Run Control - Slow Control Integration

Michele Gulmini
INFN - Laboratori Nazionali di Legnaro

Agata Week 2009 – Cologne, April 1st 2009

Outline

- Run Control and Slow Control: Overview
  - Run Control Structure
  - Slow Control subsystems

- Run Control – Slow Control integration
  - Agata Slow Control (WSDL)Interface
  - Current Status
  - Ongoing work
    - New: Global Slow Control Supervisor

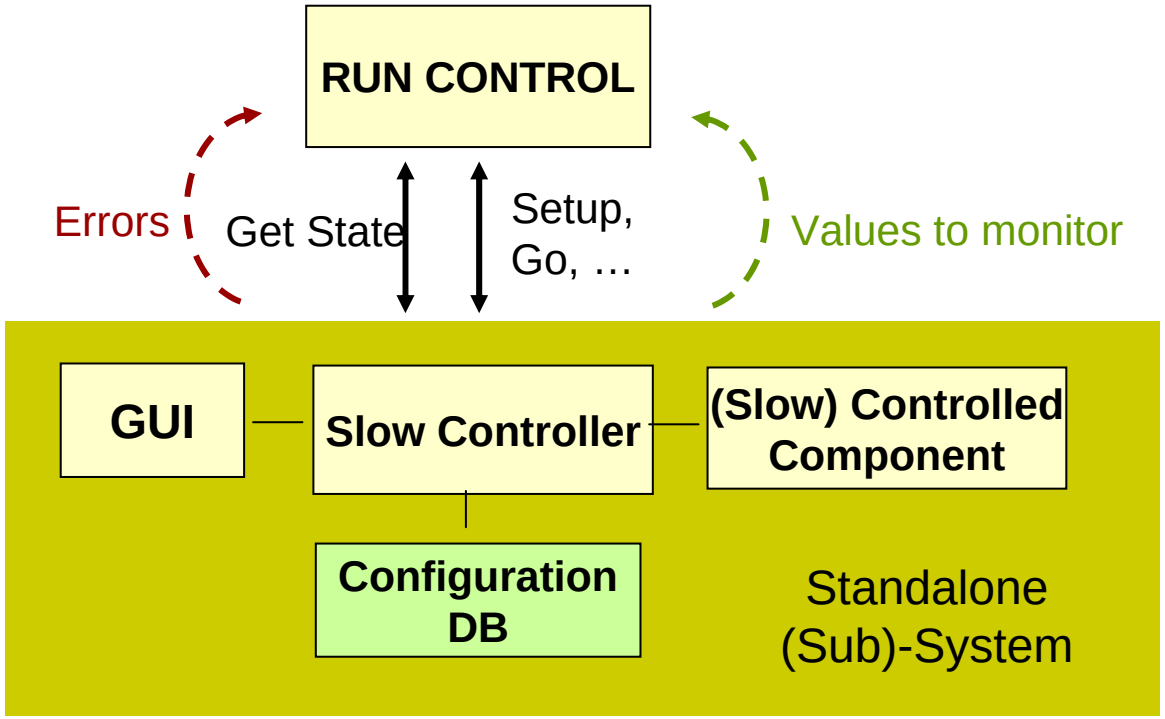
Run Control and Slow Control

- The Run Control System coordinates the several activities to put the Agata detector, its data acquisition system and its ancillary detectors into operational state.

- The Slow Control System controls the hardware devices, taking care of their correct configuration and setup.

- The Run Control interacts with the Slow Control System, in order to assure the correct configuration and setup of the hardware before a data taking is started.

