Software Training: Random Notes

Murrough Landon – 1 December 2003

`http://www.hep.ph.qmul.ac.uk/~landon/talks`

Overview

- Jargon
- ATLAS DAQ Overview
- Online Software Overview
- L1Calo Software Overview
- Development Environment

Jargon (1)

Jargon	System	Explanation
CondDB	On/Offline	Conditions Database: data used offline, eg calibrations
ConfDB	Online	Configuration Database: describes Online setup (via OKS)
CORBA	Online	COmmon Object Request Broker: protocol for objects in one process to invoke methods of objects in other processes on other nodes
DAL	Online	Data Access Library: C++/Java software layer giving more convenient access to database data than just using underlying OKS
DFM	Dataflow	DataFlow Manager: manages connections from ROS to SFI

Jargon (2)

Jargon	System	Explanation
ILU	Online	??: Implementation of CORBA used by Online (to be replaced)
IPC	Online	InterProcess Communication: Online package (based in CORBA/ILU) to handle communication between objects on different nodes, also implements partitioning
IS	Online	Information Service: Online package for exchanging data
L2PU	Trigger	Level2 Processing Unit: PC running Level2 trigger algorithm
L2SV	Trigger	Level2 SuperVisor: sends RoIs to L2PUs (similar to SFI/SFO)
MRS	Online	Message Reporting System: Online package for reporting errors
OKS	Online	Object Kernel System: XML file database used by Online software

Jargon (3)

Jargon	System	Explanation
PMG	Online	Process ManaGer: Online package for starting/stopping processes
RC	Online	Run Control
ROB	Dataflow	ReadOut Buffer: memory sitting in the ROS
ROBin	Dataflow	ROB input: custom PCI card hosting ROBs, sitting in ROS PC
ROD	Detectors	ReadOut Driver: collect data from frontends, sends to ROS
RoI	Trigger	Region of Interest: pointer to Level1 trigger object in eta-phi space, sent to Level2
RoIB	Trigger	Region of Interest Builder: custom hardware, collects RoIs from calo and muon triggers and CTP, sends to L2SV

Jargon (4)

Jargon	System	Explanation
ROL	Dataflow	ReadOut Link: SLink connection from ROD to ROS
ROS	Dataflow	ReadOut (Sub)System: PC collecting data from RODs, sends to Level2 and SFI
SFI	Dataflow	SubFarm Input: PC at input to the Event Filter, receives full events, sends to processing task on Event Filter “worker node”
SFO	Dataflow	SubFarm Output: PC at output of the Event Filter, receives events passing the trigger, sends to mass storage
SLink	Dataflow	Simple Link?: data protocol used on ReadOut Links from ROD to ROS

ATLAS DAQ Overview (1)

Things you probably know already...

- Huge number of channels in several subdetectors
- Front end (on detector) electronics read out via RODs in USA15
- Dataflow: RODs to ROS to SFI/Event Filter/SFO to mass storage
- Three level trigger: level 1 (hardware), level 2 and event filter software in large processor farms
- Processor farms grouped into subfarms controlled by one PC
- Large number (200-500?) of PCs and VME crates to be configured, controlled and monitored together

ATLAS DAQ Overview (2)

Three main DAQ areas

- Dataflow: moving data around fast
- Online: controlling and monitoring the DAQ system
- Detector Control System (DCS): monitor the detector (gas, HV, LV, etc)

ATLAS Online Software Overview

Control a large distributed system

- Need to bring several hundred distributed systems from cold start to taking ATLAS physics data
- Databases to describe the whole (or partial) configurations: OKS, confdb, DALs
- Start and stop many processes on many different nodes: PMG
- Lots of communication between (objects in) different processes: CORBA implementation (ILU) plus higher level packages (eg IPC, IS)
- Synchronised actions across the whole system: run control state model
- Central collection and filtering of error messages: MRS
- User interface to control the whole system: IGUI
- Facilities for monitoring: IS, Monitoring, OH
- Facilities for testing the system: Test manager, DVS

Online Software (1)

