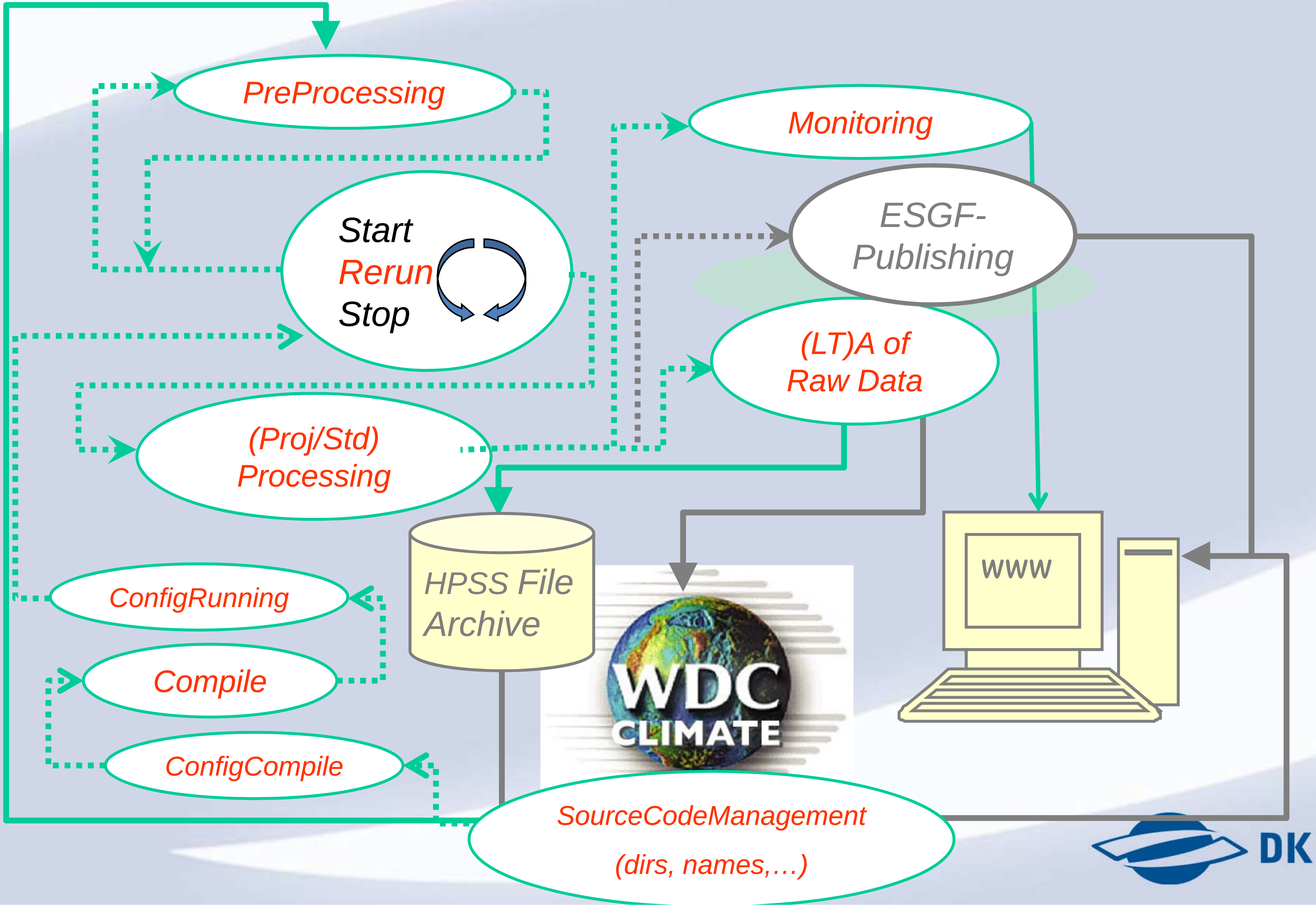
Automated Meta Data Harvesting in the IMDI workflow