Slow Control (Sub-)System



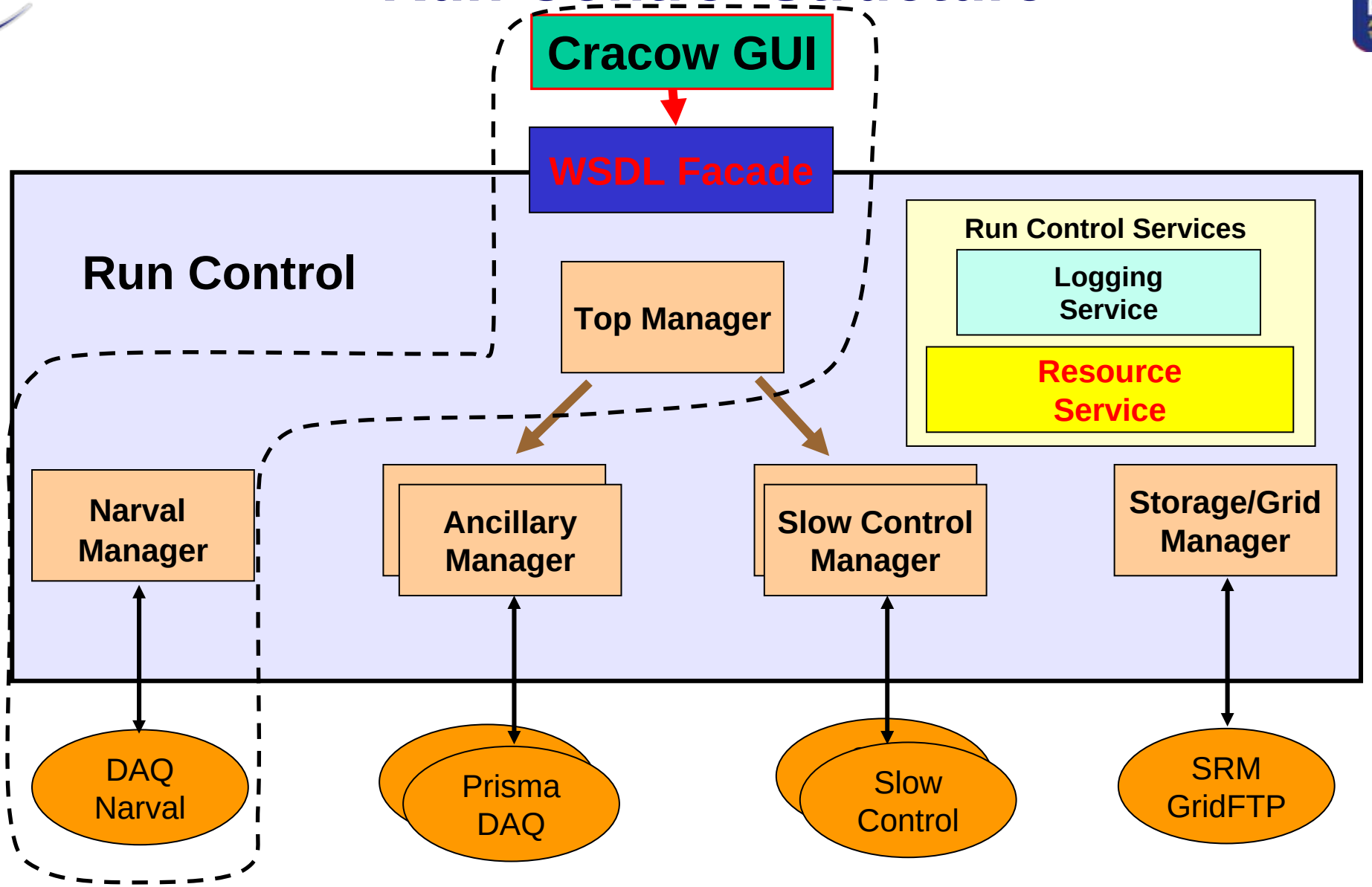
- From the Run Control point of view the Slow Control is a “Standalone System”, providing an interface for:
  - Send commands (Setup, Start, etc.);
  - Retrieve the Subsystem State;
  - Retrieve Errors and values to monitor.

Agata Run Control

- The Agata Run Control System is based on the GRIDCC middleware (http://www.gridcc.org), adopted by the LHC CMS experiment
  - Log Collector: log4j messages
  - Resource Service: configuration management
  - Run Info Database: Run Numbers and information about the run
  - WSDL (Web Service Description Language) interface for GUIs
- Instrument Manager:
  - A "customizable" Java component running on a Tomcat servlet container;
  - State machine support;
  - Event driven behaviour
    - Events (commands from the operator, error notifications from the instruments, …) trigger the execution of custom control and monitor procedures;
- Instrument Managers are organized in a control hierarchy

- In Agata we have a two-level hierarchy of Instrument Managers.

Run Control Structure



Slow Control Subsystems

- Agata Slow Control Subsystems:
  - Digitizers
  - Core and Segment Mezzanines
  - Carrier boards
  - GTS

- The Slow Control Subsystems are “Standalone Systems”, developed in different institutes with different technologies:
  - They have their own configuration DB/files, their own GUI, etc.

