

The PRISM infrastructure

System Architecture and User Interface

Kristian Mogensen, ECMWF



ECMWF



FUJITSU



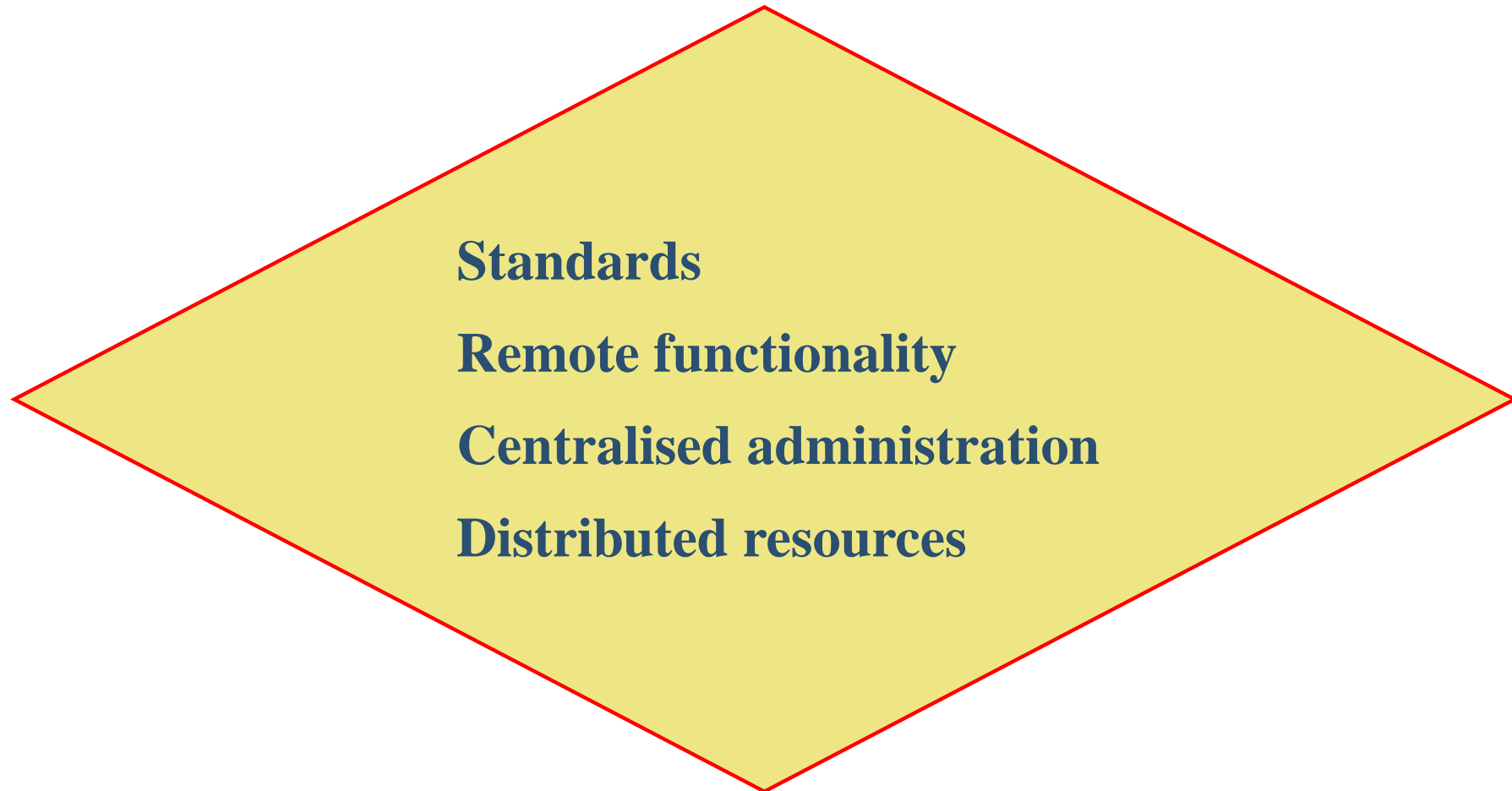
MPI Model/Data



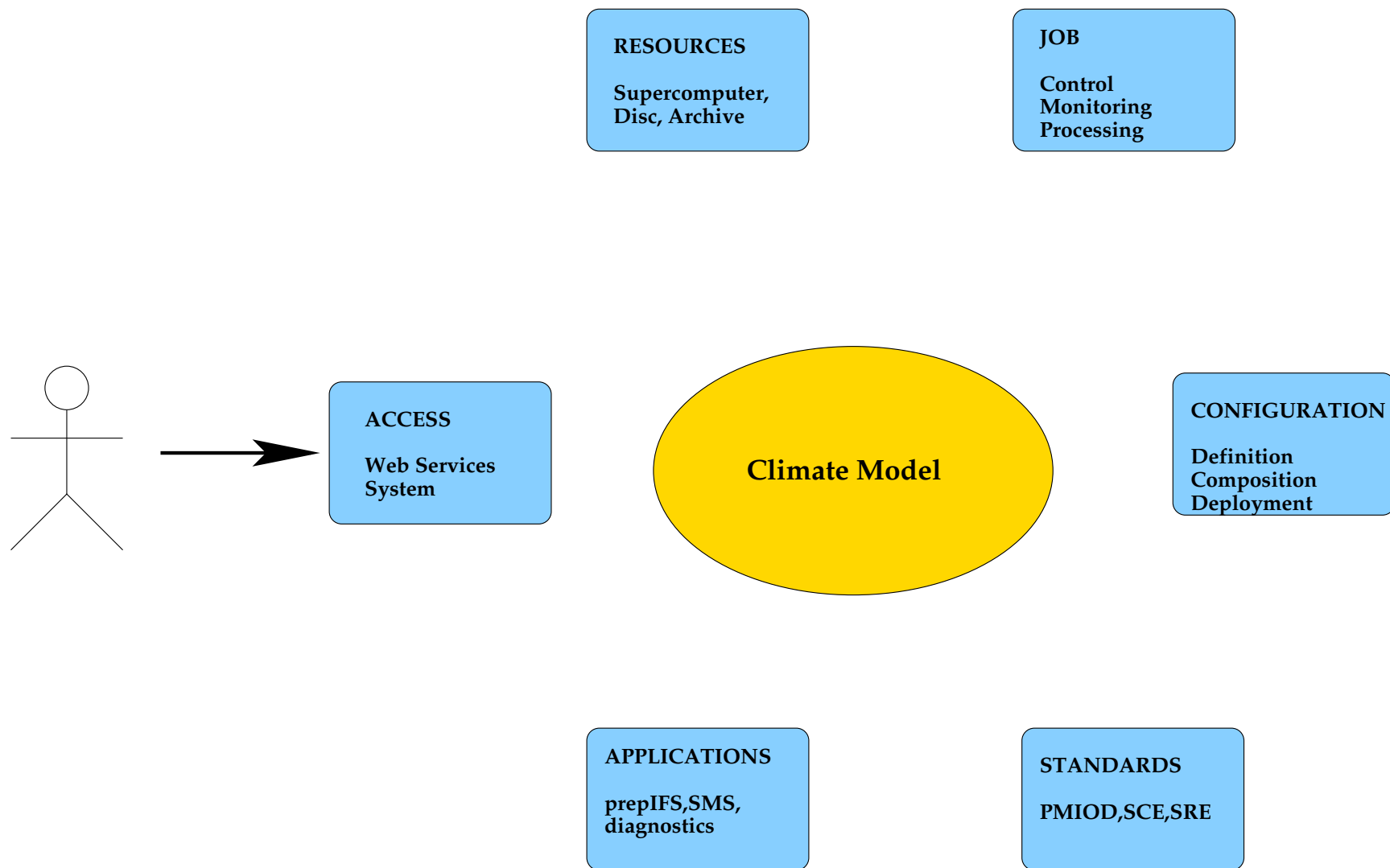
PIK

Architecture goals

The PRISM architecture provides an efficient climate modelling infrastructure through:



Infrastructure



Architecture features

