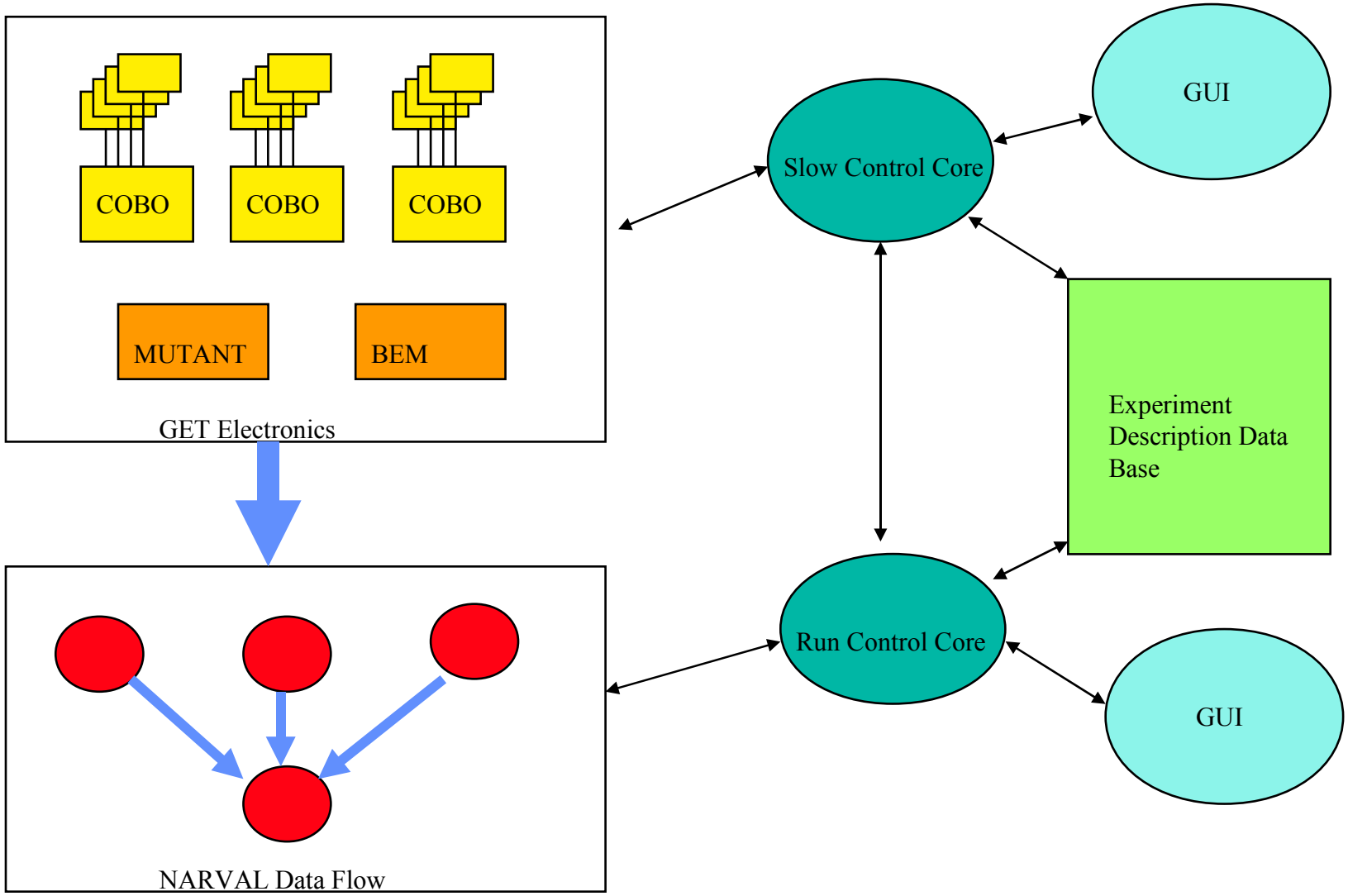
GANIL Run Control System

Frédéric Saillant

GANIL – Groupe Acquisition Physique

GANIL Run Control system



