Discussion questions and points

DUNE framework taskforce and developer workshop

September 2024

Access to hardware resources

Many of the hardware-related requirements were very general. To be able to verify that a requirement is satisfied, it must be specific:

Access to hardware resources

Many of the hardware-related requirements were very general. To be able to verify that a requirement is satisfied, it must be specific:

6) Individual modules within the framework MUST be allowed to specify particular hardware requirements.
    a) For example, modules MUST be allowed to require GPU resources.

Access to hardware resources

Many of the hardware-related requirements were very general. To be able to verify that a requirement is satisfied, it must be specific:

Too general	6) Individual modules within the framework MUST be allowed to specify particular hardware requirements.
Specific	a) For example, modules MUST be allowed to require GPU resources.

Access to hardware resources

Many of the hardware-related requirements were very general. To be able to verify that a requirement is satisfied, it must be specific:

Too general	6) Individual modules within the framework MUST be allowed to specify particular hardware requirements.
Specific	a) For example, modules MUST be allowed to require GPU resources.

From the requirements document, we see the following examples of hardware:

- "CPU cores"
- "memory"
- "GPU"
- "GPU memory"
- "Other coprocessors"
- etc.

Access to hardware resources

Many of the hardware-related requirements were very general. To be able to verify that a requirement is satisfied, it must be specific:

Too general	6) Individual modules within the framework MUST be allowed to specify particular hardware requirements.
Specific	a) For example, modules MUST be allowed to require GPU resources.

From the requirements document, we see the following examples of hardware:

- “CPU cores”
- “memory”
- “GPU”
- “GPU memory”
- “Other coprocessors”
- etc.

We’ll start by refining these terms.

Fermilab

CPU cores

Does CPU “cores” mean CPU threads?

Assumed requirement: The framework shall allow the requesting of CPU threads.

What might this mean (e.g.)?

- The job-runner shall be able to specify a maximum number of CPU threads used by the job.
- The job-runner shall be able to specify a maximum number of CPU threads used by a given algorithm.
- An algorithm shall be allotted a specific number of CPU threads it can use for execution. ***(Not best practice)***

(CPU) Memory

Assumed requirement: The framework shall allow the requesting of CPU memory.

This could mean many things. Some considerations:

- Memory is a process-wide property.
- Very difficult to specify “this algorithm will require n MB of memory to execute”.
- Memory usage is reactive—the inputs to an algorithm almost always determine what objects are to be created and thus how much memory is needed.

An easy-to-meet requirement could be:

- The framework shall attempt a graceful shutdown when the maximum process-wide requested memory usage has been exceeded.

A valid, but probably infeasible, requirement might be:

- The framework shall reschedule the execution of algorithms to optimize the memory use of the framework job.

GPUs and co-processors

Assumed requirement: The framework shall allow the requesting of GPUs.

- What should the framework be expected to do? Must the framework:
  - Initialize the necessary library runtime?
  - Allocate one or more GPUs?
  - Do you have other things in mind?
- What specific parallelism technology do you intend to use?
  - Options include CUDA, HIP, oneAPI, Kokkos, SYCL, etc.
- Are there any specific concerns about GPU memory usage?
- For what other co-processors is support needed?

Other hardware resources related questions

- Are multiple compute nodes required for a single program?
- How should collaborators (job runners, algorithm authors, etc.) interact with the framework to specify resource requirements?
- Why do you need resource monitoring, what are the goals?
- What do you need to monitor for each resource? e.g. memory and time trackers

Time-ordering

What are your needs for time-ordering?

What other sorts of ordering or adjacency are relevant?

> ***See next slides for example of processing adjacent data.***

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()) {



  }

priv
  So
};
```

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()) {

  }

priv
  So
};
```

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.

Supernova data

Do you need the offline framework to process supernova data?

1. What is your time budget for processing supernova data?
2. Do both ND and FD need to process supernova data?
3. How large are the input data?
4. Will the framework need to answer “yes” or “no” to whether a supernova exists?
5. Will the framework need to filter input data to determine the direction of the supernova?
6. What are your RAM constraints?
7. What hardware will you have available?
    1. How close in proximity does that hardware need to be to the detector?
8. How have you been expecting to process this data?