Extendability - Standards

Collaboration - Exchange of configurations

Support - Tested configurations and sw

Accessibility - Remote configuration and job control

Target groups

PRISM targets:

Users - Running model experiments on remote hosts

Developers - Software developers creating models

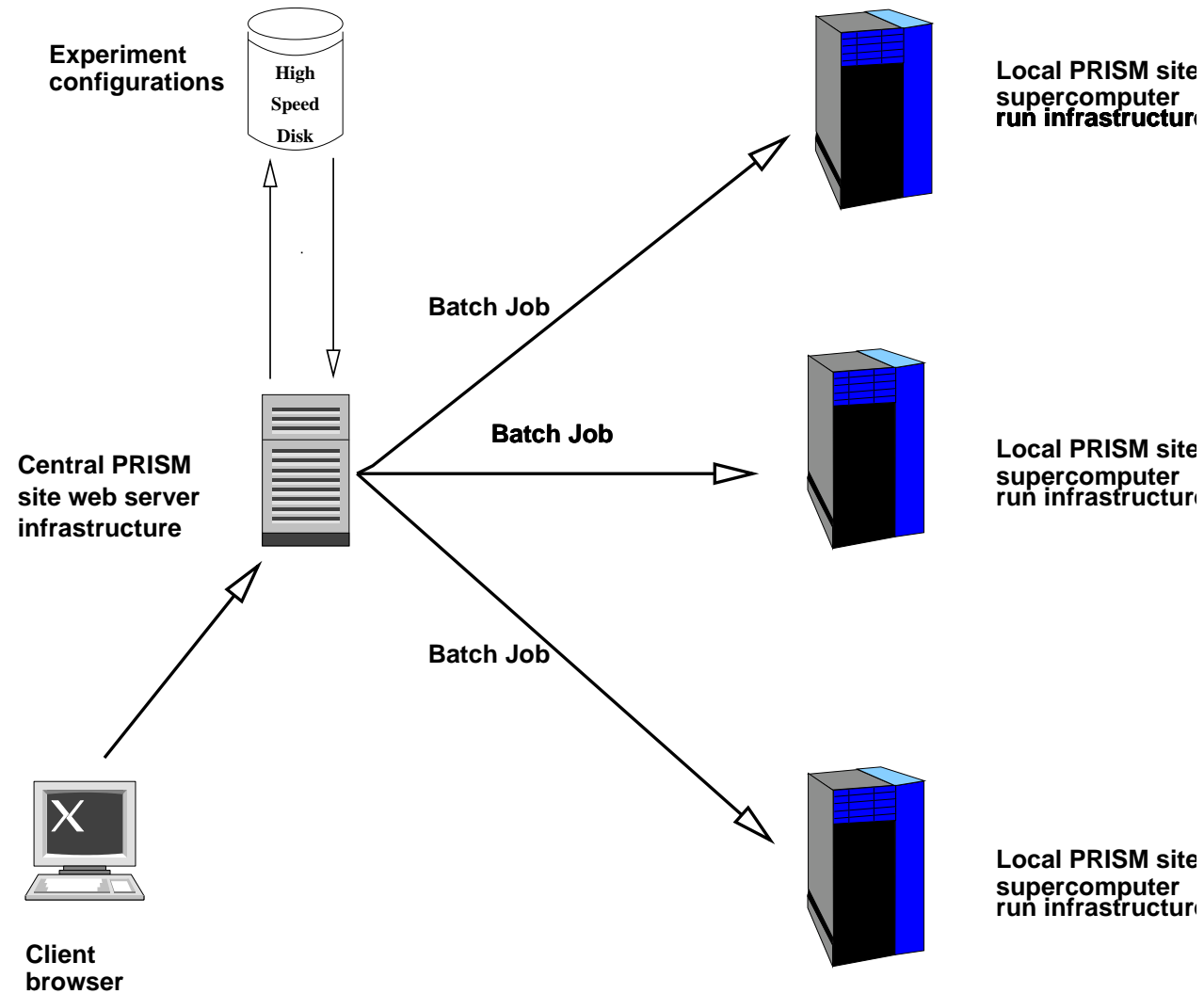
Requirements:

Users - Support and pre-tested configurations

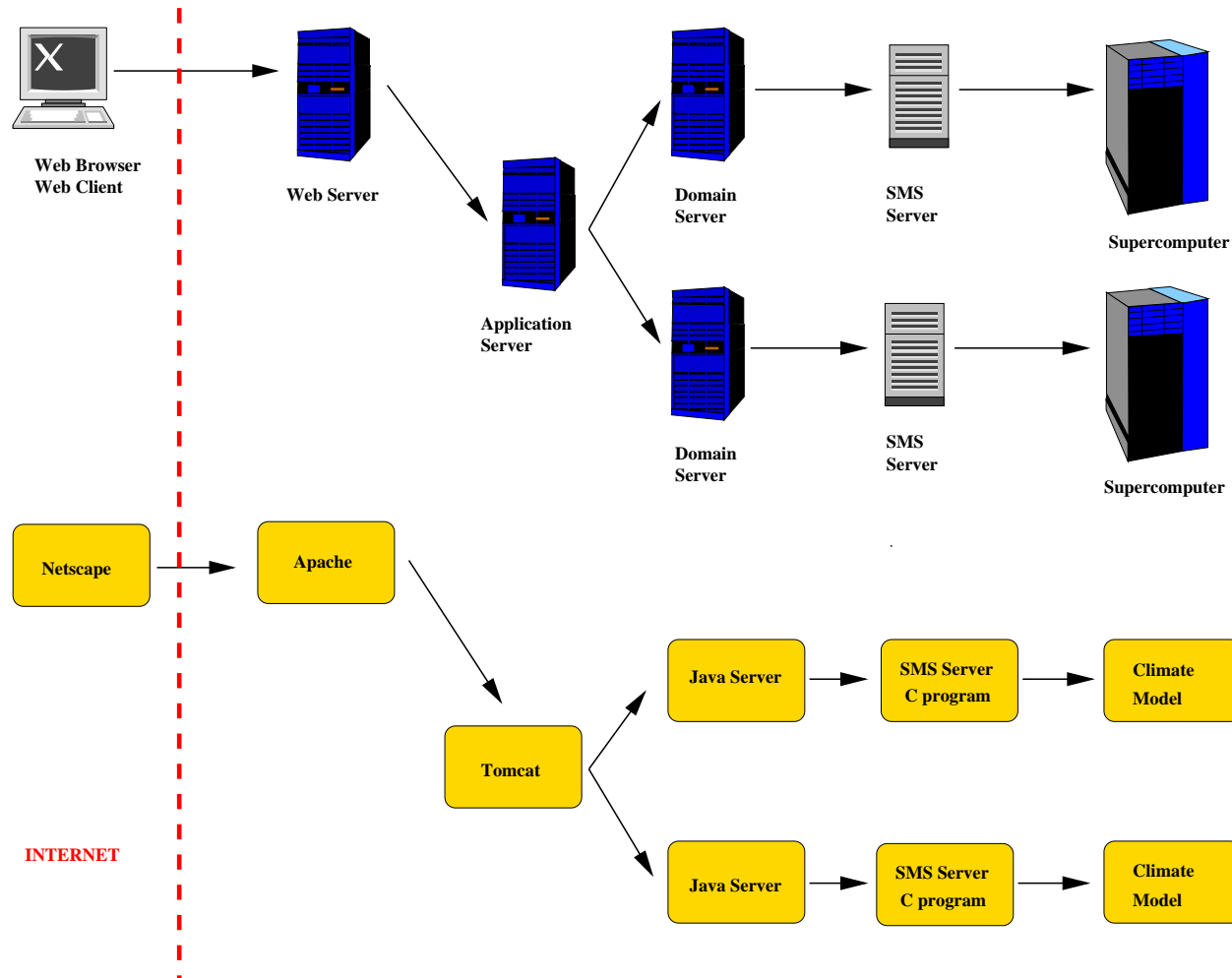
Developers - Full control of the environment

Web Services System (WSS) Architecture

Delivering remote applications to the users.



Software Components of the WSS



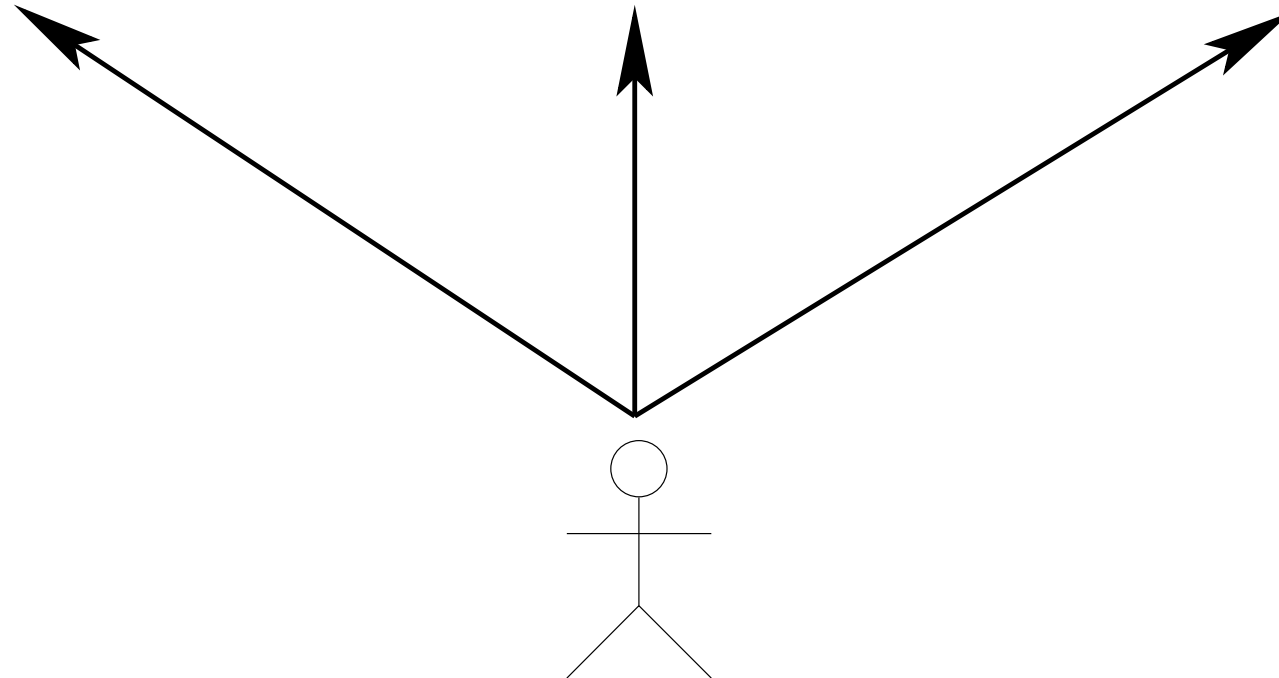
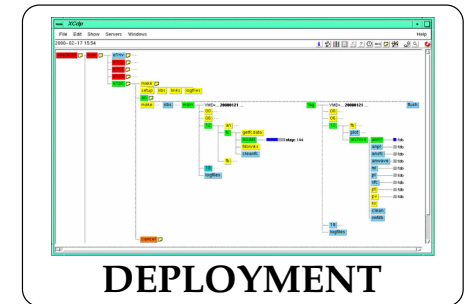
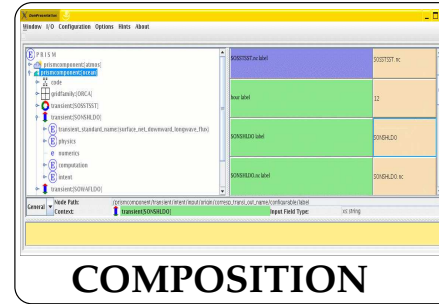
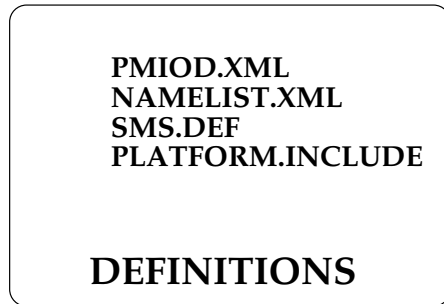
Local System Architecture

Software installed on local host. Developers have control. Model developer delivers standardised :

- Compile environment, build instructions
- I/O description, PMIOD and environment
- Run script
- Model code

Model is installed and run through standardised Web Services infrastructure.