Databases

- No hard coded configuration information - databases for everything!
- ConfDB describes hardware (crates, modules), software (what programs are available and how to run them) and overall partition/configuration (choice of which of the available hardware and software is required)
- Implemented as an object oriented database via the OKS package and associated higher level " data access libraries"
- Common "schema" extended by L1Calo (and others) for specific needs
- Data is stored in XML files, can be edited via GUIs (or by hand)
- Hardware configuration can contain some static settings
- More variable data (calibrations, trigger menu) is considered to be separate – conditions database

Online Software (2)

Interprocess Communication

- Basic aspect of almost all Online software
- Uses the CORBA standard: one object can call methods of other objects anywhere in a distributed system
- Requires a basic naming/directory system to be setup: IPC_REF_FILE
- This naming system allows different "partitions" to run in parallel (eg L1Calo can take data while the DAQ is also running other detectors separately)
- Online software hides details of CORBA implementation (soon to be changed) from the rest of the software via an intermediate package: IPC

Online Software (3)

Information Service

- Aim: allow processes to share data
- Operates like a dictionary: look up a name, get some information
- The data can be any object (which inherits from IS base classes)
- Processes can ask to be automatically notified (callback) if some information they are interested in is updated by another process

Online Software (4)

Message Reporting Service

- Aim: central handling of error messages
- Any process in the system sends out a message
- One central server process logs them, can filter them
- Other processes can ask to be notified of all messages (or some subset – eg only WARNING, ERROR, FATAL)

Online Software (5)

Process Management

- Aim: operating system independent way to start, stop and monitor processes on different nodes
- First need a "PMG agent" running on each node (started via ssh)
- After that any process can ask the agent to start or stop other processes
- If a process dies the agent can notify the parent process
- Some limitations on the environment that the child process sees (need to declare environment variables in the database)
- Standard output and standard error go to log files
  $PMG_PROCESS_LOGS_PATH set in $HOME/.onlinerc

Online Software (6)

Run Control

- Aim: synchronise startup and shutdown of many different systems via a state model with fixed transitions
- Present model (to change soon) has transitions: Load, Configure, Start, Pause, Resume, Stop, Unconfigure, Unload
- Typically one “run controller” process controls one VME crate or one ROS or one level2 or event filter subfarm
- Run controllers can be organised in hierarchies

Online Software (7)

Integrated Graphical User Interface (IGUI)

- Public face of the Online software: how the user interacts with the whole Trigger/DAQ system
- Buttons to start/stop the system, error message display, many panels for setting run parameters, dispaying system status

Online Software (8)

Development Environment

- The Online have a detailed “software process”: a formalism for how software should be developed
- Basic idea: think first, document the ideas, code later, then test
- For each package or new development provide a “requirements document” (NB not the CMT requirements file!) listing the features the package has to implement. Get feedback (review)
- Next think how to implement them and write down the “high level design” in a short document. Get feedback (review)
- Next write the code and test it (should write and get feedback on a “test plan”)
- Provide a user guide
- Regular releases of the complete set of Online software packages

Online Software (2)

Finding out more

- Website: http://cern.ch/atlas-onlsw
- For user guides to components, look under components
- Download new releases and patches to current release

L1Calo Software (1)