Configuration

Overview:

- Configuration **system**: how the framework and its users interact with configuration:
  - Configuration sources and relative precedence
  - Persistence
  - Scope, and component encapsulation
  - Reproducibility

- Configuration **language**:
  - Kinds of values that are representable
  - Equivalence of configurations
  - Structure
  - Dynamic features

Configuration system

1. What configuration sources are required, and what is the expected order of precedence?
   - Human-readable input files
   - Framework output files
   - Command-line specification
   - Other?

2. In what forms should the configuration be recorded?
   - As-seen (pre-evaluation), or post-evaluation, or both?
   - Binary, human-readable, and/or searchable?
   - Original files, compactified/aggregated files, framework output files, database, data catalog?

Configuration system

3. What are your requirements for how the configuration document applies to the distinct elements of the configured program?
    - Application of configuration to algorithms of a certain kind
    - Specific instances of an algorithm
    - Overall program control, etc.
4. What are your requirements for visibility control: should an algorithm be able to "see" the configuration of other algorithms, and/or higher-level configuration?
5. How hard should the configuration system work to create reproducible framework programs?

– Environment variable usage

– Contents from other files

– Contents from databases

Configuration language

1. What kinds of entity need to be represented distinguishably in the configuration?
   - Basic types, e.g. strings, numbers (or integers vs decimals), …
   - Structural elements, e.g. ordered vs associative arrays, comments, …
   - Intangible or platform-dependent entities: units, infinities, NaN, nil, …

2. What language features do you need to enable maintainable configurations?
   - Fragment inclusion (external files)
   - Entity replacement and/or elision
   - Substitution of entities defined elsewhere: what kinds? Granularity (maps, map members, splices)?
   - Conditional evaluation
   - Loops with variable substitution
   - Arithmetic evaluation
   - …

Configuration questions

For the configuration system and language you currently use to configure your DUNE code:

- Which behaviors are helpful/necessary?
- Which behaviors are unhelpful/harmful?

Random Number Generators: problems with stateful PRNGs

- We are talking about what the C++ standard calls *random engines* or *random bit generators*, not *distributions*.
- Stateful generators (all the ones in the current C++ standard library) require control over access to the RNG state.
  - Multithreaded use requires locking.
  - This can make the use inefficient in MT programs.
- To obtain reproducibility one must capture and allow the user to recover the state of the RNG at a specific point in the workflow
  - And make sure the capturing is done at all the right places so that recovery of the workflow allows reproduction
  - One needs to deal with issues like recovering a job with a different number of threads... what exactly are the reproducibility requirements?
- Stateful RNGs of good quality often have large state (e.g. Mersenne Twister)

Random number generators: Counter-based PRNGs

- Counter-based PRNGs are designed for supporting concurrency.
  - Small state (therefore many instances are feasible)
  - Immutable state
- The value of the *counter* is enough to determine what the next value produced in the sequence will be.
- C++26 will (probably) contain at least one CPRNG
  - It may or may not be best to use this directly, depending on constraints imposed by the requirements of the <`random`> library.
- Direct framework support might be helpful (or needed?) to make parallel use efficient
  - if there are many different algorithms using RNGs that could run in parallel, and
  - if the random number generation facilities can support this.

Random number generators: questions

- We will need to specify what “reproducibility” means in various circumstances
  - For RNGs, as opposed to distributions
  - When running on different hardware
  - When using differing numbers of threads... etc.
- Does DUNE require the ability to use “old favorite” RNG algorithms (e.g. Mersenne twister, or James’s algorithm)?
  - If so, then we need a solution that provides thread safety with whatever efficiency can be achieved.
- Would DUNE make use of counter-based RNGs?
  - If not, then it would be a waste of time to exert design effort on integrating them.
- Does DUNE foresee a need for “true” random number sequences?
  - If the answer is yes, then reproducibility becomes very expensive (because the framework would have to capture the entire sequence of numbers generated).
  - There are considerations of execution speed because of the connection from the “source of entropy” to the generated stream of numbers.