Model configuration



Model run and deployment

- **Job creation** - modularised SMS scripts for :
 - ◆ setup
 - ◆ run
 - ◆ postprocess
 - ◆ diagnostics
 - ◆ archive
 - ◆ cleanup
- **Job deployment** - mixed host and platform support through the SCE and SRE
- **Job control** - GUI support and monitoring

WSS Applications

Applications supporting the running of PRISM climate models:

- PrepIFS - GUI model configuration tool
- PrepOASIS - OASIS-4 configuration tool
- SMS/Xcdp/WebCDP - Scheduler and monitoring GUI
- Web Security System

A beta release of the system is available through the PRISM CD.

PrepIFS - What does it do?

It configures parameters and submits model runs locally or to a remote host.

Features:

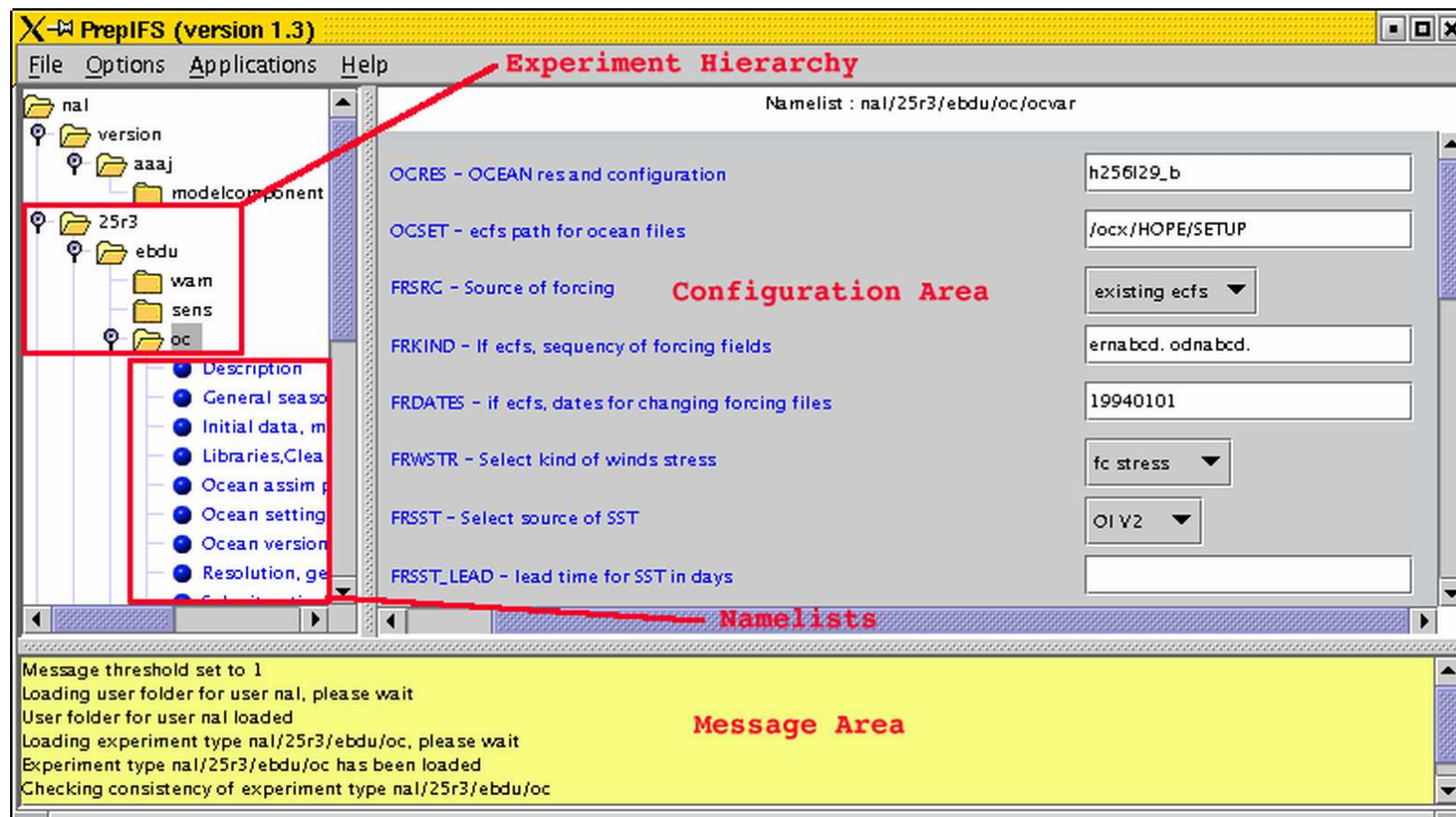
- It is easy to learn with a graphical user interface
- It ensures consistency and cloneability of model runs
- It links to documentation and provides help
- It records setups in a database

PrepIFS - continued

- It is low in maintenance and can integrate new model components
- It has a rule-based knowledge representation system acting on the user input to prevent errors and to modify the configuration
- Can prevent invalid setups from submission and failure

