PhD Thesis

HIERARCHICAL CONTROL OF THE ATLAS EXPERIMENT

Àlex Barriuso Poy

URV
Universitat Rovira i Virgili
Departament d'Enginyeria Electrònica,
Elèctrica i Automàtica

CERN
European Organization for
Nuclear Research
Physics Department

Co-directors: Dr. Helfried J. Burckhart
Dr. Eduard Llobet Valero

Genève
May 2007

CERN-THESIS-2007-037
14/05/2007

Als meus pares.
A la memòria del meu avi.

Acknowledgements

First of all, I would like to express my sincere acknowledgement to my joint supervisors. At CERN, I thank Dr. Helfried Burkhart for giving me the opportunity of working for the ATLAS DCS, his committed guidance, support and patience over these last three years. At the University Rovira i Virgili, I must thank Dr. Eduard Llobet Valero, my BEng, MEng, and now PhD co-supervisor for his constant help and guidance over the years.

I would also like to thank my colleagues Jim Cook, Slava Khomoutnikov, Slava Filimonov, Sebastien Franz, Stefan Schlenker and Fernando Varela for the many things I've learnt working with them. It has been really a pleasure to be a member of this team.

For the many fruitful discussions and good human relation I would also like to thank all the members of the different DCS sub-detectors groups and, in particular, I thank (from the inner to the outer part of the detector) Joachim Schultes, Pamela Ferrari, Anna Sfyrla, Heidi Sandaker, Zbigniew Hajduk, Elzbieta Banas, Sergey Chekulaev, Leonardo Carminati, Michail Bachtis, Tarem Shlomit, Charilaos Tsarouchas and Giulio Aielli.

I would also like to thank the JCOP Framework working group and, in special, Clara Gaspar for the technical support and competent advice. Your work made mine easier and much effective.

For their patience in reading this thesis, I would like to thank Stefan Schlenker, Helfried Burkhart and Sergey Chekulaev. Their selfless and rigorous work help me to improve this thesis.

I would also like to underline how lucky I feel for having had the opportunity to meet countless new good friends during these last four years in Geneva. Andres, Bobo, Cova, Dave, Defi, Esther, Eva, Federico, Ivan, Juan, Luis, Manolo, Mirko, Montse, Oscar, Richar, Rocio, Toni, Valeria and many others, it was a great time, isn't?

I would also like to thank Gigi and Ari for adopting me within an ambience where the warm of a family is omnipresent.

També voldria expressar el meu agraïment a l'amic Eloi i als seus pares Antoni i Angelina, per la seva hospitalitat. A Farena vaig trobar un indret on l'escriptura d'aquesta tesi va esdevenir una tasca complaent.

Hi ha sentiments difícils d'expressar amb paraules: a la meva família pel seu constant afecte i recolzament, menció concreta al meu germà Iñaki per l'esperit de superació que m'ha contagiat, a tots no puc dir-los res més que moltíssimes gràcies.

Per concludere, vorrei ringraziare Cristina per rendere la mia vita un po' più incasinata ma molto più felice. Ti voglio bene.

Resum

Els sistemes de control emprats en nous experiments de física d'altes energies són cada dia més complexos a conseqüència de la mida, volum d'informació i complexitat inherent a la instrumentació del detectors. En concret, aquest fet resulta visible en el cas de l'experiment ATLAS (A Toroidal LHC ApparatuS) situat dins del nou accelerador de partícules LHC (Large Hadron Collider) al CERN. ATLAS és el detector de partícules més gran mai construït fruit d'una col·laboració internacional on participen més de 150 instituts i laboratoris d'arreu del món. L'experiment estudia col·lisions protó-protó i, seguint l'estructura clàssica d'un detector de partícules, es composa d'una sèrie de sub-detectors especialitzats i d'un sistema d'imants superconductors que confereixen camp magnètic a l'experiment. A la part més interna de ATLAS, 3 tipus de detectors reconstrueixen les trajectòries que segueixen les partícules amb càrrega just després d'una col·lisió. A la part central, trobem 2 sistemes de calorímetres que absorbeixen i mesuren l'energia de quasi totes aquelles partícules que travessen. Finalment, a la part més externa, 4 tipus de detectors estudien les trajectòries d'un tipus de partícula concreta capaç de no ésser detectada ni absorbida en cap dels detectors precedents, el muó. Engalzant la informació extreta de cadascun d'aquests sub-detectors cada partícula pot ser completament caracteritzada i l'esdeveniment produït per la col·lisió pot ser reconstruït amb gran precisió.

Concernent l'operació de ATLAS, existeixen dos sistemes integradors principals. Per una banda, el sistema DAQ (Data AdQuisition) realitza l'adquisició de dades per els conseqüents estudis de física. Per altra banda, el DCS (Detector Control System) s'encarrega d' assegurar la coherent operació de tot l'experiment. Tot i ser dos sistemes independents, ambdós es complementen. Mentre un gestiona les dades utilitzades per als consegüents estudis de física, l'altre gestiona tota la infrastructura relacionada amb l'estat operacional del detector assegurant així la correcta extracció de informació.

El DCS, principal argument d'aquesta tesi, supervisa tot el hardware dins al complex de l'experiment incloent tots els serveis dels sub-detectors (ex. alta i baixa tensió, refrigeració, etc.) i la infrastructura general de l'experiment (ex. condicions ambientals). El DCS també és la interfície amb els sistemes externs a l'experiment com per exemple els serveis tècnics CERN (ex. ventilació o electricitat) o, encara més crucial, amb l'accelerador LHC o el DAQ de ATLAS. En total, al voltant de 200.000 canals d'informació seran supervisats en tot moment per el DCS.

Un dels principals problemes existents en anteriors experiments era la manca d'estanderdització en moltes àrees. Per exemple, degut a l'escenari tècnic de l'època, els sistemes de control a l'era LEP (1989-2000) utilitzaven diferents llenguatges de programació, diferents protocols de comunicació i hardware 'fet a mida'. Com a conseqüència, el desenvolupament i manteniment del DCS era en molts casos una tasca difícil. Amb la intenció de solventar els problemes del passat, el projecte JCOP va ser creat al CERN a finals de 1997. Els diferents sub-detectors de ATLAS (així com dels 3 altres principals experiments del LHC) estan composats de múltiples equips de persones treballant en paral·lel. L'objectiu principal del JCOP és treballar

en comú per reduir duplicitat i, al mateix temps, facilitar la integració i futur manteniment dels experiments. D'aquesta manera, components sovint utilitzats per al control de plantes industrials com PLCs, 'fieldbuses', el protocol OPC o SCADA han estat instaurats i són utilitzats amb èxit als experiments. Al mateix temps, el JCOP combina els productes comercials existents amb elements hardware i software específicament creats per al seu ús dins el món del control d'experiments de física d'altes energies. Aquest és el cas del software anomenat FSM (Finite State Machine).

El modelatge i integració dels molts dispositius distribuïts que coexisteixen al DCS es realitza utilitzant la FSM. D'aquesta manera, el control s'estableix mitjançant una gran quantitat d'entitats software distribuïdes, autònomes i cooperatives que són organitzades jeràrquicament i segueixen una lògica de màquines finites d'estats. Aquesta tecnologia, emprada per a la integració del control de ATLAS, s'anomena a la literatura 'agents intel·ligents'. Tot i demostrar ser una tecnologia molt útil per la integració de grans sistemes de control com ATLAS, poques solucions comercials existeixen per l'automatització de plantes industrials o manufactura. La raó principal d'aquest fet no deu ser relacionada amb limitacions tecnològiques sinó més aviat amb la naturalesa del mercat. Mentre el control a experiments de física d'altes energies evoluciona constantment amb nous procediments per l'operació i solucions radicalment noves, el mercat de l'automatització industrial és històricament més lent a acceptar canvis: operadors i experts tenen procediments de treball establerts associats a un procés industrial estable. A més, els costos econòmics associats a la re-enginyeria de una planta en producció, fa del control industrial un sector conservador. Tot i això, l'evolució de la tecnologia de la informació afegeix dia a dia funcionalitat a plantes industrials existents encaminant la recerca cap a 'agents intel·ligents' que proporcionen un disseny més modular i distribuït dels processos.

L'eina FSM combina dues tecnologies principals: SMI++ (State Manager Interface toolkit) i un producte SCADA comercial. SMI++ (escrit en C++) ja ha estat utilitzat amb èxit en dos experiments de física d'altes energies anteriors a ATLAS proveint la següent funcionalitat: un llenguatge orientat a objectes, una lògica de màquina finita d'estats, un sistema expert basat en regles, i un protocol de comunicació independent de la plataforma utilitzada. Aquesta funcionalitat s'aplica doncs a tots els nivells d'operació de l'experiment. D'aquesta manera, el control de ATLAS es modela mitjançant objectes software que es comporten com a màquines finites d'estats i representen diferents nivells d'abstracció de l'experiment (ex. des d'una vàlvula d'un sistema de refrigeració fins a tot ATLAS). Així i, basant-se en regles establertes i acurades inter-connexions que organitzen els objectes jeràrquicament, s'assoleix l'automatització global de l'experiment.

Aquesta tesi presenta la integració del ATLAS DCS dins una jerarquia de control seguint la segmentació natural de l'experiment en sub-detectors i sub-sistemes. La integració final dels molts sistemes que formen el DCS a ATLAS inclou tasques com: l'organització del software de control, la identificació de models dels processos, l'automatització de processos, la detecció d'errors, la sincronització amb DAQ, i la interfície amb l'usuari.

Tot i que l'experiència adquirida al passat amb la utilització de SMI++ és bon punt de partença per al disseny de la jerarquia de control de ATLAS, nous requisits han aparegut degut a la complexitat i mida de l'experiment. Així, l'escalabilitat de l'eina ha estat estudiada per afrontar el fet de què la jerarquia de control final a ATLAS serà centenars de cops més gran que cap dels dos antecedents existents. Una solució comú per a tots els sistemes que formen el DCS ha estat creada amb el principal objectiu d'assolir una certa homogeneïtat entre les diferents parts. Així, una arquitectura basada en 3 nivells funcionals organitza els sistemes pertanyents als 12 sub-detectors de l'experiment. Seguint aquesta arquitectura, les diferents funcions i parts del DCS han estat modelades amb una 'granularitat' similar entre sub-detectors, la qual cosa, ens ha portat a l'obtenció de jerarquies de control isomorfes.

La detecció, monitorització i diagnòstic d'errors és una part essencial per l'operació i coordinació de tasques de qualsevol experiment de física d'altes energies o planta industrial. La presència d'errors al sistema distorsiona l'operació i pot invalidar els càlculs realitzats per a la recerca de física. Per aquest motiu, una estratègia estàndard i una interfície estàndard amb l'usuari han estat definides donant èmfasi a la ràpida detecció, monitorització i diagnòstic dels errors basant-se en un mecanisme dinàmic de tractament d'errors. Aquests nou mecanisme es basa en la creació de dos camins de comunicació (o jerarquies paral·leles) que, al mateix temps que tracten els errors, donen una descripció més clara de les condicions d'operació de l'experiment. Així, un dels camins de comunicació està poblat per objectes dedicats a la detecció i anàlisi dels errors, mentre a l'altre, els objectes comanden l'operació de l'experiment. Aquests dos camins paral·lels cooperen i contenen la lògica que descriu l'automatització de processos al DCS. Així, els diferents objectes segueixen unes màquines finites de estats preestablertes per ATLAS que faciliten la comprensió i futur desenvolupament del DCS. A més, el fet de què l'estratègia proposada agrupi i resumi els errors d'una forma jeràrquica, facilita notablement l'anàlisi d'aquests errors en un sistema de la mida d'ATLAS. L'estratègia proposada, modular i distribuïda, ha estat validada mitjançant nombrosos tests. El resultat ha estat una substancial millora en la funcionalitat mantenint, al mateix temps, una correcta gestió dels recursos existents. Aquesta estratègia ha estat implementada amb èxit i constitueix l'estàndard emprat a ATLAS per a la creació de la jerarquia de control.

Durant l'operació de l'experiment, el DCS s'ha de sincronitzar amb els sistema DAQ a càrrec del procés de presa de dades per als conseqüents estudis de física. L'automatització de processos d'ambdós sistemes, DAQ i DCS, segueixen una lògica similar basada en una jerarquia de màquines finites d'estats (similituds i diferències han estat identificades i presentades). Tot i això, la interacció entre els dos principals sistemes integradors de ATLAS ha estat fins el moment limitada, però aproximant-se a l'inici d'operacions, esdevé cada dia més important. Així, un mecanisme de sincronització que estableix connexions entre els diferents segments dels sistema DAQ i la jerarquia de control del DCS ha estat desenvolupat. La solució adoptada insereix automàticament objectes SMI++ dins la jerarquia de control del DCS. Aquests objectes permeten a les aplicacions del DAQ comandar diferents seccions del DCS d'una forma independent i transparent. Així, la posta en marxa de diferents parts del detector pot ser planificada per DAQ depenent dels diferents estudis de física que es vulguin realitzar. Al mateix temps, el mecanisme no permet prendre dades per física quan una part del detector funciona d'una forma incorrecta evitant així l'extracció d'informació corrupta mentre l'experiment torna a un estat segur. Un prototip que assoleix la sincronització dels dos sistemes ha estat implementat i validat, i ja està llest per a ésser utilitzat durant la integració dels sub-detectors.

Finalment, la interfície situada a la sala de control entre el DCS i l'usuari ha estat implementada. D'aquesta manera, es completa la integració de les diferents parts del DCS. Els principals reptes solventats durant les fases de disseny i desenvolupament de la interfície han estat: permetre a l'operador controlar un procés de la mida de ATLAS, permetre la integració i manteniment dels molts diferents 'displays' d'operador que pertanyen als diferents sub-detectors i, donar la possibilitat a l'operador de navegar ràpidament entre les diferents parts del DCS. Aquestes qüestions han estat solventades combinant la funcionalitat del sistema SCADA amb la eina FSM. La jerarquia de control es utilitzada per la interfície per estructurar d'una forma intuïtiva els diferent 'displays' que formen el DCS. Llavors, tenint en compte que cada node de la jerarquia representa una porció susceptible de ser controlada independentment, hem assignat a cada node un 'display' que conté la informació del seu nivell d'abstracció dins la jerarquia. Tota la funcionalitat representada dins la jerarquia de control és accessible dins els 'displays' SCADA mitjançant dispositius gràfics especialment implementats. Utilitzant aquest dispositius gràfics, per una banda possibilitem que

els diferents 'displays' s'assimilin en la seva forma, i així, facilitem la comprensió i utilització de la interfície per part del usuari. Per altra banda, els estats, transicions i accions que han estat definits per els objectes SMI++ són fàcilment visibles dins la interfície. D'aquesta manera, en cas de una possible evolució del DCS, el desenvolupament necessari per adequar la interfície es redueix notablement. A més, un mecanisme de navegació ha estat desenvolupat dins la interfície fent accessible a l'operador ràpidament qualsevol sistema dins la jerarquia. La jerarquia paral·lela dedicada al tractament d'errors també és utilitzada dins la interfície per filtrar errors i accedir als sistemes en problemes de una manera eficient. La interfície és suficientment modular i flexible, permet ésser utilitzada en nous escenaris d'operació, resol les necessitats de diferents tipus d'usuaris i facilita el manteniment durant la llarga vida de l'experiment que es preveu fins a 20 anys. La consola està sent utilitzada des de ja fa uns mesos i actualment totes les jerarquies dels sub-detectors estan sent integrades.

Abstract

Control systems at High Energy Physics (HEP) experiments are becoming increasingly complex mainly due to the size, complexity and data volume associated to the front-end instrumentation. In particular, this becomes visible for the ATLAS experiment at the LHC accelerator at CERN. ATLAS will be the largest particle detector ever built, result of an international collaboration of more than 150 institutes. The experiment is composed of 9 different specialized sub-detectors that perform different tasks and have different requirements for operation. The system in charge of the safe and coherent operation of the whole experiment is called Detector Control System (DCS).

This thesis presents the integration of the ATLAS DCS into a global control tree following the natural segmentation of the experiment into sub-detectors and smaller sub-systems. The integration of the many different systems composing the DCS includes issues such as: back-end organization, process model identification, fault detection, synchronization with external systems, automation of processes and supervisory control.

Distributed control modeling is applied to the widely distributed devices that coexist in ATLAS. Thus, control is achieved by means of many distributed, autonomous and co-operative entities that are hierarchically organized and follow a finite-state machine logic. The key to integration of these systems lies in the so called Finite State Machine tool (FSM), which is based on two main enabling technologies: a SCADA product, and the State Manager Interface (SMI++) toolkit. The SMI++ toolkit has been already used with success in two previous HEP experiments providing functionality such as: an object-oriented language, a finite-state machine logic, an interface to develop expert systems, and a platform-independent communication protocol. This functionality is then used at all levels of the experiment operation process, ranging from the overall supervision down to device integration, enabling the overall sequencing and automation of the experiment.

Although the experience gained in the past is an important input for the design of the detector's control hierarchy, further requirements arose due to the complexity and size of ATLAS. In total, around 200.000 channels will be supervised by the DCS and the final control tree will be hundreds of times bigger than any of the antecedents. Thus, in order to apply a hierarchical control model to the ATLAS DCS, a common approach has been proposed to ensure homogeneity between the large-scale distributed software ensembles of sub-detectors. A standard architecture and a human interface have been defined with emphasis on the early detection, monitoring and diagnosis of faults based on a dynamic fault-data mechanism. This mechanism relies on two parallel communication paths that manage the faults while providing a clear description of the detector conditions. The DCS information is split and handled by different types of SMI++ objects; whilst one path of objects manages the operational mode of the system, the other is dedicated to handle eventual faults. The proposed strategy has been validated through many different tests with positive results in both functionality and performance. This strategy has been successfully implemented and constitutes the ATLAS standard to build the global control tree.

During the operation of the experiment, the DCS, responsible for the detector operation, must be synchronized with the data acquisition system which is in charge of the physics data taking process. The interaction between both systems has so far been limited, but becomes increasingly important as the detector nears completion. A prototype implementation, ready to be used during the sub-detector integration, has achieved data reconciliation by mapping the different segments of the data acquisition system into the DCS control tree. The adopted solution allows the data acquisition control applications to command different DCS sections independently and prevents incorrect physics data taking caused by a failure in a detector part.

Finally, the human-machine interface presents and controls the DCS data in the ATLAS control room. The main challenges faced during the design and development phases were: how to support the operator in controlling this large system, how to maintain integration across many displays, and how to provide an effective navigation. These issues have been solved by combining the functionalities provided by both, the SCADA product and the FSM tool. The control hierarchy provides an intuitive structure for the organization of many different displays that are needed for the visualization of the experiment conditions. Each node in the tree represents a workspace that contains the functional information associated with its abstraction level within the hierarchy. By means of an effective navigation, any workspace of the control tree is accessible by the operator or detector expert within a common human interface layout. The interface is modular and flexible enough to be accommodated to new operational scenarios, fulfil the necessities of the different kind of users and facilitate the maintenance during the long lifetime of the detector of up to 20 years. The interface is in use since several months, and the sub-detector's control hierarchies, together with their associated displays, are currently being integrated into the common human-machine interface.

Outline

This thesis presents the work carried out by the author within the central group for controls of the ATLAS experiment. The basic goal of this thesis is twofold: first, to describe and implement the organization of the ATLAS DCS back-end in a distributed control hierarchy formed by autonomous but co-operative software objects; and second, to develop the corresponding graphical user interface. My contribution to the DCS development has been divided in several different projects at several stages of the construction. In total, they represent most of the main tasks needed when building the ATLAS DCS back-end automation and operation. This report is divided in three main parts:

1. **Introduction**. The first four chapters introduce the context of this thesis work going from a more abstract to a more concrete discussion.

   Chapter 1 gives a brief overview of high energy physics today, describes the CERN accelerator complex and presents the principle of modern-day particle detectors.

   Chapter 2 introduces the general infrastructure and the different specialized sub-detector systems of the ATLAS experiment which have to be supervised by the DCS. The aim of this chapter is to discuss the ATLAS detector in the context of the analogy to a large scale control 'plant'. At the end of this chapter the ATLAS data acquisition system and the envisaged overall control of the experiment are discussed. This chapter also introduces terminology and elements which are used subsequently when expanding the DCS and its hierarchical control.

   Chapter 3 describes the responsibilities, the architecture and the constituent parts of the ATLAS DCS. This chapter begins discussing the evolution occurred in controls during the transition from the Large Electron Positron (LEP) to the Large Hadron Collider (LHC) era. The description of the different DCS components includes: (1) front-end instrumentation, (2) communications, and (3) the back-end system running on PCs. Finally, a more general discussion about back-end controls outside the HEP world, listing major vendors and some proprietary integrated engineering platforms, motivates the generalization of control mechanisms for large scale projects in the industrial control market.

   Chapter 4 finishes the introductory part by presenting the Joint COntrol Project Finite State Machine (JCOP FSM), the main component for creating the control hierarchy of the ATLAS DCS. Complex systems are broken down into more simple units that are hierarchically controlled. Using this approach, the experiment will be decomposed and described in terms of software objects which behave following a finite state machine logic. These objects can represent concrete devices, such as a high-voltage crate, or abstract groups of this devices, like a sub-detector or a gas system. This chapter ends by first discussing the presence in the industry of commercial tools which could provide the functionality of the JCOP FSM and, second, presenting the antecedents of the tool in the HEP world.

2. **Design Phase**. Chapter 5 presents the first prototype of a small control tree implemented for the ATLAS DCS. During that work, design considerations for structuring the hierarchy were established. The aim of this chapter is to motivate all those implementations presented in the following chapters.

   In Chapter 6 the overall philosophy used to represent the sub-detectors, sub-systems and hardware components that constitute the hierarchical experiment control system is presented. This philosophy is much influenced by the results presented in the previous chapter. The main goal of this work is to support the developers, as well as to set common guidelines suited to the scale of ATLAS. These guidelines offer a new strategy to handle eventual faults that can appear on the experiment through the usage of the JCOP FSM. At the end, a validation test of the proposed strategy concerning performance is discussed. An overview of some of this work was presented at the ICALEPCS conference during spring 2005[1].

3. **Implementation**. Once the basic design considerations about the back-end organization have been described, the integration phase is detailed. This integration involves several tasks including process model identification and optimization, data reconciliation and fusion, fault detection and supervisory control. An internal note summarizes the overall DCS integration guidelines [2].

   The first part of Chapter 7 shows two examples of sub-detector process model identification integrated in a control tree. Both follow the standard architecture previously discussed in Chapter 6. The second part of Chapter 7 focusses on the data reconciliation of the two main integration systems of ATLAS, the DCS and the Trigger and Data AcQuisition system (TDAQ). A close interaction between these two systems is of prime importance for the coherent overall operation of the experiment. The result of this work was presented at the IECON conference during autumn 2006[3]. Moreover, additional information can be found in an ATLAS technical note[4].

   Chapter 8 presents the DCS top level human-machine interface. The top level interface displays and controls the information from the DCS. This information will be available both in the ATLAS control rooms and externally via the internet. This chapter introduces the available tools for the ATLAS DCS operation with emphasis on the ATLAS DCS Operator Interface (DCSOI), the main human-computer interface for the operation of the DCS. Part of this work has been presented at the SAAEI conference during autumn 2006 [5]. In addition, specific guidelines for building the complete back-end control hierarchy of ATLAS can be found in the ATLAS internal note [6].