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

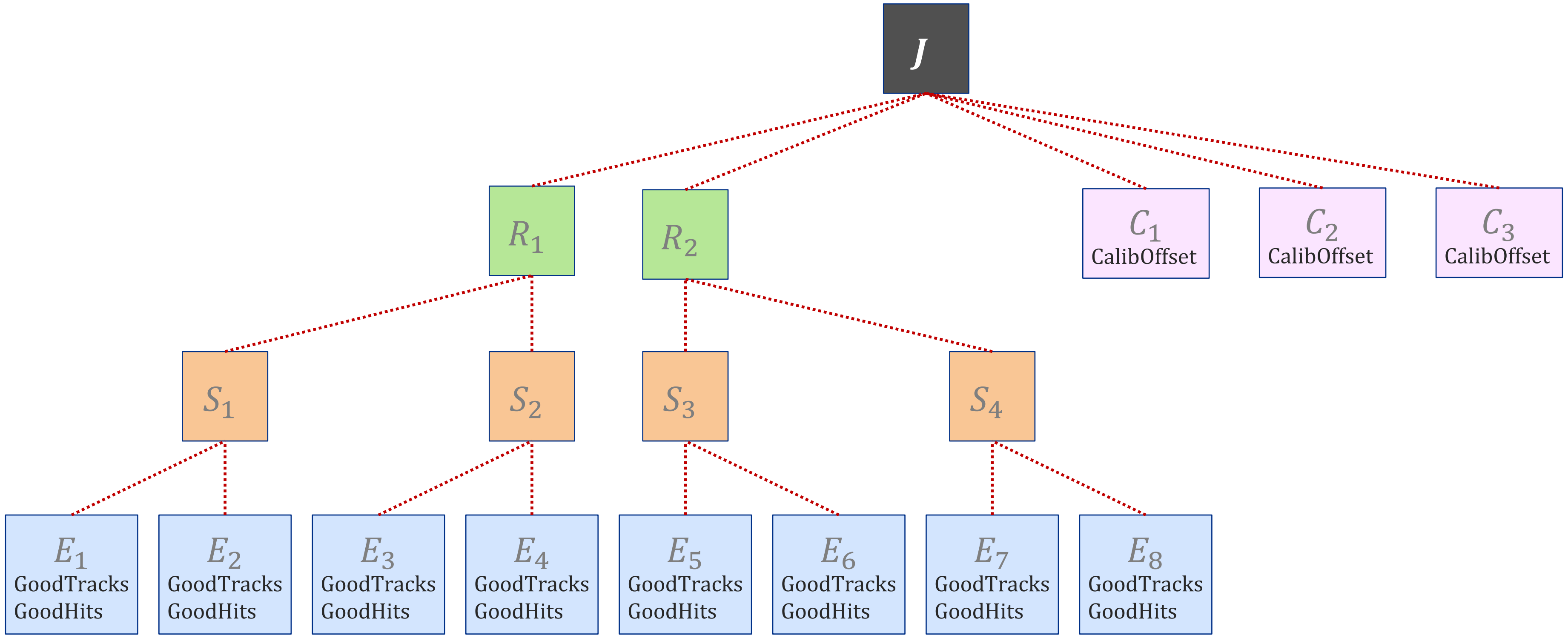
A hierarchy of data families (e.g. run $\supset$ subrun $\supset$ event)