Stephanie Legutke, DKRZ/DM

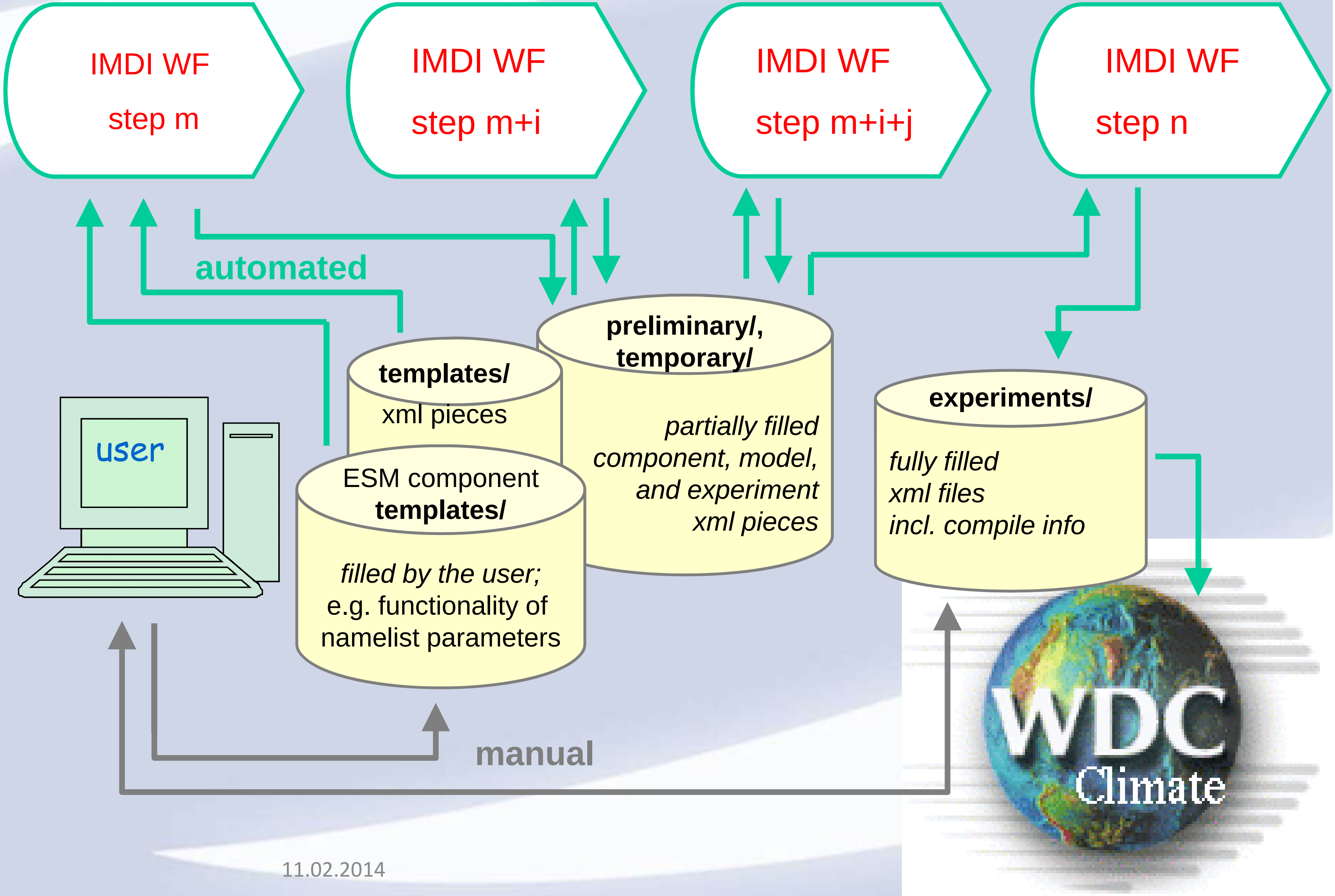
21 January 2014, Hamburg

IMDI = Integrating Model and Data Infrastructure



DKRZ

IMDI MD harvesting: high level design (I)