Abbreviations and acronyms

ACR	ATLAS Control Room
AI	Artificial Intelligence
ALICE	A Large Ion Collider Experiment
API	Application Program Interface
ATLAS	A Toroidal LHC ApparatuS
BCM	Beam Condition Monitor
BE	Back-End
CAN	Controller Area Network
CERN	Conseil Européen pour la Recherche Nucléaire
CHSM	Current Hierarchical State Machine
CIC	Common Infrastructure Control
CMS	Compact Muon Solenoid
COM	Component Object Model
COTS	Components Off-The-Shelf
CRC	Cyclic Redundancy Check
CSC	Cathode Strip Chambers
CU	Control Unit
CV	Cooling and Ventilation
DCS	Detector Control System
DCSOI	DCS Operator Interface
DDC	DAQ-DCS Communication
DDC_DU	DDC Device Unit
DEN	Device Editor and Navigator
DIM	Distributed Information Management
DIP	Data Interchange Protocol
DP	Data Point (PVSS)
DPE	Data Point Element (PVSS)
DSS	Detector Stafety System
DU	Device Unit
EASY	Embedded Assembly SYstem
EF	Event Filter
ELMB	Embedded Local Monitor Board
EM	Electromagnetic Calorimeter
EPICS	Experimental Physics and Industrial Control System
EV	Event Manager
FCAL	LAR Forward and Calorimeter
FE	Front-End

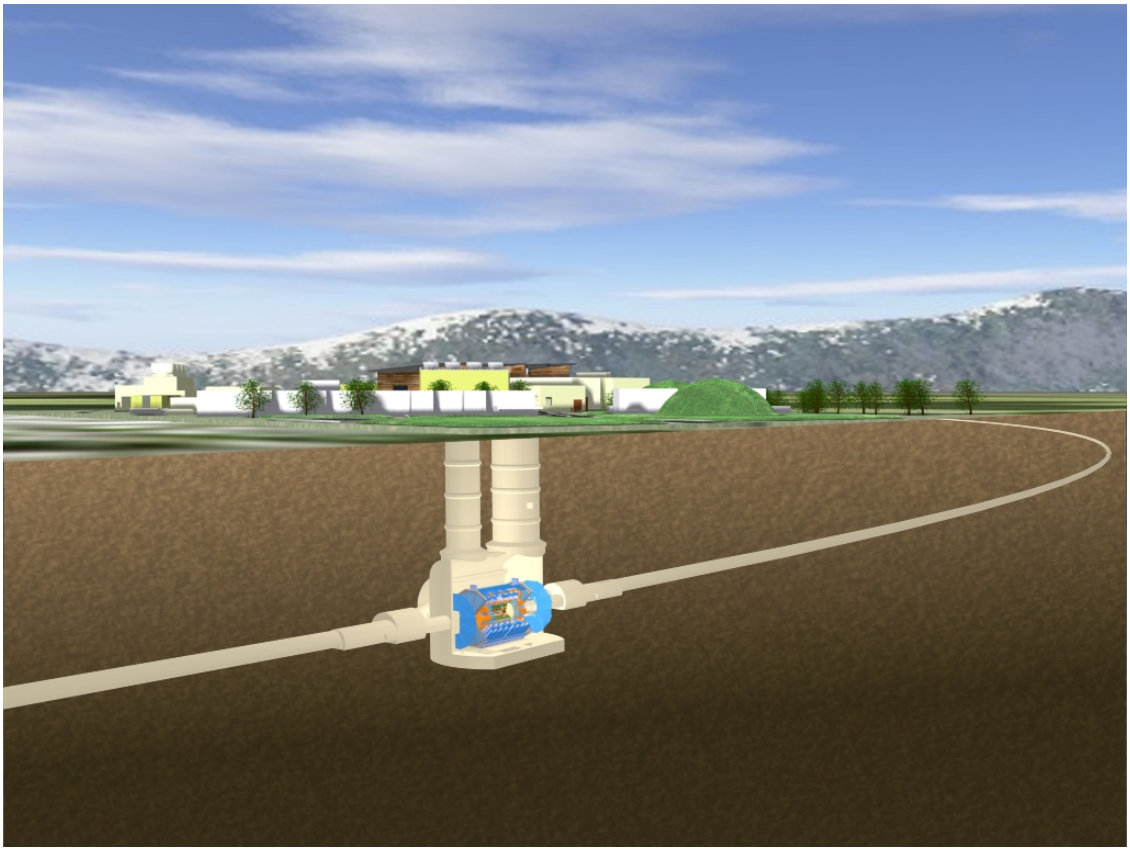
FEC	Front-End Crate
FPIAA	Finfing People In ATLAS Areas
FSM	Finite State Machine
FW	FrameWork
GCS	Global Control Station
GUI	Graphical User Interface
GUT	Grand Unified Theories
HEC	Hadronic End-Cap Calorimeter
HEP	High Energy Physics
HLT	High Level Triggers
HMI	Human Machine Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HV	High Voltage
IDE	Inner DEtector
IPMI	Intelligent Platform Management Interface
JCOP	Joint COntrols Project
LAN	Local Area Network
LAR	Liquid ARgon
LCS	Local Control Station
LEP	Large Electron Positron
LHC	Large Hadron Collider
LHCb	The Large Hadron Collider beauty experiment
LINAC	LINear ACcelerator
LU	Logical Unit
LV	Low Voltage
MARATON	MAgnetism and RAdiation TOlerant New
MB	Minimum Bias
MDT	Monitored Drift Chambers
NMR	Nuclear Magnetic Resonance
NTC	Negative Temperature Coefficient
OLE	Object Linking and Embedding
OPC	OLE for Process Control
OPCUA	OPC Unified Architecture
OSI	Open System Interconnection
PIX	PIXel detector
PLC	Programmable Logic Controller
PS	Proton Synchrotron
PVSS DB	PVSS DataBase manager
PVSS EV	PVSS EVent manager
PVSS II	Process Visualization and control system II
PVSS UI	PVSS User Interface
RADMON	RADiation MONitor
RC	Run Control
ROB	Read-Out Buffer
ROD	Read-Out Driver
ROS	ReadOut System

RPC	Resistive Plate Chambers
SCADA	Supervisory Control And Data Acquisition
SCS	Sub-detector Control Station
SCT	SemiConductor Tracker
SLIMOS	Shift Leader In Matter of Safety
SMI	State Management Interface
SML	State Manager Language
SR1	SuRface 1
SUSY	SUperSYmmetry
TDAQ	Trigger and Data AcQuisition
TDQ	Trigger and Data acQuisition sub-detector control
TDR	Technical Design Report
TGC	Thin Gap Chambers
TIA	Totally Integrated Automation
TIL	TILe calorimeter
TRT	Transition Radiation Tracker
TTC	Trigger and Timing Control
UI	User Interface
UML	Unified Modeling Language
UPS	Uninterruptable Power Supply

Contents

Part I

Introduction



Chapter 1

CERN and Particle Detection

The scale of large experiments that search for new particle physics are out of reach of individual research institutes. In this context, the European Centre for Nuclear Research (CERN) provides an international facility where researchers from all the world can work together towards new discoveries. The present challenge at CERN is the new hadron collider LHC, which is expected to be operational by the end of the present year, 2007.

This chapter gives an overall view of high energy physics today, describing briefly the CERN accelerators complex and presenting the principle of functioning of modern-day particle detectors.

1.1 CERN

The creation of an European laboratory was recommended at a UNESCO meeting in Florence in 1950, and less than three years later a Convention was signed by 12 countries of the "Conseil Européén pour la Rechèrche Nucléaire", CERN.

CERN was born as the prototype of a chain of European institution in space, astronomy and molecular biology. Nowadays, it is the world's largest particle physics laboratory. The laboratory sites are located on both sides of the Franco-Swiss border west of Geneva at the foot of the Jura Mountains. CERN operates with a yearly budget of 1.000 million Swiss francs which is raised by its 20 European member states. Furthermore, CERN employs 2500 people (out of which around 1000 are scientific personnel) and collaborates with 500 universities and institutes and 6500 people world wide.

The CERN complex provides accelerators and detectors for sub-atomic particles for high energy physics experiments to physicists around the world. The experiments are used to study the building blocks of matter or conditions which have happened shortly after the big bang. Besides physics another important discovery at CERN for daily life was the invention of the HTTP protocol and the HTML language by Tim Berners Lee in 1989 which led to the world wide web.

1.2 High Energy Particle Physics Today

Particle physicists have found that they can describe the fundamental structure and behavior of matter within a theoretical framework called the Standard Model. This model incorporates all the known particles and forces through which they interact, with the exception of gravity. It is currently the best description we have of the world of quarks and other particles. However, the

Standard Model in its present form cannot be the whole story. There are still missing pieces and other challenges for future research to solve [7].

The masses of the particles vary within a wide range. The photon, carrier of the electromagnetic force, and the gluons, that carry the strong force, are completely massless, while the conveyors of the weak force, the W and Z particles, each weight as much as 80 to 90 protons or as much as reasonably sized nucleus. The most massive fundamental particle found so far is the top quark. It is twice as heavy as the W and Z particles, and weights about the same as a nucleus of gold. The electron, on the other hand, is approximately 350,000 times lighter than the top quark, and the neutrinos may even have no mass at all.

Why there is such a range of masses is one of the remaining puzzles of particle physics. Indeed, how particles get masses at all is not yet properly understood. In the simplest theories, all particles are massless which is clearly wrong, so something has to be introduced to give them their various weights. In the Standard Model, the particles acquire their masses through a mechanism named after the theorist Peter Higgs. According to the theory, all the matter particles and force carriers interact with another particle, known as the Higgs boson. It is the strength of this interaction that gives rise to what we call mass: the stronger the interaction, the greater the mass. If the theory is correct, the Higgs boson must appear below 1 TeV. Experiments at Tevatron and at the Large Electron Collider (LEP) have not found anything below 110 GeV.

Another open question is the unification of the electroweak and strong forces at very high energies. Experimental data from different laboratories around the globe confirm that within the Standard Model this unification is excluded. When scaling the energy dependent constants of the electroweak (α_1 and α_2) and strong (α_3) interactions to very high energies, the coupling constants do not unify. Grand Unified Theories (GUT) explain the Standard Model as a low energy approximation. At energies in the order of 1016 GeV, the electromagnetic, weak and strong forces unify. One of the GUT theories is the supersymmetry (SUSY) that predicts new particles to be found in the TeV range. Many other GUT theories predict new physics at this energy scale.

These and other questions like the elementarity of quarks and leptons, the search of new quark families and gauge bosons or the origin of matter-antimatter asymmetry in the Universe, will be addressed by CERN's next accelerator, the Large Hadron Collider, which is currently under construction.

1.3 The Large Hadron Collider

Already in the mid-1980's, before the Large Electron Positron collider (LEP) [8] was operating, scientists around the world started to think about an accelerator which would be capable to exceed the energies provided by LEP in order to prove even deeper into the properties of matter. Finally, in 1994, the construction of CERN's Large Hadron Collider (LHC) was officially approved. This year marks the birth of the world's largest and most powerful accelerator which will provide proton-proton collisions at energies 10 times greater and at a rate 100 times higher than any previous machine. LHC will be installed in the existing LEP tunnel. According to the current schedule the LHC is planned to be operational in the year 2007.

At present, the LHC is being installed in a tunnel 27 km in circumference, buried 50-175 m below ground. Located between the Jura mountain range in France and Lake Geneva in Switzerland, the tunnel was built in the 1980s for the previous big accelerator, the LEP. The tunnel slopes at a gradient of 1.4% towards Lake Geneva. The LHC will produce head-on collisions between two beams of particles, either protons or lead ions. The beams will be created in CERN's existing

chain of accelerators and then injected into the LHC (see Figure 1.1,). These beams will travel through a vacuum comparable to outer space. Superconducting magnets operating at extremely low temperatures will guide them around the ring. The LHC will generate about 800 million collisions per second and it will provide collisions at the highest energies ever observed in laboratory conditions. Four huge detectors - ALICE, ATLAS, CMS and LHCb - will observe the collisions so that the physicists can explore new territory in matter, energy, space and time. ATLAS and CMS are general-purpose proton-proton detectors with similar physical goals but different design. The main difference lies in the magnet system which strongly influences the geometry of the detectors. ALICE and LHCb are two specialized detectors. More details about the detectors can be found in the technical design reports: for ATLAS [9], for CMS [10], for ALICE [11] and for LHCb [12]. The LHC is a machine for concentrating energy into a very small space. Particle energies in the

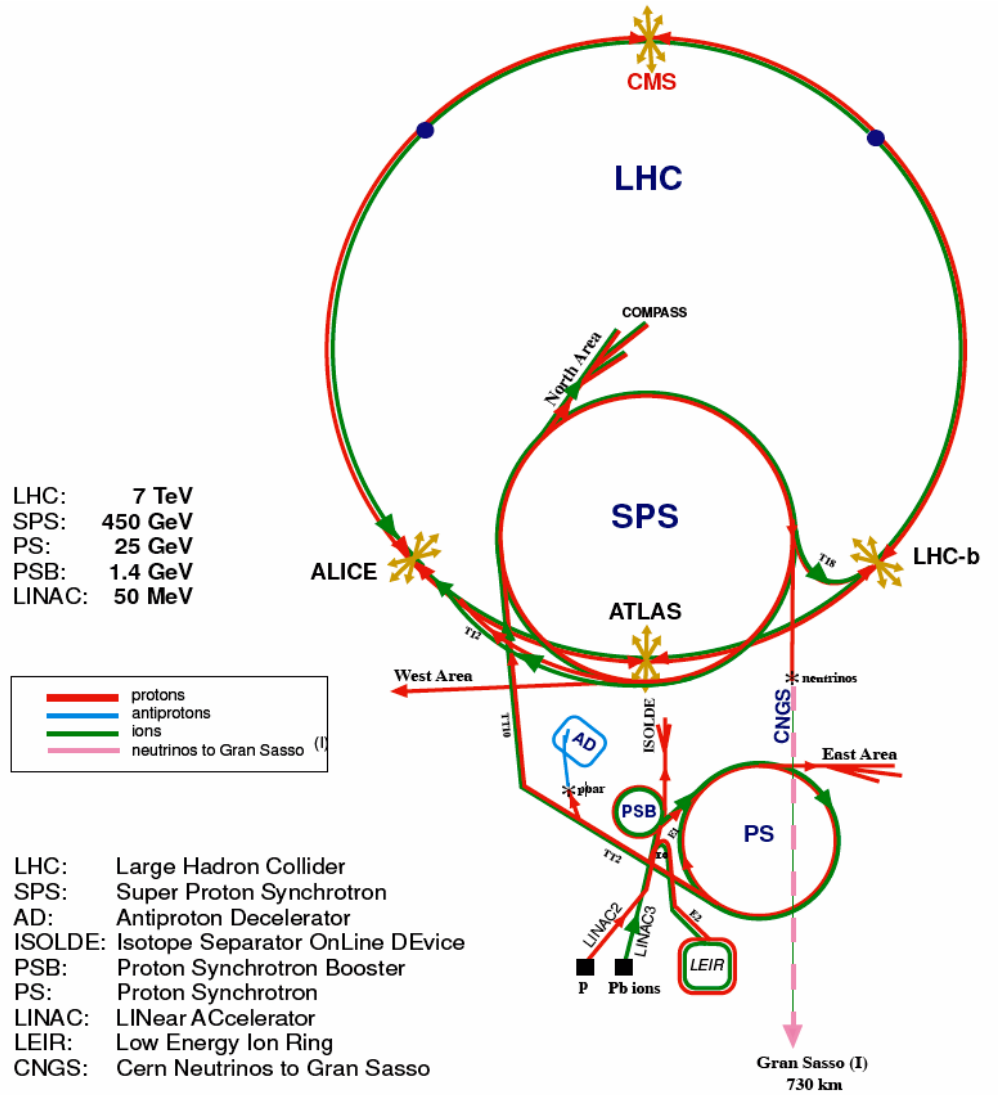


Figure 1.1: Overview of the CERN accelerator complex (not to scale). A succession of machines with increasingly higher energies injects the beam each time into the next one, which takes over to bring the beam to an energy even higher, and so on. The flagship of the complex will be the LHC where the main four experiments (ATLAS, CMS, LHCb and ALICE) will study the head-on collisions between two beams of particles. Picture taken from [13].

LHC are measured in Tera electronVolts (TeV). 1 TeV is roughly the energy of a flying mosquito, but a proton is about a trillion times smaller than a mosquito. Each proton 'flying' round the LHC will have an energy of 7 TeV, so when two protons collide the collision energy will be 14 TeV. Lead ions have many protons, so they can be accelerated to even greater energy: the lead ion beams will have a collision energy of 1150 TeV. At near light-speed, a proton in a beam will make 11245 turns per second. A beam might circulate for 10 hours, traveling more than 10 billion kilometers - far enough to get to the planet Neptune and back again. To control beams at such high energies the LHC will use some 7000 superconducting magnets. These electromagnets are built from superconducting materials: at low temperatures they can conduct electricity without resistance and so create much stronger magnetic fields than ordinary electromagnets. The LHC's niobium-titanium magnets will operate at a temperature of only 1.9 K (-271°C). The LHC will operate at about 8 tesla, whereas ordinary 'warm' magnets can achieve a maximum field of about 2 tesla. If the LHC used ordinary 'warm' magnets instead of superconductors, the ring would have to be at least 120 km in circumference to achieve the same collision energy.

1.4 Particle Detectors - An Overview

Detectors are used to examine tracks made by the new particles that are produced when other accelerated particles collide. In the early days photographic films, spark chambers and bubble chambers were used, but since the late 1960s, electronic detectors have taken over these old methods. There are two basic kinds of electronic detectors: tracking detectors, which reveal the trajectories of individual charged particles, and calorimeters, which measure energies. A modern electronic detector is built like an onion, with layers of trackers and calorimeters to give as much information as possible about the particles produced in each collision. Figure 1.2 shows the cross-section of the ATLAS detector after a simulated collision, the main parts are the Inner Detector, surrounded by a super-conducting solenoid, the Calorimeters and the Muon Spectrometer with its superconducting air-core toroids.

Tracking chambers make the path of the particle directly visible. Just the track of the particle gives a lot of useful information, especially if the detector is placed inside a magnetic field: the charge of the particle, for instance, will be obvious since particles with positive electric charge will bend one way and those with negative charge will bend the opposite way. Also the momentum of the particle can be determined: very high momentum particles travel in almost straight lines, while low momentum particles make tight spirals. Tracking chambers are similar to every-day effects: high-flying airplanes seem invisible, but in the right conditions, you can see the trails they make. It is the same for particles: creating the right conditions inside a detector allows you to see their trails, i.e. their tracks. This is achieved when particles pass through suitable substances that are able, in different ways, to visualize the interaction of the passing particle with the atoms of the substance itself. Early research in particle physics used, for instance, detectors called cloud chambers, where trails of droplets formed along the tracks left by particles. In a bubble chamber, particles leave trails of bubbles in a liquid. In a spark chamber, sparks occur when a gas is ionized by a passing particle. Most modern tracking devices do not make the tracks of particles directly visible. Instead, they produce tiny electrical signals that can be recorded as computer data. A computer program then reconstructs the patterns of tracks recorded by the detector, and displays them on a screen.

However, more information is needed and usually a tracking device is associated with a calorimeter. Calorimeters stop and fully absorb most of the particles, providing a measurement of their energy. A calorimeter measures the energy lost by a particle that goes through it. It is

usually designed to fully stop or 'absorb' most of the particles coming from a collision event, forcing them to deposit within the detector all of their energy. Calorimeters typically consist of layers of 'passive' or 'absorbing' high density material (lead for instance) interleaved with layers of 'active' medium such as solid lead-glass or liquid argon. Interactions between the particles and the high-density material of the calorimeter, with a high atomic number Z, which means a large number of protons, induce the production of 'showers' of low energy particles, a process that quickly uses all of the available energy. The active medium samples this lost energy, from which, physicists can determine the total energy of the incoming particle. Two kinds of calorimeter are typically needed to absorb the different kinds of particles.

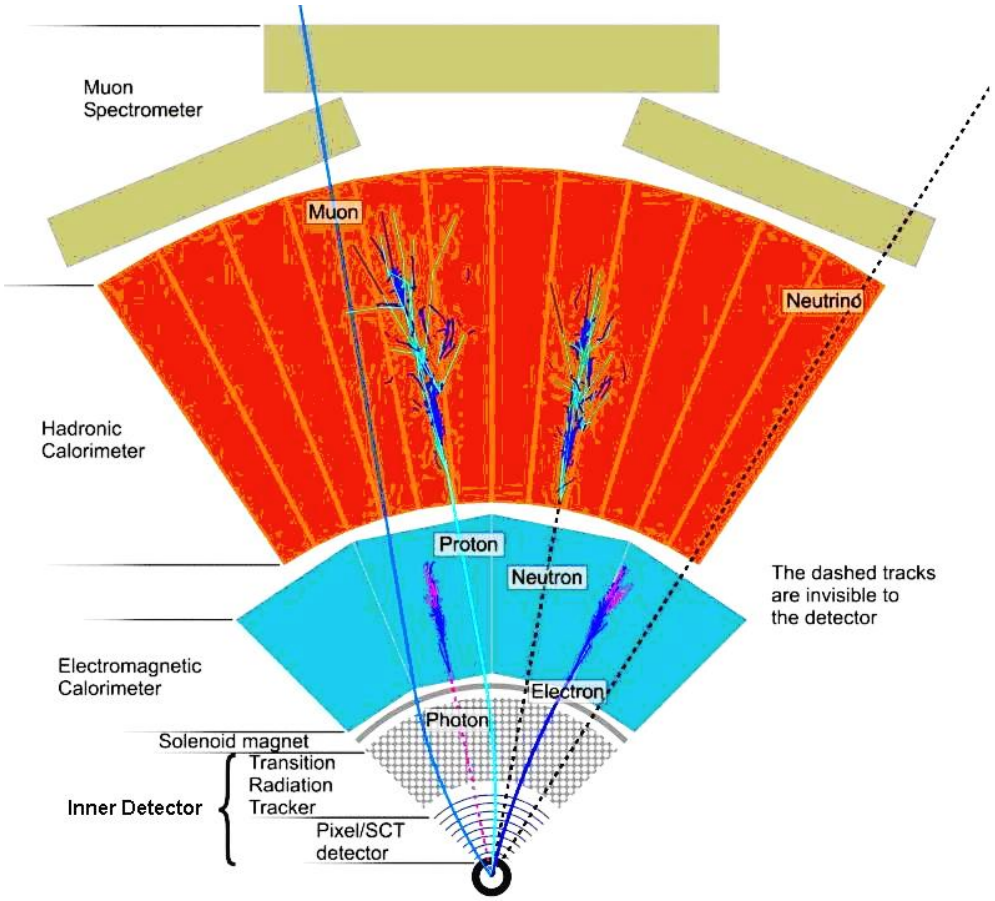


Figure 1.2: Cross-section of the ATLAS detector after a collision. The experiment is composed of three main detector systems, the tracker, the calorimeter, and the muon system

- Electromagnetic calorimeters measure the energy of light particles - electrons and photons - as they interact with the electrically charged particles inside matter.
- Hadronic calorimeters sample the energy of hadrons (particles containing quarks, such as protons and neutrons) as they interact with atomic nuclei.

Muons and neutrinos are often the only particles capable of escaping a calorimeter. Muons can hardly be stopped, but at least they can be identified: special muon detectors are located outside the calorimeter, and only muons can emerge and leave a track there. Muon detectors can

be gas-filled chambers that detect the passage of charged particles in a similar way to the central tracking detectors.