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



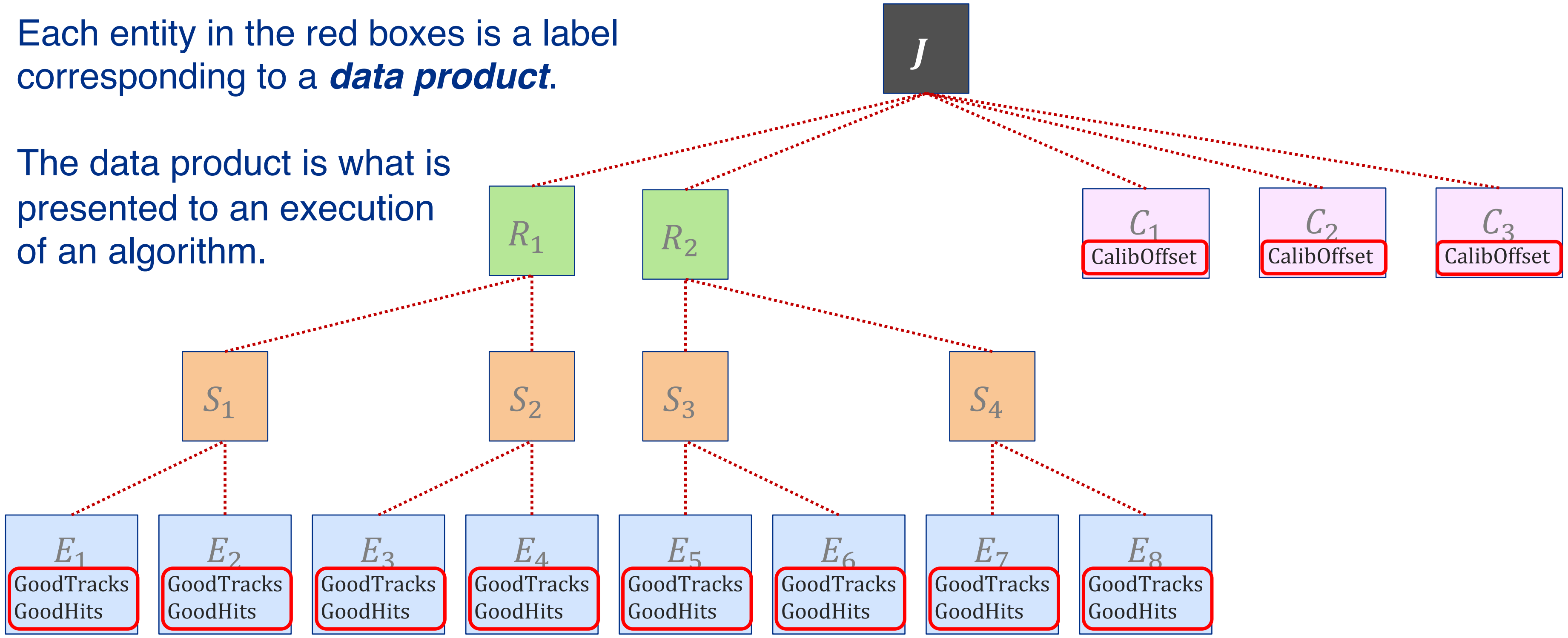
J
R_1
R_2
C_1 CalibOffset
C_2 CalibOffset
C_3 CalibOffset
S_1
S_2
S_3
S_4
E_1 GoodTracks GoodHits
E_2 GoodTracks GoodHits
E_3 GoodTracks GoodHits
E_4 GoodTracks GoodHits
E_5 GoodTracks GoodHits
E_6 GoodTracks GoodHits
E_7 GoodTracks GoodHits
E_8 GoodTracks GoodHits

Definitions (in pictures)

From services discussion on April 26

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

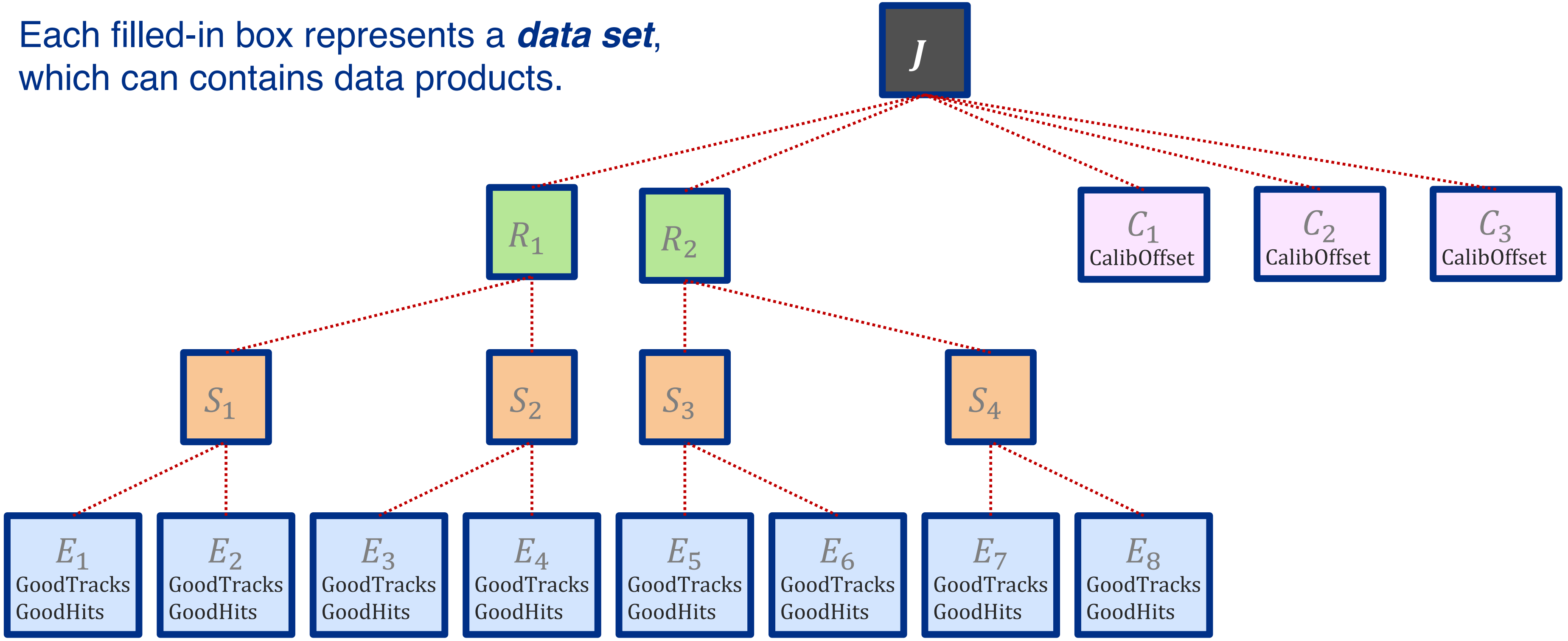
The data product is what is presented to an execution of an algorithm.



