The CTADIRAC Production System : recent developments

A. Faure, L. Arrabito

LUPM, CNRS-IN2P3, France

F2F DIRAC, Montpellier, April 5th

I. The Production System and its user interface

DIRAC systems for workflow management

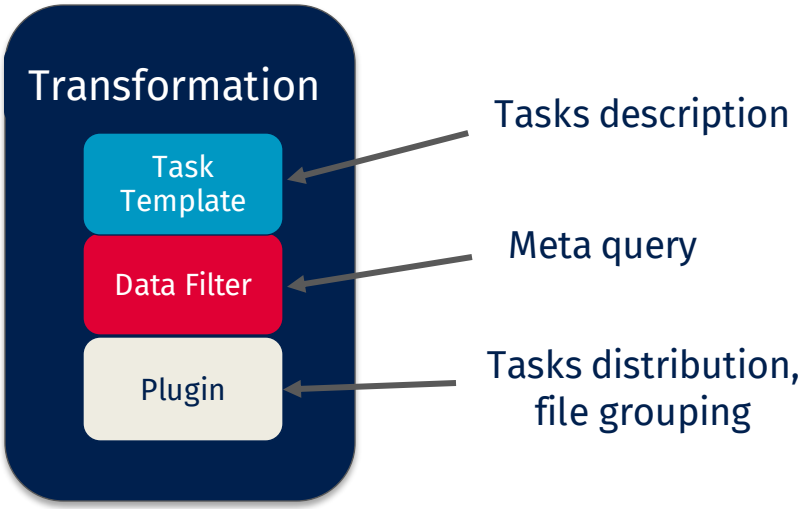
Workflow Management is based on the DIRAC **Transformation and Production systems.**

A **workflow** is a series of transformations operated on data.

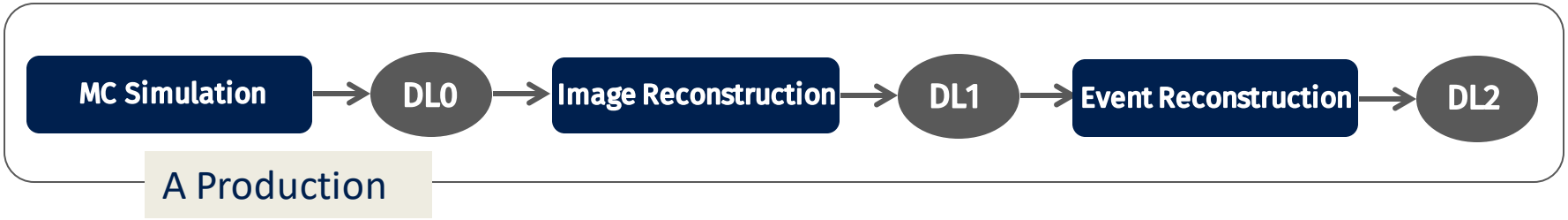
Example of a simple workflow used in CTAO

The Transformation System

A transformation is identified by : a **task**, a **query on metadata** to select input/output data, and **rules** to distribute tasks and group input files.



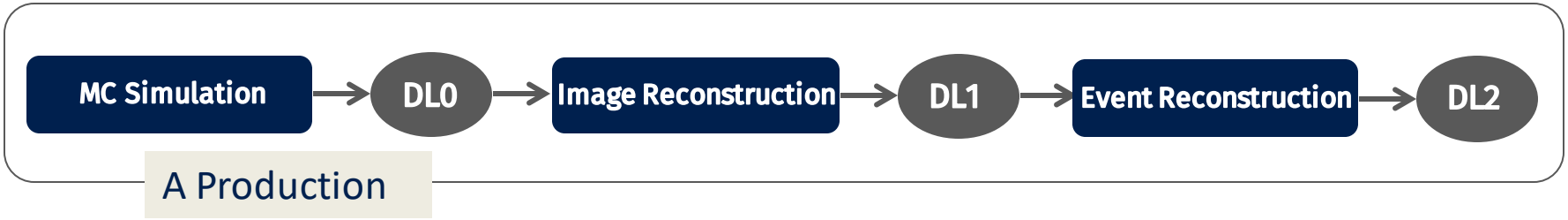
The Production System



It is a high-level system built on top of the Transformation System.