Neutrinos, by contrast, escape and do not even leave a track, going through all the detectors undetected. However, as they are the only known particles that can escape, their presence can be inferred from an imbalance of the initial and final energies of the event.

Assembling all the pieces of information from each track, physicists can fully characterize each particle, and by arranging all the tracks coming from a collision, they can reconstruct the event with great precision.

Chapter 2

The ATLAS Experiment

The ATLAS experiment is a general-purpose experiment for proton-proton collisions which has been built by a consortium of institutes, and laboratories all around the world. The experiment is composed of three detector systems, the tracker, the calorimeter and the muon system. These are sub-divided into 9 different specialized sub-detectors that perform different tasks such as track reconstruction and particle identification.

This chapter begins giving an overview of ATLAS and its general infrastructure. Then, it follows going through its different specialized sub-detector systems. A brief description of the operational principle and the required infrastructure for operation are given for each sub-detector system. Finally, the discussion of the envisaged overall experiment control motivates the following chapters. At this point, the data acquisition system of ATLAS, which will be later synchronized with the DCS, is introduced. This chapter provides terminology and elements which are useful later when expanding the DCS and its hierarchical control.

2.1 Overview of the ATLAS Experiment

The ATLAS (A Toroidal LHC ApparatuS) detector is a general-purpose experiment for proton-proton collisions designed to investigate the full range of physical processes at the LHC. With its length of 45 m and its height of 22 m it is the largest particle physics experiments ever built. It follows the classical structure of a particle detector. The main parts are shown in Figure 2.1, the Inner Detector, surrounded by a super-conducting solenoid, the Calorimeters and the Muon Spectrometer with its gigantic superconducting air-core toroids. These main three parts are divided into 12 different specialized sub-detectors that perform different tasks such as track reconstruction and particle identification. All those components together have a weight of 7000 t, which is similar to the weight of the metallic structure of Eiffel Tower is 7300 tons. The ATLAS detector can be also logically divided into two parts: the central part, also called barrel part, and the forward or end-cap part. Most of the sub-detectors discussed later exist in both areas.

The ATLAS Collaboration currently consists of 151 institutions from 34 countries, counting approximately 1800 scientific authors. The estimated cost of the detector is about 0.5 billion CHF.

2.2 Infrastructure of the ATLAS Experiment

The main part of the equipment needed to operate the detector will be geographically distributed in three areas (See Figure 2.2). The main control room SCX1 at the surface of the installations,

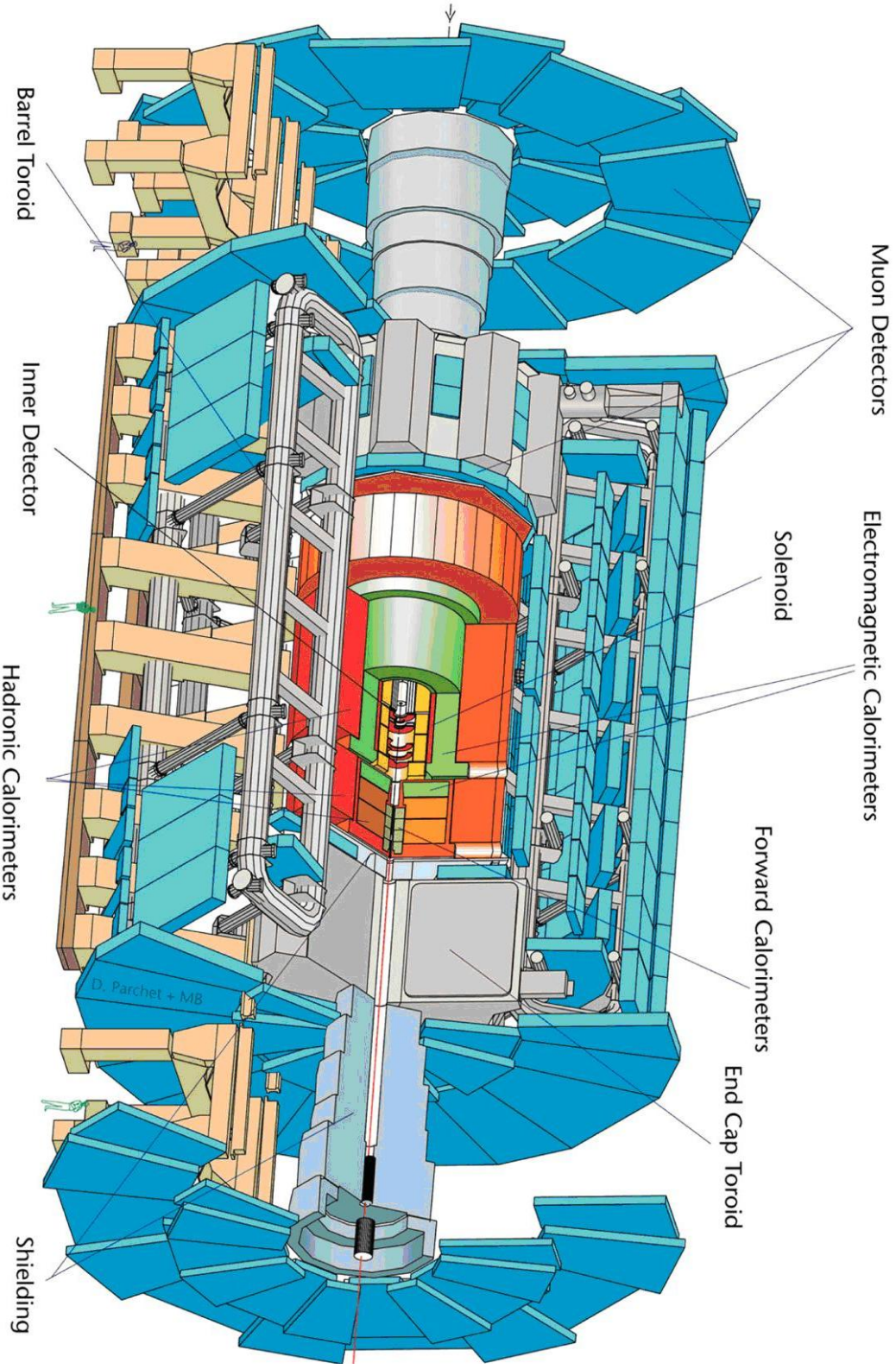


Figure 2.1: Overall layout of the ATLAS detector. With its length of 45 m, its height of 22 m and its weight of 7000 t it is the largest particle physics experiments ever built. Picture taken from [13].

the underground electronics rooms USA15 and US15, and the cavern of the detector, UX15. Because of high radiation and high magnetic fields in the area immediately around ATLAS, it seems infeasible to have certain equipment close to the detector (i.e. the High Voltage power supplies) and so space has been reserved in the USA15. USA15 is equivalent to US15 except that the first is accessible to personnel during beam operation and the second is not.

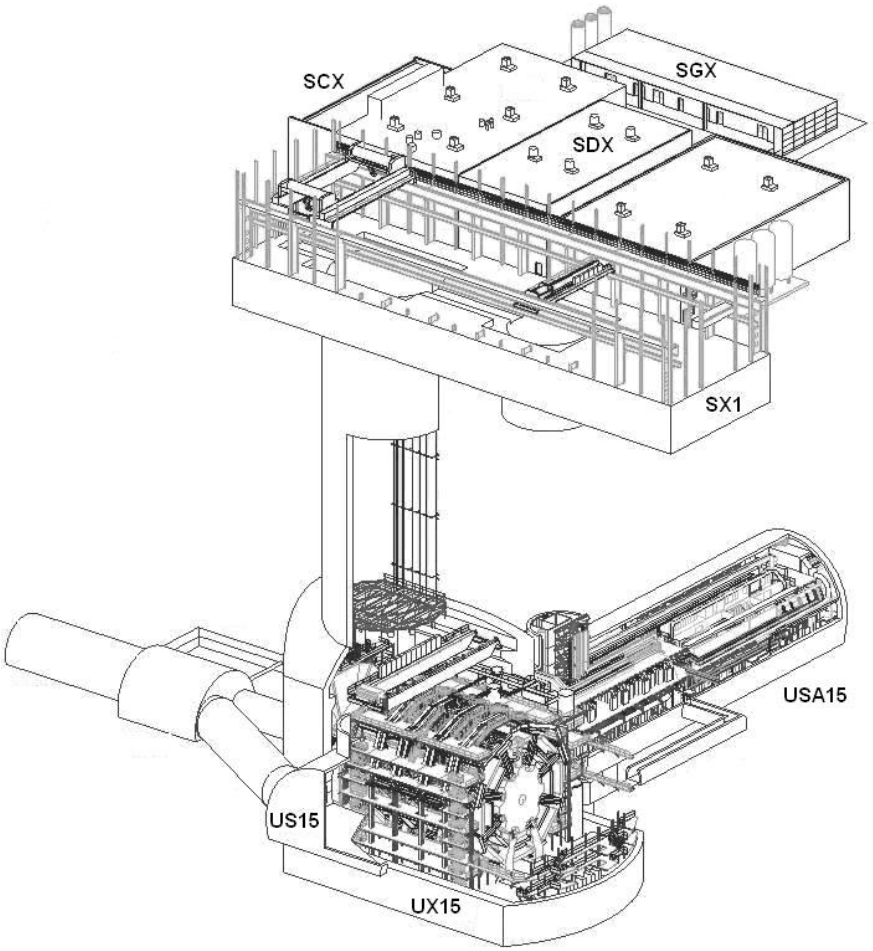


Figure 2.2: ATLAS cavern infrastructure. The ATLAS controls are distributed in three main areas, namely SCX1 (surface control room), UX (detector cavern), US15 and USA15 (electronics rooms). Picture taken from [13].

At present the detector is being installed in the cavern UX15 which floor is located 92 m below ground where most of the detectors parts will be assembled for the first time underground. The cavern dimensions are 53 m long, 35 m high and 30 m wide.

Civil engineering started in 1997 when the LEP accelerator was still operational; during the excavation, 300.000 tons of rock was extracted and 50.000 tons of concrete used. Due to the weight of the detector, a 5 m thick concrete floor steel-enforced slab was put in place. Presently, the floor moves slightly upwards $\sim$1 mm per year.

The infrastructure installation started in November 2003 and the detector assembly in November 2004. The absence of a sufficiently large surface hall to pre-assemble the barrel toroid magnet or even a significant fraction of the muon detector leads to a very complicated sequential assembly in the UX15 cavern. Throughout the installation it is essential to find the correct sharing between

the heavy handling operations to put in and assemble the very large and heavy components with meticulous testing to ensure that each sub-system is working before moving on to the next. The installation process is organized into six sequential phases: (1) surface and underground infrastructure, (2) barrel toroid and barrel calorimeters, (3) barrel muon chambers and end-cap calorimeters, (4) inner detectors and muon "big wheels", (5) end-cap toroids and muon "small wheels" and (6) vacuum pipe, shielding and closing.

As we will see later in this chapter, each specialized sub-detector is responsible of its own dedicated control systems. However, it also exists a common infrastructure that need to work in close interaction with the sub-detectors. This common infrastructure consists in a set of systems internal (i.e. environmental conditions) and external (i.e. LHC accelerator) to the ATLAS detector. They are listed and described briefly below:

- **CERN Technical Services**: The CERN technical infrastructure contains several basic services that are necessary to run the laboratory, such us cryogenics, conventional cooling, ventilation, gas system and electricity distribution. Since these systems are very similar over the different experiment, they are developed, maintained and controlled CERN-wide by specialized groups for homogeneity and reliability. However, some of the systems need feedback from the detector. This is the case for the gas system and cooling which work in close interaction with the detector.

  - **Cryogenics**: When the LHC will start up, it will operate the largest 1.8 K helium refrigeration and distribution systems in the world, and the two biggest experiments, ATLAS and CMS, will deploy a wide range of cryogenic techniques. The ATLAS refrigerating plants are independent from the system required to cool the LHC to 1.8 K. ATLAS will use two helium refrigerators to cool at 4.5 K the spectrometer magnets and one nitrogen refrigerator to cool at 84 K the electromagnetic calorimeter. Figure 2.3 shows the toroid magnet cryogenics system set-up at the cavern UX15.

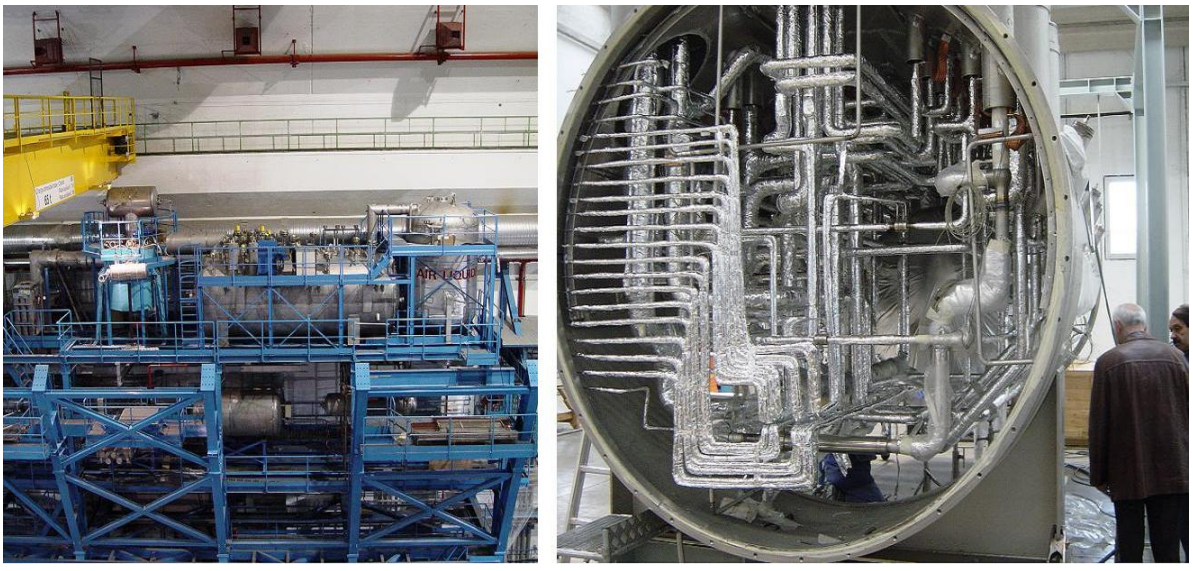


Figure 2.3: Left: Helium cryogenics system for the toroid magnet installed at the caver UX15. Right: Cross-section of the main valve box for the proximity cryogenics system. [13].

  - **Cooling and Ventilation**: It ensures the operation of the ATLAS detector and its infrastructure within safe temperature ranges. Most of the specialized sub-detectors have its own cooling plant controlled by an external PLC that interacts with the detector data. Sub-detectors retrieve relevant data and simultaneously are able to specify set points on the operating parameters, flow, pressure and temperature. In contrast, other systems such as power cable, racks or beam pipe cooling work in stand-alone mode without interaction with the detector. In addition, heating, ventilation, air conditioning, smoke removal and gas mixture removal installations are operational for the construction of the surface and underground structures of the experiment.

  - **Gas Systems**: They provide different gas mixtures to the gaseous detectors. The ATLAS gas detectors are the Transition Radiation Tracker (in the inner detector) and the 4 muon sub-detectors. The gas systems have several control loops, ratio, pressure, and temperature. The most critical control loop deals with the pressure regulation which need to be accurate for good physic data taking. In addition, depending on each sub-detector the pressure has to be within a specific range to prevent damaging the system.

  - **Electricity**: CERN is operating a large electrical network for the supply of electricity to the accelerators, experiments and infrastructure. The electrical network operates on voltage levels from 400V to 400KV with a total yearly consumption of near to 1000 GWh. The electrical network supervision is based on an industrial control system completely developed and supported by an external contractor. Thus, the maintenance and further development will be under the responsibility of the system provider. In addition, for certain equipment the usage of Uninteruptable Power Supplies (UPS) is needed. There are various reasons to use a UPS ranging from avoiding inconveniences of power cuts up to assuring the safety of the experiment (i.e. providing the possibility to switch down equipment in a ordery fashion in case power is cut and is not re-established within a certain time).

  - **Vacuum**: An extremely high vacuum must be maintained inside the beam pipe to avoid parasitic particles from interacting with the beam. The vacuum chamber, or beam pipe, is made of beryllium which is 20 times more transparent to particles than steel. The beam pipe is the interface between the accelerator and the experiments, and must therefore meet the requirements of both. Experimental physicists require the straightest and most transparent chamber possible so that every single particle produced in the collisions is visible to their detectors. On the other hand, accelerator physicists want a strong, reliable beam pipe with no leaks and maximum electrical conductivity.

- **Environment Infrastructure**: General environmental parameters including temperature, humidity and atmospheric pressure in the detector cavern, electronics rooms and surface will be available to ensure the safety of the experiment and a good operation. The nominal values of the temperature can vary over a wide range, from 4.5 K for the super-conducting magnets, to 88 K for the liquid argon calorimeters, to 253 K for the silicon parts of the inner detector to 293 K for the TRT, tile calorimeter and muon chambers. Radiation probes will be spread over the experiment geometry being mainly supervised by the sub-detectors themselves. The instrumentation in the cavern must be radiation-hard or tolerant to levels of 1-300 Gy per year in the muon sub-detector and inner tracker, respectively.

- **LHC**: The phases of the LHC define a multitude of states required for the internal functioning of ATLAS. A sub-set is of direct interest for the interaction with the experiment control, in particular those states and parameters that describe the condition of the beam with consequences for the operation of the detector. Phases with stable beam and low background may indicate that it is safe to bring the detector to the operational state for physics data-taking, whereas abnormally high background conditions may dictate that the detectors be turned off for safety reasons.

- **Rack Control**: Most of the underground electronics are placed within racks in rooms USA15 and US15. Figure 2.4 shows an schematic view of the rack distribution at the level 1 of the USA15 counting room. The primary functions of a rack are: mechanical housing for sub-racks (i.e. the HV crates) and other instrumentation (i.e. PCs), and the provision of a water/air cooling system to evacuate the heat generated by the electronics housed within. The purpose of the rack control system is to provide monitoring of the rack's internal environment and function, and to provide a limited amount of control. The control system comprises several functional components: the rack sensors, the power distribution unit, the safety system and the associated safety devices, and optional I/O. A rack is not in itself a complicated piece of equipment and requires very few variables to describe its state.

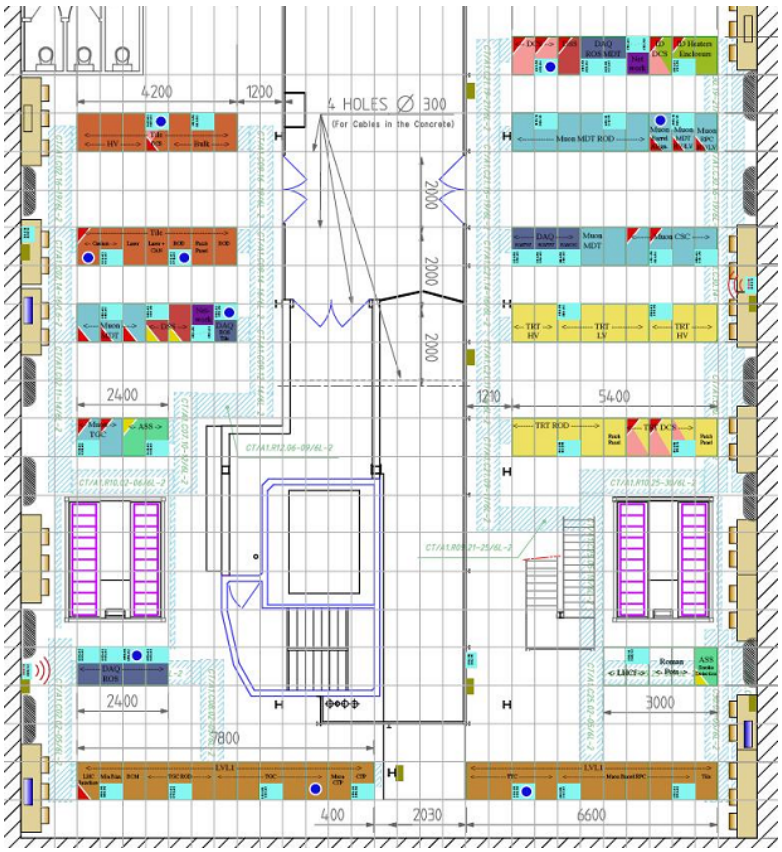


Figure 2.4: Electronics racks distribution in the underground USA15 level 1. Different sub-detectors' racks (represented by colors) that will allocate the instrumentation, mainly rack mounted PCs and the power systems. [14].

- **Finding People Inside ATLAS Areas** (FPIAA): During the maintenance periods it is expected that up to 150 people could be present in the cavern at the same time, most of them working inside the intricacies of the detector, and completely hidden and invisible from outside. In case of emergency, especially if smoke or fog (leaks of cryogenic fluids) are present, it could be extremely difficult and dangerously long for a rescue team to locate every person who could be in danger. Under these circumstances, a granular system for finding persons is then mandatory. FPIAA is based on a large number (at the present about 400) of passive infrared sensors, each one detecting the presence of a person in a relatively small volume ($\sim$30 m^3) and distributed to cover the most critical locations in the cavern.

- **Magnet system**: The magnet system is crucial for the well functioning of the experiment and it supposes one third of the total budget of ATLAS. It is based on a thin superconducting central solenoid which generates a uniform longitudinal field for the inner tracker and three superconducting toroid magnets (the barrel and two identical end-caps), each of them made of eight coils producing a tangential field around the central axis for the muon spectrometers (see Figure 2.5 and 2.6). The peak magnetic field for the central solenoid is of 2.6 T, for the barrel toroid and end-cap toroid they are 3.9 and 4.1 T respectively. To achieve the desired high magnetic fields an aluminum-stabilized NbTi superconductor is used. The external envelope of the magnets is 20 m in diameter and 26 m in length with a total cold mass of 660 tons and a stored energy of 1.6 GJ when magnets are powered up to 20 kA.

  In order to refrigerate the magnet system, two separate helium refrigerators have been installed in a side-cavern close to the detector to allow the four operational modes of magnets, namely: the cool-down from ambient temperature, the steady-state operation at 4.5 K, the thermal recovery (from $\approx$ 60 K) after a fast energy dump and the warm-up.

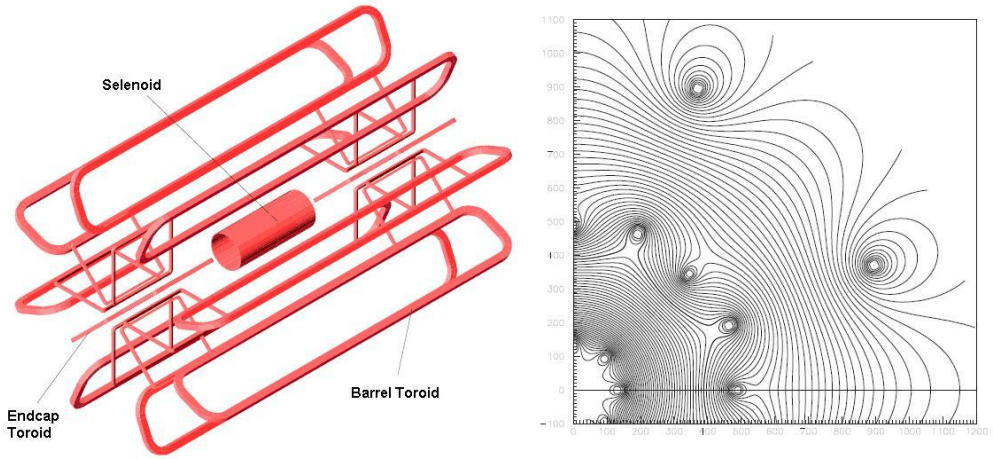


Figure 2.5: Left: Atlas magnet system. At the center of the magnet configuration is a super-conducting selenoid producing a field of 2 T in the inner detector region. In the muon spectrometer three separate toroid magnets, each consisting of 8 coils, create a field azimuthal to the detector axis. Right: Magnetic field map in plane perpendicular to the beam axis in the region between the barrel and one end-cap toroid

Information about the overall detector concept can be found in the ATLAS Technical Design Report (TDR) for Technical Co-ordination [9] and in the (already older) ATLAS technical proposal [15]. More detailed information about the ATLAS subsystems is presented in the TDR of each subsystem ([16], [17], [18], [19], [20], [21]).