Definitions (in pictures)

From services discussion on April 26

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



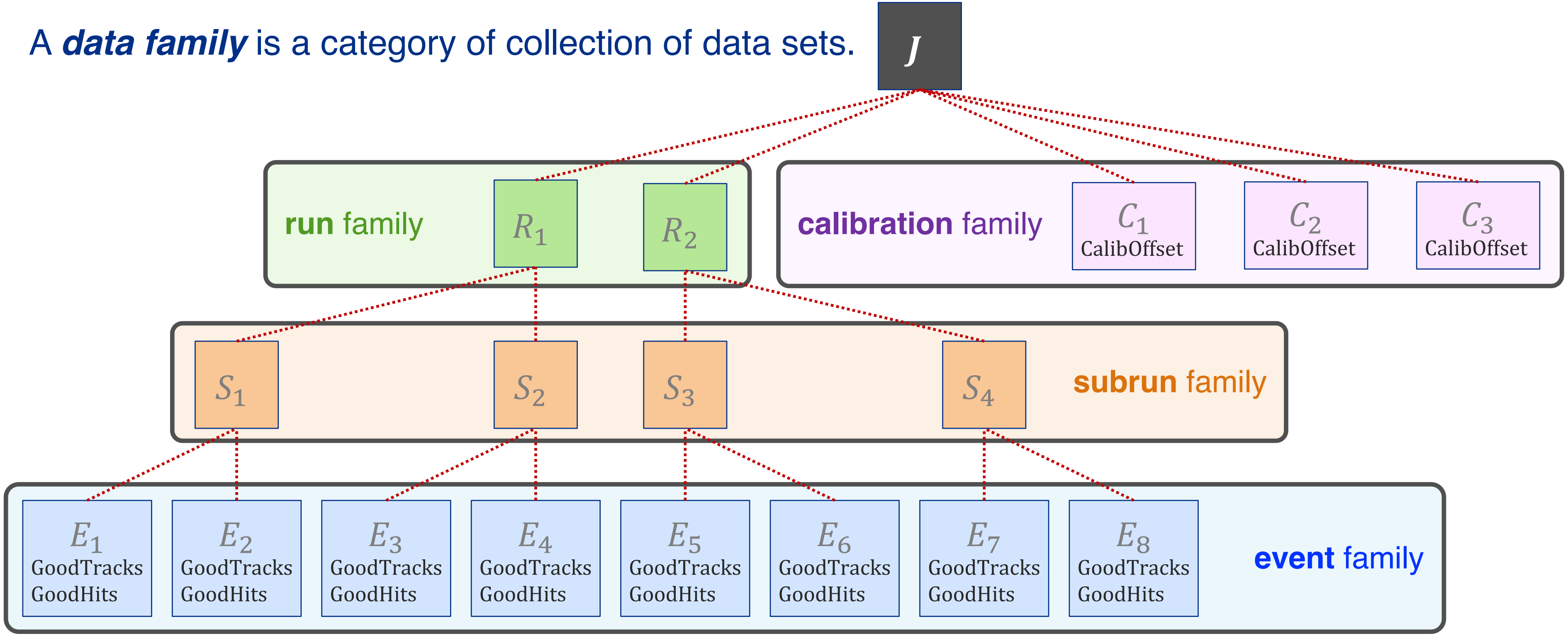
J
R_1
R_2
C_1 CalibOffset
C_2 CalibOffset
C_3 CalibOffset
S_1
S_2
S_3
S_4
E_1 GoodTracks GoodHits
E_2 GoodTracks GoodHits
E_3 GoodTracks GoodHits
E_4 GoodTracks GoodHits
E_5 GoodTracks GoodHits
E_6 GoodTracks GoodHits
E_7 GoodTracks GoodHits
E_8 GoodTracks GoodHits