IMDI WF step m

IMDI WF step m+i

IMDI WF step m+i+j

IMDI WF step n

automated

templates/

xml pieces

preliminary/, temporary/

partially filled component, model, and experiment xml pieces

experiments/

fully filled xml files incl. compile info

user

ESM component **templates/**

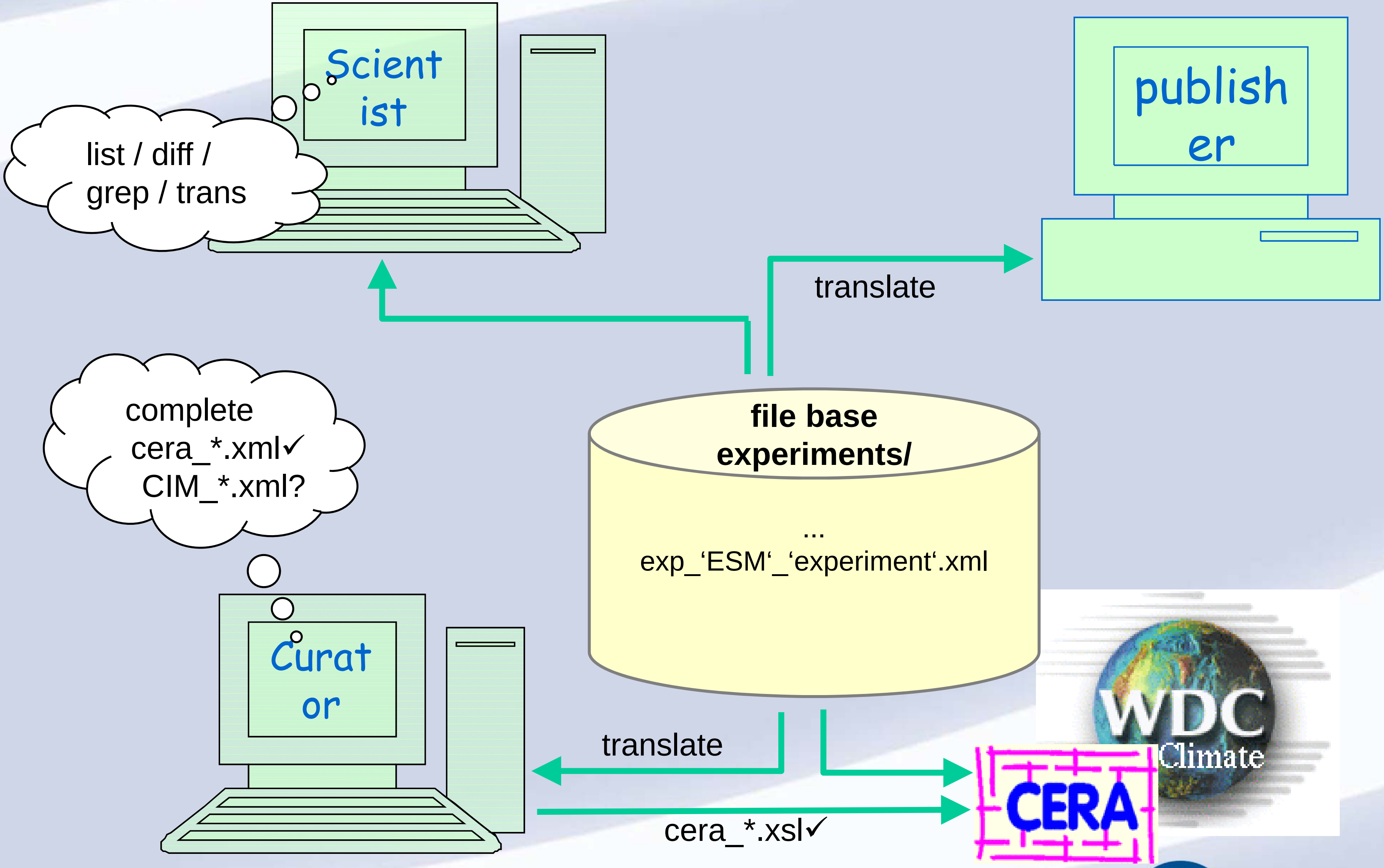
filled by the user; e.g. functionality of namelist parameters

manual

IMDI MD harvesting: high level design (II)

- Create xml formatted MD files for numerical experiments incl. information from compilation
- Fill automatically as far as possible
- Consistency checks where needed and possible
- Allow manual filling if wished
- Anticipate translation to other MD model forms
- Allow seamless later extension
- Multi-purpose use