Overview

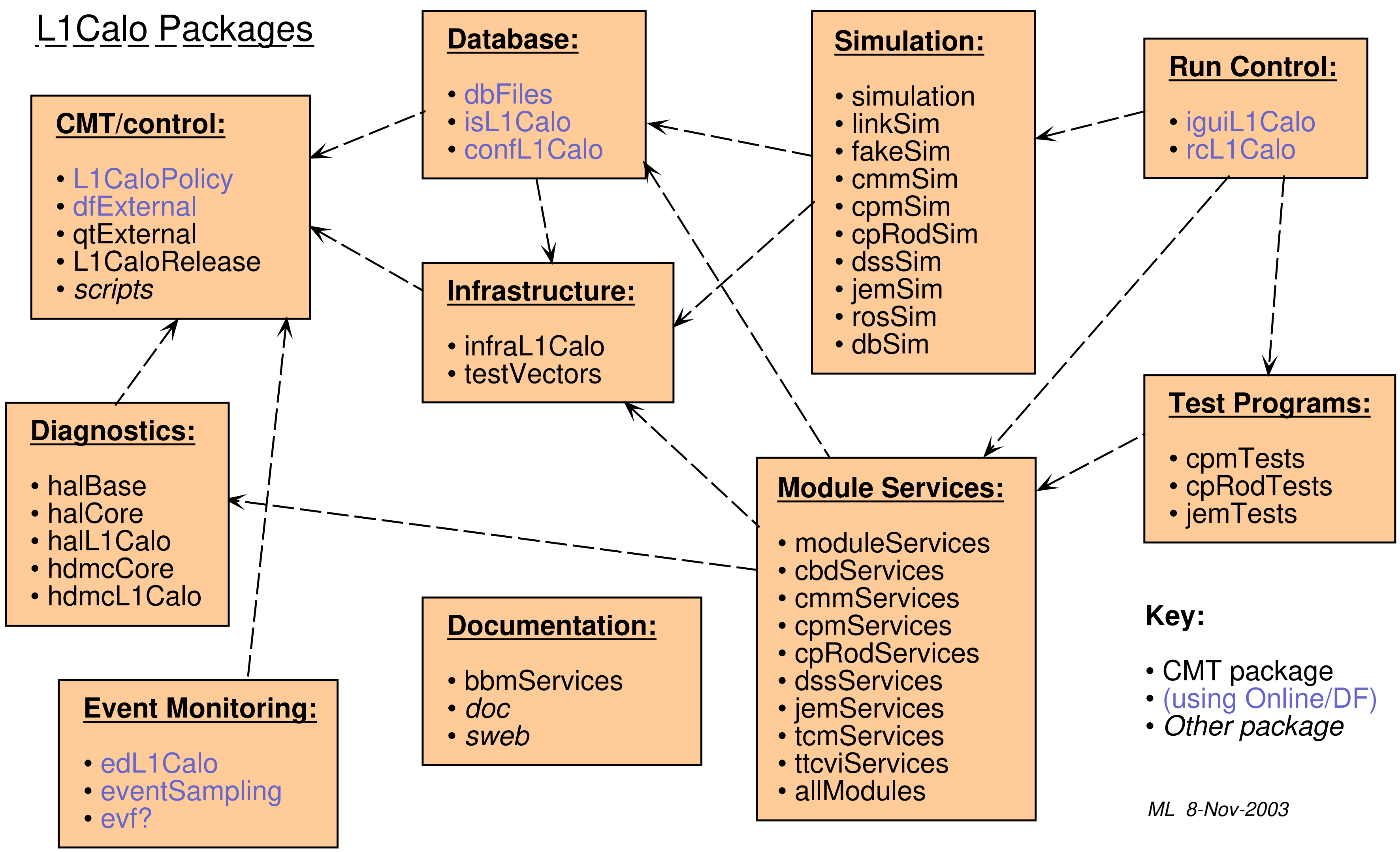
- Software for testing, controlling, configuring, monitoring and simulating the calorimeter trigger hardware
- (Offline software in ATHENA framework is outside this scope)
- Operating in the context of the ATLAS Online software
- Tried to follow (a light version) of the Online software development process
- Overall requirements document (SW note 7), some user guides and design suggestions, not so many reviews

L1Calo Software (2)

Main Components

- Low level (detailed) description of the modules: HDMC (also provides GUIs for diagnostics)
- Higher level functions of the modules, in particular the interface to the run control state machine: Module Services
- Simulation of the system
- Interface with and customisation of the Online software: eg database, run controllers, IGUI panels
- Some common infrastructure, scripts, etc

L1Calo Components and Packages



L1Calo Software (3)

HDMC

- Two branches: description of hardware (Parts) GUIs to control them
- Parts are organised in hierarchies: Bus contains Modules which contain SubModules, Registers and Memories etc
- Hierarchy maintained dynamically and stored/read from files by PartManager
- Bit fields in registers described by a configuration file, can be used to make Register subclasses via `regbuild`
- NB register layouts can be dynamically variable – this feature was used by Heidelberg modules in the past - does the PPM still need it (would like to remove it for easier maintenance)?