Fermilab

Definitions (in pictures)

From services discussion on April 26

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



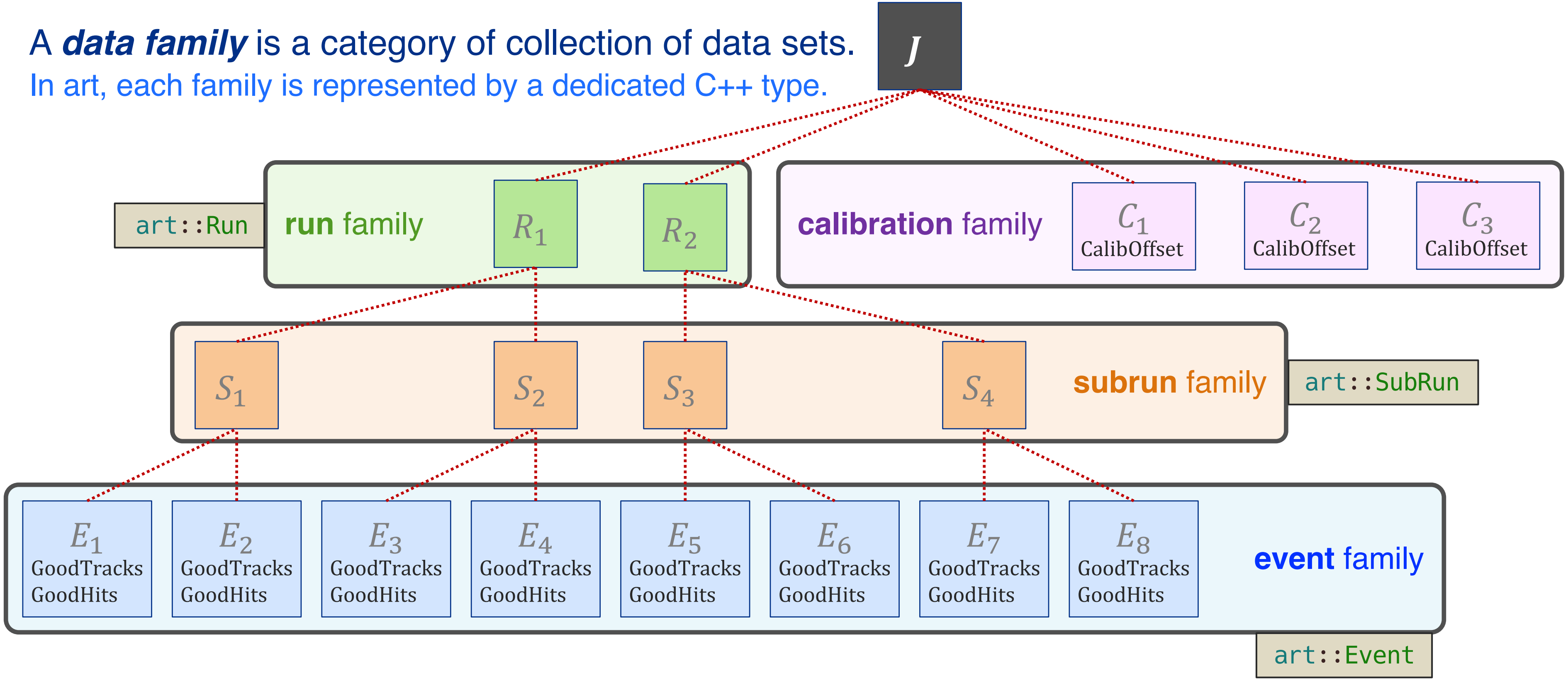
J
run family
R_1
R_2
calibration family
C_1 CalibOffset
C_2 CalibOffset
C_3 CalibOffset
S_1
S_2
S_3
S_4
subrun family
E_1 GoodTracks GoodHits
E_2 GoodTracks GoodHits
E_3 GoodTracks GoodHits
E_4 GoodTracks GoodHits
E_5 GoodTracks GoodHits
E_6 GoodTracks GoodHits
E_7 GoodTracks GoodHits
E_8 GoodTracks GoodHits
event family