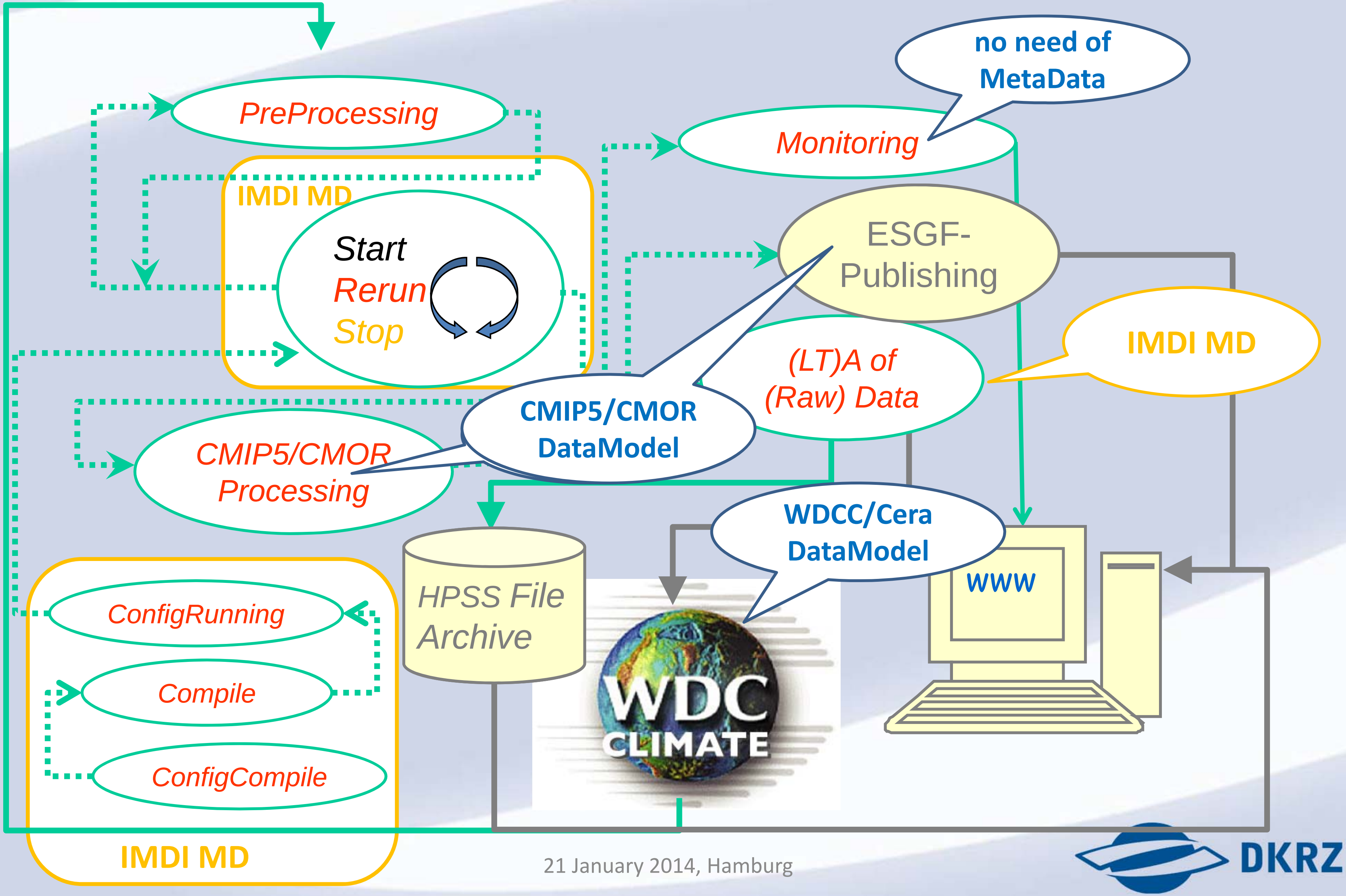
IMDI MD file base usage



Story of MD Generation in IMDI

- Work started in Aug 2007
- Achievements:
  - prototype ready for the COSMOS-ao model of the MPI-Met *(predecessor of CMIP5 MPI-ESM=ECHAM6/JSBACH/MPIOM/HAMOCC*)
  - translation script transform_imdi_to_cera.xsl (Hans-Herman Winter)
- Work stopped in Jul 2009 (SVN history)
  - staff shortage
  - new remits (CMIP5)
  - no way interface with CIM
- positive feedback
- planned resumption for CMIP5 never happened

