Provenance Data in the Context of Earth System Modelling

Kerstin Fieg, Luis Kornblueh, Deike Kleberg, Pavan Siligam

DKRZ & MPI-M

1. Difference Metadata – Provenance Data
2. Why Provenance Data?
3. How to collect Provenance Data
4. Requirements => Luis‘ talk

DKRZ

Max-Planck-Institut für Meteorologie

1. Metadaten vs. Provenance Data

Metadat

- … are struct ata products
- … describing th itself
- … becoming increas t with growing amount of data

Possible future demand f

… extended mach adability tadata

… and / or au c evaluation p res of the metadata.

No data archiving without Metadata

e.g. NetCDF as self describing model metadata format

Provenance Data

“The provenance of a piece of data is ocess that led to that piece of data”, Moreau, 2010

- … describing the origin results
- … is more than an “ on” to Metadata
- … includes proc details (history / used tools / methods / procedure .....)
- … ideally enting the complete development process

No automatic self describing tool available

2. Why Provenance Data ?

- we have to - because of “Gute wissenschaftliche Praxis”

Benefit of Provenance Data – in addition to Metadata

- making Provenance data available
  … can help to evaluate quality of data
  … can document the scientific standard
  … enables error – tracking - *even years after data production and leave of the scientist*

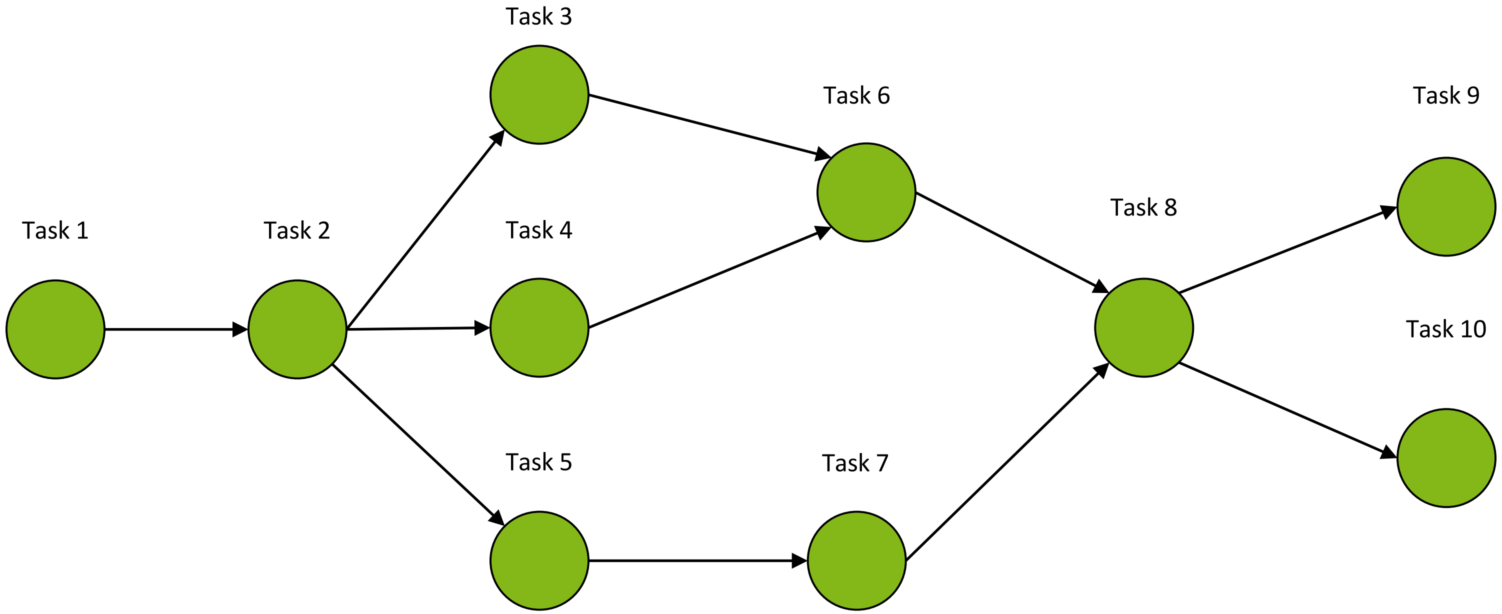
- degree of reliability is documented and traceable (how / who / why / where /what…)
  => *for possible discussions in future facts are available*!