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.

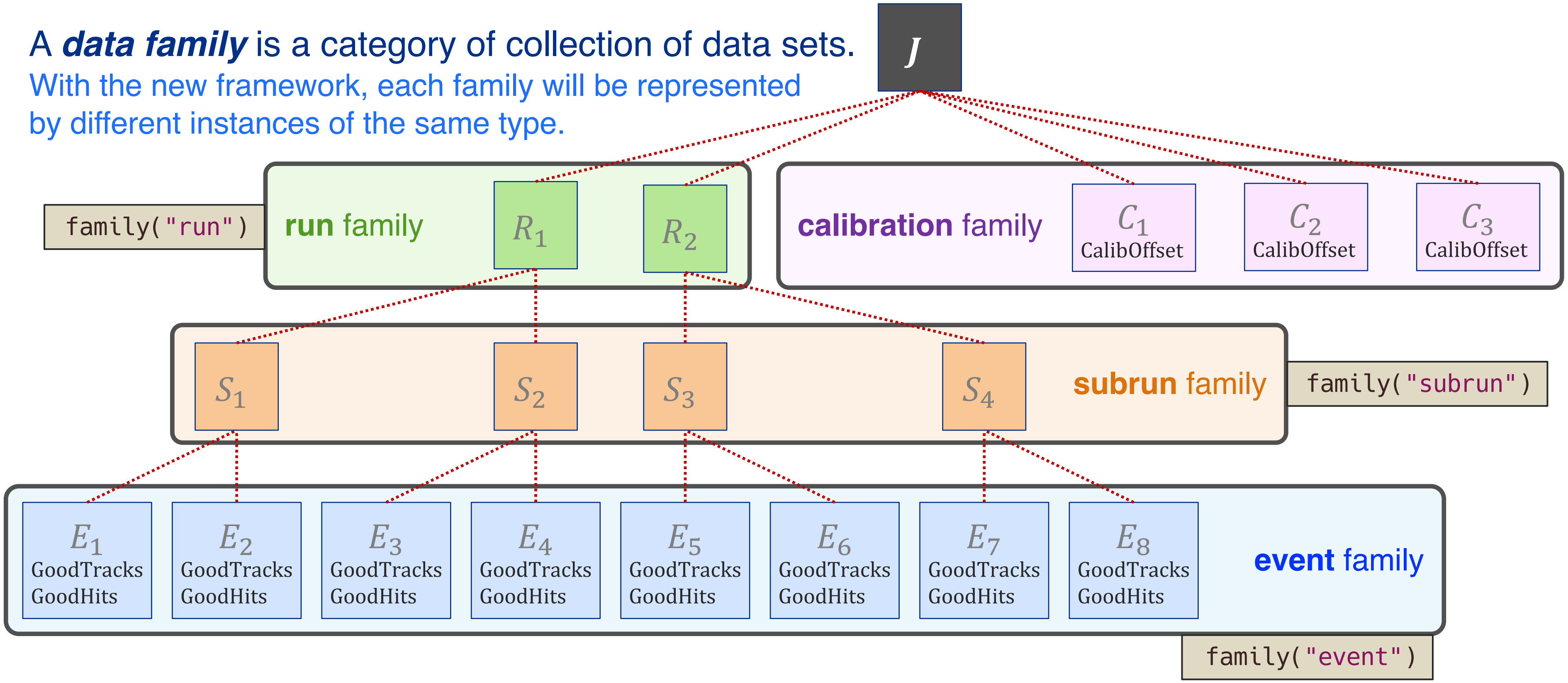


J
art::Run
run family
R_1
R_2
calibration family
C_1 CalibOffset
C_2 CalibOffset
C_3 CalibOffset
S_1
S_2
S_3
S_4
subrun family
art::SubRun
E_1 GoodTracks GoodHits
E_2 GoodTracks GoodHits
E_3 GoodTracks GoodHits
E_4 GoodTracks GoodHits
E_5 GoodTracks GoodHits
E_6 GoodTracks GoodHits
E_7 GoodTracks GoodHits
E_8 GoodTracks GoodHits
event family
art::Event

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.