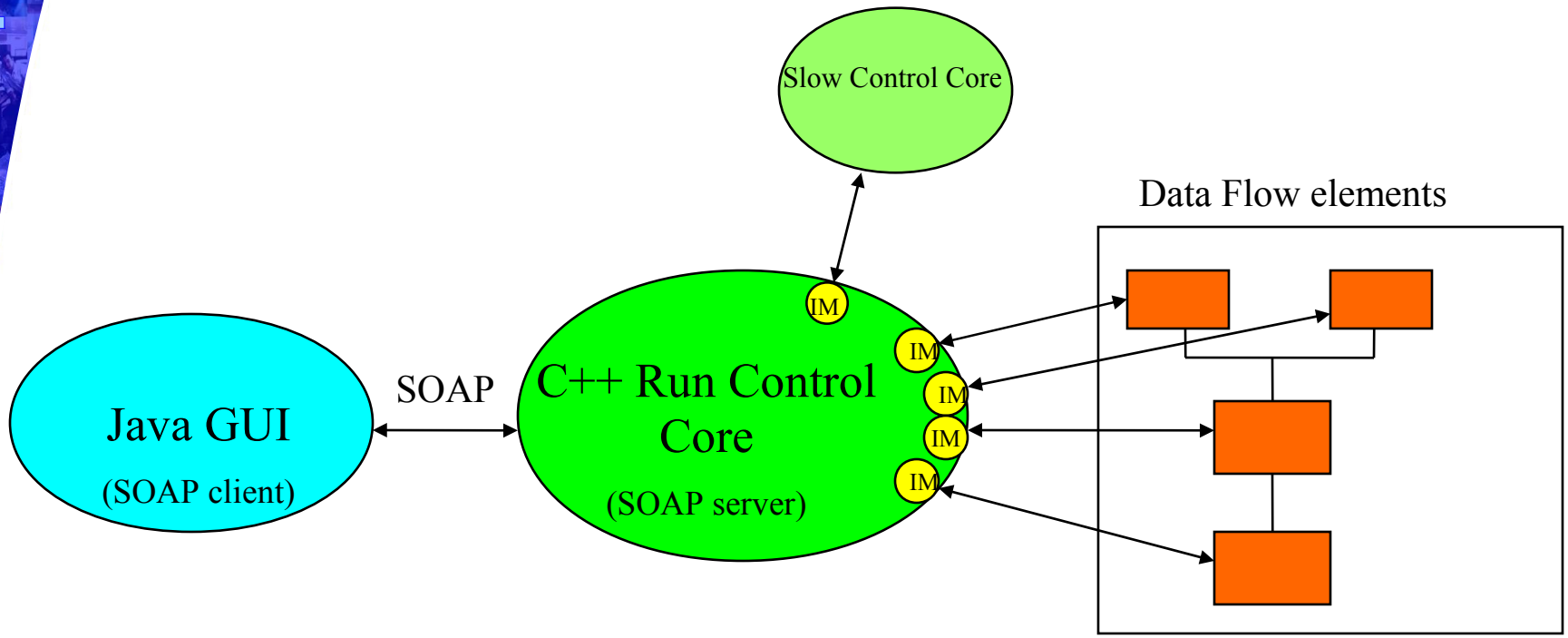
General overview of Slow and Run Control for GET

GANIL Run Control System

Run Control main tasks :