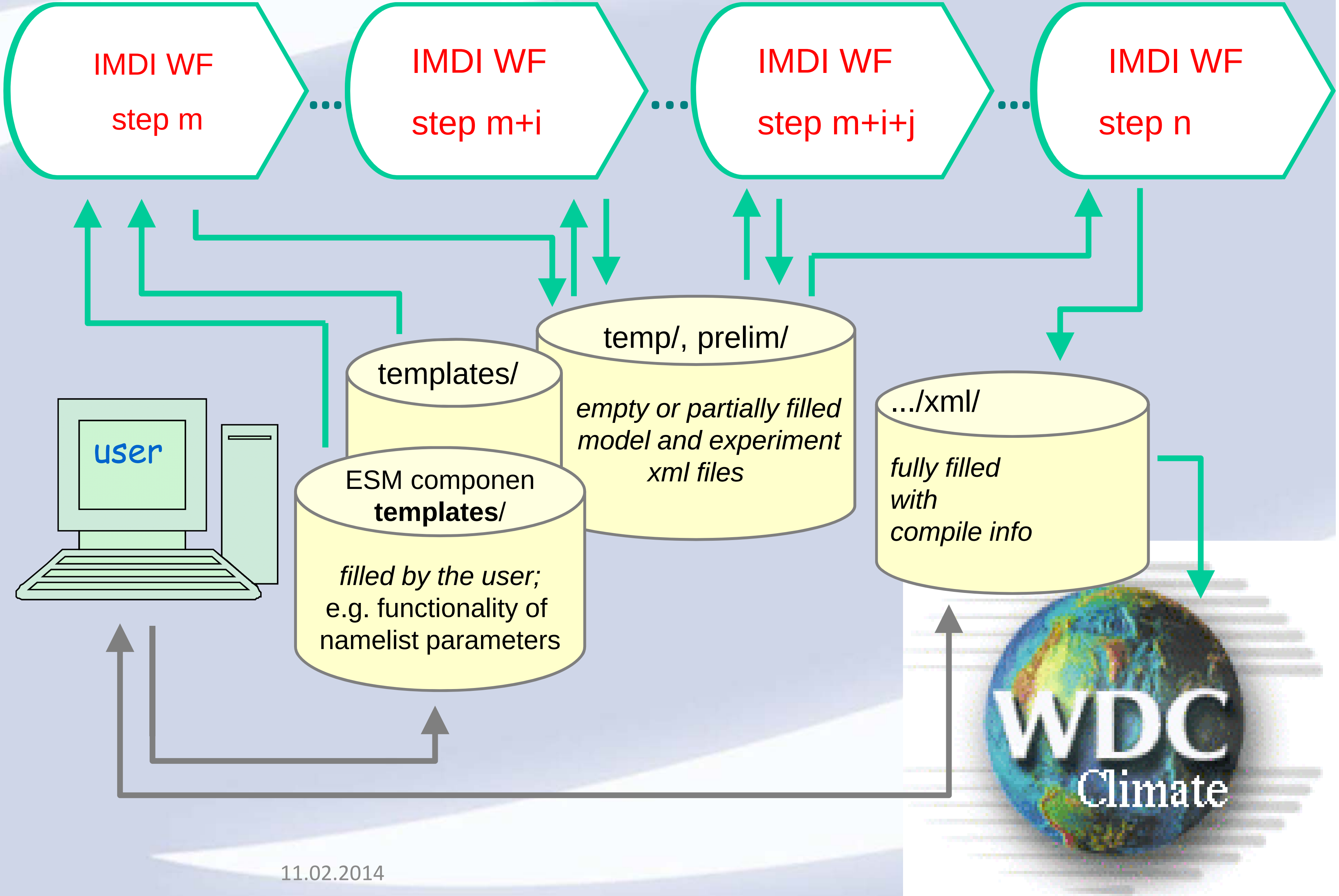
CMIP5 IMDI Meta Data Harvesting: Where ?