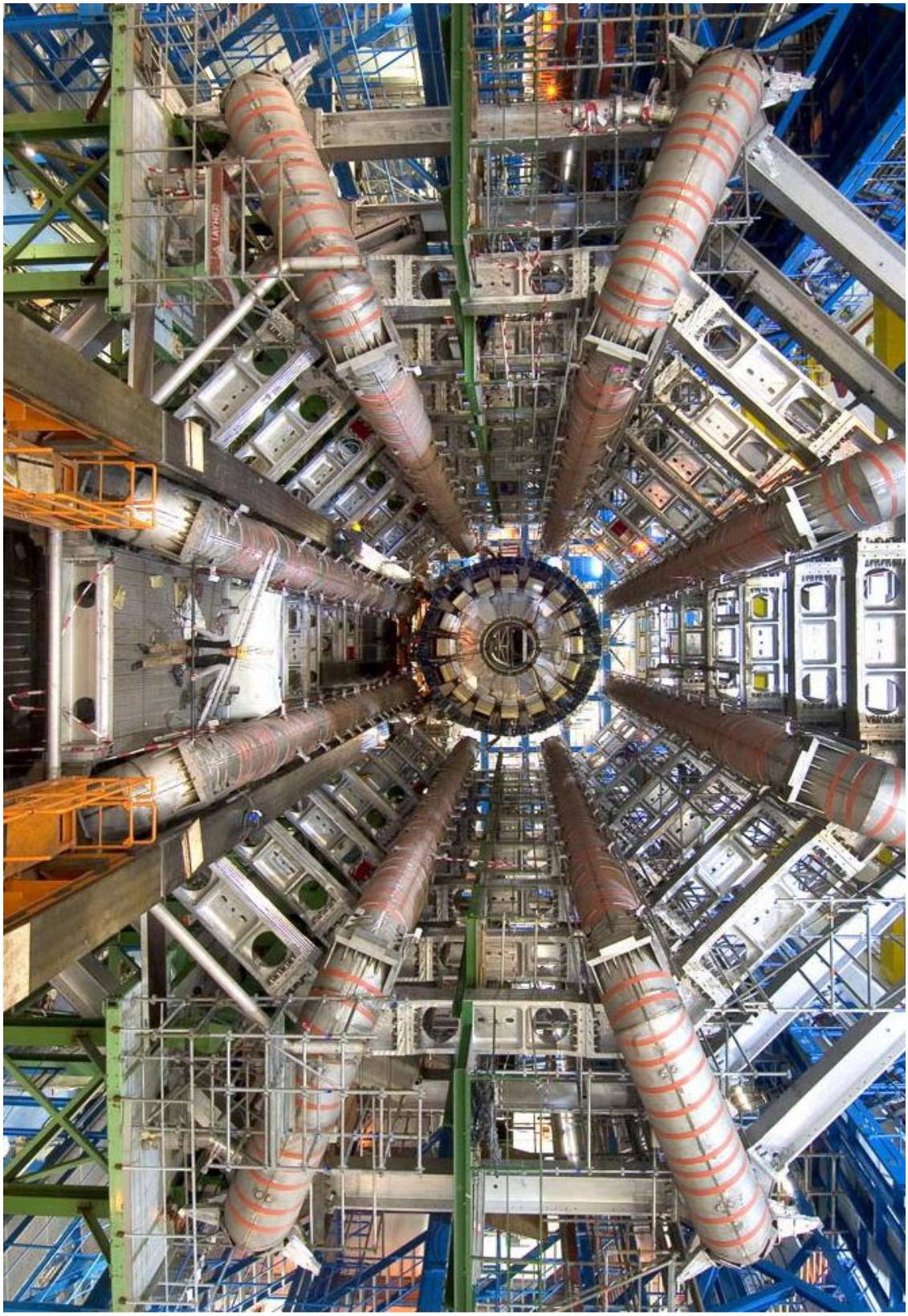


Figure 2.6: ATLAS set-up in October 2005 with the barrel toroid magnet completely assembled.[13]

2.3 Inner Detector

The ATLAS Inner DEtector (IDE) consists of three different types of detectors. At the inner radii two high-resolution detectors are used to perform high-precision measurements whereas at the outer radii continuous tracking elements are installed. The three parts of the IDE are contained in the central solenoid which provides a nominal magnetic field of 2 T. Its purpose is to provide pattern recognition, momentum and vertex measurements and electron identification. The IDE is shown in Figure 2.7.

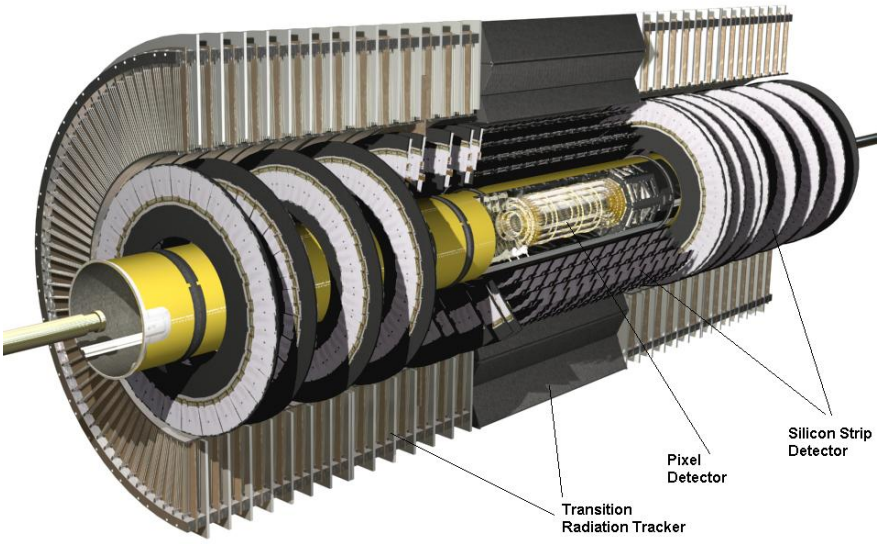


Figure 2.7: Cut through the ATLAS Inner Detector. It consists of two different high-resolution detectors at the inner radii (Pixel Detector, Semiconductor Tracker (Silicon Strip Detector)) and continous tracking elements (Transition Radiation Tracker) at the outer radii.

The outer radius of the IDE cavity is 115 cm and its total length is 7 m. It is divided into one barrel part and two identical end-caps on either side. In the barrel, the high-precision layers are arranged in concentric cylinders around the beam axis. In the end-caps they are mounted on discs perpendicular to the beam axis. Similarly the straw tubes are adjusted parallel to the beam axis in the barrel and radially in the end-caps.

Given the momentum and vertex resolution requirements and the very large track density at the LHC, fine-granularity detectors are needed to be able to work with sufficient precision. Therefore semiconductor tracking detectors were chosen, applying pixel and silicon microstrip (SCT) technologies. Since the amount of material should be minimal and because of the high costs the number of precision layers must be limited. At the outer radii it is placed a gaseous detector, a Straw Tube Tracker (TRT) is implemented to provide continuous track-following with little material per point and at low costs. A typical particle track crosses three pixel layers and eight strip layers (equal to four space points). Additionally around 36 tracking points are provided by the straw tubes. Although the straw tubes have a lower precision per point compared to the silicon trackers, they compensate that by a large number of measurements and a higher average radius. Those techniques combined give a robust pattern recognition and a very good resolution.

Operating Principle of Silicon Detectors

The basic operating principle of semiconductor detectors (Pixel and SCT) is analogous to gaseous ionization devices (TRT) which are discussed later more in Section 2.5.

Figure 2.8 below shows how the passage of ionizing radiation through a semiconductor medium creates electron-holes pairs (electron-ion pairs in a gas medium) along its track, being the number proportional to the energy loss. An externally applied electric field separates the pairs before they recombine: electrons drift towards the anode and holes to the cathode. The collected charge produces a current pulse on the electrode, whose integral equals the total charge generated by the incident particle.

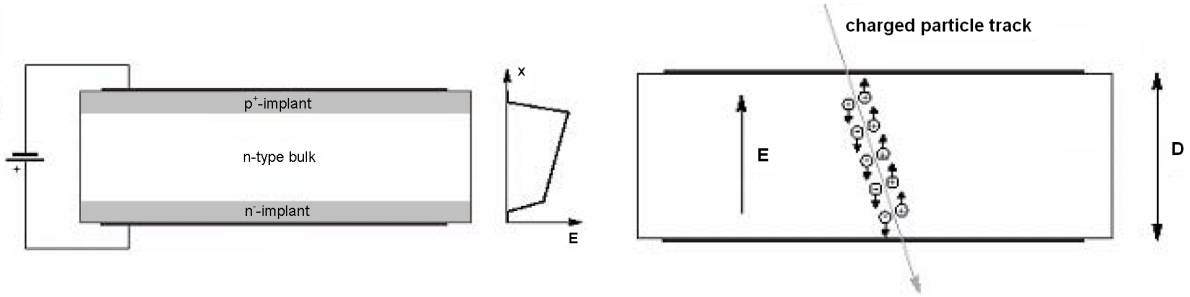


Figure 2.8: Left: Schematic of a Silicon Detector based on n-type bulk material. Once the junction is under reversed bias, all free carriers in the bulk are drained by the electric filed. Right: A charged particle traversing the detector generates free electrons-hole pairs along its track which are moved by electric field. Picture taken from [22].

The advantage of the semiconductor is that the average energy required to create an electron-hole pair is 10 times smaller than for gas ionization. The result is an increase in the energy resolution but also in cost of the detector. Moreover, because of their greater density, they have a greater stopping power than gas detectors. They are compact in size and can have very fast response times. However, semiconductors generally require cooling to low temperatures before they can be operated. This, of course, implies a critical cooling system which adds to detector overhead. One of the problems in current semiconductors detectors research, in fact, is to find and develop new materials which can be operated at room temperature. Being crystalline materials, they also have a greater sensitivity to radiation damage which limits their long use.

IDE Controls

From the point of view of controls Pixel, SCT and TRT have specific requirements. The pixel detector has a very high power density across the active area which needs a sophisticated fast interlock system between power supplies and the cooling system. The SCT covers a large area with silicon detectors which have to be very stable and well aligned. The TRT being a gaseous detector needs the gas system controls closely coupled to the high voltage system. However, since these sub-detectors are assembled in little room, close to the interaction point, they share a set of common services.

- **Beam Condition Monitor** (BCM): It is an ATLAS dedicated system which detects anomalous beam events that would cause background to the normal events of colliding protons, or even more drastically, cause danger to the experiment. The radiation hard BCM detectors will be placed at $\pm$z on both sides of the interaction point along the beam pipe.

- **Evaporative Cooling**: Semiconductors detectors, Pixel and SCT, require cooling to low temperatures before they can be operated. Both share a sophisticated evaporative cooling system that allows a silicon operating temperature of $\sim$-6°C and is able to remove in total 60 KW of heat. Figure 2.9 shows a schematic view of the system. A unique cooling plant is being built for both sub-detectors having 4 compressors working in parallel, one condenser and one liquid pump at the outlet of the condenser. Then, the different detector structures will be cooled by 204 independent cooling circuits which are tuned according to the sub-detector operational scenario. Each cooling circuit consists of one recuperative heat exchanger, 1, 2 or 3 capillaries, detector structure, one heater on the return vapor line and piping connecting all the components to the distribution racks via one pressure and one back pressure regulator.

- **Nuclear Magnetic Resonance** (NMR): The NMR system monitors the magnetic field strength at a small number of points in the IDE volume. It provides a precise and absolute field value that can be used to scale a field map. The map will be measured with a mapping machine and possibly improved with track curvature studies.

- **Radiation Monitor** (RADMON): The IDE has to be operated in a highest radiation environment of the experiment area being inaccessible for servicing during long periods. The RADMON measures the total ionising dose and the non-ionising energy loss at various locations in the detector. These measurements are vital for understanding the changes in performance during the operation of the experiment, verifying simulations and thus giving a chance to plan a better operation scenario. Moreover, it also provides on-line information about the degradation of the bipolar transistors performance.

- **Environment**: Three main measurements provide with data about the environment within the IDE volume. Firstly, integrated humidity and temperature sensors monitor the IDE conditions, specially through the TRT where exist cold cables. Secondly, coolant temperature sensors monitor the heat exchanger of the evaporative cooling and its efficiency. Finally, useful information about the temperatures in the IDE volume is retrieved by monitoring the temperatures on several places along the cable route.

- **Thermal Enclosure**: To separate the cold SCT volume from the TRT operating at room temperature, the SCT will be surrounded by a thermal enclosure and maintained in a dry, cold nitrogen atmosphere.

For monitoring in the high radiation environment inside the IDE, the aim is to have as few complex active elements as possible but the choice of technique also has to take into account the amount of material which will be introduced into the IDE.

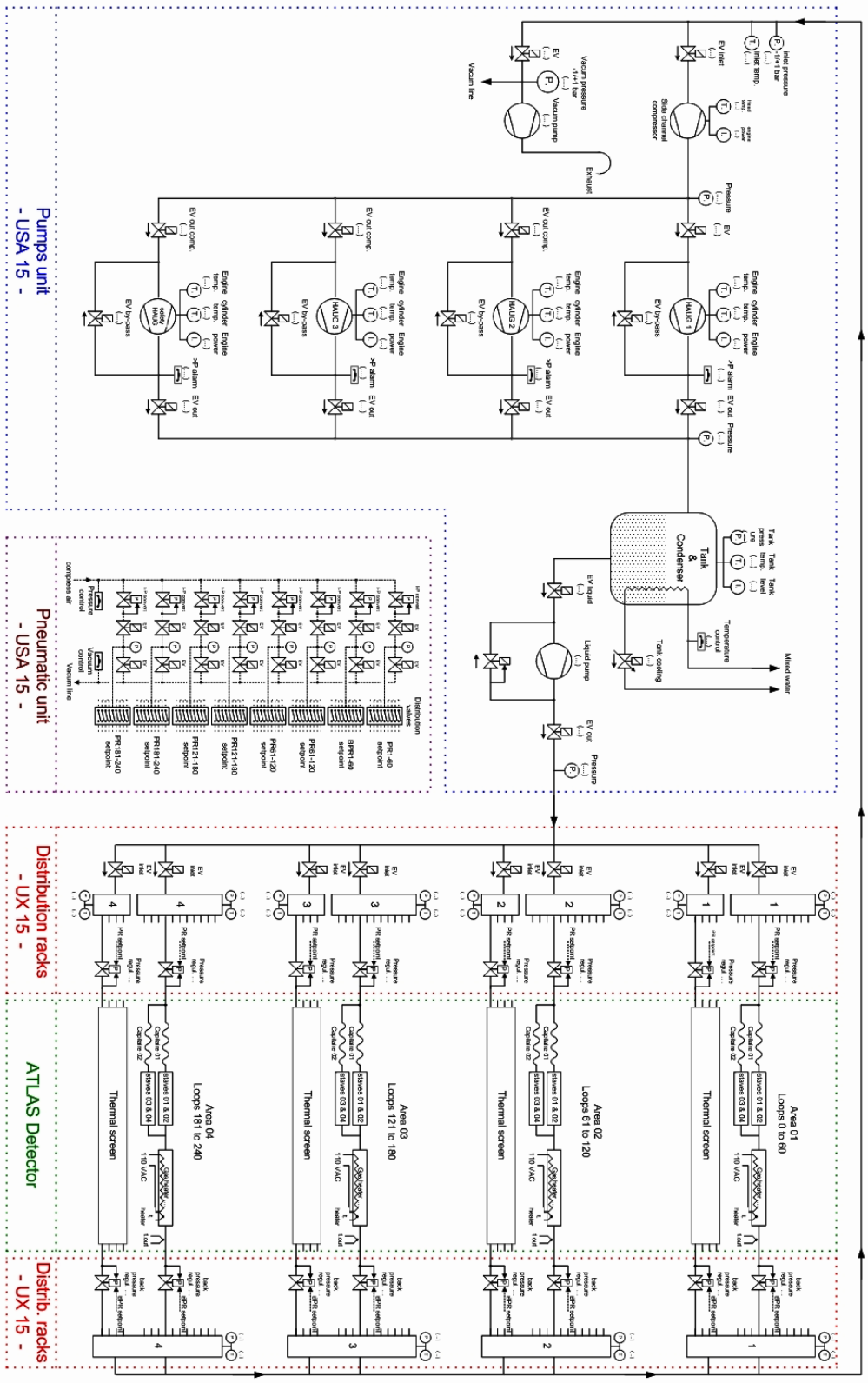


Figure 2.9: Evaporative Cooling Schema. The system works as a standard industrial fridge, the fluid is delivered in a liquid phase at room temperature from the condenser to the capillaries, through which the fluid expands and remains in saturation conditions in the detector structure. Then, heaters evaporate the residual liquid at the exhaust. The efficiency of the cycle is increased by a recuperative heat exchanger by decreasing the required flux at the inlet of the detector structures. Picture taken from [23].

2.3.1 Pixel Detector

The PIXel detector (PIX) is the innermost component of ATLAS. It consists of three barrels at radii of 4 cm, 10 cm and 13 cm and five discs on each side between radii of 11 cm and 20 cm. A total of 140 million pixels will provide the required resolution for the highly crowded LHC environment (50 ηm in Rø direction and 300 ηm in z). The PIX detector need to deal with two main constraints to ensure its operation during more than 10 years[ref]:

- The high power density of the pixel front-end electronics (up to 0.7W/cm^2, and $\sim$20KW total heat dissipation), requires a powerful cooling system.

- The harsh radiation environment (up to 300 Gy per year) puts another constraint on the design of the control system. As heat-ups of irradiated silicon detectors can cause irreparable damage to the sensor material, a thermal interlock system is implemented, acting on the power supplies and boards where active components like regulators will be located.

PIX Controls

The PIX detector consists of 1750 individual detector modules (Figure 2.10 shows an schematic view of a module). Each module has an optical link for the data transfer. From a control point of view a detector module is the smallest unit to act on. This modularity is determined by the different levels of radiation in the detector which rises the need of different voltages and currents to be applied to the different parts of the detector and by the amount and thickness of the cables needed.

- **Evaporative Cooling**: As the pixel detector is very sensitive to heat-ups, each module is equipped with an on-board Negative Temperature Coefficient (NTC) resistor for temperature monitoring and read out independently of the data acquisition system. The cooling systems is shared with the SCT detector and has been designed to evacuate 0.7 W/cm^2. It is based in a evaporative fluorocarbon (C_3F_8) [23], a nonflammable, nonconductive and radiation resistant liquid. An evaporation temperature of $\sim$-20°C in the on-detector cooling channels will allows a silicon operating temperature of $\sim$-6°C.

- **Power**: It uses commercial power supplies from the German company Iseg®. The system is able to take decisions autonomously, not relying on the functionality of a network and reducing the field-bus traffic in order to improve the safety of the detector. The requirements vary from the depletion bias of up to 700V to several low voltages, some with low power consumption, others drawing 2-5 A.

- **Interlocks**: A dedicated thermal interlock acting on the power supplies will be implemented to avoid damage to the detector in case of temperature increase due to the harsh radiation environment to which the pixel sensors are exposed. This is a pure hardware solution capable to operate in stand-alone mode or integrated within I/O modules. The signals from the 10 kΩ NTC sensors is compared to some given thresholds and sets, in case of large discrepancies it disables the power to either all or to a single module.

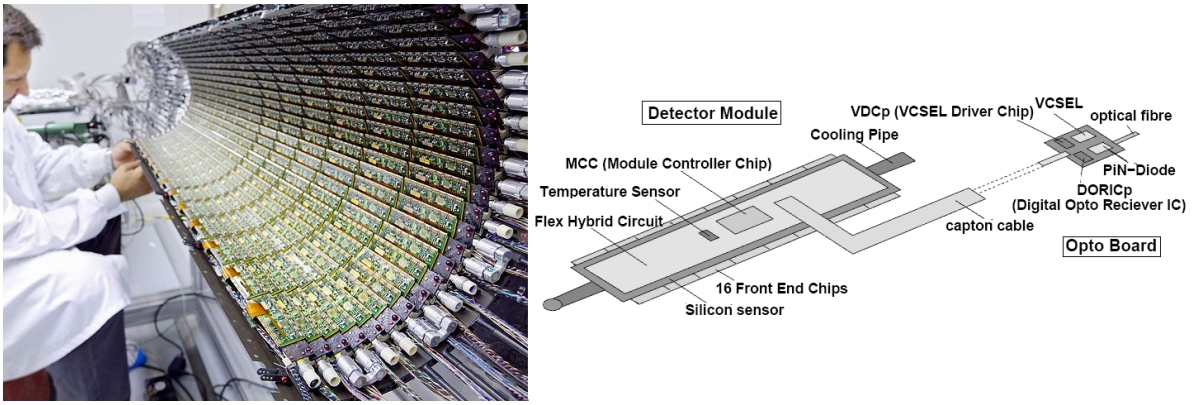


Figure 2.10: Left: Assemblage of half of the pixel tube located at the outer part of the pixel detector[13]. Right: Layout of a pixel detector module connected with an opto board. The detector module consists of a silicon sensor and 16 front end chips as well as a module controller chip gathering hit data and servicing trigger request. This is the smallest controllable unit. Picture taken from [24]

2.3.2 Silicon Strip Detector

The silicon strip detector (SemiConductor Tracker - SCT) consists of 4088 silicon microstrip modules which take up an area of 61 m^2 and counts with 6.3 million readout channels. As a result, the SCT gives a resolution of 16 ηm in Rø direction and 580 ηm in z. These are arranged into 4 concentric barrel layers at radii of 30.0, 37.3, 44.7 and 52.0 cm and 2 end-caps of 9 disks each. Most modules consist of 4 silicon sensors assembled in 2 daisy-chained pairs that are glued back-to-back with a small stereo angle (40 mrad) and bonded to the Front-End (FE) electronics hybrid. Each module has 1536 readout channels for physics data taking. Optical communication has been chosen to minimize the electrical pickup and to reduce the material. Figure 2.11 shows a picture of the detector modules and assemblage.

SCT Controls

Since the technology and location within the detector is similar to the Pixel sub-detector, both sub-detector have similar constraints from the control point of view. The amount of parameters to control is very large, each of the 4088 SCT modules requires several power lines for the electronics hybrid and silicon sensors, plus 3 fibers for the optical readout. Each of the electrical lines is individually controlled and monitored. Once again, the reliable operation of the cooling system is mandatory for the stable detector operation [25][26].