- The Run Control Structure foresees one Instrument Manager for each Slow Control Subsystem

NEW: Global Slow Control

- NEW (yesterday):
  - It was decided to develop a “Global Slow Control Supervisor “ that integrates the slow controls of the four subsystems.
  - The “Global Slow Control” will take care of the dependencies between the slow control subsystems and will coordinate the configuration and setup procedures
  - A Run Control Instrument Manager (“Slow Control Manager”) will interact with it

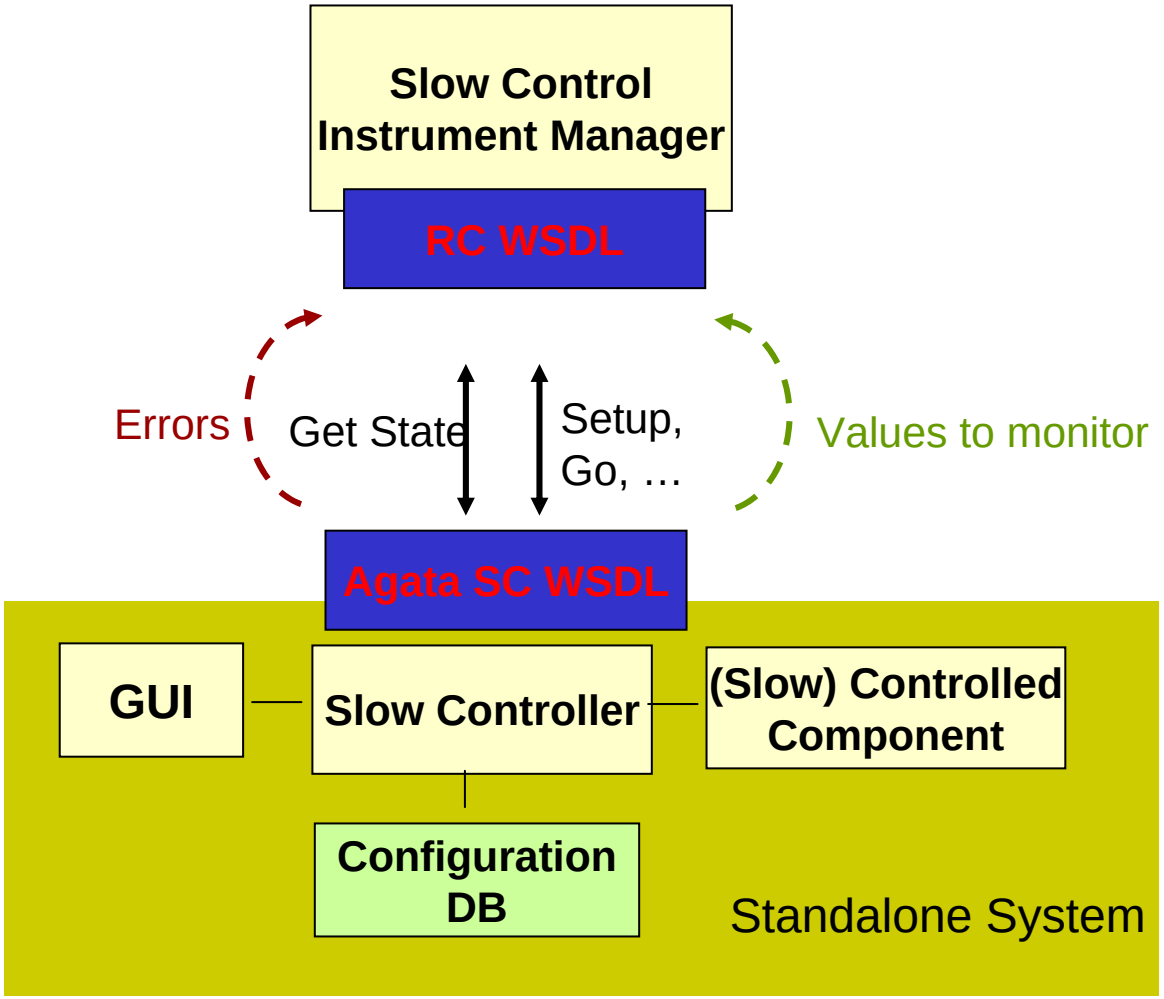
Agata Slow Control WSDL interface

- In order to simplify and homogenize the Run Control - Slow Control integration it was decided to:
  - Use SOAP/HTTP as a communication protocol
  - Each Slow Control Subsystem exports to the Run Control a “high level” WSDL (Web Service Description Language) interface
  - Use the same WSDL interface for all the Slow Control Subsystems

- The WSDL interface defined and implemented by V.Pucknell for the digitizers has been chosen (Agata Slow Control WSDL – version 1)

- A second version has been recently defined.