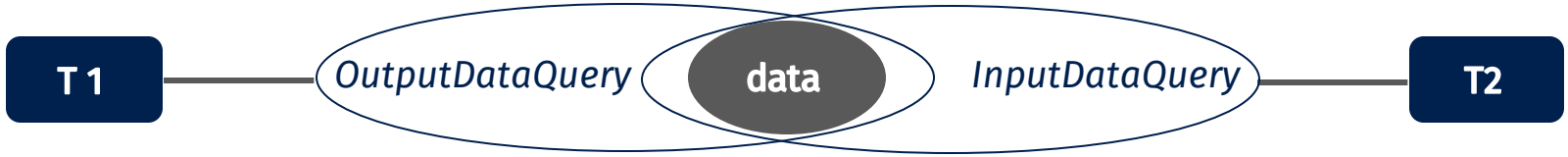
It is used to automatically instantiate the different transformations that compose a workflow.

The Production System



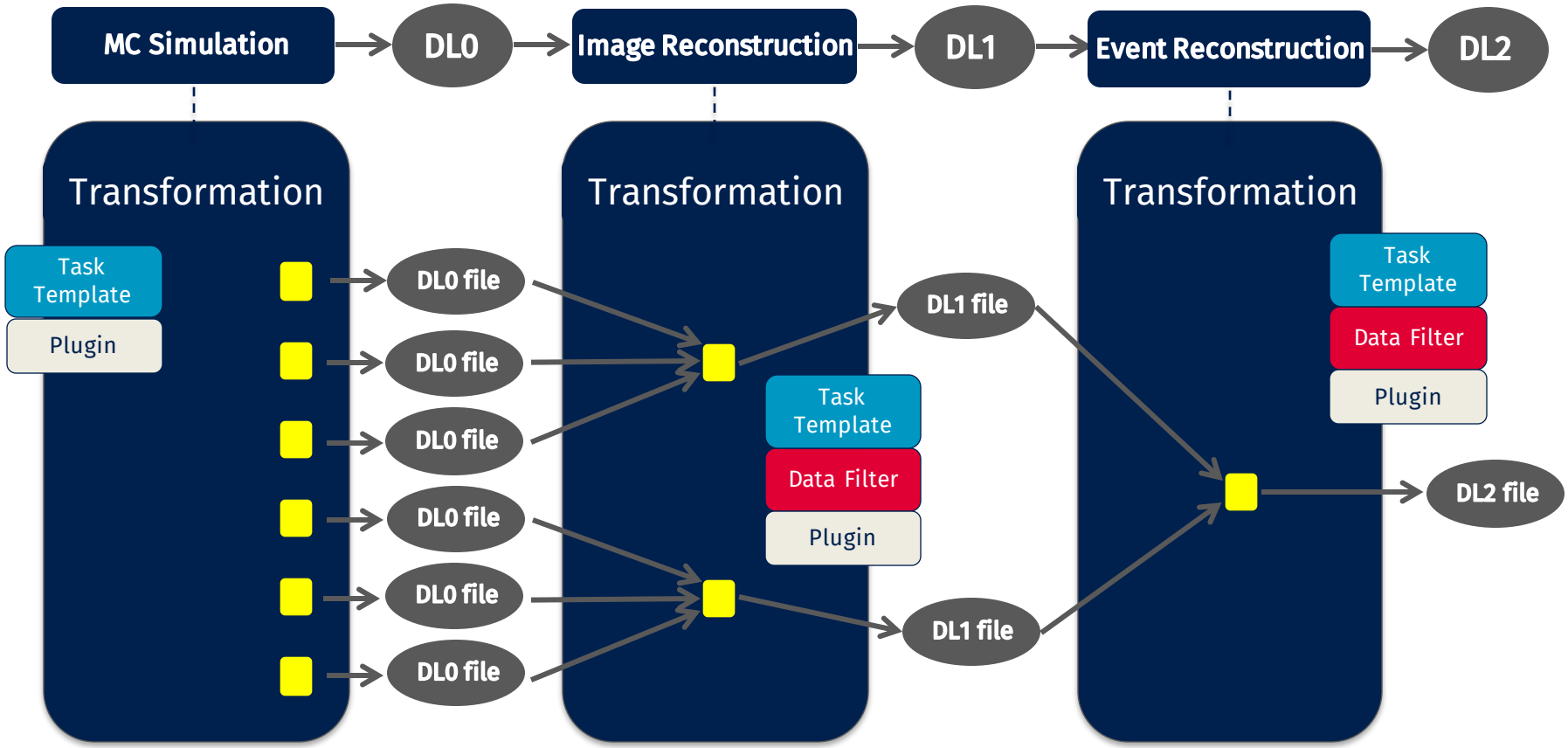
It is a high-level system built on top of the Transformation System.