- **Evaporative Cooling**: The radiation-induced leakage current and doping changes affecting the depletion voltage of the silicon modules depend on the operating temperature. The operating temperature of the SCT varies from +15°C, while startup and commissioning, to -7°C, during operation in the ATLAS pit. Thermal stability of better than 2°C has to be ensured for all the operations.

- **Power**: Like the pixel tracker, the power system will control and monitor High Voltage (HV) and Low Voltage (LV) supplies. Each LV card controls 4 electrically independent LV channels and each HV card controls 8 channels, providing bias voltage (up to 500 V) to the

detector modules. The monitoring of voltages and currents is done at the power supplies with an expected precision of 2.5% of its nominal value.

- **Interlock**: It makes a direct hardware connection between the cooling system and the power supplies. An interlock signal is generated if the temperature on the cooling pipes are above a predefined threshold. In addition, it connects the readout system to the power supplies. The interlock ensures that the power is switched off if the optical fibres are disconnected from their receivers.

- **Environment**: A total number of almost one thousand sensors, placed at various points through the SCT, provide information on the environment conditions. NTC thermistors are used for the temperature monitoring while radiation hard Xeritron sensors are used for the humidity monitoring.

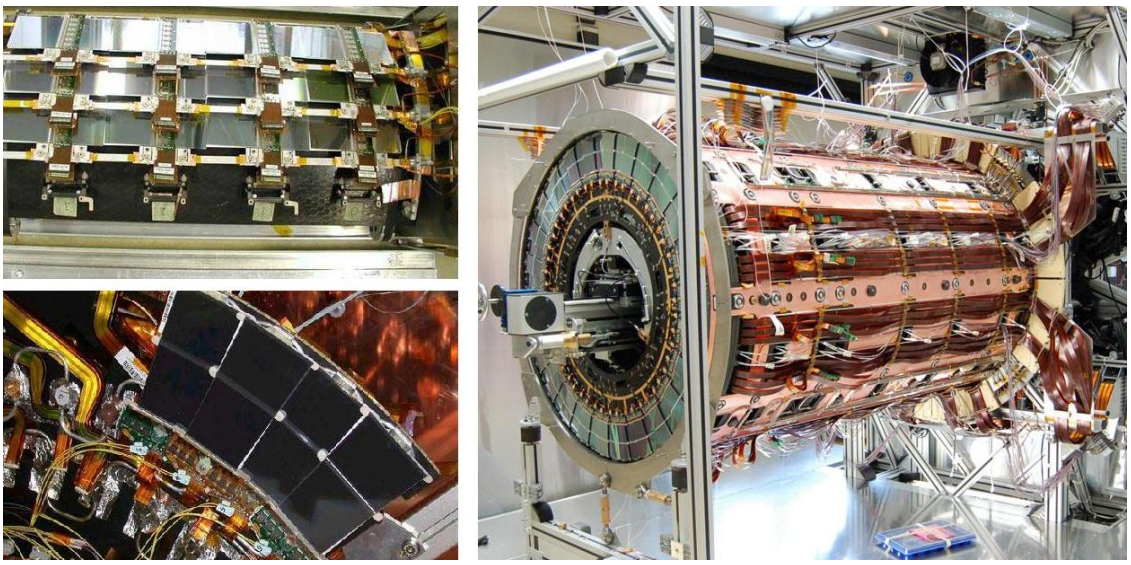


Figure 2.11: Left: Detail of both barrel (top) and end-cap (bottom) modules being assembled. Right: SCT detector with all the silicon disks installed. [13].

2.3.3 Transition Radiation Tracker

The transition radiation tracker (TRT) is divided into a barrel section and several wheels in each end-cap. It consists of 420.000 straws, each with a diameter of 4 mm. Transition-radiation photons are created in a radiator between the straws and are used for electron identification. The drift-time measurement gives a spatial resolution of 170 ηm per straw and two independent thresholds. These allow the detector to distinguish between transition-radiation hits and tracking hits. Figure 2.12 shows a picture of the half of a barrel completed and detail on the TRT straws.

TRT Controls

Being a gaseous detector, the controls of the TRT imply very specific features respect to the silicon detectors. The TRT forms the external layer of the IDE, facing in the internal side the cold SCT and

externally the cryostat wall. The mechanical construction optimized for low mass and robustness against radiation has to maintain good performance and mechanical stability over a big volume for a long period [27].

- **Cooling**: The cooling system uses two cooling fluids; gaseous CO_2 for cooling and ventilation of the TRT end-cap detectors, and water which removes heat from the gas produced by the ionization avalanche current and cools both front-end electronics and the barrel. Thus, the entire volume of the TRT is kept at the constant temperature of about 25°C.

- **High Voltage Supply**: It uses commercial power supplies from the Italian company CAEN®. The system is also needed for calibration purposes, the gas system and the HV are interfaced via software to obtain a stable gas gain. The system tunes the values of the high voltage applied to the straws to maintain constant gas amplification in a changing environment which is crucial for having good data for physics.

- **Low Voltage Supply**: It delivers low voltage in 2 - 8 V DC range to the front-end electronics. In total, 24 kW of electrical power over a distance of 40-50 m. The system has ∼3000 individual channels which are all monitored and controlled.

- **Temperature System**: The operational temperature of the TRT is of about 25°C. Standard resistive temperature sensors (Pt100 and Pt1000) with two-wire read-out measure temperatures within the TRT volume. The areas where the temperature is monitored are: a) detector mechanics (for stability of structures), b) detecting elements (for gas gain stabilization), c) electronics (for life time of the components), and, d) cooling circuitry both gaseous and liquid (for efficiency of the cooling and control of gradients). The temperatures of the gas will be monitored to ±0.5°C and those of the front-end electronics to ±2.0°C. The gradients along the detecting elements should not be higher than 10°C to ensure quality of performance. Moreover, the system also monitors temperatures external to the detector parts (i.e. cooling liquid and gas plants, electronics racks and power supplies).

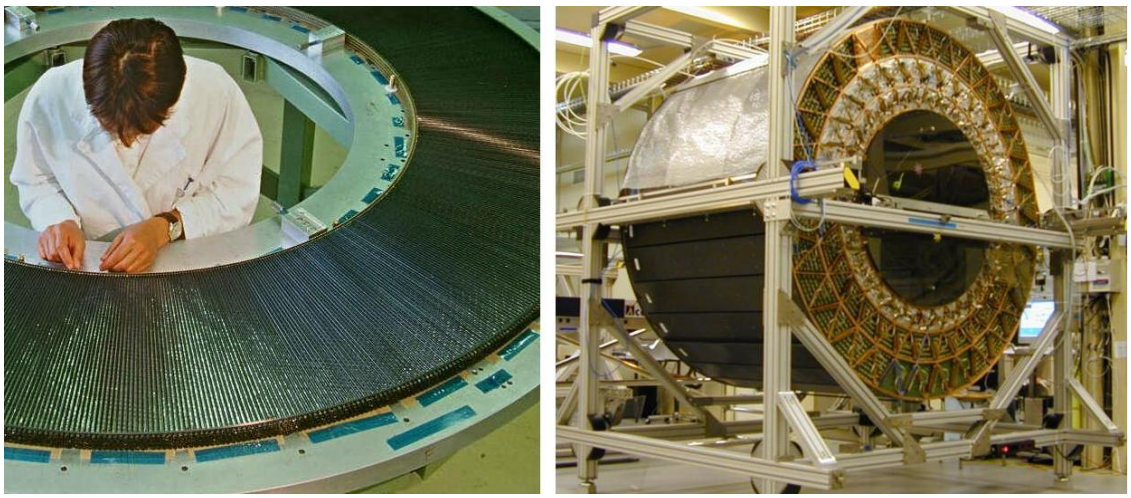


Figure 2.12: Left: Working on the TRT straws. Right: Barrel module completely assembled [13].

- **Gas System**: The TRT is operated with a special Xe-CO_2-CF_4 gas mixture. A primary concern is to obtain good performance at high occupancy and counting rates. This is achieved with a small straw diameter and with isolation of the sense wires within individual gas volumes. A stand-alone control system will monitor and control the quality of the gas and its distribution as well as its operating conditions (flow, pressure and status of the hardware) at different points.

- **Interlock system**: The interlock system plays a very important role in safety of detector operations. Many different failure modes require urgent intervention. Thus, hardware interlocks will exits between all the systems above mentioned. Moreover, the status of the interlock system hast to be carefully monitored.

2.4 Calorimeters

The calorimeter system in ATLAS is a composite detector based on the total absorption of particles to measure the energy lost by all the particles and jets formed in the collision. The energy measurement of calorimeters is based on the formation of a shower, a cascade of particles, when relativistic particles traverse dense matter.

The radiation and interaction length are material dependent. Generally, for dense materials, the interaction length is up to an order of magnitude longer than the radiation length. This property is used in the ATLAS calorimeters to separate the electromagnetic and the hadronic showers.

The calorimeter system consists of an inner electromagnetic and an outer hadronic calorimeter. The layout of the ATLAS calorimeters is shown in Figure 2.13.

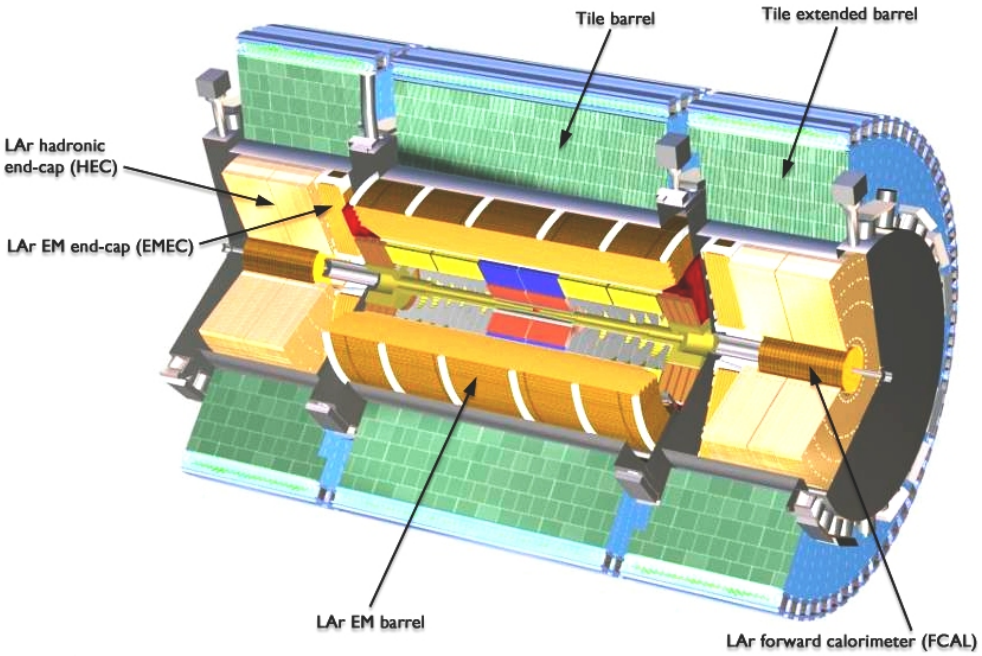


Figure 2.13: Layout of the ATLAS calorimeters. After the inner detector the next layer of detectors is formed by the electromagnetic calorimiter. The tile barrel and liquid argon end-caps of the hadronic calorimeter form the outer layer.[13]

The dense and compact electromagnetic calorimeter is built with accordion-shaped absorbers and liquid argon scintillating material in the barrel region and in the end-cap region. The electromagnetic calorimeter follows the outer envelope of the inner detector and is contained inside the ATLAS cryostat. The cryostat keeps the argon liquid at a cryogenics temperature of 84 K. It is cylinder-shaped with 5.5 m diameter and 7 m long.

The hadronic calorimeter follows the electromagnetic calorimeter and is split into different subsystems applying different technologies. In the barrel region a tile calorimeter is located with iron absorbers and plastic scintillator as active material. The hadronic calorimeter in the end-cap is based on copper absorbers and liquid argon as active material. A particulary challenging detector is the forward calorimeter close to the beam pipe, due to the high radiation level of this region. It also uses liquid argon as sensitive medium.

The principle of functioning and the controls of the calorimeter is briefly discussed in the following sections. From the point of view of controls, the grouping of the different calorimeters systems has been done following the two main used technologies: tile and liquid argon.

2.4.1 Liquid Argon Calorimeter

Based on Liquid ARgon (LAR) as a sensitive medium the calorimeter is divided into several components:

- **LAR ElectroMagnetic Calorimeter** (EM): This is an electromagnetic sampling calorimeter with 'accordion-shaped' lead electrodes in the barrel and in the end-caps (see Figure 2.14). The barrel EM calorimeter is placed inside a barrel cryostat, which surrounds the IDE. The central solenoid which provides the magnetic field for the IDE is integrated in the barrel cryostat in order to minimize the material in front of the EM calorimeter and to achieve the desired calorimeter performance. The end-cap EM calorimeters are contained in two end-cap cryostats together with the hadronic end-cap, EM calorimeter and the forward calorimeter. In total, for physics data taking, the EM has 110.208 read-out channels in the barrel region and 63.744 in the end-caps.

- **LAR Hadronic End-Cap Calorimeter** (HEC): The HEC modules are located in the end-cap cryostats at either end of the IDE volume and share the cryostats with the end-cap EM and forward calorimeters. The HEC sits behind the end-cap EM and is completely shadowed by it. It is a sampling calorimeter with some flat copper absorbers. The HEC consists of two doughnut-shaped wheels which each weigh about 67 tons and consist of 32 wedge-shaped modules each. The outer radius of the wheels is 2090 mm. All in all, the HEC has about 8600 read-out channels for physics reconstruction. Figure 2.14 shows also a recent picture of the HEC detector.

- **LAR Forward CALorimeter** (FCAL): On each side, the FCAL consists of three modules. An additional fourth module is realized as a passive plug which serves to shield the forward muon chambers. All modules are located within the end-cap cryostat. Keeping in mind that the radiation deposited in the FCAL is huge, the device has been designed as a mechanically simple, high-density and radiation resistant detector. The first module (FCAL1) is an electromagnetic calorimeter consisting of copper; FCAL2 and 3 are hadronic calorimeters built from tungsten/tungsten alloy. All modules have an outer radius of 455 mm and a length in z of 450 mm. Mechanically, the modules consist of single absorber matrix bodies which carry arrays of tube electrodes in holes in the matrix body. The tube axes are oriented along the

beam line. FCAL1 holds about 12.000 tubes; FCAL 2 and 3 holds about 10.000 and 8000 tubes respectively.

In addition, presamplers consisting of one layer of LAR in front of the electromagnetic calorimeter help to correct for the energy loss in front of the calorimeter (mainly due to cryostat walls and the barrel solenoid).

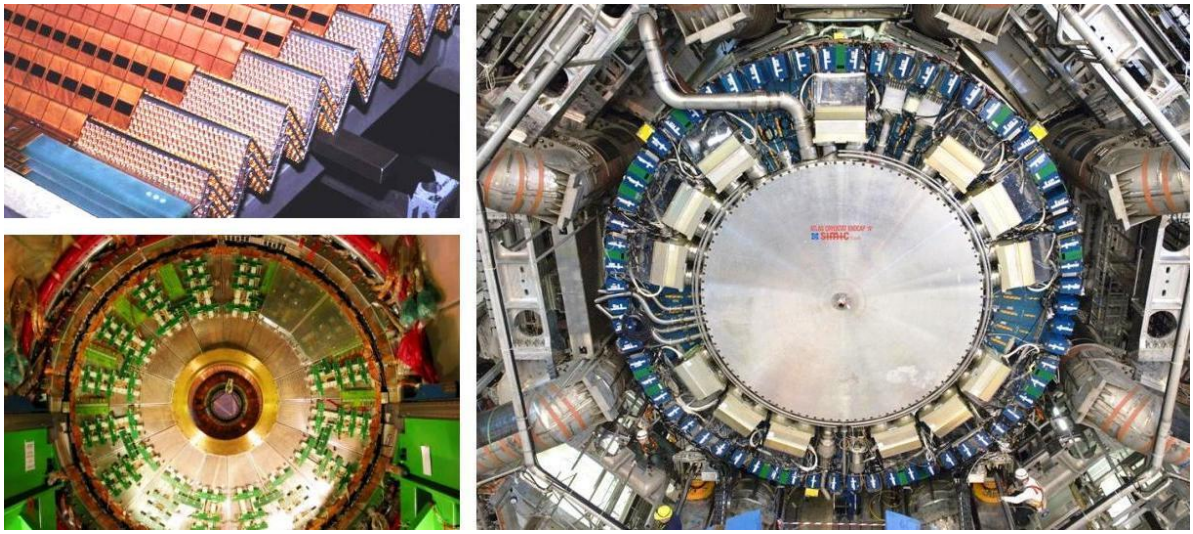


Figure 2.14: Left in the top, detail of the EM calorimeter accordion-shaped absorbers. Left in the bottom: back of the rear HEC wheel (in the inner opening the FCAL assembly will be installed, in the outer part the cryostat surround the HEC. Right: End-cap calorimeter installed in the ATLAS cavern.)[13]

LAR Controls

- **Cryogenics**: The response of the liquid argon calorimeters depends on the temperature and varies roughly by 2% per Kelvin. In addition, there is a large amount of material to cool, the cryostat vessel weight is about 600 tons, including the calorimeters, and about 80 tons of liquid argon. A 25.000 l storage dewar filled by a liquid nitrogen liquifier has been put in place close to USA15. The cryogenic system is designed to allow a maximum cool down time of 30 days, transfer of LAR into the cold cryostat in less than 24 hours, transfer of LAR out of the cryostat in less than 2 hours and a maximum warm-up time of 30 days. Cooling down to LAR temperature is achieved by circulation of helium gas cooled in an external and moveable liquid nitrogen heat exchanger. Stable operation at a temperature of 84 K is ensured by a regulated flow of liquid nitrogen through a set of coils.

- **Front-End Crates Power System** (FEC): The 58 LAR FEC are located directly on the detector (16 at each end of the LAR barrel, and 13 at each end of the EC) and read-out around 190.000 calorimiter channels for physics reconstruction. A total of 7 voltages and currents, 4 temperatures probes measuring the cooling water and a number of flowmeters are monitored for each crate.

- **High Voltage**: About 150 HV power supply units are located in racks in USA15. The full set-up for the HV system in LAR is made up of more than 5000 channels that will feed the absorbers of the detector.

- **HEC Low Voltage**: The hadronic EC (HEC) has active electronics inside the EC cryostats, on the modules themselves. The low voltages are produced by DC/DC-converters in the power boxes and split into 320 channels, each of them containing a +7.2V, +3.3V and -1.6V line.

- **High Precision Temperature System**: The accuracy needed to preserve the energy resolution of the calorimeter is very fine, around +-50 mK. The temperature gradients inside of the cryostats are monitored by 38 temperature monitor boards and 508 PT-100 probes, which are connected by a 4-wire readout. In addition, the temperature monitoring allows to prevent large deformations of the cryostat due to temperature gradients during cool-down and warm-up in different parts of the calorimeter.

- **Purity Monitors**: LAR detectors need to monitor the purity of the liquid argon since electronegative impurities (mainly O_2) capture ionization electrons, and hence degrade the performance of the detector. In order to have a good performance for physics reconstruction, it is necessary to purify the liquid argon from to a level below 0.3 ppb (oxygen equivalent). The monitors are essentially mini-calorimeters containing radioactive sources, and the responses to different types of sources can be used to deduce the liquid argon purity.

2.4.2 Tile Calorimeter

The TILe Calorimeter (TIL) uses a sampling technique with iron as absorber and scintillating tiles as active material. The tiles are placed perpendicular to the beam axis and are staggered in depth. The periodical structure consists alternately of a 3 mm thick tile and a 14 mm thick iron plate. Two sides of the scintillating tiles are read out by wavelength shifting fibers into two separate photomultipliers. Figure 2.15 shows the operating principle of a scintillator/photomultiplier detector.

The calorimeter is divided into one barrel and two extended barrels, each of the three made up of 64 modules. The gap between barrel and extended barrel provides space for cables, feedthroughs and service pipes.

TIL Controls

- **High Voltage**: The HV system of the TIL calorimeter feeds the photomultipliers with a voltage ranging from 400 to 1000 V and a current of 20 mA per channel. Each of the 256 modules (or super-drawers) of the calorimeter is equivalent from a HV point of view.

- **Low Voltage**: The LV system feeds two main components of the drawers: the HV distributor electronics and the readout front-end. The analogue readout electronics, i.e. motherboards, are very sensitive and the peak-to peak noise level must be kept below 2 mV.

- **Cesium System**: The system is only used for quality check and inter-calibration of the TIL modules. It only runs during long shutdowns ($\geq$ 8 hours) and does not permit any action during a physics run. A movable radioactive source (Cesium) is carried along by a liquid flow and travels through the calorimeter body deposing a known energy to the calorimeter cells. The corresponding variation of a photomultiplier current reflects the tiles and fibers optical quality in the cells.

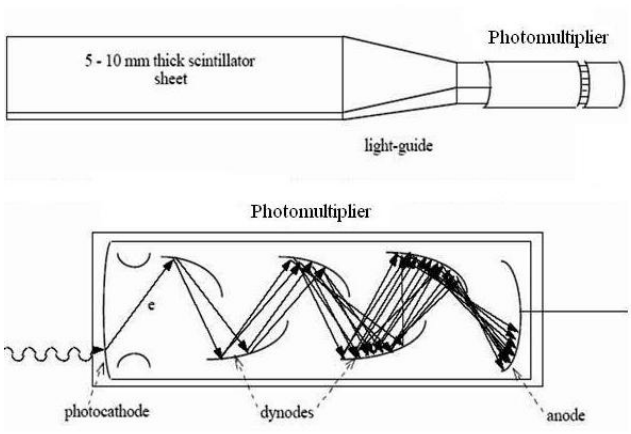


Figure 2.15: Left: Schematic diagram of a Scintillator/Photomultiplier. Certain materials, when struck by radiation, emit a small flash of light (i.e. a scintillation). When coupled to an amplifying device, such as a photomultiplier, these scintillations can be converted into electrical pulses. A main characteristic of scintillators is that the light output is directly proportional to the exciting energy. This, together with a very fast time response made them one of the most often and widely used particle detection devices in nuclear and particle physics today. Picture taken from [28]. Right: Working on the Tilecal barrel.

- **Cooling System**: The large iron mass of the TIL calorimeter contributes to the temperature stability inside the modules, but a dedicated cooling system must evacuate heat dissipated by the electronics, maintaining stable local temperatures to within 1°C. A stable temperature inside the drawers is required for stability of the electronics, gain stability of the photomultipliers, and cesium calibration. A system of heaters compensate for the temperature gradient in the cavern. Moreover, by means of water circulating the low voltage power supplies that feed the tile drawers are cooled.

- **Minimum Bias** (MB): The MB scintillator is made of a proprietary blend of polystyrene doped with a small amount of fluorescing agent. Each MB scintillator should provide enough light to detect a single minimum ionizing particle with high efficiency. Due to radiation damage during the lifetime of operation, the light budget should allow a degradation of the light output by a factor of 2.

2.5 Muon Spectrometer

The ATLAS muon spectrometer measures the magnetic detection of muon tracks in the three large superconducting air-core toroid magnets. The layout is shown in Figure 2.16. For this measurement it uses two types of trigger chambers and two types of high-precision tracking chambers.

The magnetic field for the muon spectrometer can be seen consisting of three parts. In the barrel region the magnetic field is produced by the large barrel toroid. In the end-cap region the tracks are bent by two smaller end-cap magnets. In the transition region between barrel and end-cap the magnetic bending is provided by the combination of the other two magnetic fields.