L1Calo Software (4)

Module Services

- Set of packages providing standard access to the hardware
- Sits between HDMC layer and the run control
- Base class DaqModule: subclass of Module and implements abstract DaqInterface
- Each module has to subclass DaqModule and implement the run control methods: load, configure, start, pause, resume, stop, un/deconfigure, unload as well as some others like initModule, readOut, updateModuleStatus
- Implementing DaqModule subclass with HDMC Parts is expected (but not required)

L1Calo Software (5)

Module Services (continued)

- Typical approach: create custom DaqSubModule subclasses for major components such as FPGAs and ASICs
- These may contain (lists of) Memories and `regbuild` generated register classes
- DaqSubModules and SubModuleLists also implement the run control methods so the DaqModule can delegate all actions to its components
- Doxygen reference documentation (for moduleServices package) and example package (bbmServices)

L1Calo Software (6)

Database

- Extension of Online database packages – which are changing :-(
- Customised to add descriptions of our modules and their parameters
- Also their calibration data and trigger menu (should change)
- Other L1Calo additions include descriptions of cabling, firmware test vectors and sets of “run types”
- Design guide (very out of date, SW note 5), User guide (SW note 12) and doxygen reference docs (confL1Calo package)

L1Calo Software (7)

Simulation

- Set of packages providing detailed simulation of the hardware
- Generic base classes (simulation package), common L1Calo classes (linkSim), individual module simulations (xxxSim and fakeSim)
- Complete simulation configured from the same database as used to load the hardware (dbSim package)
- User guide (SW note 11) and doxygen reference docs (for simulation package)

L1Calo Software (8)

Run Control

- Implements a “skeleton” provided by the Online - customised to send state transition commands to our modules (and separately to the simulation)
- NB this is all due to change too with move to “ROD crate DAQ”
- At the moment we have one generic crate controller that can be used to control all the modules in any of our different crate types after reading the configuration from the database (ROD crate DAQ controller will handle any subdetector)
- Not much syncrhonisation between crates: we have to be careful what each module does in each step – follow common standard

L1Calo Software (9)

Run States (changing slightly next year)

- Load: reset modules, reload firmware (if needed), set basic timing (eg TTCrx clock phase)
- Configure: load most configuration parameters, eg most register settings, calibration data, test vectors (if any) trigger menu, etc
  Simulation run controller runs the test vector generation and the simulation at this point - controllers synchronised so this happens before loading resulting test vectors!
- Start: not much apart from some TTC commands
  (run controller itself starts “kicker” process)
- Stop: not much – maybe other TTC commands, stop “kicker”
- Unconfigure, Unload: nothing for the modules
- Periodically (every few seconds?) run controller calls updateModuleStatus to read status from modules and publishes it in the Information Service

L1Calo Software (99)

Software Development Process

- CVS repository at RAL (may move to CERN some day)
- Roughly follow ATLAS standard C++/Java coding conventions
  http://cern.ch/Atlas/GROUPS/SOFTWARE/OO/Development/qa/General
- Use CMT: common ATLAS tool for building sets of separate software packages with complex interdependencies
- There are several working models for using CMT – we follow that developed by the Online group (bit different from DataFlow and Offline)
- Online model via http://atddoc.cern.ch/cmt
  our suggestions (out of date SW note 8)
- Useful scripts: http://www.hep.ph.qmul.ac.uk/l1calo/sweb/scripts
- Separate work into (fairly) small packages: generally one person per package
  http://www.hep.ph.qmul.ac.uk/l1calo/sweb/packages

L1Calo Software (99)

Software Development Process

- Encourage you to commit (compilable) code to CVS. Also to tag packages from time to time and especially before/after major changes
- Use Doxygen style comments for reference documentation
  http://www.doxygen.org
- Nightly builds of all L1Calo software (and documentation)
- Website http://www.hep.ph.qmul.ac.uk/l1calo/sweb