Gaudi Event Data Model Integration

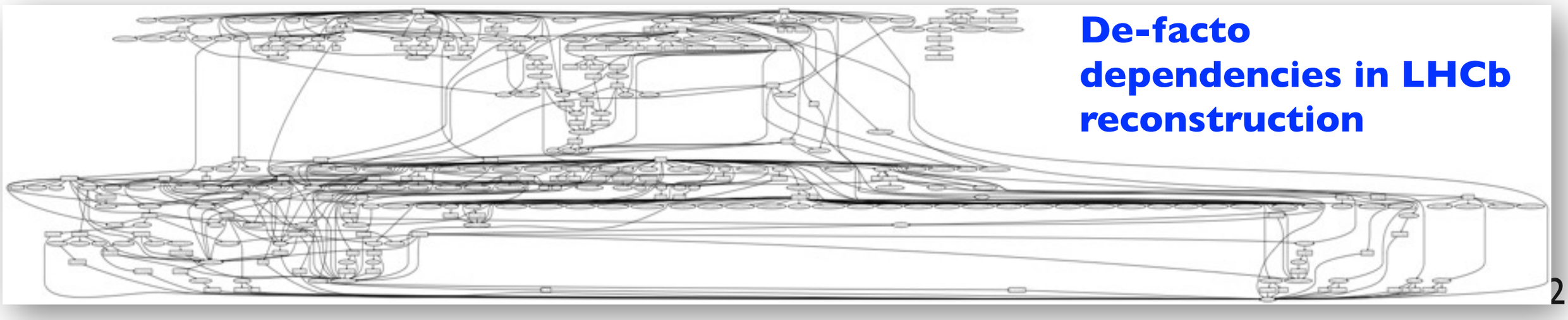
Benedikt Hegner

FCC SW Meeting
23.10.2014

Driving considerations

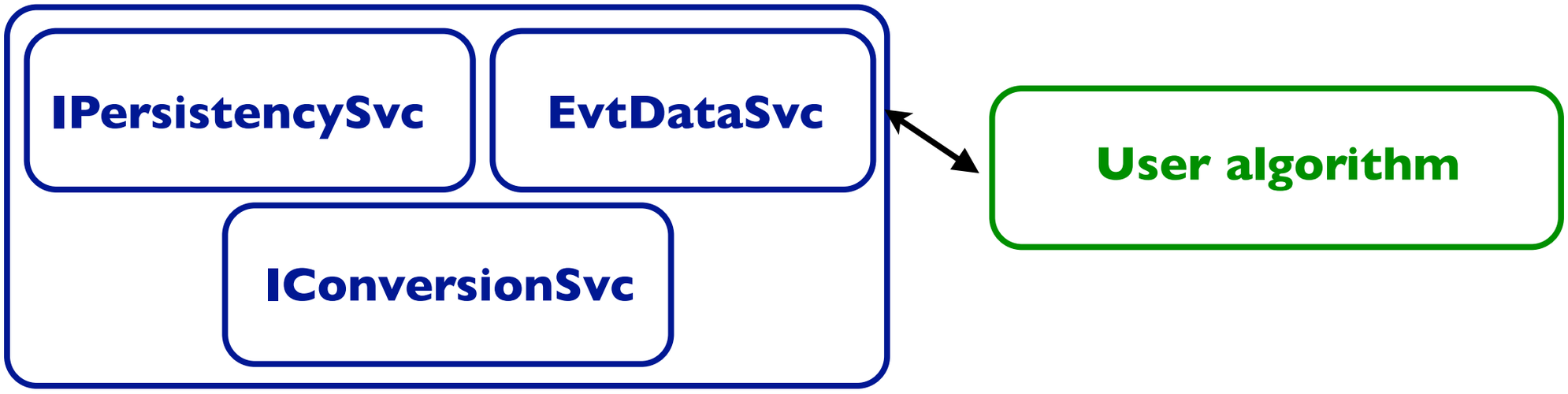
- Integrate new I/O and Data Model as smoothly as possible
  - Avoid big revolutions on Gaudi design (for now ;-))