21 January 2014, Hamburg

DKRZ

IMDI MD harvesting: high level design (I)



IMDI WF step m

IMDI WF step m+i

IMDI WF step m+i+j

IMDI WF step n

temp/, prelim/

empty or partially filled model and experiment xml files

templates/

ESM componen **templates/**

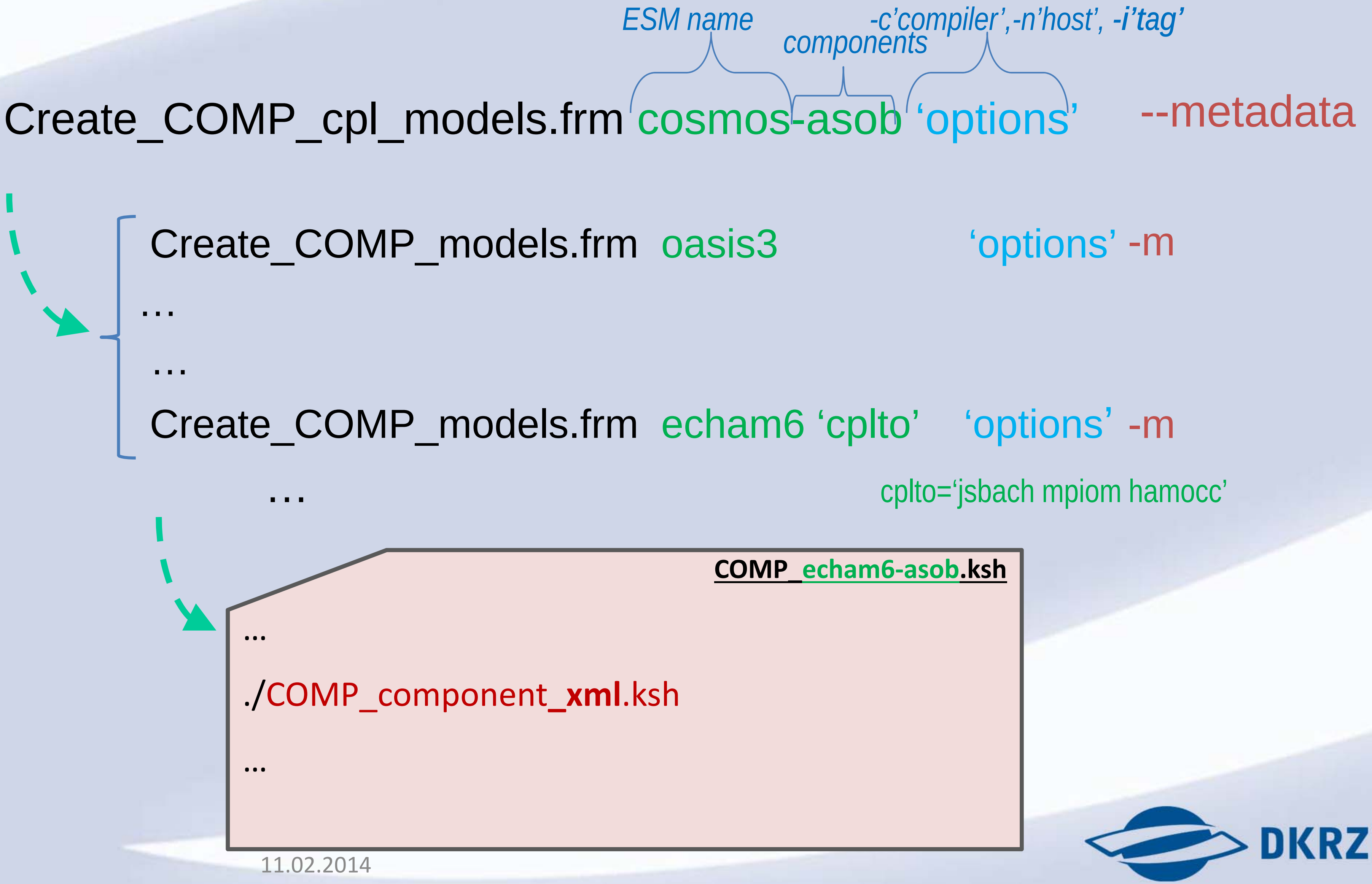
filled by the user; e.g. functionality of namelist parameters

.../xml/

fully filled with compile info

11.02.2014

Configure Compilation for ESM MD generation



ESM name *components* *-c'compiler',-n'host', **-i'tag'***

Create_COMP_cpl_models.frm cosmos-asob 'options' --metadata

Create_COMP_models.frm oasis3 'options' -m

…

…

Create_COMP_models.frm echam6 'cplto' 'options' -m

…

cplto='jsbach mpiom hamocc'