It is used to automatically instantiate the different transformations that compose a workflow.



Two transformations are connected if the output data of T1 intersects the input data of T2 and the workflows are **data-driven**.

The Production System



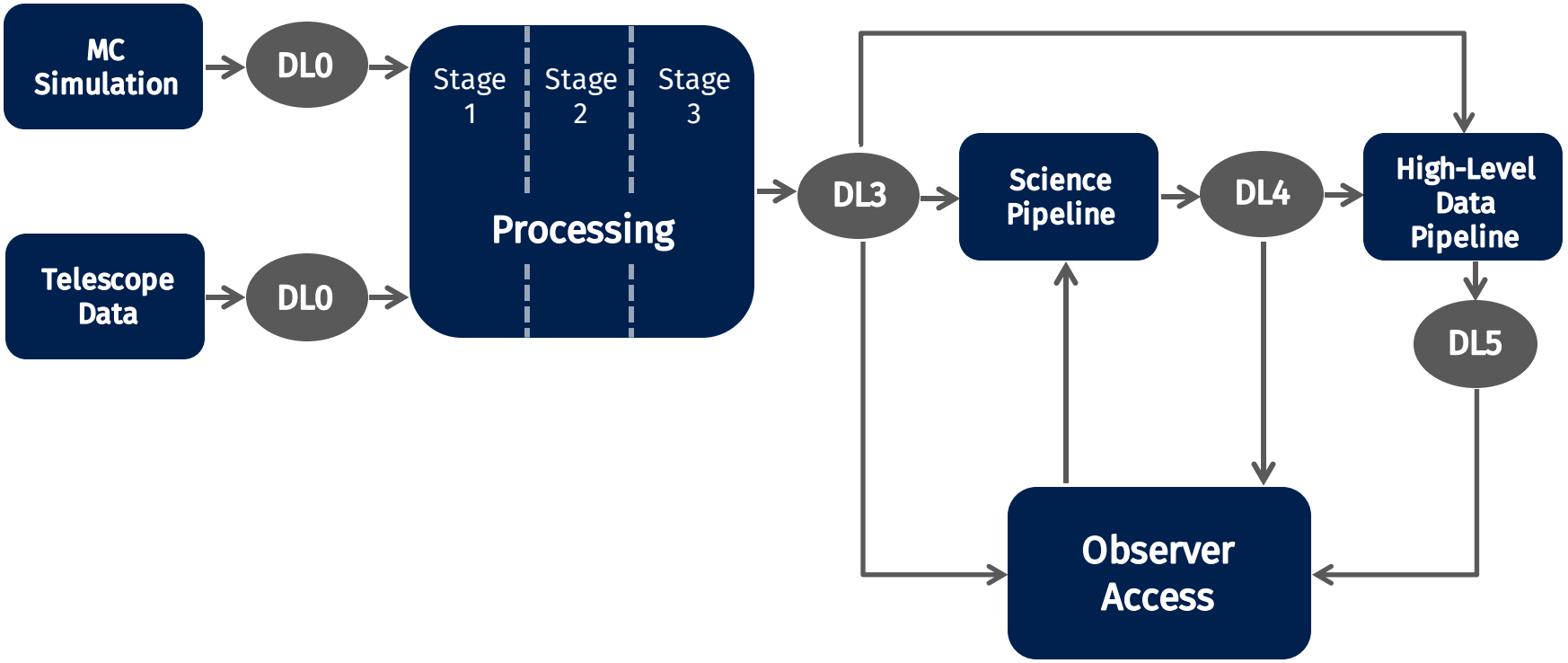
The Production System

We have developed a **high-level interface** for the configuration and submission of productions.

With this interface we are able to submit **complex workflows** more easily.