In the barrel region the muon chambers form three cylinders concentric with the beam axis. They are positioned at radii of about 5, 7.5 and 10 m. The end-cap chambers are arranged in four

vertical discs at distances of 7, 10, 14 and 21-23 m from the interaction point. The outer muon chambers define the size of the ATLAS detector.

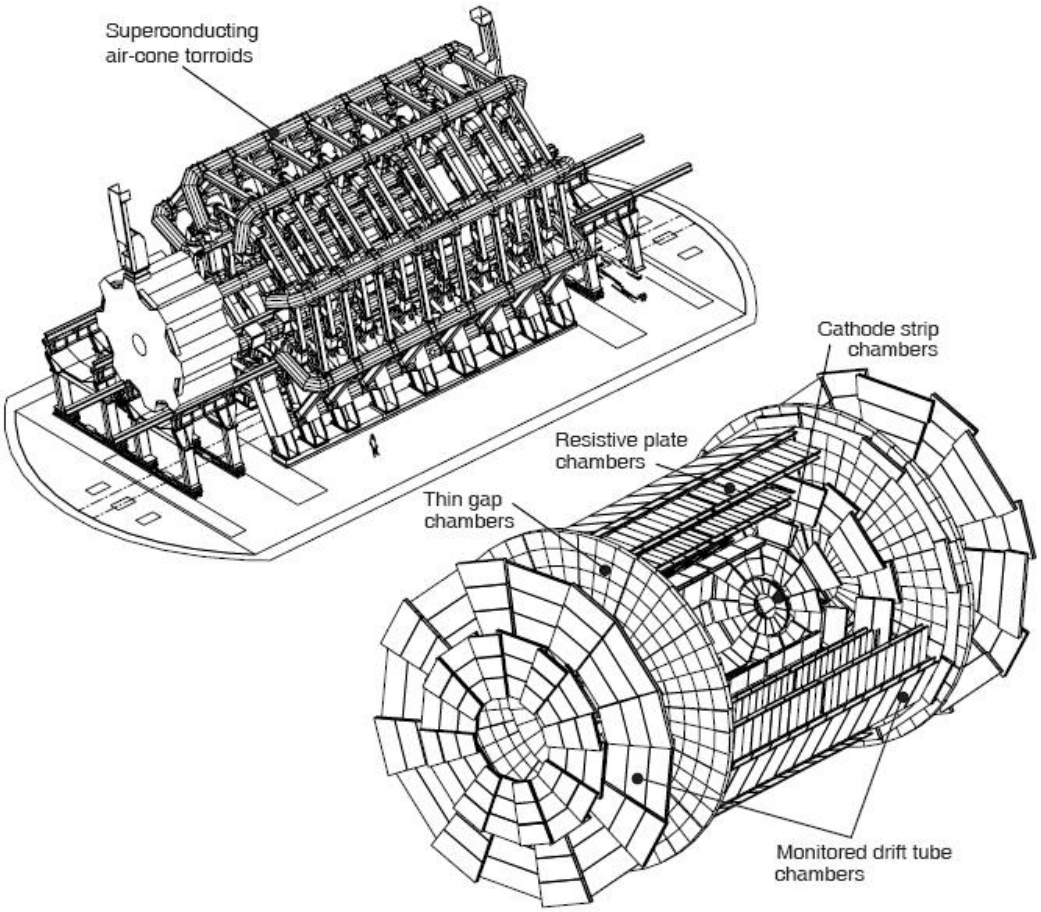


Figure 2.16: Layout of the muon spectrometer. It uses four different chamber technologies to measure the deflection of muon tracks in the magnetic filed created by the three superconducting air-core toroids.

The high-precision muon track measurement is provided by the Monitored Drift Tubes (MDT). Close to the interaction point, Cathode Strip Chambers (CSC) are used because of the demanding rate and background conditions. For the trigger system Resistive Plate Chambers (RPC) are used in the barrel and Thin Gap Chambers (TGC) in the end-cap regions.

Operating Principle of Gaseous Detector

The four sub-detectors that compose the muons spectometer share the same operating principle, they are gaseous ionization detectors which work at room temperature and pressure. Ionization detectors were the first electrical devices developed for radiation detection. These instruments are based on the direct collection of the ionization electrons and ions produced in a gas by passing radiation. Because of the greater mobility of electrons and ions, a gas is the obvious medium to use for the collection of ionization from radiation.

The basic configuration consists of a container with conducting walls and a thin end window (see Figure 2.17). The cylinder is filled with a suitable gas, usually a noble gas. Along its axis is suspended a conducting wire to which a positive voltage, +Vo, relative to the walls is applied.

A radial electric field is established. If radiation now penetrates the cylinder, a certain number of electron-ions pair will be created, either directly, if the radiation is a charged particle, or indirectly through secondary reactions if the radiation is neutral. The mean number of pairs created is proportional to the energy deposited in the counter. Under the action of the electric field, the electrons will be accelerated towards the anode and the ions towards the cathode where they are collected.

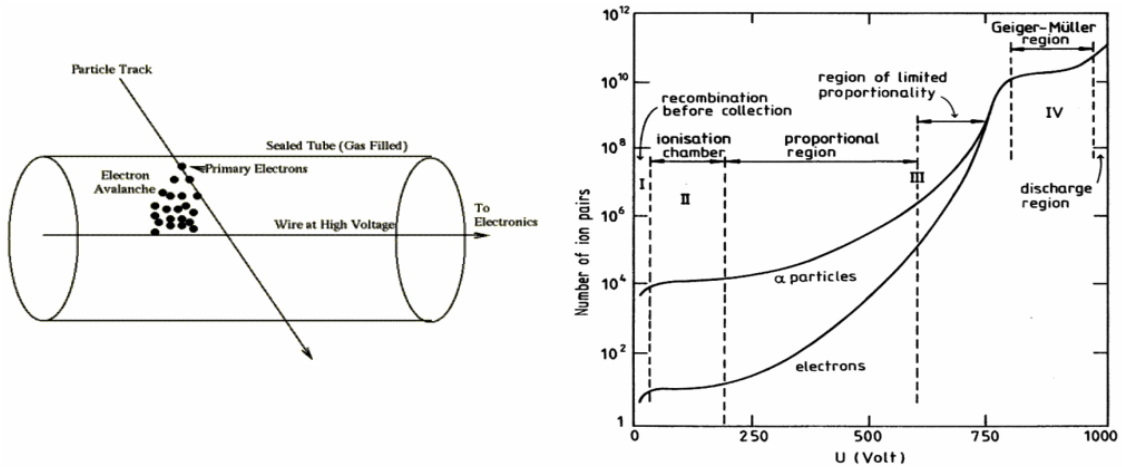


Figure 2.17: Left: Operating principle of a gaseous ionization detector. When a radiation penetrates the gas filled cylinder, a certain number of electron-ions pair is created. Under the action of the electric field, the electrons are accelerated towards the anode and the ions towards the cathode. The mean number of pairs created is proportional to the energy deposited in the counter. Right: Number of ions collected versus applied voltage in a single wire chamber. The voltage applied to the anode determines the operational mode of the gaseous detector. Picture taken from [28].

Muons Controls

Different parameters have to be considered for a reliable operation of the spectrometer. Temperature gradients and vibrations can affect the performance of the detector. In addition, the large field of the toroidal magnets requires significant precautions, notably in the design and layout of the electrical equipment and electronics installed on or near the chambers.

The sub-detectors forming the muon spectrometer present common needs and can be arranged in the same manner from the point of view of controls. The main systems to control each of the sub-detectors that form the muon spectrometer are discussed below.

- **Alignment System**: The relative position of the several thousand of muon chambers is essential for the good performance of the spectrometer. To achieve the required momentum resolution a stabilization of the dimensions and positions of the chambers at the 30 μm level has to be provided.

  However, due to the large dimensions of the spectrometer, it will not be possible to keep stable the geometry of the chambers and their position in the spectrometer.

  The approach chosen to cope with movements is to measure their effect on the measurements of tracks and to correct for them off-line using the information from the alignment system which monitors the chamber deformations and positions (no attempt is made at

physical chamber repositioning). The alignment of the CSC, RPC and TGC chambers will be performed using resistive proximity sensors; the MDT modules uses an optical alignment system. This alignment system, consist of different arrangement of Charge-Coupled Device (CCD) cameras, lens and masks providing a precision in the order of tens of μm. A dedicated processor reads the images of CCD cameras and calculates the alignment constants of the individual chambers.

- **Gas System**: The gas systems provides the different gas mixtures needed for the operation of the four sub-detectors of the muon spectrometer. The gas mixture must be supervised and then analyzed to provide feedback on the mixing to the system. The same analysis station also monitors the return gas. The mixed gas is delivered to gas racks residing in the detector hall. Each of these racks supplies a number of chamber cluster of the different detectors. Each cluster gas supply is monitored and controlled. Additionally, it is monitored the pressure and flow at different points of the installation, and the oxygen and water contamination of the mixture.

- **Environmental**: Temperature gradients can affect the uniformity and stability of the drift velocity (velocity of the electrons coming from the ionizing event), the gas gain across the detector and the mechanical deformations of the chambers. Thus, the tracking chambers must be continuously monitored. In addition, for the RPC and TGC chambers, monitoring of the temperature of experimental hall at the gas distribution manifolds and of the front-end electronics is performed. All the temperatures are monitored with a resolution of 0.5°C by NTC thermistors.

  Environment monitoring in terms of radiation level in the vicinity of the detector, magnetic field using Hall probes, humidity and commonly provided services are also required.

- **Power System**: The HV system works in a close interaction with the gas and environment systems. The working voltage scales proportionally to the inverse of the gas density, for small density variations, thus both the knowledge of the environmental condition and the high voltage distribution must be granular enough to guarantee the detector working point uniformity. For example, the MDT gas mixture, Ar-CO_2 93% - 7% with $2{\cdot}10^4$ gas gain, demands a working voltage of 3080 V which can increase up to 3.6 kV using other gas mixtures or increasing gas gain. In contrast, TGC using CO_2-ηC_5H_{12} 55% - 45% has a working voltage of 2.9 kV.

  In addition a LV system supplies the front-end and readout electronics. In the case of RPC and CSC the power system hardware is all based on the industrial CAEN EASY technology which integrates in the same power crates all the needed functions: HV, LV, ADC, DAC.

- **JTAG**: A digital interface allows setting of threshold voltages, bias currents and loading of various parameters in order to improve the detector performance online when data taking.

2.5.1 Monitored Drift Tube chambers

The MDT is composed of a total of 1172 chambers covering an area of 5500m^2. In total, 656 of these chambers are placed in the Barrel and 516 in the end-caps. The MDT chambers consist of a 30 mm diameter aluminum tube with a central W-Re wire. Figure 2.18 shows the operational principle of a drift chamber. The MDT chambers are operated at 3 bar absolute pressure with a non-flammable mixture of Ar-CO_2. They provide a maximum drift time of $\sim$700 ns, excellent

ageing properties and a single-wire resolution of ∼80 μm. The tube lengths vary from 70 to 630 cm and the tubes are positioned orthogonal to the R-z plane in both the barrel and end-cap region. That allows a high-precision measurement of the axial coordinate (z) in the barrel and the radial coordinate (R) in the transition and end-cap region. A total of 372 000 channels are read out during physics data taking.

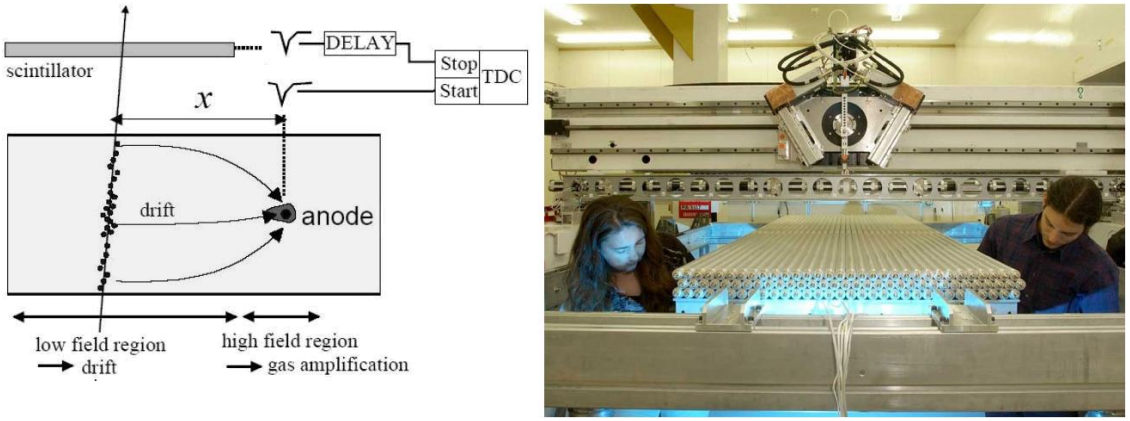


Figure 2.18: Left: Drift Chamber. The drift cell is defined at one end by a high voltage electrode and at the other end by the anode of a simple proportional counter. To signal the arrival of a particle, a scintillation counter is placed before and after the chamber. A particle traversing the chamber and scintillator, now, liberates electrons in the gas which then begin drifting towards the anode. At the same time, the fast signal from the scintillator starts a timer. The signal created at the anode as the drifting electrons arrive then stops the timer to yield the drift time. Picture taken from [29]. Right: Working on the assembly of MDT tubes [13].

2.5.2 Cathode Strip Chambers

The CSC is composed of a total of 32 chambers, 16 for each end-cap. The CSCs are multi-wire proportional chambers with symmetric cells. Figure 2.19 shows the operational principle of a multi-wire proportional chamber. The cathode is segmented into strips orthogonal to the anode wires. Due to the avalanche effect around the anode wire, charge is induced into the cathode and by charge interpolation between neighboring strips a high-precision measurement can be accomplished, resulting in resolutions better than 60 μm. The chambers are operated with a non-flammable mixture of Ar-CO_2-CF_4. The measurement of the transverse coordinate is performed by the second cathode which consists of strips parallel to the anode wires.

2.5.3 Resistive Plate Chambers

The RPC is composed of a total of 1088 chambers covering a total surface of about 3500 m^2. The RPCs consist of a narrow gap formed by two parallel resistive plates. The gap is filled with non-flammable gas based on $C_2H_2F_4$. Through a high electric field between the plates, primary ionization electrons are multiplied into avalanches and form a current signal. The signal is read out via capacitive coupling of metal strips on both sides of the chamber. Similar to the CSCs the cathode strips of one side are orthogonal to the ones on the other side to achieve a two coordinates measurement. A total of 383 000 channels are read out for physics reconstruction.

2.5.4 Thin Gap Chambers

The TGCs are similar to multiwire proportional chambers but have slightly different dimensions. The chambers are operated with a highly flammable gas mixture of CO_2-ηC_5H_{12}, therefore adequate safety precautions have to be taken. The electric field configuration and the small dimensions provide a short drift time and thus a good time resolution. 440 000 channels are read out during physics data taking. Figure 2.19 below shows a recent picture of the TGC assemblage.

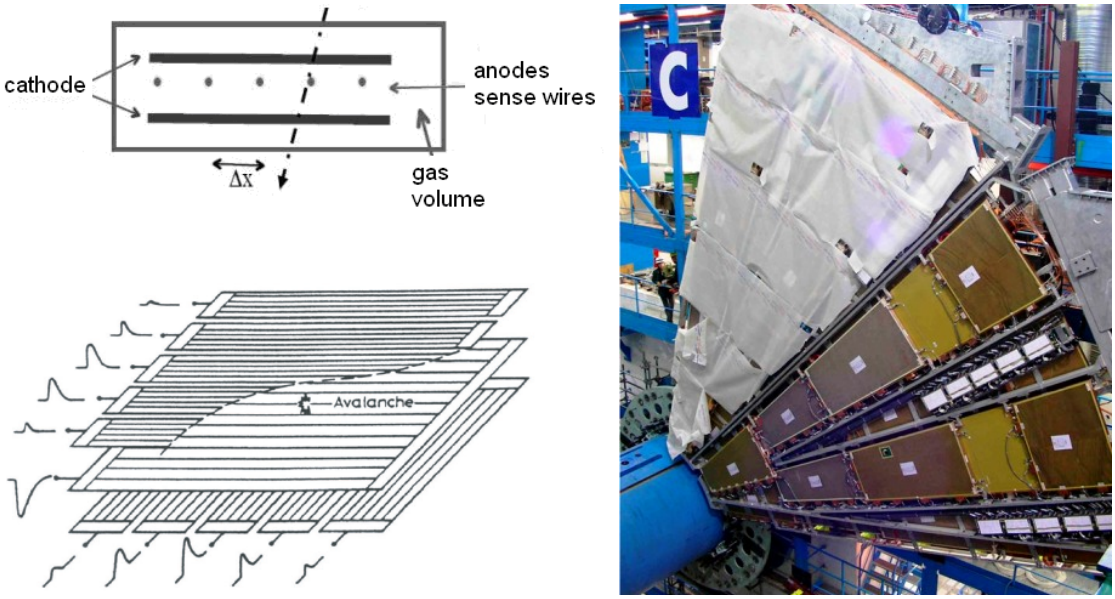


Figure 2.19: Left: Multi-Wire Proportional Chambers are detectors for position sensing. The basic structure consists in an anode sandwich on cathode bread. When a negative voltage is applied to the cathode planes a electric field arises. Except for the region close to the anode, the field lines are parallel and almost constant. When a particle cross the detector, electrons and ions are liberated in the constant field region drifting along the field lines toward the nearest anode wire. Upon reaching the high field region, the electrons are accelerated to produce an avalanche. The positive ions liberated in the multiplication process then induce a negative signal on the anode wire. In a similar manner, a positive signal is induced on the cathode. The signal from one anode plane gives information on one coordinate of the ionizing event, the second coordinate is obtained by using a second detector whose anodes wires are oriented perpendicularly to the first. Picture taken from [29]. Right: Second section of the TGC being assembled.

2.6 Trigger and Data Acquisition System (TDAQ)

Leaving aside the different sub-detectors systems, which will be later supervised by the DCS, this section introduces briefly the TDAQ (Trigger and Data Acquisition) system. The TDAQ system drives the overall experiment to the physics data-taking process and, together with the DCS, it can be stated that they represent the two main integration systems of ATLAS. Consequently, for the final operation of the experiment, it is required the synchronization of both systems. This interaction will be discussed in detail in Chapter 7.

2.6.1 The Trigger System

The trigger system selects bunch crossings containing interesting interactions. The bunch crossing rate at LHC will be of 40 MHz, and at design luminosity, there will be about 23 interaction per bunch crossing leading to an interaction rate of about 10^9 Hz. The online triggering system will be capable of selecting interesting physic signatures to approximately 100 Hz. Therefore, within seconds the data flowing from the detector has to be reduced by a factor $\sim 10^7$ while retaining an excellent efficiency for the rate for new physics, such as Higgs boson decays. This is achieved by defining different trigger levels (LVL1, LVL2 and LVL3 also called Event Filter) as shown in Figure 2.20.

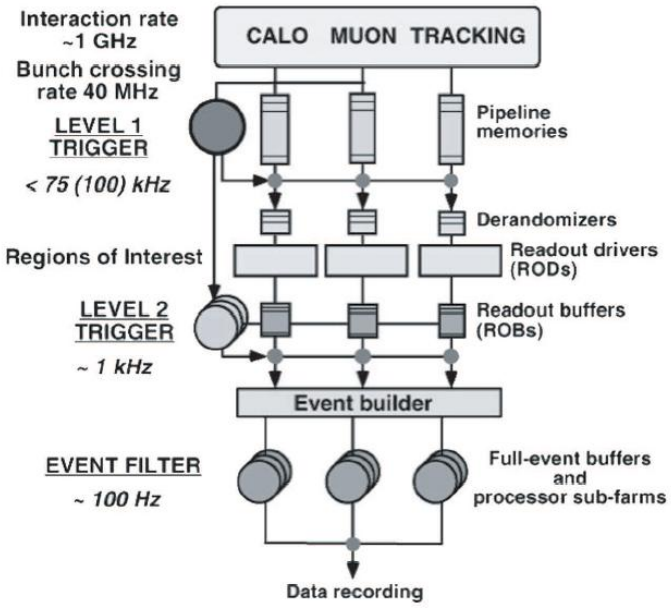


Figure 2.20: Block diagram of the ATLAS trigger and data acquisition system (TDAQ). The ATLAS TDAQ is based on three levels of online event selection. Each level refines the decisions made at the previous level and, where necessary, applies additional selection criteria. Starting from a initial bunch-crossing rate of 40 MHz, the rate of the selected events must be reduced to 100Hz for permanent storage.

The LVL1 trigger uses reduced-granularity data from a subset of detectors (muon trigger chambers and calorimeters). The LVL1 trigger accepts data from these detectors at the full LHC bunch-crossing rate of 40 MHz (every 25 ns). At this stage, the sub-detectors are treated individually. The latency, which is the time to form and distribute the LVL1 trigger decision, is 2 μs and the

maximum output rate is limited to 100 kHz by the capabilities of the sub-detector readout systems and the LVL2 trigger.

During the LVL1 processing, the data from all parts of the ATLAS detector are held in pipeline memories of the front-end electronics. The LVL1 trigger must identify unambiguously the bunch crossing containing the interaction of interest and introduce negligible dead-time. The LVL1 trigger decision is based on combination of objects required in coincidence.

Events selected by LVL1 are read out from the front-end electronics systems of the detectors into Read Out Drivers (RODs) and from there forwarded to Read Out Buffers (ROBs). At this states, full granularity and full precision data from most of the detectors is available. To minimize the latency, only data from regions of interest, defined by LVL1, are transferred to the trigger processors. The LVL2 trigger reduces the rate from about 100 kHz after LVL1 to 1 kHz with a latency ranging from 1 to 10 ms depending on the event.

All data for the selected bunch crossing from all the detectors are held in the ROBs until the LVL2 trigger takes the decision to either discard the event or to accept it.

After an event is accepted, the full data are sent to the Event Filter processors via the event builder. The Event Filter trigger uses the full event data together with the latest available calibration and alignment to make the final selection of events for the off-line analysis. At LVL3 a complete reconstruction is possible with decision times up to about 1 s. The Event Filter must achieve a data storage of 10-100 MB/s by reducing the event rate and the event size.

2.6.2 The Data AcQuisition System

The DAQ system handles the distribution of data from the ROD to mass storage and the overall monitoring and control of the data taking. For this reason, the system has been factorized in two major components: DataFlow and Online Software[30].

The *DataFlow* provides the functionality of receiving and buffering detector data from the ROD, distributing events to the High Level Triggers (HTL) and forwarding selected events to mass storage. These functions are handled by the 4 components of the DataFlow subsystem:

1. *ReadOut*: it receives and buffers data coming from the ROD.

2. *LVL2*: it manages flow of events and control messages within the second level event selection system.

3. *Event Builder*: it collects all data fragments corresponding to the same bunch crossing from the ROD and builds a complete and formatted event.

4. *EF I/O*: it passes the events assembled by the Event Builder to the Event Filter and sends the selected events to mass storage.

The *Online Software* system controls the overall experiment. It provides run control, configuration of the HLT and DAQ system and manages data taking partitions. This component is essential for the coordination with the DCS since it constitutes the interface point between both systems.

Next it is discussed the envisaged overall experiment control. This is a brief discussion that intends to put in context the operation of ATLAS (driven by the DCS and the TDAQ systems) within the LHC accelerator.