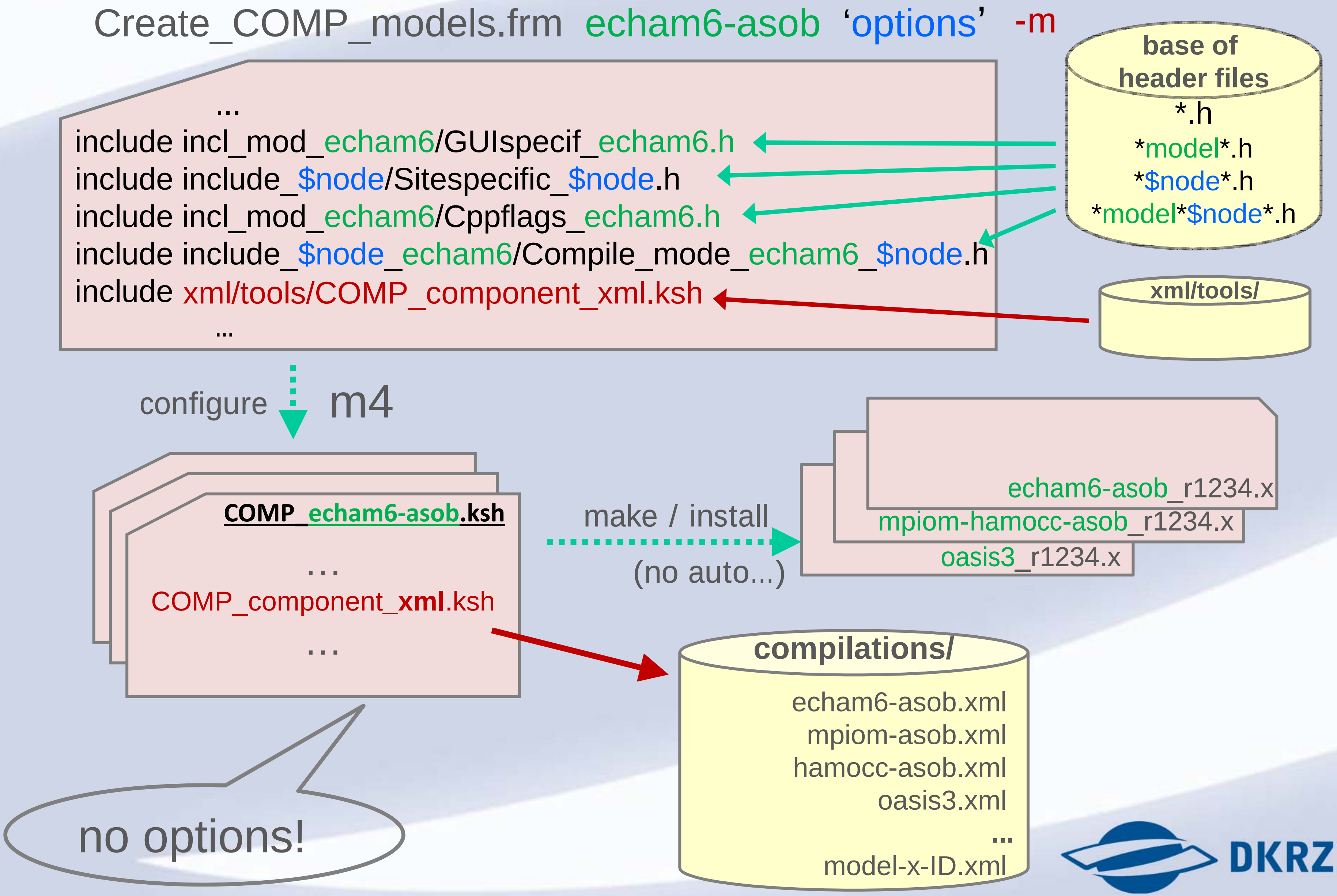
COMP_echam6-asob.ksh

…

./COMP_component_**xml**.ksh

…

Creating Configured (Compile) Scripts



Create_COMP_models.frm echam6-asob ‘options’ -m

```
...
include incl_mod_echam6/GUIspecif_echam6.h
include include_$node/Sitespecific_$node.h
include incl_mod_echam6/Cppflags_echam6.h
include include_$node_echam6/Compile_mode_echam6_$node.h
include xml/tools/COMP_component_xml.ksh
...
```

base of header files

*.h

model.h

$node.h

model$node*.h

xml/tools/

configure m4

COMP_echam6-asob.ksh

…

COMP_component_**xml**.ksh

…

make / install

(no auto...)