Example of a complex workflow

The Production System



The Production System Interface

Simple workflow description in a YAML file :

```yaml
ProdSteps:
  - ID: 1
    input_meta_query:
    job_config:
      type: MCSimulation
      version: 2022-08-03
      site: LaPalma
      particle: electron
      pointing_dir: South
      zenith_angle: 20
      n_shower: 100
  - ID: 2
    input_meta_query:
      parentID: 1
    job_config:
      type: CtapipeProcessing
      version: v0.17.0
  - ID: 3
    input_meta_query:
      parentID: 2
    job_config:
      type: Merging
      version: v0.17.0
      group_size: 5

Common:
  MCCampaign: Prod6Test
  configuration_id: 15
  base_path: /vo.cta.in2p3.fr/user/a/afaure/prod6/
```

The Production System Interface

Simple workflow description in a YAML file :

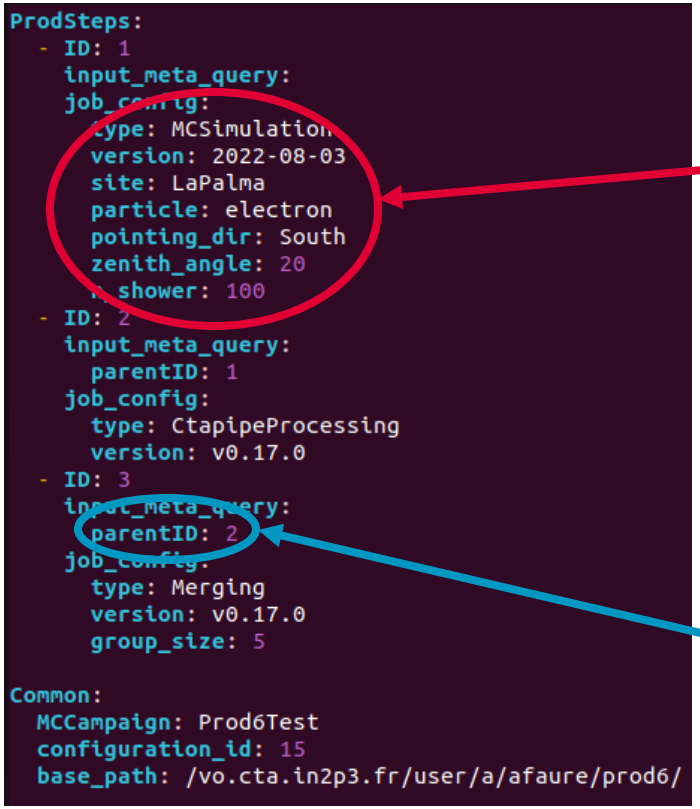
```yaml
ProdSteps:
  - ID: 1
    input_meta_query:
    job_config:
      type: MCSimulation
      version: 2022-08-03
      site: LaPalma
      particle: electron
      pointing_dir: South
      zenith_angle: 20
      n_shower: 100
  - ID: 2
    input_meta_query:
      parentID: 1
    job_config:
      type: CtapipeProcessing
      version: v0.17.0
  - ID: 3
    input_meta_query:
      parentID: 2
    job_config:
      type: Merging
      version: v0.17.0
      group_size: 5

Common:
  MCCampaign: Prod6Test
  configuration_id: 15
  base_path: /vo.cta.in2p3.fr/user/a/afaure/prod6/
```

Transformation input/output defined by meta-data specification

The Production System Interface

Simple workflow description in a YAML file :