- Provenance Data
  - … enable reusebility of data *for a longer time*
  - … enable runtime recovery
    - ✓ repeat / reproduce model runs

3. How to collect Provenance data?

We have a defined model **workflow**

… which describes a chain of chronological and logical connected data steps or processing steps - or both in combination



To perform the workflow, we need a model **framework**

… which is a transparent runtime environment, that drives the workflow and supports it during the entire lifetime of a model run or project

Complete Workflow in climate research context (= Reinhards graph)

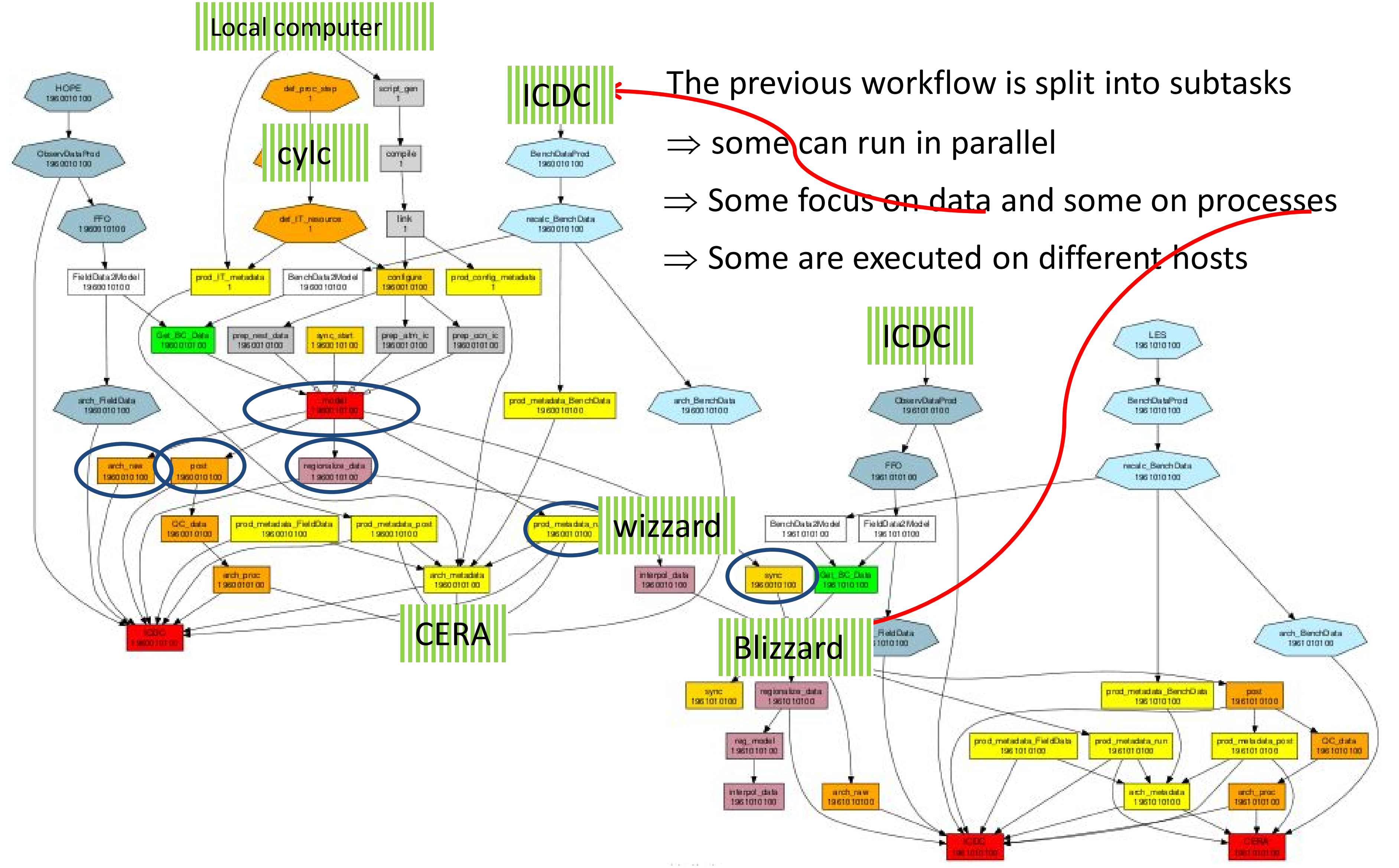
1. Prepare model (dynamic, physics, optimization)
2. Set up components (site, machine, compiler)
3. Define experiment
4. Build model
5. Compute host directory structure and input file setup
6. Restart and time dependent input data preparation
7. Model run step
8. Restart and log handling
9. Archiving of raw data and restart
10. Post-processing
11. In-house quality control
12. Archiving of derived data products
13. Data publishing process (CIM, ESGF data publication)
14. ESGF (data and index nodes)
15. Data search and data access interface
16. Community analysis