echam6-asob_r1234.x

mpiom-hamocc-asob_r1234.x

oasis3_r1234.x

compilations/

echam6-asob.xml

mpiom-asob.xml

hamocc-asob.xml

oasis3.xml

...

model-x-ID.xml

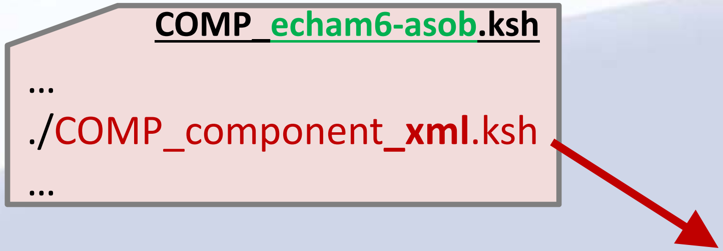
no options!

IMDI MD harvesting: high level design (III)

- compile and run scripts assembled from (model and platform specific) common header files with no branches (‘if constructs’)
  => shell-script (compile, execution) parsing is facilitated
  => make use of general IMDI structures
- no hidden info used in IMDI (e.g. dyn. libraries)
  => all information available at compile+run time
- all info is harvested
- no need to touch the models
- sufficient info for scientist’s notebook
- translation needed for external/non-scientific data user
- possibility to include hand-written annotations; e.g.
  - NAMELIST parameter functionalities
  - cpp flag functionalities

facilitate translation into xml of other DM

Filling ‚compilations/' file base (xml)



COMP_echam6-asob.ksh

```
…
./COMP_component_xml.ksh
…
```

compilations/echam6-asob.xml :

- -f && continue filling || ! -f && start with empty template
- dateStart (!-f)
- SVN URL and revision numbers of : codebase (**check**), IMDI tool kit
- software packages (compiler (**check**), CDI, MPI-2, NetCDF, lapack, ...):

  path, name, version (**replace**)
- cpp flags (add)
- compiler, loader options (incl. object files) from compiler protocols
- executable name (**replace**)
- dateEnd (**added**)

IMDI MD harvesting: high level design (IV)

- no info classification (relevant/non-relevant)
- no discussion on meta data model
  - namespace: imdi
  - long tag names
  - as little structure as possible
  - as much structure as needed (performance?)
  - redundant information
- assembled from xml pieces/sections filled during IMDI WF with (optional) manual extension
  - at experiment start and
  - at experiment end only

IMDI MD harvesting: functions/

Functionalities

- Set
  set element to given value; overwrite if already set
- SetNew
  set element to given value; no check
- SetNewCheck
  Sets value of element if not yet done; checks against existing value (equality required)
- Add[Diff]
  add new line [only if new content is different] to xml file without check
- GetContent
  Example: GetContent "CouplerName" setup-file => CouplerName=oasis3
- ReadAssignment
  in ifile: param=$param

IMDI MD harvesting: functions/

Functionalities

- GetComponents
  get [exec]names, class, config-tags, grid acronym, type of restart/output file
- CheckCodelists
  analyse code/pressure lists of component output in setup file;
  add xml formatted ‘diagnostic output’ section to ‘cplmod’_’experiment’_.xml*
- GetAdjunctFile
  retrieve NAMELIST files (as here doc in run script or from IMDI directory)
- GetInputFiles
  write one xml element for each input file (in/out, path, basename, actial name)
- ReadNamelist
  create for each component xml sections for all namelists and groups incl. given value
- Software
  libraries, tool kits, submodels or processes
- AssembleXML
  assemble filled pieces into valid xml file describing the complete experiment

IMDI MD Assembly

Last call before the simulation is started:

function AssembleXML

- *assemble pieces into valid xml file (complete experiment set up)*

When simulation is finished

- *last simulation year:*
  - *include performance statistics (saved for all (re)starts (even unsuccessful ones))*
  - *document ‘disturbed’ restarts (time and size of disturbance)*

➢ The final xml file with model/experiment description is ready. It may be completed manually (e.g. parameters for Cera interfacing)

The End