Run Control & Slow Control WSDL



Agata Slow Control WSDL

- Agata Slow Control WSDL (V. Pucknell): version 2

int DoReset (char *parameter, int *rc);
int DoSetUp (char *parameter, int *rc);
int DoStop (char *parameter, int *rc);
int DoGo (char *parameter, int *rc);
int DoOption (char *parameter, int *rc); **// new in version 2**

int GetState (int *rc, int *code, char *state, char *reason);
int GetCounters (int *rc, char *counters);
int GetRates (int *rc, char *rates);
int GetOption (char *parameter, int *rc, char *options); **// new in version 2**

```
<definitions name="DataAcquisitionControlServer"
      targetNamespace="http://npg.dl.ac.uk:8015/DataAcquisitionControlServer.wsdl">
…
      <message name="GetStateRequest">
      </message>
      <message name="GetStateResponse">
      <part name="ResponseCode" type="xsd:int"/>
      <part name="Code" type="xsd:int"/>
      <part name="State" type="xsd:string"/>
      <part name="Reason" type="xsd:string"/>
      </message>
```

Status

- This WSDL (version 1) has been implemented for the digitizers, for the carrier, and in the ENX software (segment/core mezzanines).
  - Need to move to version 2
- No implementation for the GTS yet.

- A prototype Slow Control Instrument Manager (based on version 1) has been implemented and integrated in the Run Control hierarchy.
  - Preliminary tests with MIDAS (digitizers) and ENX performed

- The Run Control requirements (Run Number dispatching, Configuration Key mechanism) have been addressed in the Agata Slow Control WSDL version 2
  - No more extensions and/or modifications should be necessary

- NEW: ENX will be probably used to develop the new “Global Slow Control Supervisor”
  - Run Control-Slow Control interaction will be based on the WSDL version 2 implemented in ENX

Summary

- The Run Control and Slow Control integration is in progress
  - SOAP/HTTP protocol
  - Agata Slow Control WSDL version 2 defined
  - Agata Slow Control WSDL version 1 implemented by some Slow Control subsystems (digitizers, carriers, ENX)
  - Prototype Run Control Instrument Manager (based on version 1) ready

- Need to move WSDL version 2

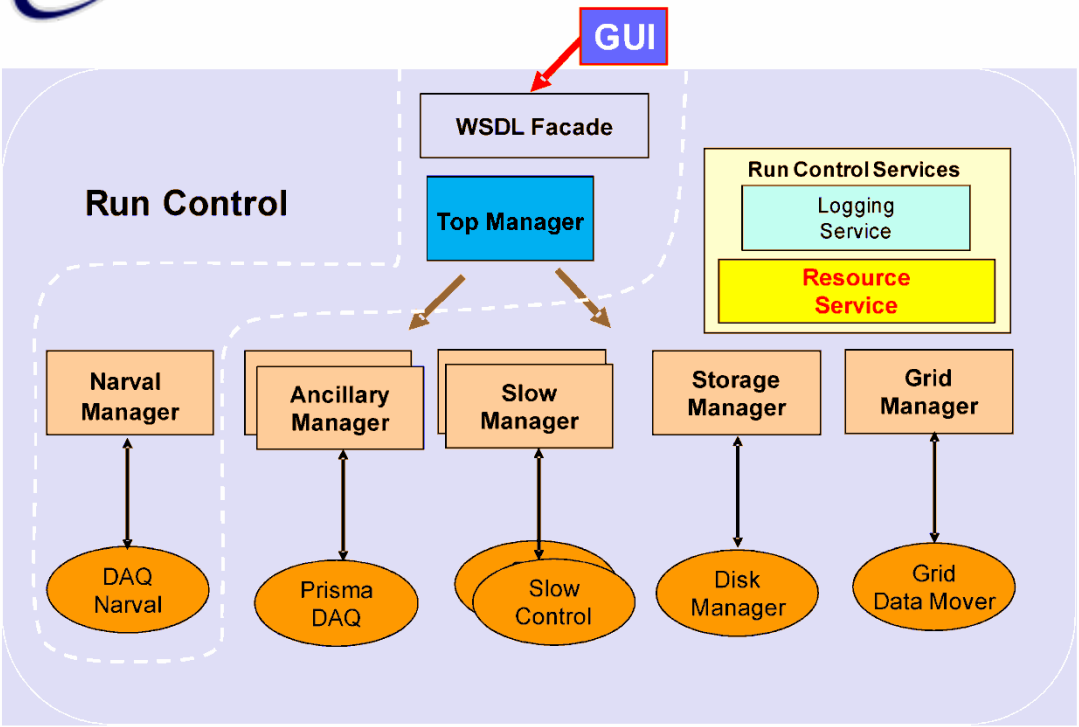
- NEW: A Global Slow Control Supervisor is going to be developed.
  - It will implement the Agata Slow Control WSDL Interface
  - The Run Control will interface to it