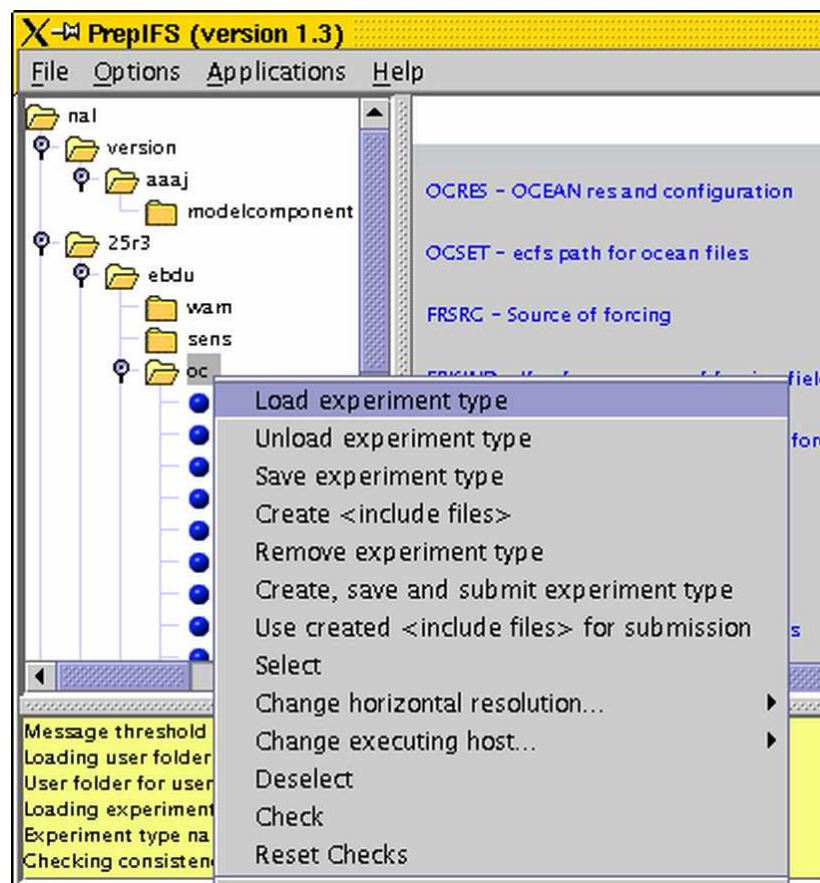
PrepIFS Graphical User Interface

Main window



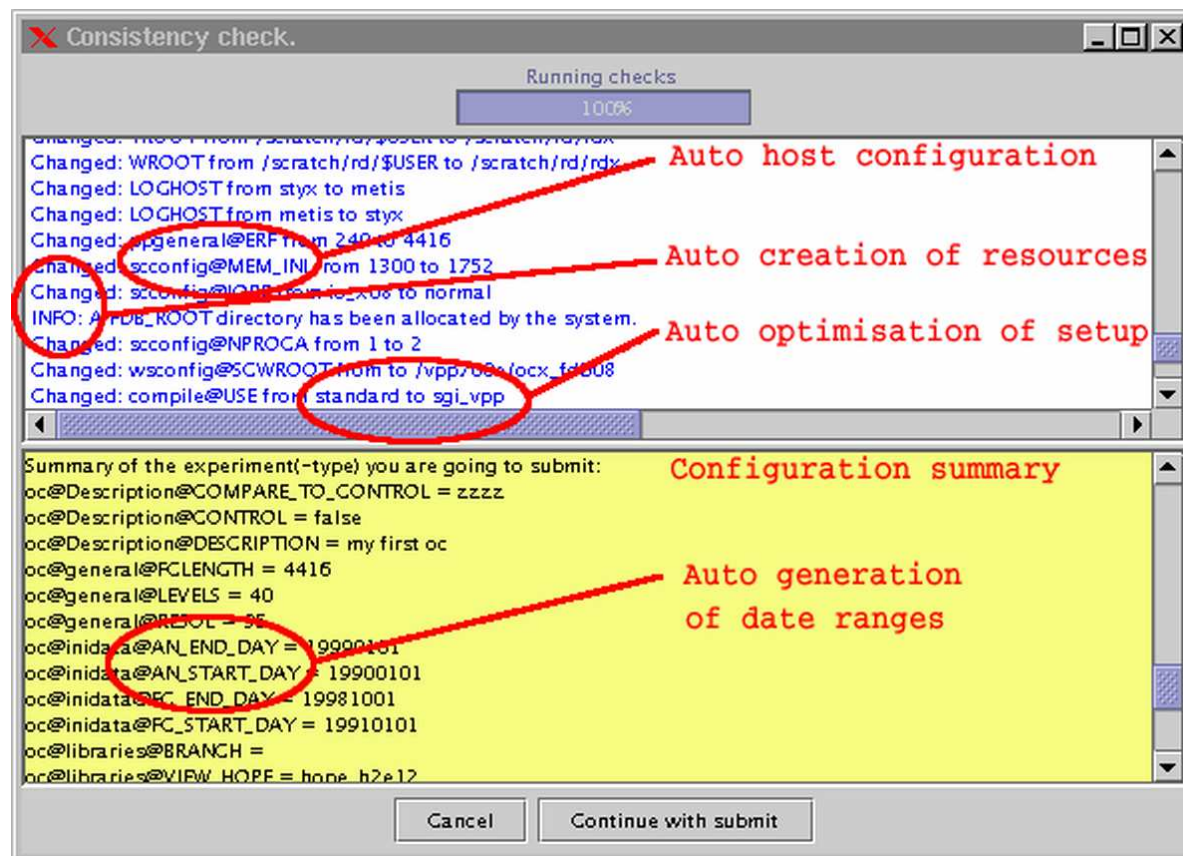
PrepIFS - Functionality

- Loading
- Saving
- Deleting
- Checking
- Submitting



PrepIFS - Check of configuration

- Optimisations
- Host specifics
- Resource creation
- Information on problems
- Corrections



```

Consistency check.
Running checks
100%
Changed: WROOT from /scratch/rd/$USER to /scratch/rd/rdx
Changed: LOGHOST from styx to metis
Changed: LOGHOST from metis to styx
Changed: ngeneral@ERF from 240 to 4416
Changed: sconfig@MEM_INI from 1300 to 1752
Changed: sconfig@IOPE from io_x06 to normal
INFO: ANDB_ROOT directory has been allocated by the system.
Changed: sconfig@NPROCA from 1 to 2
Changed: wsconfig@SCWROOT from /vpp/oc/locx_fd008
Changed: compile@USE from standard to sgi_vpp

Summary of the experiment(-type) you are going to submit:
oc@Description@COMPARE_TO_CONTROL = zzzz
oc@Description@CONTROL = false
oc@Description@DESCRIPTION = my first oc
oc@general@FCLENGTH = 4416
oc@general@LEVELS = 40
oc@general@RESOL = 25
oc@inidata@AN_END_DAY = 19990101
oc@inidata@AN_START_DAY = 19900101
oc@inidata@FC_END_DAY = 19981001
oc@inidata@FC_START_DAY = 19910101
oc@libraries@BRANCH =
oc@libraries@VIEW_HOPE = hope_h2e12
    
```