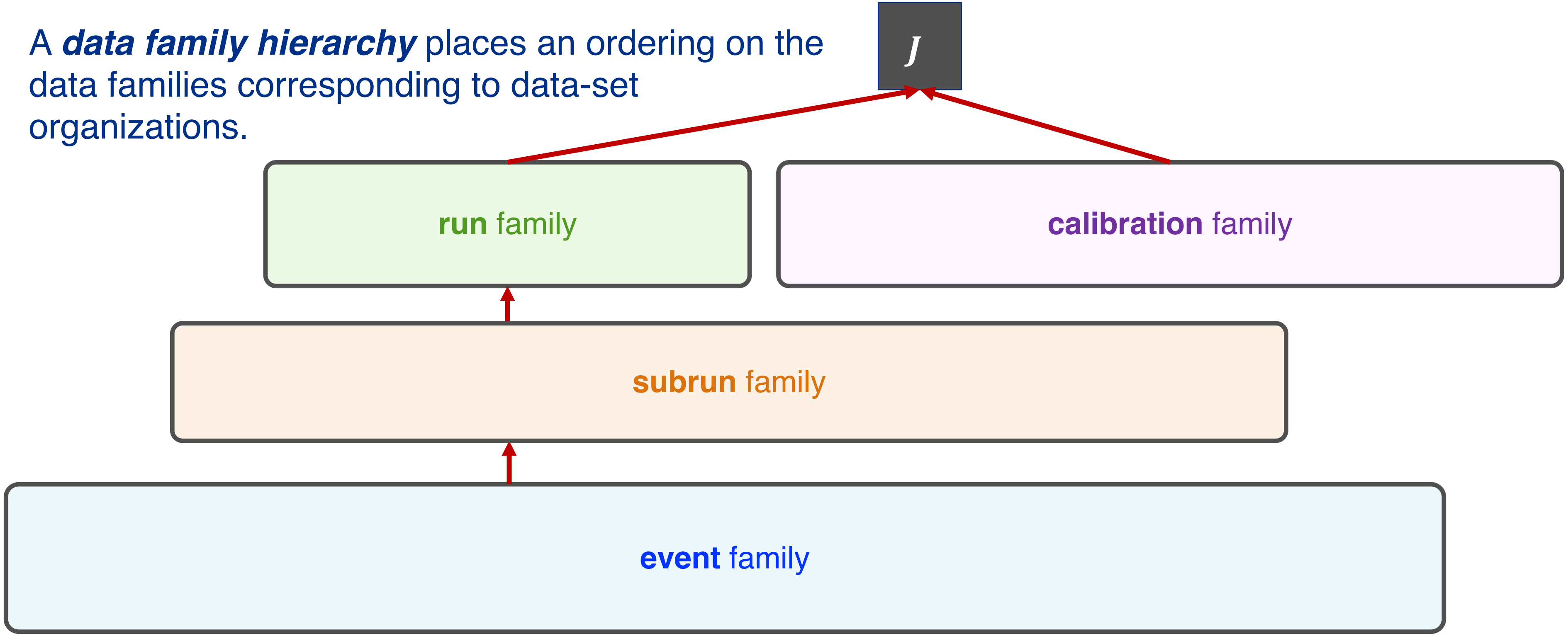


J
family("run")
run family
R_1
R_2
calibration family
C_1 CalibOffset
C_2 CalibOffset
C_3 CalibOffset
S_1
S_2
S_3
S_4
subrun family
family("subrun")
E_1 GoodTracks GoodHits
E_2 GoodTracks GoodHits
E_3 GoodTracks GoodHits
E_4 GoodTracks GoodHits
E_5 GoodTracks GoodHits
E_6 GoodTracks GoodHits
E_7 GoodTracks GoodHits
E_8 GoodTracks GoodHits
event family
family("event")

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.



J
run family
calibration family
subrun family
event family