2.7 Overall Experiment Control

The overall control of the ATLAS experiment includes the monitoring and control of the operational parameters of the detector and of the experiment infrastructure as well as the supervision of the software involved in the event readout. This functionality is provided by two independent but interacting systems that perform complementary functions, as shown in Figure 2.21. The Online Software controls all TDAQ processes needed for physics data taking. The DCS handles the control of the detector hardware and the related infrastructure.

The TDAQ and the DCS systems have different operational requirements. Whilst the DAQ control is only required during physics data taking or calibration procedures, the DCS has to function with no interruption to ensure the safe operation of the detector. However, an intense interaction between both systems is of prime importance for the coherent overall operation of the experiment.

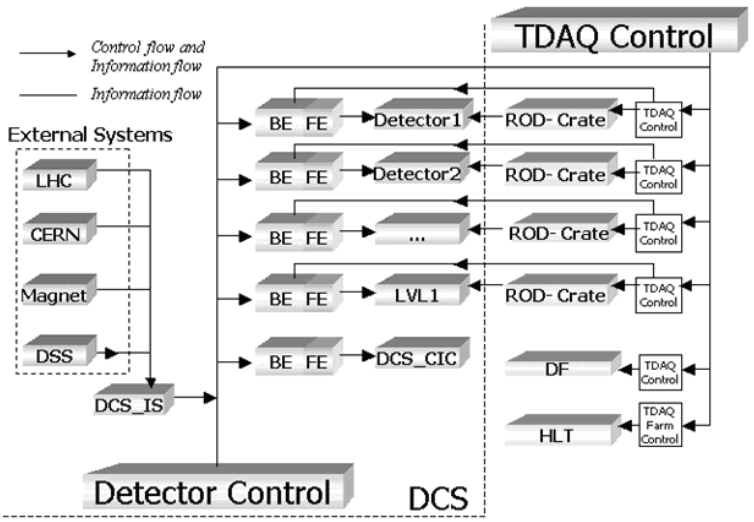


Figure 2.21: The overall control of the ATLAS experiment includes the monitoring and control of the operational parameters of the detector and of the experiment infrastructure as well as the supervision of the software involved in the event readout. Picture taken from [31].

Watching the experiment at the top control abstraction level, a third element would be involved in the experiment control. This is of course the LHC accelerator. An example of this inter-relationship is illustrated in Figure 2.22.

At the global operation level of the experiment, only two states of the DCS would be relevant to TDAQ, these are: *DCS Ready*, the detector can be used for data taking and; *DCS Not Ready*, the detector cannot be used for data taking.

On the other hand, for TDAQ purposes, only three conditions could summarize the state of the LHC machine, these are: *Non Stable Beam*, the LHC accelerator is not in conditions which enable the operation of ATLAS; *Detector Safe*, the LHC machine is in conditions which enable the safe operation of the detector, but beams are not available and; *Stable Beam*, the LHC machine is in conditions which enable the safe operation of ATLAS and beams are available.

Next chapter discusses the ATLAS DCS, the main system related to the author's work.

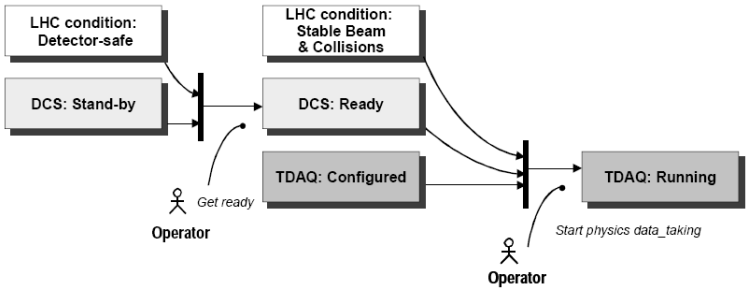


Figure 2.22: Inter-relationships of the experiment conditions, the detector state (via DCS) and the TDAQ system. Stand-by, in this example, is an internal DCS state representing the condition where the detectors' high-voltage, for example, is not ramped up. Picture taken from [32].

Chapter 3

The ATLAS Detector Control System (DCS)

Different hardware and software systems are needed to control the ATLAS experiment. One of the main tasks of the DCS work during the last years has been to provide such systems, including the supervisory control and the distribution of the processes' automation, that will enable the coherent operation of ATLAS.

This chapter describes the scope, the architecture and the building blocks of the ATLAS DCS. The chapter begins discussing the evolution occurred in controls during the transition from the LEP to the LHC Hera. Later, the description of the different components used for the DCS are organized as follows: (1) front-end, (2) communications, and (3) the back-end. At the end, there is a discussion about back-end controls outside the HEP world, the major vendors in the industrial control market today are listed, and some proprietary integrated engineering platforms are discussed briefly.

3.1 Introduction to the DCS - A Short Story

Although the experience gained during LEP accelerator (1989-2000) is an important input to the design of the detector controls, the next generation of detectors for the LHC experiments brings further requirements onto the DCS due to their complexity and size. Furthermore, taking into account the technological evolution between the LEP and the LHC era, it implies the necessity of re-engineering the controls of the experiments.

In the 1960s, the 4-20 mA analogue signal standard was introduced for instrumentation. The development of digital processors in the 1970s sparked the use of computers to monitor and control a system of instruments from a central point. At that time, the specific nature of the tasks to be controlled called for instruments and control methods to be custom designed. In the 1980s smart sensors began to be developed and implemented achieving a digital control. This prompted the need to integrate the various types of digital instrumentation into field networks and, consequently, fieldbus standards were developed to formalize the control of smart instruments. During the 1990s the Supervisory Control And Data Acquisition (SCADA) systems evolved allowing full distributed control facilities using the Internet more as a communication tool.

Having as a main reference the LEP experiments, one main problem was the lack of standardization in many areas. Many different programming languages, custom hardware and protocols were employed due to the technical scenario at that time. Therefore, the development and maintenance during the lifetime of the experiment was, in same cases, difficult. New 'plant' constructions,

upgrades of existing facilities, or long-planned 'plant' expansions took plenty of time, money, and control expertise. During the evolution of LEP then many things were learnt, two examples are[33]:

- The re-engineering during the nineties of the L3 Muon control system[34] showed how industrial products can be interfaced into an existing experiment, in this case, into an experiment where, some years before, it was necessary to write custom code.
- The NA48[35]experience in 1998 underlined how the object-oriented paradigm maps well onto external hardware. Each device being controlled was represented as a software object in the front-end. At that time, using a SCADA package, the NA48 experiment was able to implement not only the user interface but also its alarm handling, trending and logging capabilities. Consequently, the construction of the DCS using a SCADA tool typically required configuration-style work rather than detailed coding.

In the mid-90s, when the engineering of the new LHC experiments was started, and having the experience gained at LEP, a decision had to be taken. The engineering of the controls for the LHC experiments could be implemented in-house, through a system integrator, and/or in conjunction with a process automation suppliers. The final decision taken at CERN relied as much as possible in Commercial Off-The Shelf (COTS) components (e.g PLCs, fieldbuses or SCADA products) while keeping a certain degree of freedom through the implementation of an integrated engineering platform suited for the specific requirements of a HEP experiment. This integrated engineering platform was later implemented within the context of the Joint COntrols Project (JCOP) [36].

The JCOP at CERN was set up at the end of 1997 to address common issues related with the controls of the LHC experiments. The LHC will have in total 50 different sub-detectors many of which are comprised of multiple teams working in parallel. However, as far as controls are concerned, these various groups will in many cases be using similar equipment and require very similar functionality. Thus, the aim of the JCOP is to reduce duplication and to ease integration by developing and supporting control systems centrally. Products such as PLCs, fieldbus or SCADA tools, that have all been used successfully in existing HEP laboratories, were adopted by JCOP. PLCs are effective at performing autonomous and secure local process control. The fieldbus is an ideal solution in a geographically dispersed environment. The advantage of using a standard bus relates to its ease of use having no drivers to write and maintain. If the fieldbus is then connected with a SCADA product or to a PLC component any communications software have to be written. OLE (Object Linking and Embedding) for Process Control (OPC) is an emerging non-proprietary software standard to interface components from different vendors which offers several advantages: first, it acts as glue between independent layers of the system allowing to change an individual layer without breaking the entire system. Second, it permits separately developed components to be readily incorporated into the overall DCS. Regardless the advantages offered by OPC, initially it was developed as a platform-dependent protocol running on Windows. Thus, in order to complement OPC, the Data Interchange Protocol (DIP) was adopted as the standard solution for the DCS exchange information with external systems (e.g. LHC machine, TDAQ, CERN Technical Services). DIP is based on the Distributed Information Management (DIM) protocol already used at the DELPHI experiment [37] during the nineties. This is a suited solution for exchanging information between heterogeneous systems running in different platforms.

Another major issue tackled by JCOP was the choice of a supervisory and control software. An evaluation of the widely employed Experimental Physics and Industrial Control System (EPICS) was done at CERN in 1997/1998. EPICS [38] is a collection of three things: an architecture for building scalable control systems; a collection of code and documentation comprising a software

toolkit; and a collaboration of major scientific laboratories and industry. However, the evaluation suggested that whilst this had certain strengths, it would not be appropriate for experiments as complex as those for LHC which would not start until 2007 and then run for 10 to 20 years. This led to a decision by the CERN controls board to sponsor an in-depth survey of the SCADA market[39]. SCADA vendors release one major version and one to two additional minor versions once per year. These products evolve thus very rapidly so as to take advantage of new market opportunities, to meet new requirements of their customers and to take advantage of new technologies. Thus, in 1999, the four LHC experiments chose together the commercial SCADA tool PVSS-II to construct their back-end control systems. At that time, the next step was the creation of a software framework (based on the selected SCADA system) in order to be used commonly at the LHC experiments.

The Framework is one of the sub-projects of the JCOP and represents a collaboration between the four LHC experiments and the CERN IT controls group. By sharing development, the overall effort required to build and maintain the experiment control systems is reduced. As such, the main aim of the Framework is to deliver a common set of software components, tools and guidelines that can be used by the four LHC experiments to build their control systems (e.g. interfaces to power supplies, configuration tools, etc). Originally, the JCOP Framework was influenced by the Software Engineering Standard PSS-05 [40] on which development started at the European Space Agency (ESA) in 1984. The PSS-05 guides provide an easy to understand set of guidelines covering all aspects of a software development project, and following the standard leads to discipline and quality.

3.2 Scope of the DCS

The principal task of the DCS is to enable the coherent and safe operation of the ATLAS detector[41]. The DCS supervises the hardware in the experiment set-up including all the detector services (e.g. HV, LV, cooling) and the common experimental infrastructure (e.g. racks, environmental conditions). Moreover, the DCS also serves as interface to external systems such as the CERN Technical Services (e.g. electricity, ventilation) and most notably to the LHC accelerator (e.g. for beam conditions and backgrounds).

The DCS provides online status information to the level of detail required for global system operation. The interaction of detector experts with their detector is also done via DCS. It monitors all operational parameters, gives guidance to the operator, and signals any abnormal behavior. It must also have the capability to take appropriate actions automatically if necessary and to bring the detector to a safe state.

Finally, supervision by the DCS is required continuously. Some parts of the experiment cannot stop running because any interruption can be costly in time or money, or may even be detrimental to the performance of that experiment.

DCS, Exempted Responsibilities

Concerning safety aspects, the DCS only treat them at the least-severe level. This mainly concerns questions of sequencing operations or requiring specific conditions to prevail before allowing certain procedures to be executed. Monitoring and prevention of situations which could cause major damage to the detector or even endanger people's lives are the responsibility of a dedicated Detector Safety System (DSS).

Briefly, the DSS consists of a front-end system with stand-alone capability, which is based on PLCs. A back-end system, implemented also in PVSS-II, provides supervisory functions but it is not needed for the real-time operation. Bi-directional information exchange between DCS and DSS is provided on the PVSS-II level. Actions are triggered only in one direction, from DSS to DCS. In this way, the DCS can not disturb the operation of the safety system, but early information about minor problems that the DSS may detect enables DCS to take corrective actions before the problem escalates and DSS has to take a more drastic action.

Concerning the extraction of data for physics research, the DCS has not a direct contact with this matter which is handled by the TDAQ system (see Section 2.6). However, because both systems are complementary good-quality physics data requires detailed synchronization between the TDAQ and DCS[42]. Whilst the TDAQ deals with the data describing a physics event (characterized by an event number), the DCS treats all data connected with the hardware of the detector related to the operational state of the detector when the data were taken (categorized by a time interval). Moreover, correlation of data between both systems is again required for off-line analysis.

3.3 The ATLAS DCS Set-up

The architecture of the DCS and the technologies used for its implementation are constrained by both the functional requirements and the environmental aspects. As shown in Figure 3.1, the DCS consists of a distributed Back-End (BE) system running on PCs and of different Front-End (FE) systems. The BE will be implemented with a commercial Supervisory Control And Data Acquisition system (SCADA). The DCS FE instrumentation consists of a wide variety of equipment, ranging from simple elements like sensors and actuators up to complex Front-End Controllers (FECs) for specialized tasks. The connection between FE and BE is provided by fieldbus or LAN.

As already mentioned in Section 2.2, the equipment of the DCS will be geographically distributed. The SCADA components will be located at control and electronics rooms, while the FE equipment will be placed in USA15, US15, and UX15 (see Figure 2.2).

The relative independence of the operation of the sub-detectors will lead to a hierarchical organization of the experiment allowing partitioning during operation[43]. In addition to the 9 specialized sub-detectors discussed in the previous chapter, 3 other control segments exists within the ATLAS DCS:

1. The Common Infrastructure Controls (CIC) monitors services provided to the experiment as a whole. For instance, the general electricity, ventilation or environmental conditions in the UX15 cavern are in handled by the CIC and later delivered to the rest of sub-detectors. Moreover, the CIC also supervises equipment which is common to several sub-detectors like the electronics racks or fieldbuses branches.

2. The Inner DEtector (IDE) monitors and controls services allocated within the limited volume of the inner detector. Hence, systems like the thermal enclosure or the radiation monitor are shared by the assemble of sub-detectors that constitute the ATLAS tracker (i.e. PIX, SCT and TRT).

3. The TDQ (Tigger DaQ) represents a small piece of the DCS and its main purpose is to monitor and control all those electronics racks, located in the underground electronics rooms, that contain TDAQ instrumentation.

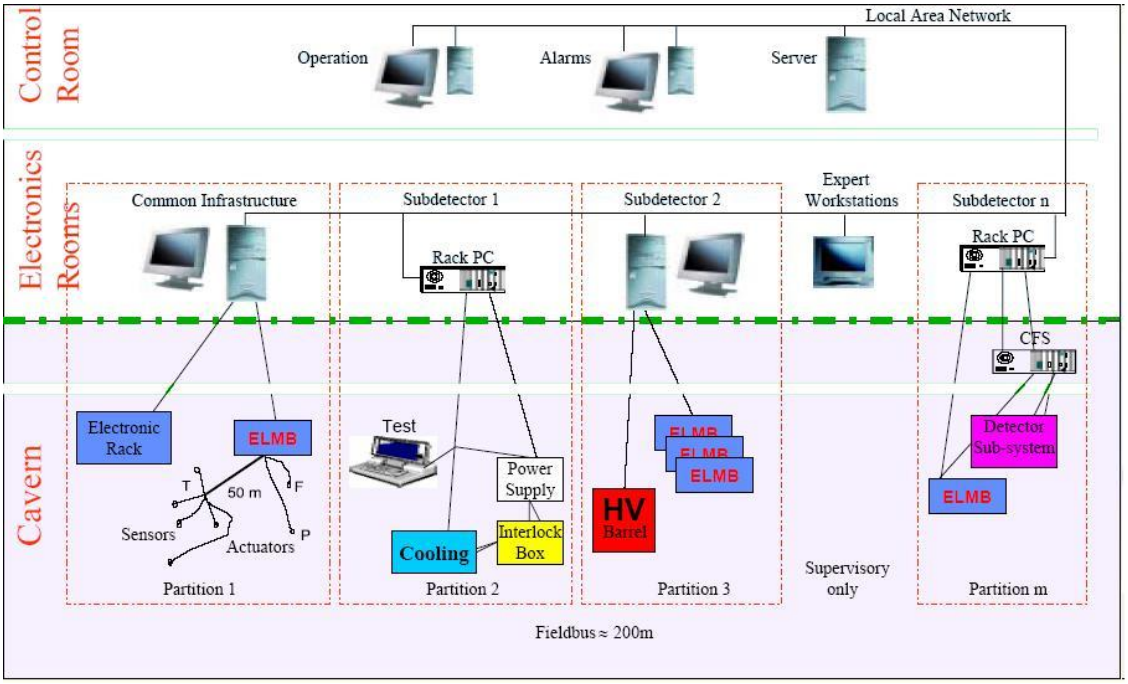


Figure 3.1: The DCS is distributed in three main areas, namely SCX1 (surface control room), UX (detector cavern), US15 and USA15 (electronics rooms). The DCS control chain is constituted by back-end and front-end elements such as an operator interface, an integrated engineering platform, a communication network, controllers and I/O sub-systems.

On the one hand and due to the nature of the experiment, the FE electronics in UX15 will be exposed to a strong magnetic field (up to 1.5 tesla) and to ionizing radiation. In order to minimize the radiation effects, the FE equipment is being located outside the calorimeters, where the dose rate is of the order of 1 gray/year or less. This permits the use of selected COTS components, which however, have to be individually qualified for radiation following the 'ATLAS policy on radiation tolerant devices'[44]. All the FE equipment which can not operate under these environmental conditions, like processor-based devices (e.g. FEC) or some types of power supplies, are being accommodated in the electronics rooms.

On the other hand, the BE system consists essentially of PCs and is organized in a tree-like structure with several levels. The highest level is situated in the control room and provides all the tools needed for the integrated operation of the detector as a whole, like the operator interface and the alarm system. The workstations in the levels below provide the online data and command handling, and full stand-alone operation capability for each sub-detector. This gives operational independence to the sub-detectors when needed, e.g. for commissioning, debugging, or calibration. The connection between all PCs of the BE is done over a LAN that is part of the SCADA software PVSS-II. At the lowest level, communications between PCs and the read-out instrumentation are normally done via fieldbus.

3.4 Front-End system (FE)

The FE equipment connects directly to the detector hardware. It comprises sensors and actuators, digitizers, controllers and processors, commercial devices, and stand-alone computer-based systems. Concerning monitoring, the FE reads and digitizes values, processes them in some cases, and transfers the data to the BE. It also executes the commands that it receives from the BE.

The FE equipment is distributed over the whole volume of the detector with cable distances of up to 150 m. Two conflicting aspects constrain the distribution underground. Because of the radiation level, the magnetic field, the space limitations, and the inaccessibility of UX15 during beam time, it is preferable to locate the equipment in the electronics rooms. However, the complexity, cost, and technical difficulties of laying analog cabling over such distances, and the difficulties of transferring large amounts of digital data over long distances in such a harsh environment, imply the digitization and compression of the data as early as possible. This should be done on the detector cavern UX15, and only results of digitization, possibly with analysis of data and compression, should be transferred to USA15 or US15.

The FE DCS of the sub-detectors is the responsibility of the sub-detector groups. It consists of two categories of equipment, one being more general purpose and widely used, like the power system, PLCs or the Embedded Local Monitor Board (ELMB)(all expanded below), and the other being more specialized such as the alignment system of the muon spectrometer and other various systems.

3.4.1 Embedded Local Monitor Board

The Embedded Local Monitor Board (ELMB) is a plug-on board that provides standard analog and digital input/output facilities for a wide range of FE control and monitoring tasks[45]. Because of the space limitations on the cavern and the large amount of channels to be supervised, a custom design was developed providing a high density of channels at low cost. Figure 3.2 shows a schematic view of this device.

Concerning the operation of the device in the harsh environment of the cavern, the ELMB board provides a radiation tolerance of up to around 5 Gy and 3×10^{10} neutrons/cm^2. This corresponds to a period of about 10 years of operation in the cavern. Moreover, the device does not contain components sensitive to a magnetic field up to 1.5 T. It can be stated then that no such commercial devices exist providing: tolerance to work in such hostile environment and high density of channels at low cost.

Mounted on a standardized motherboard, the ELMB can directly read out front-end electronics through the commercial Kvaser PCI CAN card interface[46]. The motherboard can be equipped with standard adapters such as 4-wire Pt100 sensors (Pt1000), two-wire resistive sensors and differential voltage attenuators.

The firmware loaded into the ELMB includes instructions for both driving the input/output functions and the communication over CANbus using the higher level protocol, CANopen. Standard configuration software tools provide easy 'plug and play' functionality for the user. The ELMB fulfils the majority of standard I/O requirements of the ATLAS sub-detector applications in terms of functionality, accuracy, and stability. Typical applications of the ELMB are to directly read from/write to sensors and actuators, or to control more complex devices like power supplies, gas devices, cooling systems, or whole detector elements like chambers. In the second class of applications, the ELMB is normally fully integrated in the device. In several cases, like the monitoring of the gas flow or the control of the SCT power supply system, which require additional

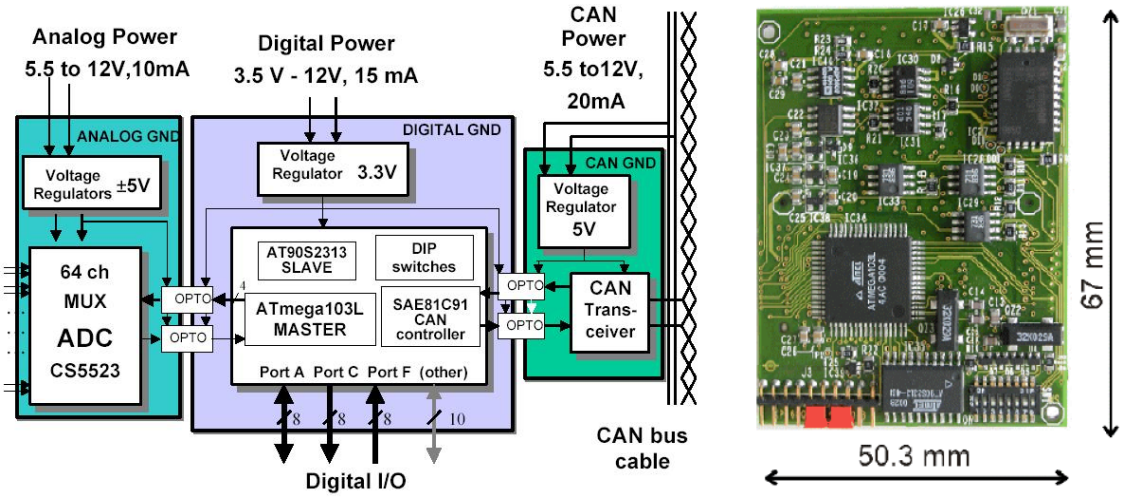


Figure 3.2: The ELMB is a single, credit-card-sized electronics board. Left: It comprises 64 high-precision analog input channels of 16-bit accuracy and it provides 8 digital inputs, 8 digital outputs and 8 configurable (either input or output) lines. Its serial port can drive additional devices like a digital-to-analog converter. As it has very low power consumption, it can be powered from the electronics rooms via the fieldbus cable. Right: Front side of the ELMB with the master and slave micro-controllers, CAN chip and DIP switches for node identification and setting of the baud rate.

functionality like histogramming, specialized ELMB firmware has been developed by the users. In these cases, the ELMB library is not used directly being fully reprogrammed using the CANopen protocol.