Auto host configuration

Auto creation of resources

Auto optimisation of setup

Configuration summary

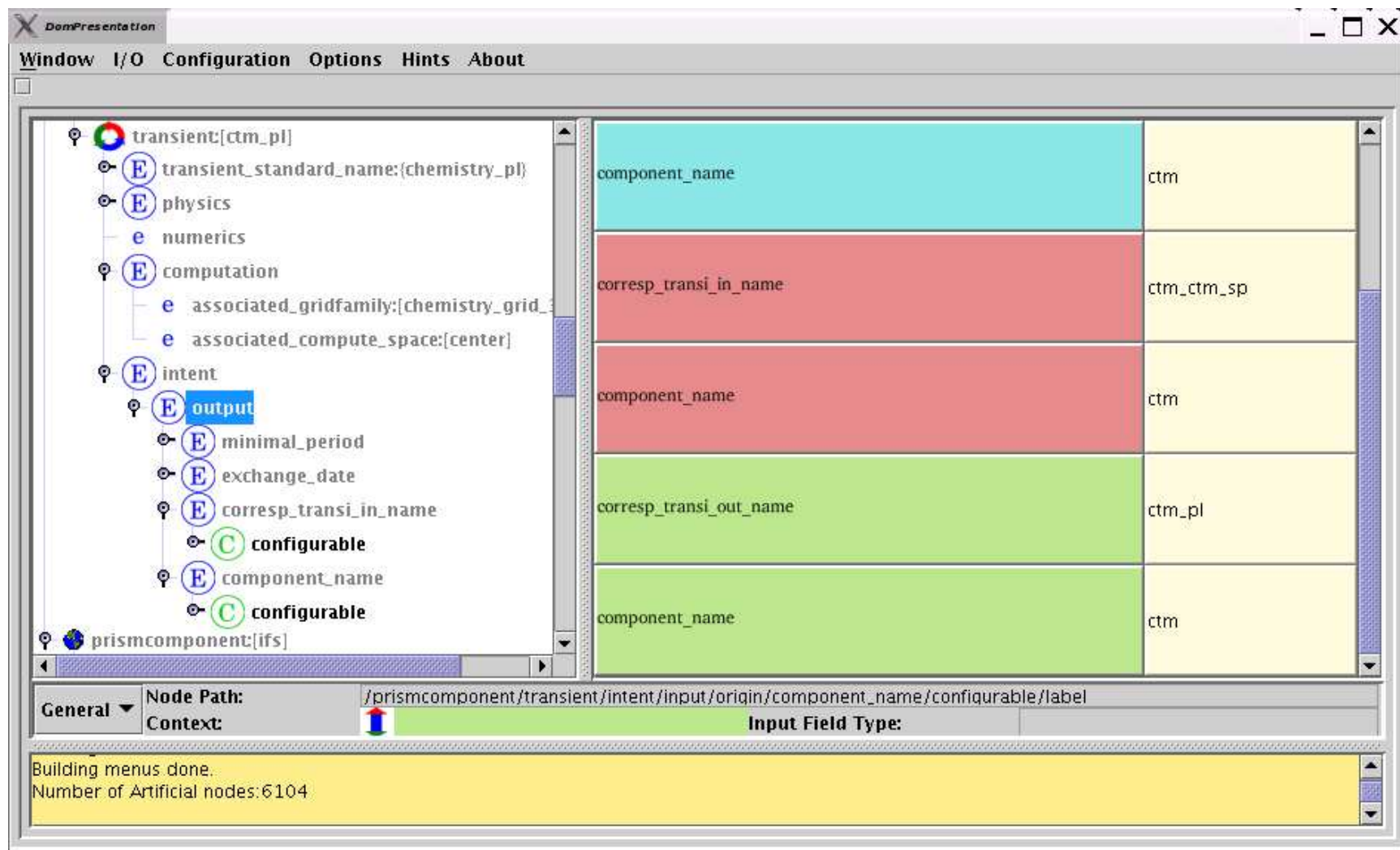
Auto generation of date ranges

Ability to override the system settings.

PrepOASIS - What does it do?

- Builds a configuration from OASIS4 PMIOD files
- GUI configuration of the coupler such as:
transformations, interpolation and IO
- Visual coupling interface (VCI) for configuration of
model variables

PrepOASIS - Configuration



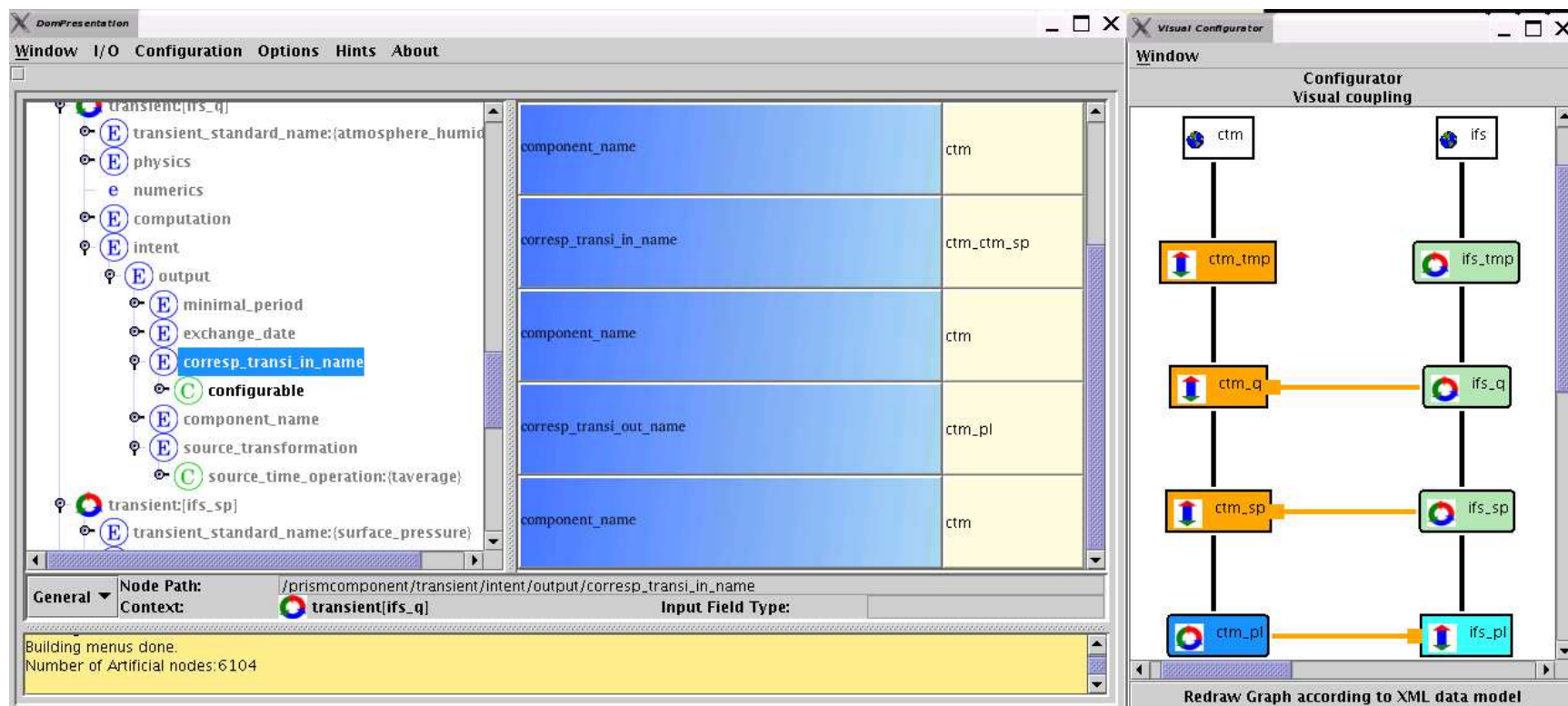
The screenshot shows the DomPresentation software interface. On the left is a tree view of the configuration structure. The 'output' node is selected. On the right is a table of configuration parameters. Below the table is a status bar showing the node path and context. At the bottom is a log window.

component_name	ctm
corresp_transi_in_name	ctm_ctm_sp
component_name	ctm
corresp_transi_out_name	ctm_pl
component_name	ctm

Node Path: /prismcomponent/transient/intent/input/origin/component_name/configurable/label
Context: Input Field Type:

Building menus done.
Number of Artificial nodes:6104

PrepOASIS - Visual coupling



The image shows two windows from the PrepOASIS software. The left window, titled 'DonPresentation', displays a configuration tree for a 'transient[ifs_q]' component. The tree structure is as follows:

- transient[ifs_q]
 - transient_standard_name:(atmosphere_humid)
 - physics
 - numerics
 - computation
 - intent
 - output
 - minimal_period
 - exchange_date
 - corresp_transi_in_name (highlighted)
 - configurable
 - component_name
 - source_transformation
 - source_time_operation:(taverage)
- transient[ifs_sp]
 - transient_standard_name:(surface_pressure)

Below the tree, a table lists the values for the selected 'corresp_transi_in_name' field:

component_name	ctm
corresp_transi_in_name	ctm_ctm_sp
component_name	ctm
corresp_transi_out_name	ctm_pl
component_name	ctm

The status bar at the bottom of the left window shows: 'General Node Path: /prismcomponent/transient/intent/output/corresp_transi_in_name Context: transient[ifs_q] Input Field Type:'. A message box at the bottom indicates 'Building menus done. Number of Artificial nodes:6104'.

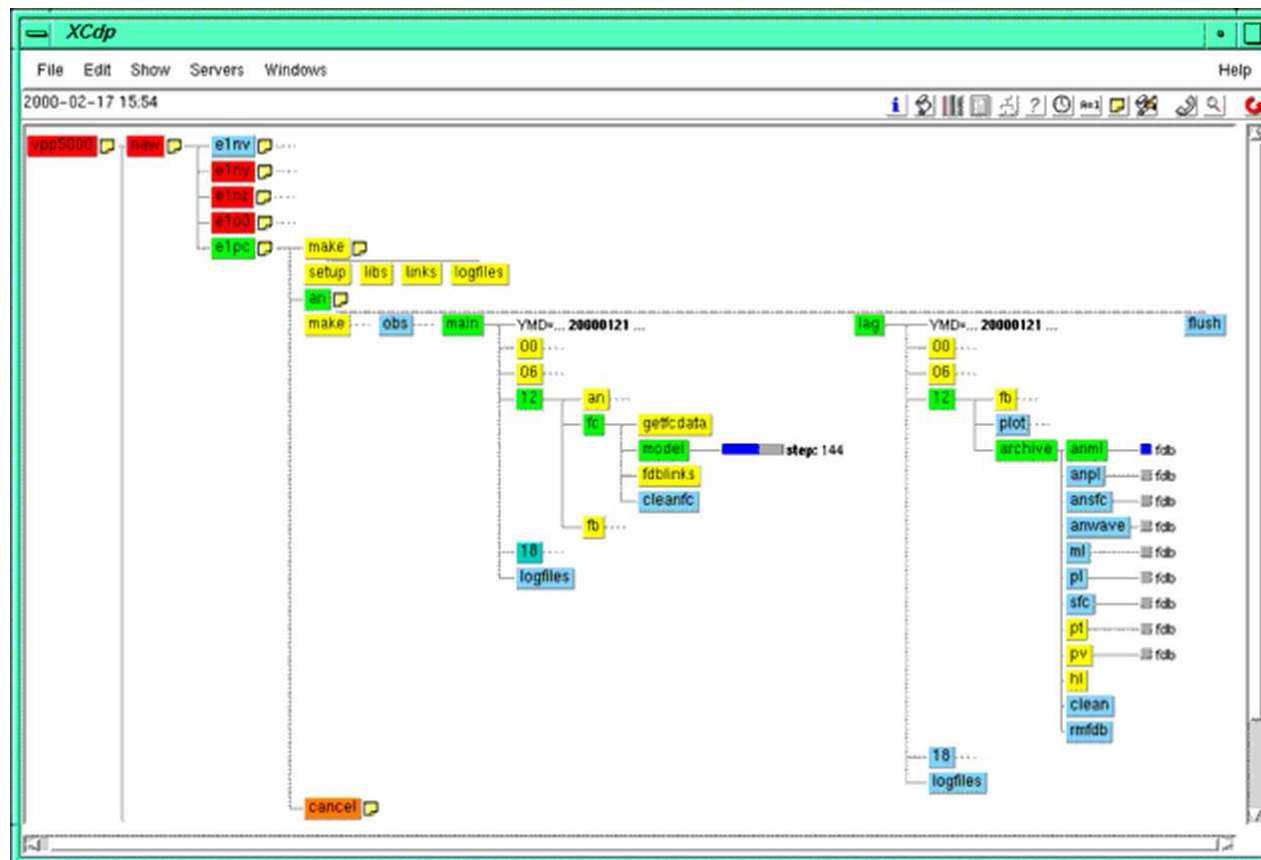
The right window, titled 'Visual Configurator', shows a 'Configurator Visual coupling' graph. It illustrates the data flow between two model components: 'ctm' (left) and 'ifs' (right). The graph consists of several nodes connected by lines:

- Top level: 'ctm' and 'ifs' (parent nodes).
- Second level: 'ctm_tmp' (orange) and 'ifs_tmp' (green).
- Third level: 'ctm_q' (orange) and 'ifs_q' (green), connected by a horizontal orange arrow pointing from ctm_q to ifs_q.
- Fourth level: 'ctm_sp' (orange) and 'ifs_sp' (green), connected by a horizontal orange arrow pointing from ctm_sp to ifs_sp.
- Fifth level: 'ctm_pl' (blue) and 'ifs_pl' (cyan), connected by a horizontal orange arrow pointing from ctm_pl to ifs_pl.

Vertical lines connect the parent nodes to their respective child nodes in each level. A button at the bottom of the right window reads 'Redraw Graph according to XML data model'.