- Take it as a chance to prepare for concurrency
  - Announce data dependencies of algorithms explicitly
  - Doing too late may be expensive



Gaudi vs. CMS

Gaudi - federation of Services

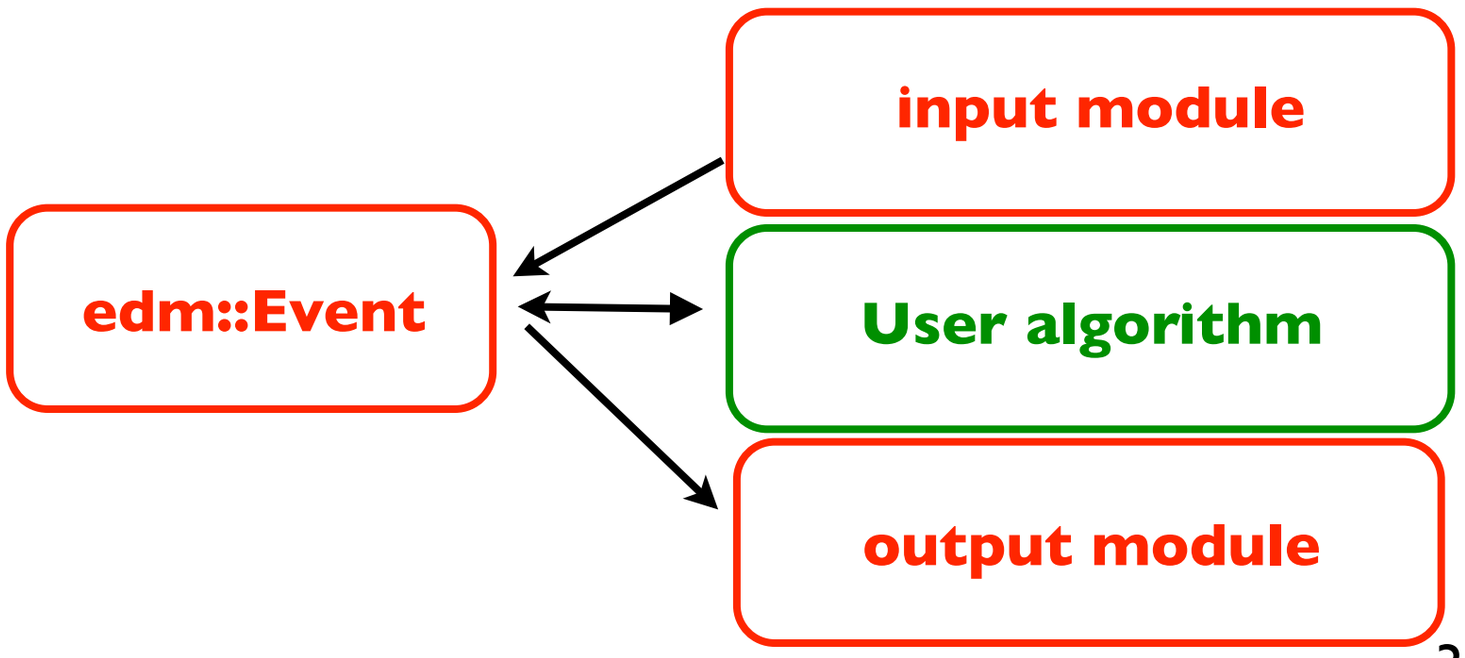


Gaudi flexible but entry level rather high

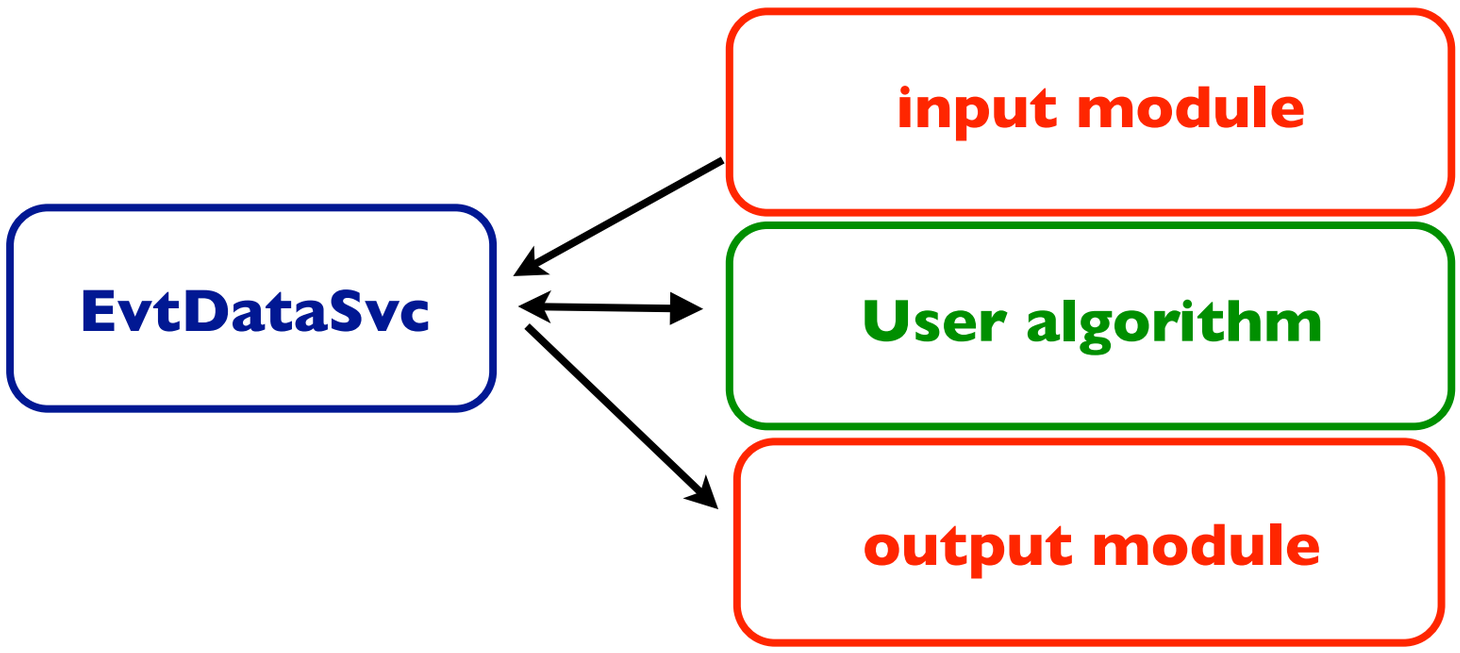
CMS exposes some details to the user

We will try the CMS way...
... but keep the EvtDataSvc

CMSSW
Services and Algorithms



Gaudi + CMS mixed



Concrete Steps

- Made albers create files in Gaudi-compatible directory structure
- Made Data Model compatible with `EvtDataSvc` of Gaudi
  - Let all `Collections` inherit from `DataObject`
  - Does not spoil the benefits of the new library!
  - Extended `EvtDataSvc` to do the required extra-bookkeeping
  - Created InputAlgorithm
  - Open item is creating the OutputAlgorithm
- This allows the new I/O for persistency + all other classes for transient access

Announcing dependencies

.h file

includes

```
#include "DataObjects/InputType.h"
#include "DataObjects/OutputType.h"
```

include type headers

private members

```
DataObjectHandle<InputType> m_inputHandle;
DataObjectHandle<OutputType> m_outputHandle;
```

EvtDataSvc

```
“in”   : InputType
“out”  : OutputType
```

.cc file

Constructor

```
declareOutput("anOutput", m_outputHandle,”in”);
declareInput("anInput", m_inputHandle,”out”);
```

Initialize handles

::execute

```
OutputType* out = new OutputType();
m_outputHandle.put(out);

InputType* in = m_inputHandle.get();
```

put data

retrieve data