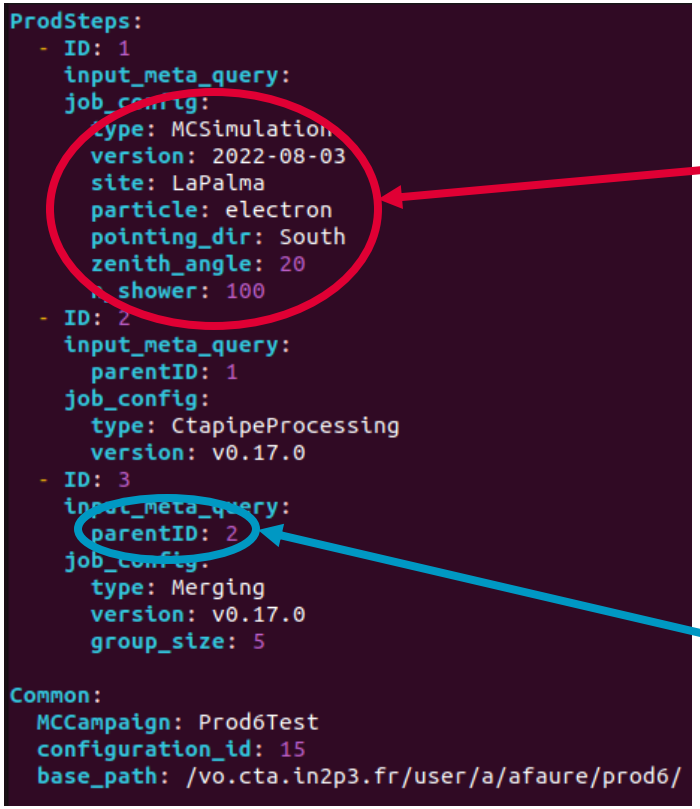


Transformation input/output defined by meta-data specification

Specify parent/child transformation connections eventually adding extra meta-data specifications

The Production System Interface

Simple workflow description in a YAML file :



Transformation input/output defined by meta-data specification

Automatically builds the queries on meta data from the workflow description

Specify parent/child transformation connections eventually adding extra meta-data specifications

The Production System Interface

Simple workflow description in a YAML file :

```yaml
ProdSteps:
  - ID: 1
    input_meta_query:
    job_config:
      type: MCSimulation
      version: 2022-08-03
      site: LaPalma
      particle: electron
      pointing_dir: South
      zenith_angle: 20
      n_shower: 100
  - ID: 2
    input_meta_query:
      parentID: 1
    job_config:
      type: CtapipeProcessing
      version: v0.17.0
  - ID: 3
    input_meta_query:
      parentID: 2
    job_config:
      type: Merging
      version: v0.17.0
      group_size: 5

Common:
  MCCampaign: Prod6Test
  configuration_id: 15
  base_path: /vo.cta.in2p3.fr/user/a/afaure/prod6/
```

- We developed the interface in the CTADIRAC software extension but we plan to port it in vanilla DIRAC
- Each users community can specify its own meta-data

II. CWL support in CTADIRAC

CWL

Standard for describing workflows

Composed of several steps (**command-line tools** / expressions)

A CWL runner (e.g. **cwltool**) **parses** the steps described in **YAML** and **runs** the workflow

CWL & DIRAC

Standard for describing workflows

Composed of several steps (**command-line tools** / expressions)

A CWL runner (e.g. **cwltool**) **parses** the steps described in **YAML** and **runs** the workflow

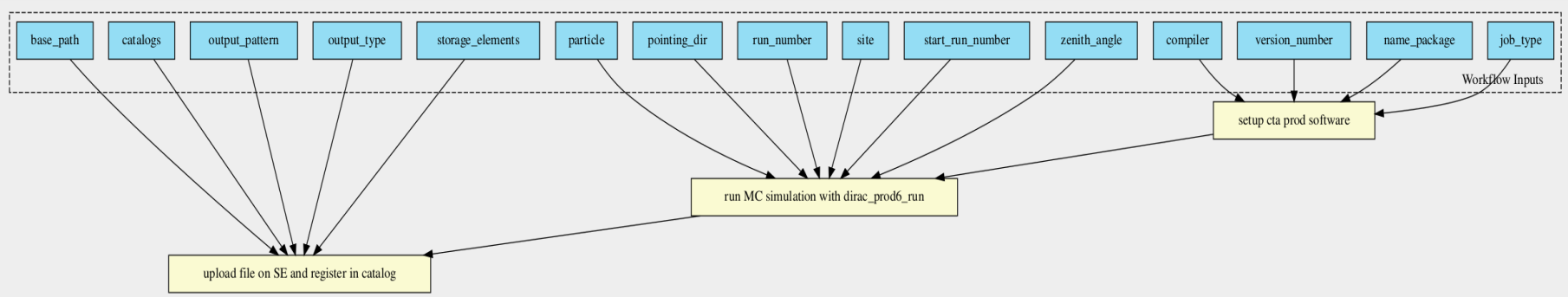
Different workflow types : **Jobs** / **Transformations** / **Productions**

In CTADIRAC we use the **Production System Interface** to parse and run large workflows described in **YAML**

Run on **distributed data centers**

Simulation workflow

- Simple simulation workflow : **setup** the MC simulation software, **run** it, and **store** the data produced in the DIRAC File Catalog.