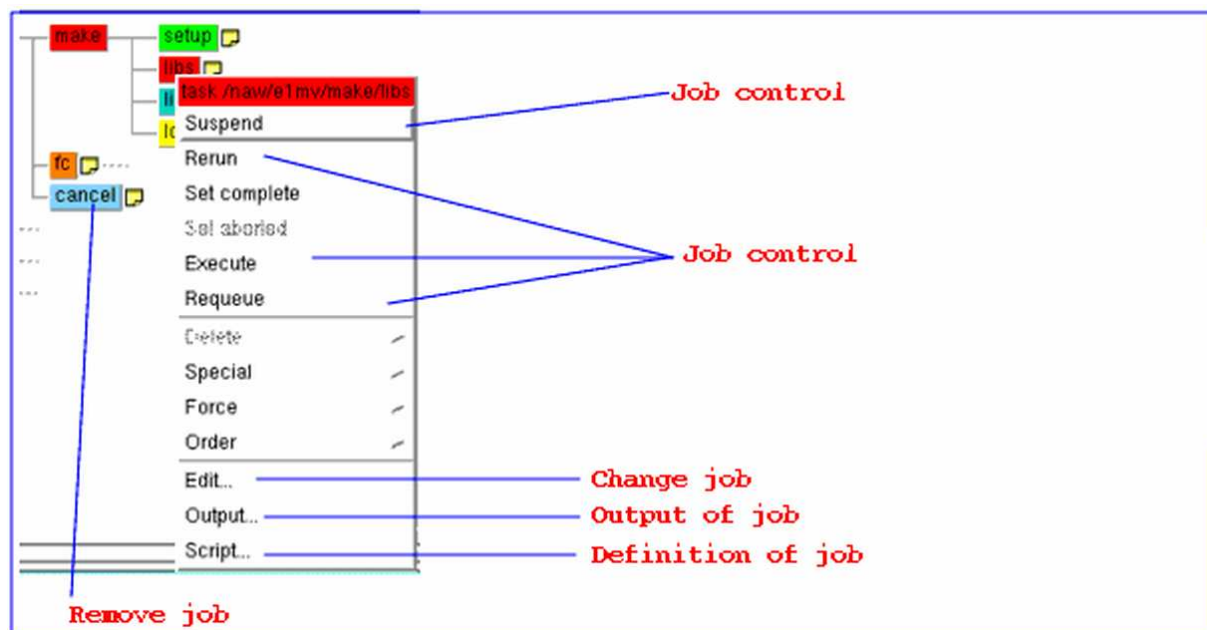
Xcdp - Job control GUI

- Colour status
- Hierarchy display
- Info access
- Control
- Alerts



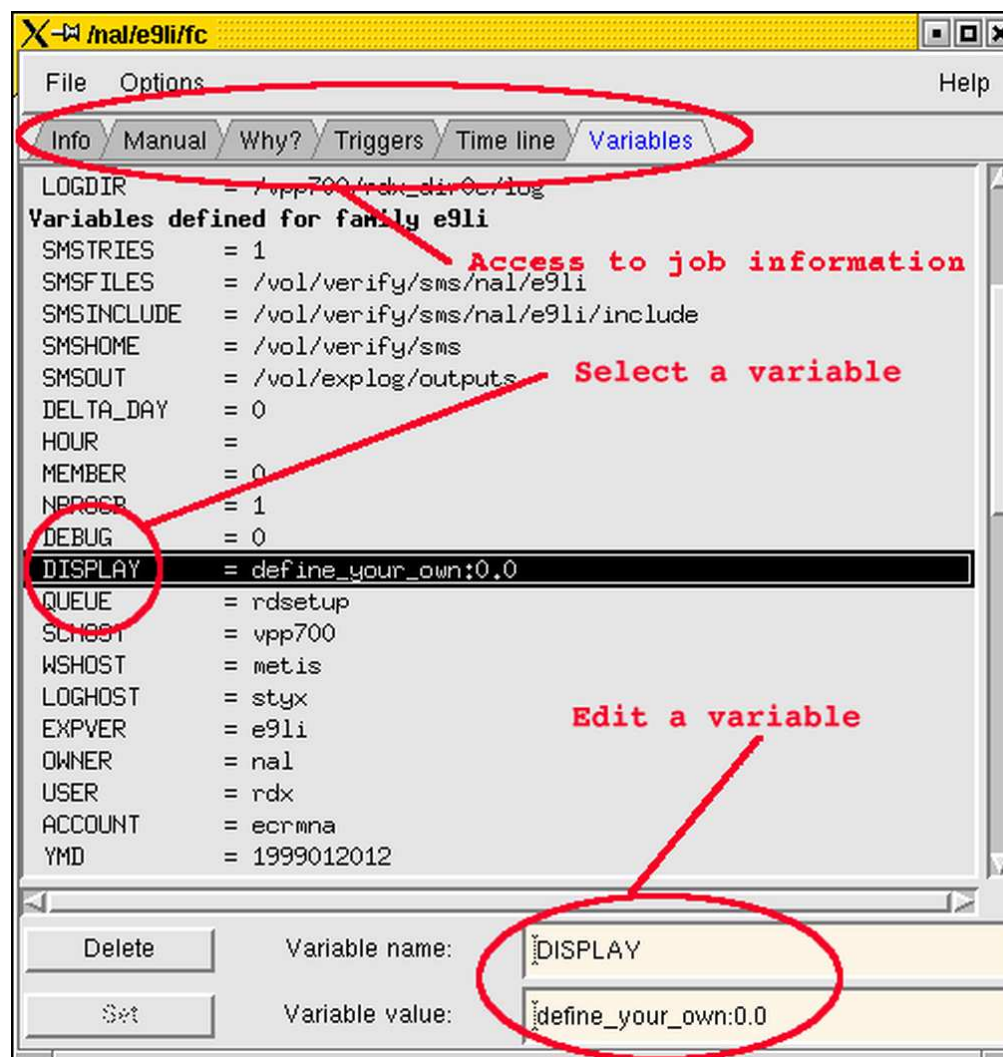
Xcdp - Job control, functionality

- Cancel job
- Rerun job
- Suspend job
- Set complete
- Edit scripts
- Browse logfiles



Xcdp - Information access

- Logfiles
- Scripts
- Definitions
- Editing
- Debugging
- Watching



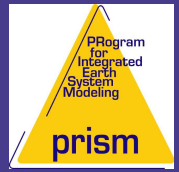
The screenshot shows the 'Variables' window in Xcdp. The window title is 'nal/e9li/fc'. The 'Variables' tab is selected and circled in red. Below the tabs, a list of variables is shown. The 'DISPLAY' variable is highlighted with a red circle. A red arrow points from the text 'Access to job information' to the 'Variables defined for family e9li' section. Another red arrow points from the text 'Select a variable' to the 'DISPLAY' variable. A third red arrow points from the text 'Edit a variable' to the 'Set' button at the bottom. The 'Set' button and the 'Variable value' field are also circled in red.

Variable	Value
LOGDIR	= /vpp700/rdx_dir/0c/log
Variables defined for family e9li	
SMSTRIES	= 1
SMSFILES	= /vol/verify/sms/nal/e9li
SMSINCLUDE	= /vol/verify/sms/nal/e9li/include
SMSHOME	= /vol/verify/sms
SMSOUT	= /vol/explog/outputs
DELTA_DAY	= 0
HOUR	=
MEMBER	= 0
NPROCP	= 1
DEBUG	= 0
DISPLAY	= define_your_own:0.0
QUEUE	= rdsetup
SCHEDT	= vpp700
WSHOST	= metis
LOGHOST	= styx
EXPVER	= e9li
OWNER	= nal
USER	= rdx
ACCOUNT	= ecrma
YMD	= 1999012012

Buttons: Delete, Set

Variable name: DISPLAY

Variable value: define_your_own:0.0



Future plans

- **Coupling consistency** - Rule based constraints
- **More visual cues** - New functionality in the VCI
- **Job control** - Better integration with prepIFS

End



ECMWF



FUJITSU



MPI Model/Data



PIK

Thanks for your attention.