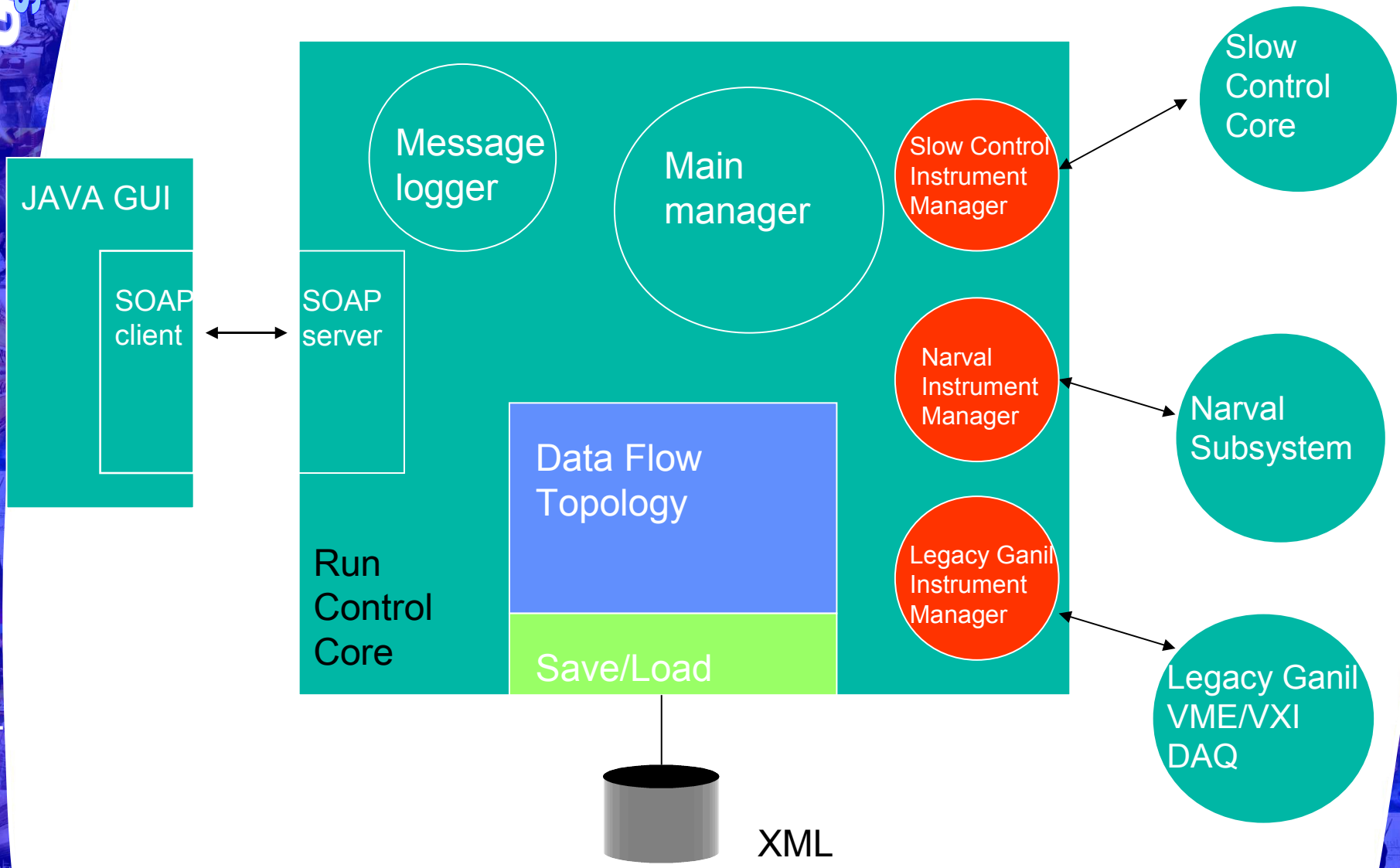
- Configure DAQ for a run by selecting data flow elements
- Save/restore a configuration
- Commands to control data flow elements
- Commands to globally setup, start, stop electronics through Slow Control Core
- Monitor DAQ (status, data rates...)
- Handle error/info messages
- Log book
- User-friendly graphical interface, separated from the core of the application

GANIL Run Control System



- Run Control Core accesses data flow components with specific communication protocols encapsulated in « Instrument Managers »
- Run Control Core written in C++ with gSOAP library
- WSDL file generated by gSOAP
- Java GUI integrates SOAP client stub thanks to WSDL file

GANIL Run Control System



Data Flow Topology Description

What are the objects involved in a topology ?