What we try to cover (processing & data)

1. Prepare model (dynamic, physics, optimization)
2. Set up components (site, machine, compiler, libs)
3. Define experiment
4. Build model
5. Compute host directory structure and input file setup
6. Restart and time dependent input data preparation
7. Model run step
8. Restart and log handling
9. Archiving of raw data and restart
10. Post-processing
11. In-house quality control
12. Archiving of derived data products
13. Data publishing process (CIM, ESGF data publication)
14. ESGF (data and index nodes)
15. Data search and data access interface
16. Community analysis

=> No metadata, but a protocol of **all actions** leading to the final data is collected and stored

“all actions”: e.g. the HD(CP)2 workflow



The previous workflow is split into subtasks

⇒ some can run in parallel

⇒ Some focus on data and some on processes

⇒ Some are executed on different hosts

Wishlist for a tool , that can handle it all

General technical requirements:

- allow modularity / flexibility / portability
- allow user interaction / experiment configuration
- enable program portability
- allow general reproducibility of a task / result
- bundle / integrate
  - heterogeneous, distributed services / software tools
  - heterogeneous and distributed data management
- collect provenance data and / or metadata

Interface of data and process workflow

Our **frameworks** tool should …

- schedule the individual subtasks of the process chain
- be platforms independent
- enable monitoring of processes
- support testing and quality checking (QC)
- ease failure handling
- enable restart / controlled repetition of an experiment
- deliver / produce provenance data(= “Gute wissenschaftliche Praxis …”)

What are we using?

⇒ **We selected CYLC as our Frameworks tool**

What is CYLC?

- Cylc ("*silk*") is a **suite engine** and **uler**, designed to manage suites of cycling tasks in weather and casting
- Using Python => platfor ent
- Enables running of ies and / or processes in parallel
- Developed by IWA, NZ) under GNU Licence

H. Oliver will be in Hamburg for the
ISENES2 Workshop on workflows, June 3 – 5 2014

CYLC

⇒ … will be our new runtime environment for climate research (e.g. for projects like MIKLIP & HD(CP)2)

⇒ … furthermore, will manage the collection of the Provenance Data

⇒ Data will be collected in a separate Provenance Data Base

DKRZ

Max-Planck-Institut für Meteorologie

Concept of a data base schema for provenance data by Deike Kleberg (MPI-M)

Target:

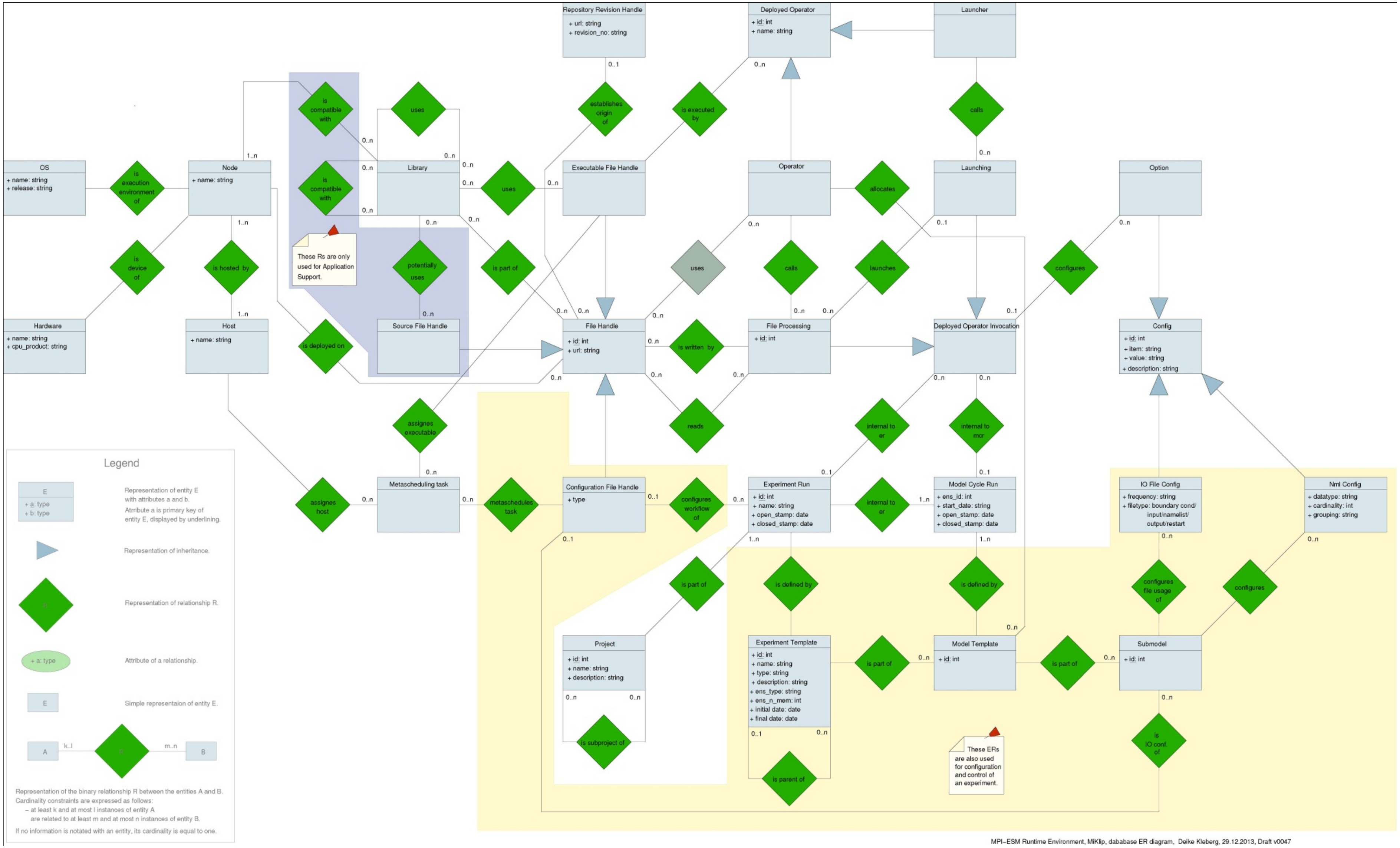
Develop a data base scheme to store all kind of Provenance data

- in an abstract and simple way
- flexible and easy expandable for possible future requirements

DKRZ

Max-Planck-Institut für Meteorologie

Concept of a data base schema for provenance data by Deike Kleberg (MPI-M)



Repository Revision Handle
+ url: string
+ revision_no: string

Deployed Operator
+ id: int
+ name: string

Launcher

OS
+ name: string
+ release: string

Node
+ name: string

Library

Executable File Handle

Operator

Launching

Option

Hardware
+ name: string
+ cpu_product: string

Host
+ name: string

Source File Handle

File Handle
+ id: int
+ url: string

File Processing
+ id: int

Deployed Operator Invocation

Config
+ id: int
+ item: string
+ value: string
+ description: string

Metascheduling task

Configuration File Handle
+ type

Experiment Run
+ id: int
+ name: string
+ open_stamp: date
+ closed_stamp: date

Model Cycle Run
+ ens_id: int
+ start_date: string
+ open_stamp: date
+ closed_stamp: date

IO File Config
+ frequency: string
+ filetype: boundary cond/ input/namelist/ output/restart

Nml Config
+ datatype: string
+ cardinality: int
+ grouping: string

Project
+ id: int
+ name: string
+ description: string

Experiment Template
+ id: int
+ name: string
+ type: string
+ description: string
+ ens_type: string
+ ens_n_mem: int
+ initial date: date
+ final date: date

Model Template
+ id: int

Submodel
+ id: int

These Rs are only used for Application Support.

These ERs are also used for configuration and control of an experiment.

Legend

Representation of entity E with attributes a and b. Atrribute a is primary key of entity E, displayed by underlining.

Representation of inheritance.

Representation of relationship R.

Attribute of a relationship.

Simple representaion of entity E.

Representation of the binary relationship R between the entities A and B. Cardinality constraints are expressed as follows:
- at least k and at most l instances of entity A are related to at least m and at most n instances of entity B.

If no information is notated with an entity, its cardinality is equal to one.

MPI–ESM Runtime Environment, MiKlip, dababase ER diagram, Deike Kleberg, 29.12.2013, Draft v0047