& e)
  {
    auto new_data = e.getProduct<SomeData>("some_data");
    auto old_data = std::exchange(data_, new_data);
    if (old_data == SomeData::invalid()) {
      return;
    }
    auto some_other_data = process_both(old_data, data_);
    e.put(std::make_unique<SomeOtherData>(some_other_data),
          "some_other_data");
  }

private:
  SomeData data_{SomeData::invalid()};
};
```

Example of dealing with adjacent data

```
SomeOtherData process_both(SomeData old_data, SomeData data) { ... }
```

art-like style

```
class MyProducer : public art::EDProducer {
public:
  void produce(art::Event& e)
  {
    auto new_data = e.getProduct<SomeData>("some_data");
    auto old_data = std::exchange(data_, new_data);
    if (old_data == SomeData::invalid()) {
      return;
    }
    auto some_other_data = process_both(old_data, data_);
    e.put(std::make_unique<SomeOtherData>(some_other_data),
          "some_other_data");
  }

private:
  SomeData data_{SomeData::invalid()};
};
```

This approach *requires* **all data** to be presented in a time-ordered fashion.

Very difficult to achieve efficient concurrent processing without extensive bookkeeping from the module author/user.

Example of dealing with adjacent data

```
SomeOtherData process_both(SomeData old_data, SomeData data) { ... }
```

art-like style

```
class MyProducer : public art::EDProducer {
public:
  void produce(art::Event& e)
  {
    auto new_data = e.getProduct<SomeData>("some_data");
    auto old_data = std::exchange(data_, new_data);
    if (old_data == SomeData::invalid()) {
```

An alternative

```
REGISTER(m)
{
  m.with(process_both)
   .transform("some_data"_in("raw"), "some_data"_in("raw"))
   .related_by(some_adjacency_criterion)
   .to("some_other_data");
}
```

Let the framework do the work.

The framework incurs responsibility of invoking the algorithm concurrently.

Example of dealing with adjacent data

```
SomeOtherData process_both(SomeData old_data, SomeData data) { ... }
```

art-like style

```
class MyProducer : public art::EDProducer {
public:
  void produce(art::Event& e)
  {
    auto new_data = e.getProduct<SomeData>("some_data");
    auto old_data = std::exchange(data_, new_data);
    if (old_data == SomeData::invalid()) {
```

An alternative

```
REGISTER(m)
{
  m.with(process_both)
   .transform("some_data"_in("raw"), "some_data"_in("raw"))
   .related_by(some_adjacency_criterion)
   .to("some_other_data");
}
```

Let the framework do the work.

The framework incurs responsibility of invoking the algorithm concurrently.

User specifies adjacency criterion used to associate the input arguments to the algorithm.