- Instruments :
  - Narval subsystem (contains actors)
  - Narval actors (producers, mergers, filters, storage actor …)
  - Legacy Ganil VME/VXI DAQ (Vamos, Lise, Must2 …)
  - MIDAS DAQ from Daresbury (Exogam, Tiara …)
- Oriented links :

  represent data flow from one instrument to another

Data Flow Topology Description

What are the objects involved in a topology ? (cont'd)

- Instrument Parameters :

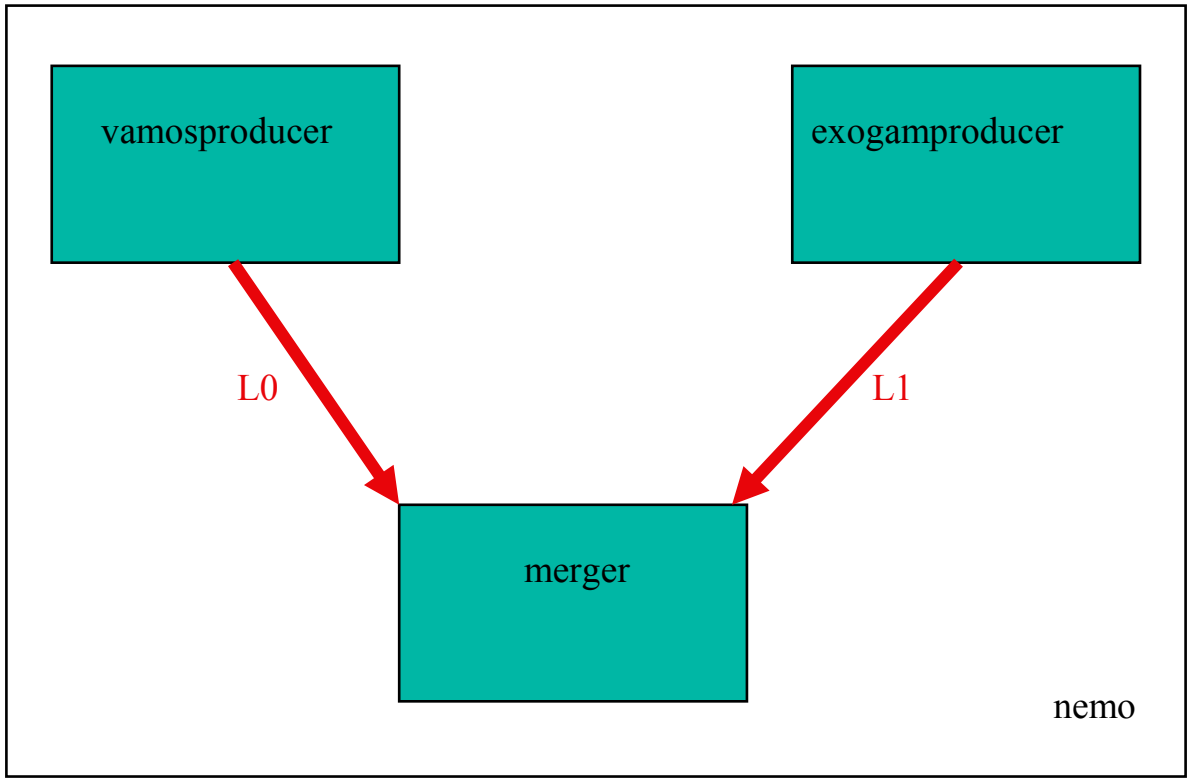
  Parameters belonging to an instrument. They have a name, a type (string, integer, boolean …), a value and some access rights (read-only, read-write …).

  Example :

  for the storage actor, name of the file where to store incoming data

Data Flow Topology Description

Example of a topology:



Data Flow Topology Description

How to save the data flow topology description ?