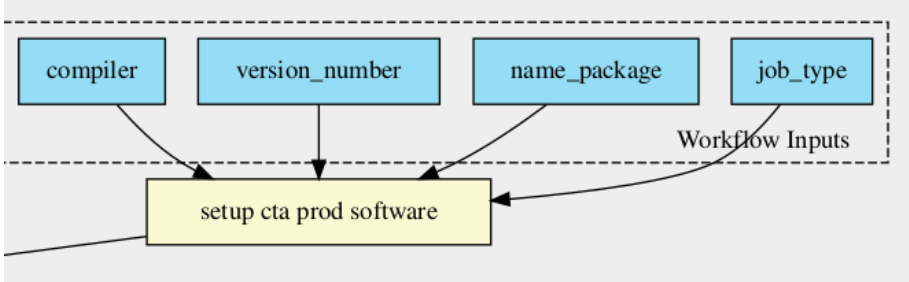


- Goal : be able to run the same workflows described in CWL locally with cwltool and on the distributed data centers with DIRAC.

- Strategy : describe each **step** in CWL and write a **parser** that converts these steps into steps of a DIRAC Transformation.

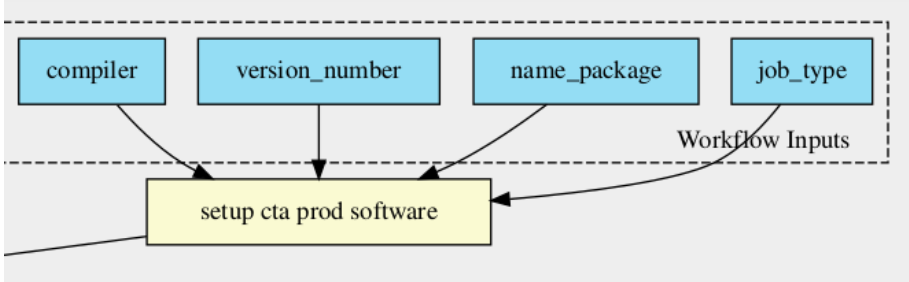
Simulation workflow

- Example with the first step :



Simulation workflow

- Example with the first step :

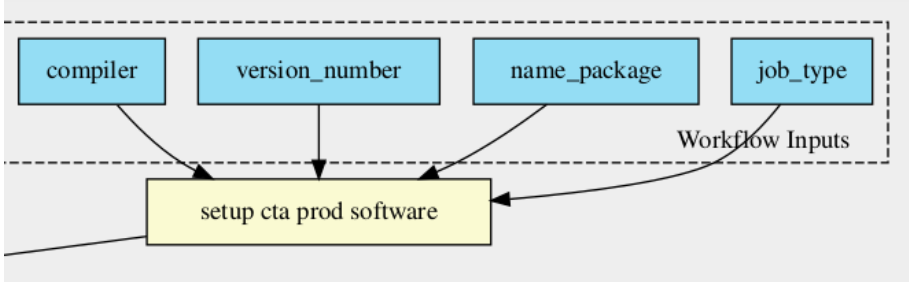


which corresponds to the command :

```
cta-prod-setup-software -p package -v version -a job_type -g compiler
```