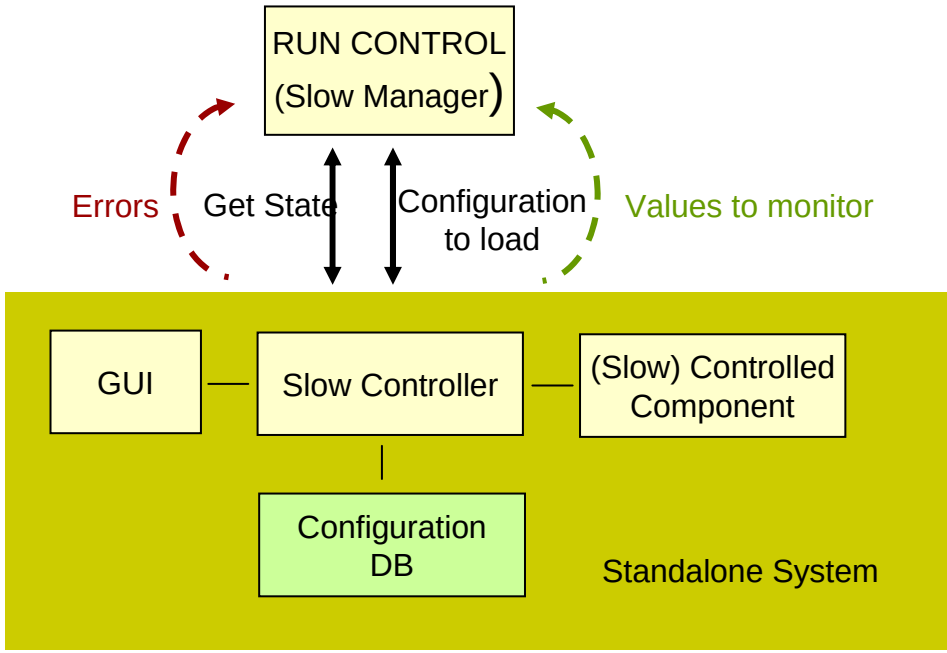
The end

Thank you

Spare

Slow Control and Overall Configuration





- Each Run Control Subsystem (Narval, Prisma, Slow Controls) has its own way to manage configurations
  - The Run Control uses the **Resource Service DB** for its own configuration
    - configurations of the controlled components might also be stored; a configuration change in a subsytems force to store a new config (for all Agata) in the RS DB
- How the shifter chooses the overall Agata configuration for a data taking?
- How the Run Control manages the configurations?

Configuration

- The concept of Global Configuration Key (a Name) is used
  - A limited amount of Keys will be available to the shifter in the main GUI
    - i.e: TRIPLE_CLUSTER, TEST_EVB, FEB_SOURCE_TEST, …
  - Subsystem experts associate the Global Configuration Keys to the proper configuration to load
  - The Run Control dispatches the Configuration Key received by the GUI to the (SC) subsystems (Configure Command)
  - Subsystems translate the Key in the proper configuration to load
  - Agata Slow Control WSDL has been extended in order to have the Global Configuration Key as input parameter at setup time