⟶ XML file

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <configuration xmlns="etest.xml">
- <narval hostname="ganp754" name="nemo">
- <actor executable_file="acqsbuf" hostname="ganp754" log_level="debug" name="vamosproducer">
<parameter dynamic="true" name="acqvmecrate" permission="read_only" type="string_type" value="ganlo1" />
</actor>
<actor executable_file="acqsbuf" hostname="ganp754" log_level="debug" name="exogamproducer" />
<actor executable_file="evtnum_eventbuilder" hostname="ganp754" log_level="debug" name="merger" />
</narval>
<link buffer_size="1000000" destination="merger" destination_port="fifo" name="L0" source="vamosproducer"
source_port="fifo" />
<link buffer_size="1000000" destination="merger" destination_port="fifo" name="L1" source="exogamproducer"
source_port="fifo" />
</configuration>
```

XML generation and parsing are based on Xerces library

(DOM for file generation, SAX for file parsing)

Data Flow Topology Description

How to build the data flow topology ?

→ a complete set of web services to create and modify instruments, links ans parameters.

• CreateNarval	adds a Narval subsystem
• CreateActor	adds a Narval actor
• CreateStorage	adds a Narval storage actor
• CreateEventBuilder	adds a Narval event builder
• CreateVmecom	adds a legacy Ganil VME crate
• CreateLink	defines a link between two instruments
• CreateParameter	adds a parameter to an instrument
• SetParameterValue	modifies parameter value
• …	

Data Flow Topology Description

How to get information about an existing topology ?

• GetEquipmentCount	number of instruments
• GetLinkCount	number of links
• GetParameterCount	number of parameters for a given instrument
• GetEquipmentByIndex	returns info on equipment of a given index
• GetLinkByIndex	returns info on link of a given index
• GetParameterByIndex	returns info on parameter of a given index in an instrument
• GetParameterType	returns parameter type
• GetParameterValue	returns parameter value
• …	

The configuration graphical user interface uses these web services to allow the user to fully manipulate the topology of the system.

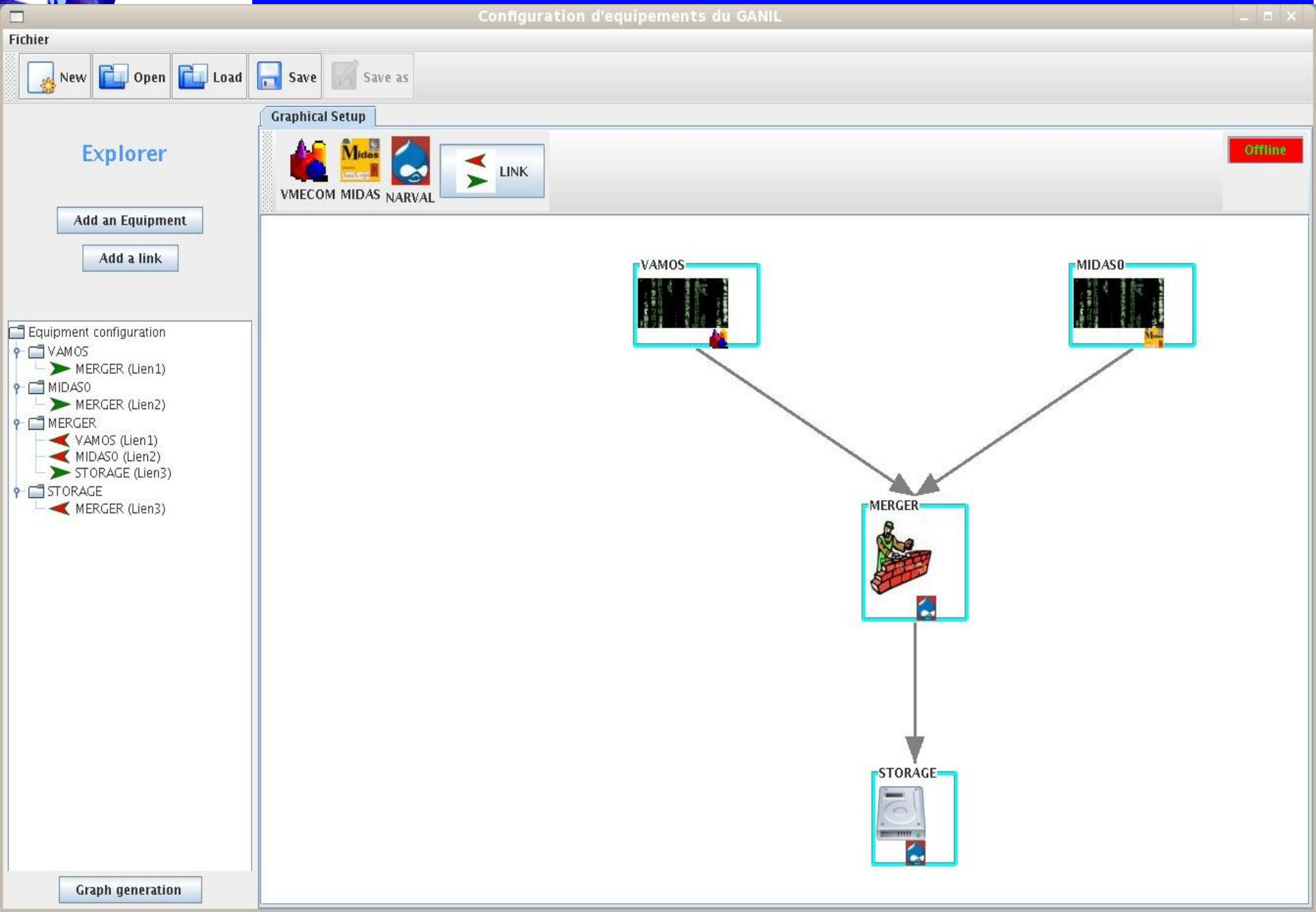
Run Control Graphical User Interface

Current GUI only for building and modifying the topology :

- developped in Java
- uses web services to get the topology from Run Control Core, modify the topology or create a new one from scratch
- SOAP client side generated thanks to WSDL file provided by RCC
- error messages handled by Log4j