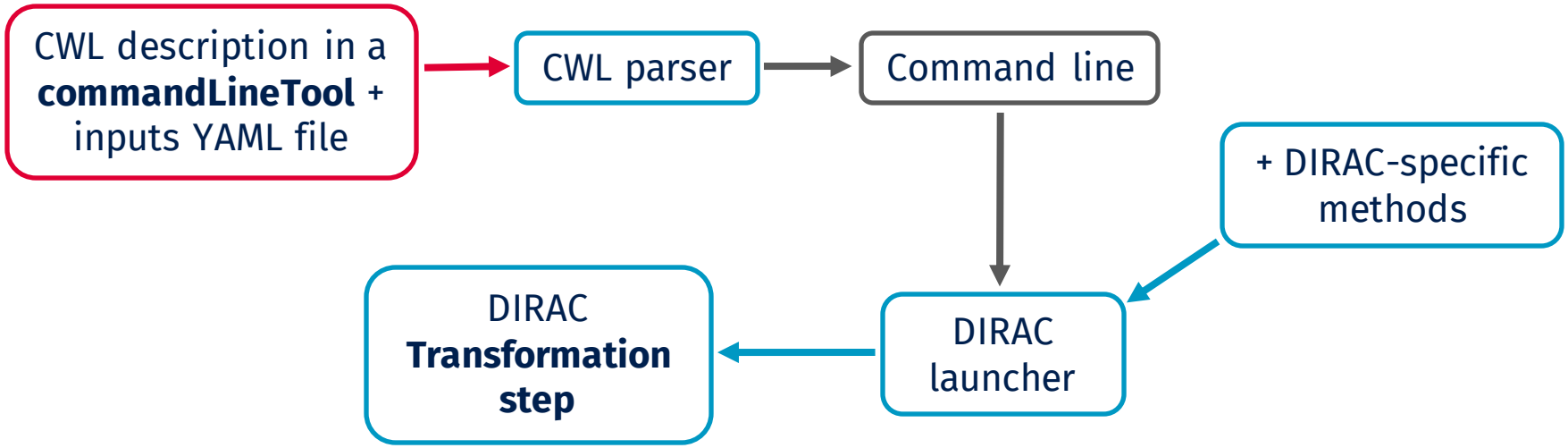
Simulation workflow

- Example with the first step :



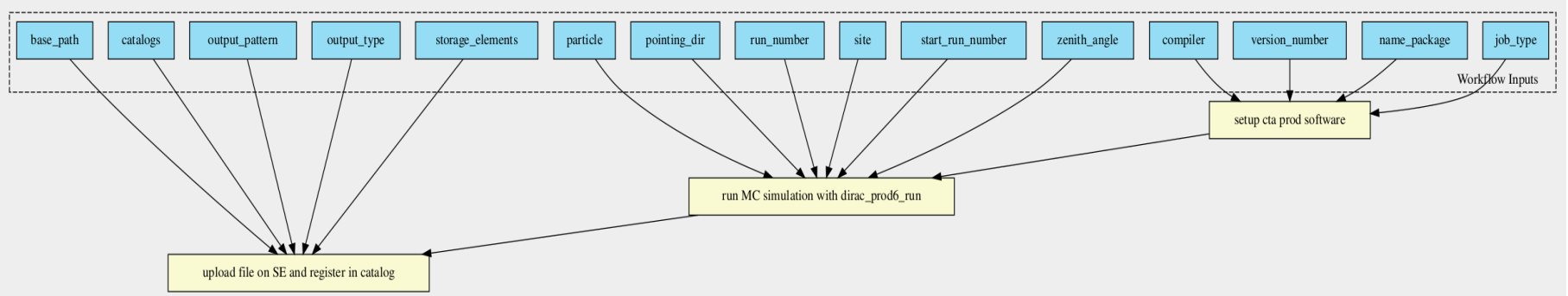
which corresponds to the command :

```
cta-prod-setup-software -p package -v version -a job_type -g compiler
```



Simulation workflow

- For a complete workflow, the steps are described independently (**atomized workflow**):



3 CWL **commandLineTool** descriptions + 3 inputs YAML file

CWL parser

DIRAC launcher

A 3-steps DIRAC Transformation run on the distributed data centers

CWL-to-DIRAC scripts

- CWL Parser :
  - Minimalist : for **command line tools** only
  - Inspired by **cwl2script**
  - Uses **cwltool** functions to parse CWL command line tools and to get the corresponding command line

- Launcher script :
  - Adds DIRAC methods between the command lines to run the steps on the distributed data centers :
  - Sets the job type, the input & output sandbox of each step
  - Translates the command lines to DIRAC executables
  - Submits the Transformation

Conclusion

- We have developed an **interface** to the Production System to easily configure and submit complex workflows.
- We are able to run workflows described in CWL with CTADIRAC using DIRAC Transformations.
- This is limited to **atomized workflows** described in CWL where each step is the execution of a command line.
- This will enable pipelines users using CWL to run their workflows both locally and on the distributed data centers with DIRAC.
- We have to investigate how to use input and output meta queries with CWL in order to launch a **Production** made of several Transformations.

Thank you for your attention