- allows the user to add an instrument of any known type in the topology
- allows the user to add a link between two existing instruments
- topology is always displayed in a graphical window and in a tree

Development of command and monitoring GUI not yet started

GANIL Run Control system



Error messages management

- Use of Log4j and Log4cxx
- Log4ada developped by Xavier Grave (Orsay) for Narval
  - → works like Log4j/Log4cxx
- Legacy Ganil VME/VXI crates use a RPC-based protocol to send messages
  - → Run Control Core will include an RPC server to translate into Log4cxx
- Message Logger centralizes all messages and log them on a disk file (XML)
- Messages can also been visualized in a graphical tool like Chainsaw, with advanced sorting capabilities

States monitoring and management

- each instrument manager contains a state machine reflecting the one of the real DAQ instrument (a monitoring thread asks for instrument state continuously)
- Run Control Top Manager contains a global state machine updated accordingly to the instrument state machines
- global state and individual states are checked when a command is issued by the user
- a command on the system might have an impact on the instrument state machines and the global state machine
- not yet implemented …

GANIL Run Control System

NARVAL

- Originally developped by IPN Orsay
- Today, collaborative development with IPNO, CSNSM, GANIL, LPC Caen
- Distributed Acquisition System
- Developped in Ada95
  - Object Oriented programming
  - Strongly typed language
  - Robust applications
  - Distributed processes by using Annex E (CORBA equivalent)
- Easy to link with C++
- Used for AGATA DAQ
- *Web site : http://narval.in2p3.fr/*

NARVAL

- **Main components :**
  - A main process to handle the state of all the configuration (Coordinator)
  - Set of actors to manage the data flow
    - Producer : input of data flow (hardware or other DAQ)
    - Intermediary : acts as a NxM soft switch that can filter data
    - Consumer : end of data flow (data storage, output to other DAQ,...)
  - Logging of error/info messages (Log4Ada)
- **Data flow transport over Unix fifo, TCP/IP, Infiniband**
- **Communication via Web Services (SOAP) with the « Coordinator »**

NARVAL

Specific actors already developped for Ganil DAQ :

- interface for legacy Ganil VME/VXI crates (Ganil development)
- event builder based on event number (Ganil development)
- storage actor to write data flow on disk (LPC Caen development)