3.4.2 Power System

As discussed in previous Chapter 2, the power system is an essential part of any HEP experiment. On the one hand, many pieces of the detector require of HV including wire chambers, photo-multipliers or silicon detectors. On the other hand, LV is also needed by many devices including read-out electronics, smart sensors or actuators. A collection of the envisaged power consumptions of the different sub-detectors modules is shown in Table 3.1.

In order to build power system at the LHC experiments, commercial COTS systems were selected by the JCOP in conjunction with the detectors. These systems provide

Sub-detector	No. of modules	Power/module(W)	No. of voltages
PIX	1000	48	7
SCT	4088	9	7
TRT	1500	50	3
LAR	58	3300	7
TIL	32	2200	7
MDT	1200	40	2/3
CSC	64	40	2/3
RPC	550	60	2/3
TGC	600	60	2/3

Table 3.1: Power supply requirements. Table taken from [47]

separately-controllable channels that are packed into a single rack-mountable VME crate. Each channel has a set of parameters to be controlled and monitored such as current limits or over/undervoltages trip points. Usually the channels are contained on modules, or cards, which plug into the crate. Between 4 to 48 modules are normally plugged into a single crate, depending on the manufacturer and model, and each module has up to perhaps 32 channels. The electrical characteristics of all channels in a module are the same, but modules of many types exist each with different properties. Moreover, specialized vendors in nuclear instrumentation already provide equipment tolerant to both radiation and magnetic field. This is the case of the Caen EASY (Embedded Assembly SYstem)[48] and the Wiener MARATON (MAgnetism and RAdiation TOlerant New)[49] power systems. Besides the products offered by these two vendors, JCOP also supports power supplies from the Iseg company[50]. All these power systems are able to take decisions autonomously, not relying on the functionality of a network and reducing the traffic to improve the safety of the detector. At the same time, they are easily interfaced by means of OPC into the SCADA system where the supervisory control is achieved. Figure 3.3 shows the example of the power system set-up used for testing purposes in the SCT detector.

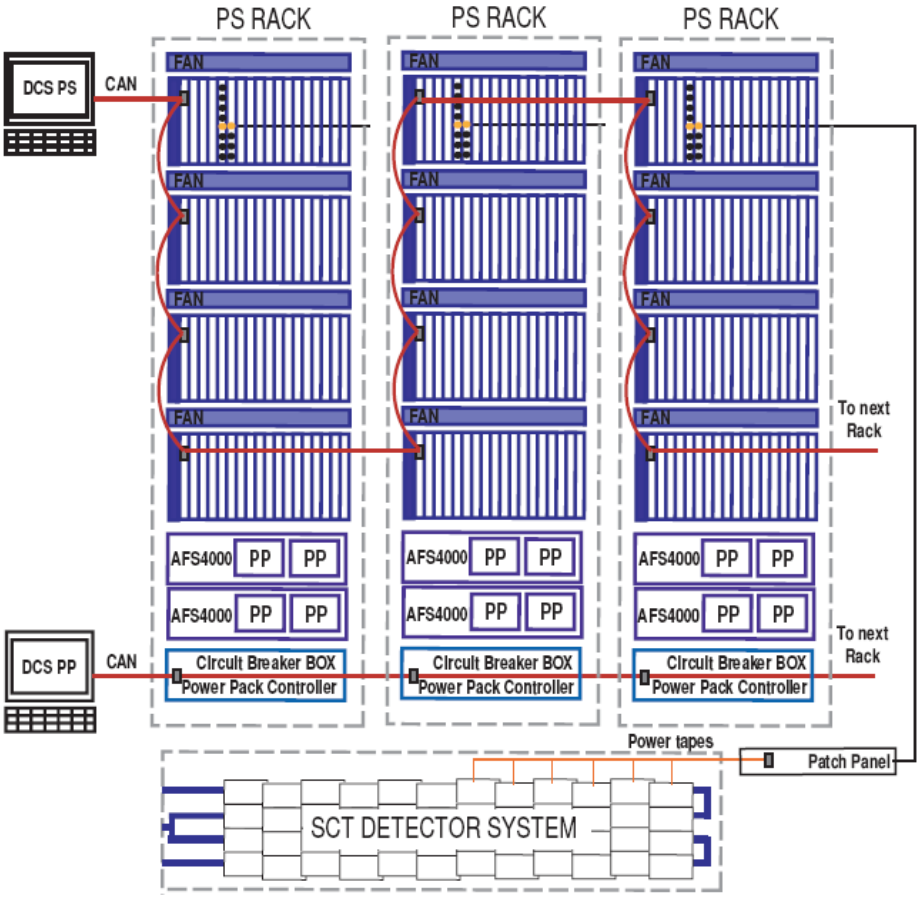


Figure 3.3: Power supply system prototype of SCT. There are four crates in each rack that are connected together and are read out by the same CANbus via the crate controller. Picture taken from [25]

Finally, the power system in ATLAS is completed by the usage of Uninteruptable Power Supplies (UPS) which increase power reliability on the off-detector instrumentation. UPS is normally used in those cases where the absence of voltage during a short-transition to a diesel generator back-up is too long. In this group, computers and control systems are included. A UPS consists essentially of a rectifier, which charges a battery, which in turn feeds a DC/AC converter to supply a standard low voltage level to the equipment (230V/400V to three phases systems, or 230V for single phases systems)[51].

3.4.3 Programmable Logic Controllers (PLC)

PLC-based solutions are well adapted to two-level control architectures where the front-end layer has to be autonomous and independent from the supervision layer. The PLC enables the process control, in terms of input/output readout or closed loop control, to not depend on the network neither on a remote computer being more secure. Hence, highly reliable systems in ATLAS such as the DSS, cooling, cryogenics, magnet or gas systems made a wide implementation of the PLC approach. Again, due to the vast usage of PLCs at CERN, a market survey was done in 1997 over European manufacturers giving an out coming list of this type of equipment to be supported by the JCOP (see Table 3.2).

Concerning communications within the PLC modules (i.e. analog, digital, TTL, RS232, etc), this are either handled by means of an internal bus or through a bus coupler that is connected with a fieldbus segment (e.g. PROFIBUS, WorldFIP or CAN). Furthermore, PLCs today provide ethernet-based communications that enable the usage of OPC for the data exchange with the supervisory level.

Regardless the advantages offered by this type of equipment, the integration of PLCs from different makers into a large control systems still presents inconveniences. Each vendor normally offers a different programming language which can make the in-home development or maintenance of the entire system difficult. In addition, even if communications are based on TCP/IP and IEEE 802.2, PLC protocols are still manufacturer specific which hinders the exchange of information between PLCs of different manufacturers or even with the SCADA system. Thus, if we assume (as discussed later in this chapter) that the trend in the control market is to homogenize the control data, it would be desirable that PLCs and SCADA software would be more mix-and-match. Thus, new generations of PLCs should provide more homogeneity when data exchanging as well as a standard PLC programming language that could be even integrated into SCADA.

Schneider Quantum and Premium	Standard modular PLCs specially recommended when supporting the connection to CANOpen devices (e.g. ELBM) is needed
Siemens S7-400 and S7-300	Standard modular PLCs recommended for application requiring redundancy and security
Wago	Small CPUs programmed like a PLC (for small applications)

Table 3.2: List of PLCs supported by the JCOP for its usage on the LHC experiments.

3.4.4 Other FE equipment

Leaving aside the presented instrumentation, the FE system in ATLAS will be also formed by many specialized, non-standard equipment like the alignment and calibration systems which are usually self-contained. There is also a large number of devices for measuring humidity, temperature, pressure, smoke, presence, etc, which are selected and integrated in the control chain by the sub-detectors experts depending on their needs. These devices however normally follow the standard control chain for the read-out (i.e. ELMB - CANbus).

Most of the equipment in ATLAS will be housed in racks (allocated in UX15, USA15 and US15) which are cabled to the detector. An inventory work is being carried out in ATLAS in order to allocate the racks to different sub-systems and to prepare the cabling work, Figure 3.3 shows an schematic view of the rack distribution in the USA15 level 1.

Finally, the DCS set-up will count with up to 200 rack mounted PCs where the back-end controls will be distributed. In order to ensure reliability these PCs are equipped with: (a) 2U high with 3 PCI slots to connect standard Kvaser PCI CAN cards, (b) dual power supplies, (c) Intelligent Platform Management Interface (IPMI), and (d) dual core CPU, and (e) redundant disks.

3.5 Communications in the DCS Environment

Communication over a network using standards middleware protocols is a key point for inter-process communications among the different DCS components. Fieldbuses are widely used for their low cost and short response time. Fieldbuses should not be confused with Local Area Networks (LANs) although in some cases their domains of application may overlap. Both are used in the DCS at different levels, fieldbuses normally establishing communications with field devices, and Ethernet LAN between computers. Figure 3.4 matches the communication standards used in the DCS with their correspondent layer on the Open Systems Interconnection model (OSI). Concerning the OSI model, from its rising, the makers of equipment started to use proprietary drivers as part of the application programme, and only through them, one could access the data stored in their equipment. In order to eliminate communication barriers between different platforms (e.g. Windows and Linux) and makers of equipment, the ATLAS DCS uses OPC and DIM. This section discusses the different middleware protocols above listed and used in ATLAS.

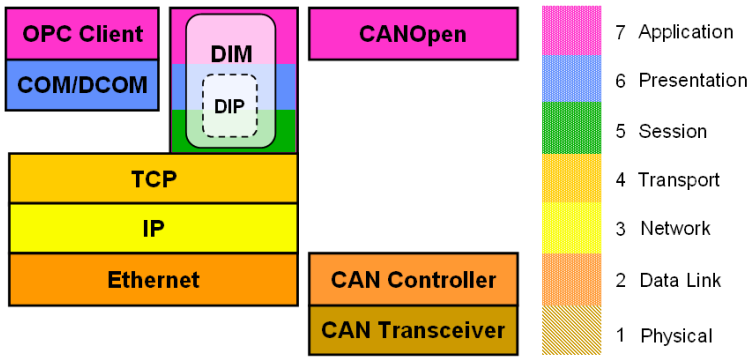


Figure 3.4: OSI model for communications in ATLAS. The CAN fieldbus is the standard used to connect with the ELMB devices and Wiener crates. OPC is the main protocol to establish communication with instrumentation that are connected through a LAN. DIM is a high-level protocol used to connect heterogeneous systems that can work in different platforms.

3.5.1 Fieldbuses

The term fieldbus refers to data transmission systems used to federate or integrate rather simple equipment to be remotely monitored and controlled. Standardization of international fieldbus specifications, notably the Foundation Fieldbus [52], has enabled users to build optimum field networks comprised of freely chosen field devices from various device vendors.

In essence, a fieldbus is a simple cable bus used to link isolated field devices, such as controllers, transducers, actuators and sensors by means of a well-defined protocol which permits to set a distributed-control network. Each field device has low cost computing power installed in it, making each device a 'smart' device (i.e. the ELMB). Each device is able to execute simple functions on its own such as diagnostic, control, and maintenance functions as well as providing bi-directional communication capabilities.

Industrial fieldbuses differ in technical characteristics such as bandwidth, network topology, length, determinism, robustness, error handling, openness, redundancy, etc. This lead to a division of fieldbuses into in three main categories (see Figure 3.5).

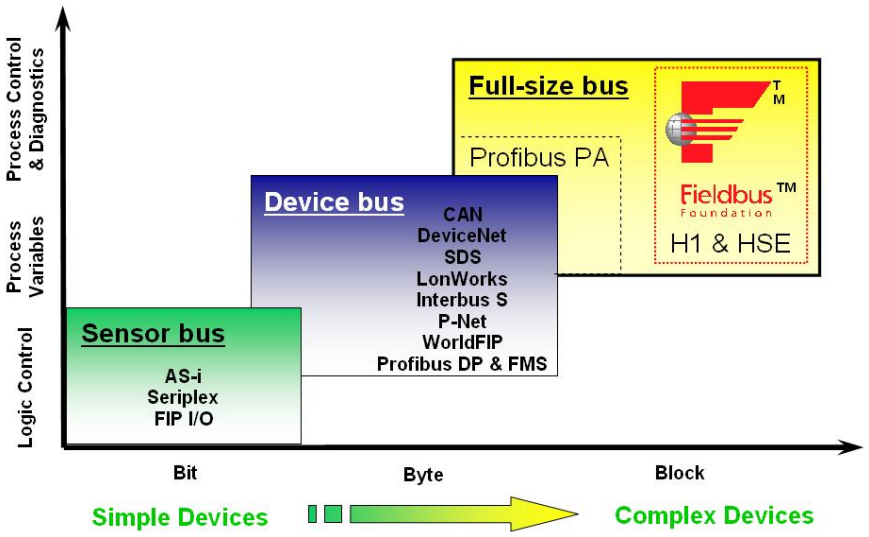


Figure 3.5: Types of buses. 1-Sensor buses: Used to connect intelligent sensors to PLC or front-end equipment. 2-Device buses: Connect I/O modules to PLCs or front-end. 3-Full-size bus: Similar to LANs are used to handle data at higher levels between the PLCs or front-end equipment and the supervisory stations.

In the two first types, communications are usually periodic and deterministic whereas in full-size buses they are frequently on-request. Device buses constitute the standard approach to connect front-end equipment to the BE in ATLAS.

In order to limit the types of fieldbuses used at CERN (more than 120 types are available in the industry), a major evaluation effort was performed and concluded with the recommendation of three fieldbuses to be used in the LHC: CAN, WorldFip and Profibus[53]. These three fieldbuses are complementary in their technical aspects and domain of application and therefore, should suffice to meet all requirements for applications at CERN in both, accelerator and experiment fields.

Profibus[54] is well developed and supported by large companies such as Siemens and it already has a wide acceptance in European and American industry. Profibus can work in multi-master or in master-slave mode. It is specially recommended in applications where a large data volume must be handled, baud rates can be selected from 9.6 kbits/s up to 12 Mbits/s.

WorldFip[55] is a system based on a centralized access method where one master continuously distributes the access right (token) to the different stations. It is appropriate for systems with critical time requirements supporting three standardized transmission rates: 2.5 Mbits/s over short distances using twisted pair cables or optical fibres, 1Mbit/s for twisted pair cable at distance less than 500 meters and a low frequency of 31.25 kbits/s for long segments up to 2000 meters. An additional frequency of 5Mbit/s was specified for optical cable transmission.

CAN[56] is specially suited for applications where high flexibility and reliability of data transmission are needed. CAN equipment is very robust, has excellent error detection and recovery, can be used over large distributed areas, does not use components sensitive to magnetic field, and has good support from industry.

From the three CERN recommended fieldbuses, CAN has been selected by the ATLAS community for its robustness and reliability. Next this fieldbus is expanded.

Communication via the Controller Area Network (CAN) fieldbus

CAN was introduced into the market by the Bosh company in 1986. This industrial bus was primarily intended for the automotive market having high requirements for the reliability of data transmission. However, it is now used in many non-automotive industrial applications (e.g. controls of production lines and machine tools, medical apparatus or nautical machinery). It can be used as an open system without the need to pay any licence fees and enjoys of a wide acceptance by industry and research laboratories. Chips are available and are mass produced so they are cheap and reliable. Due to the requirements of the automotive market they will be available for a long period of time.

Briefly, a CAN message contains an identifier field, a data field and error, acknowledgement and CRC (Cyclic Redundancy Check) fields. The identifier field consists of 11 bits for CAN 2.0A or 29 bits for CAN 2.0B. The size of the data field is variable from zero to 8 bytes. When data are transmitted over a CAN network no individual nodes are addressed, instead, the message has assigned an identifier which uniquely identifies its data content. The identifier not only defines the message content but also the message priority. Any node can access the bus and, after successful arbitration of one node, all other nodes on the bus become receivers. After having received the message correctly, these nodes then perform an acceptance test to determine if the data is relevant to that particular node. Therefore, it is not only possible to perform communication on a peer to peer basis, where a single node accepts the message, but also to perform broadcast and synchronized communication whereby multiple nodes can accept the same message that is sent in a single transmission.

Another feature of CAN is the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) mechanism that arbitrates the access to the bus. Contrary to other bus systems CAN does not use acknowledgement messages, which cost bandwidth on the bus. All nodes check each frame for errors and any node in the system that detects an error immediately signals this to the transmitter. This means that CAN has high network data security as a transmitted frame is checked for errors by all nodes. Depending on the CANbus speed the lengths of the cables are limited. Table 3.3shows the relation between the bit rate and the cable length.

Concerning the standardization of the protocol, only the two lower layers of the OSI model are commonly defined (i.e. physical and data-link layers). Although the usage of direct CANbus for low-level applications may be adequate, a higher-level communication protocol is required on top of CAN when integrating this fieldbus into the supervisory control. In order to give a solution to this need, many makers have developed layers on top of the standardized ones such as DeviceNet or CANopen being some of them still considered proprietary. From these different higher-level communication protocols available, CERN selected CANopen on the basis of flexibility and market acceptance. CANopen is defined by the CAN in Automation (CiA) organization and a detailed

Bit rate	Cable length
10 kbits/s	6.7 km
20 kbits/s	3.3 km
50 kbits/s	1.3 km
125 kbits/s	530 m
250 kbits/s	270 m
500 kbits/s	130 m
1 Mbits/s	40 m

Table 3.3: Relationship bit rate - cable length for CANbus.

description of the protocol is provided in[57]. This standard implements layer seven of the OSI communication model, the application layer(see Figure 3.4). It defines how the data-bytes of the CAN frames are used among the different nodes on the bus and also guarantees the inter-operability of devices from different manufacturers.

3.5.2 OLE for Process Control (OPC)

Conventional integrated systems in the past had to develop custom interfaces for inter-connectivity between different vendor's systems. In order to solve this lack of standardization, the OPC standard was developed in 1996 by an industrial automation industry task force[58]. The specification defines a standard set of objects, interfaces and methods for use in process control and manufacturing automation applications that facilitate the inter-operability. Then, by the usage of OPC, it is easy to integrate different vendors' systems such as power supplies or PLCs in large-scale facilities and, as so, OPC has naturally became a standard at CERN to interface the equipment with the supervisory level.

The OPC specifications were originally based on the Microsoft's OLE technology of the 1990s, from this came the meaning of OPC (OLE for Process Control). OLE however was soon replaced by the Component Object Model (COM) and Distributed COM which both were also primarily used by Microsoft for the Microsoft Windows operating system family. By the usage of these technologies, OPC provided multi-client capability (i.e. users can access an OPC server with several OPC clients) effective locally on a PC, but also remotely in distributed networks.

The next stage of OPC is the OPC Unified Architecture (OPCUA) which has been specified and tested and starts to be implemented. OPCUA can be implemented with Java, Microsoft .NET, or C, eliminating the need to use a Microsoft Windows based platform of earlier OPC versions[59]. OPCUA combines the functionality of the existing OPC interfaces with web services technologies such as XML (eXtensible Markup Language) messages or the SOAP (Service Oriented Architecture Protocol) standard to deliver higher level of support. As a result, OPCUA looks to become the standard for exchanging industrial data, replacing FactoryTalk, Archestra, some Modbus applications, and OPC Data Access. Although these promising (but not yet clear) advances, it is too early (too late) to implement these new technologies into the controls of the LHC experiments that have to start operation at the end of the present year. The functionality offered by OPCUA (e.g. multi-vendor multi-platform inter-operability) however, is ensured at the LHC experiments by the CERN standard protocol DIM.

3.5.3 Distributed Information Management (DIM)

DIM was originally developed for the DELPHI[60] experiment at LEP and it is heavily used in ATLAS and the rest of LHC experiments[37]. DIM allows inter-process communication between heterogeneous processes running on Windows systems and on Linux which interfaces available in both C and C++.

DIM, like most communication systems, is based on the client/server paradigm. The basic concept in the DIM approach is the concept of *service* which is provided by the server to the client. A service is a set of data (of any type or size) which is normally requested by the client in different ways: (a) the client requests information *only-once*; (b) the client requests the information to be updated at regular time intervals; (c) the client requests the information to be updated whenever it changes; and (d) the client sends a *command* to the server.

In order to allow for the required transparency (i.e. a client does not need to know where a server is running), as well as to allow for easy recovery from crashes and migration of servers, a name server is present.

Servers *publish* their services by registering them within the name server (normally only once, at start-up). Clients *subscribe* to services by asking the name server which server provides the service and then contacting the server directly, providing the type of service and the type of update as parameters. The name server keeps an up-to-date directory of all the servers and services available in the system. The interaction between servers, clients and the name server can be seen in the following Figure 3.6.

The underlying DIM component constitutes a key technology for the construction of the hierarchical control of the experiment that is discussed in next Chapter 4. DIM provides the mechanism for the exchange of in formation between the different nodes that constitute the control tree(e.g. new states or commands).

3.5.4 Data Interchange Protocol (DIP)

The aim of DIP is to define a single data exchange mechanism between all systems involved in the LHC operations. In ATLAS, this standard protocol is used to interface with external systems such as the LHC, the Magnet System or the DSS. DIP is essentially based on the DIM protocol and it allows relatively small amounts of real-time data to be exchanged between very loosely coupled heterogeneous systems that do not need very low latency. The data is assumed to be mostly summarized data rather than low-level parameters from the individual systems, i.e. cooling plant status rather than the opening level of a particular valve. DIP is available as a JCOP Framework component.

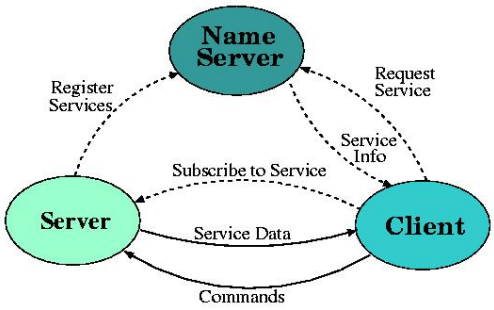


Figure 3.6: DIM's main data flow diagram. The name server receives service registration messages from servers and service requests from clients. Once a client obtains the 'service info' (i.e. the service co-ordinates) from the name server it can then subscribe to services or send commands directly to the server. Picture taken from [61].

3.6 The Back-End System (BE)

The back-end system comprises all those the software components that process the output from the front-end and interacts directly with the user offering supervisory control. Data processing and analysis, display, high-level automation and sequencing, storage and archiving of data are all functions of the BE. The BE system in ATLAS is organized hierarchically in PCs which usually runs under the Microsoft Windows operating system. This section introduces first the commercial SCADA product PVSS-II and second the JCOP Framework, which is the software engineering platform, based on PVSS-II, specially developed for the controls of the four LHC experiments[62]. The main work of this thesis is related with the BE and the hierarchical organization and automation of ATLAS DCS processes.

3.6.1 The PVSS-II SCADA System

SCADA stands for Supervisory Control And Data Acquisition. As the name indicates, it is not a full control system, but rather focuses on the supervisory level. Besides basic functionality like Human Machine Interface (HMI), alarm handling, archiving, trending or access control, SCADA products also provide a set of interfaces to: hardware (e.g. fieldbuses and PLCs) and software to communicate with external applications, e.g. Application Program Interface (API).

An evaluation of SCADA products was performed at CERN in the frame of the JCOP which ended up with the selection of the Austrian company ETM (recently absorbed by SIEMENS) and its product PVSS-II[39] which in German is the abbreviation of 'Process visualization and control system II'[63].

One of the main features offered by PVSS-II is its modularity which help us to describe the product. Figure 3.7 shows the modular design of PVSS-II which is handled by functional modules specifically created for different tasks. These modules are called *managers* and constitute separate processes in software.

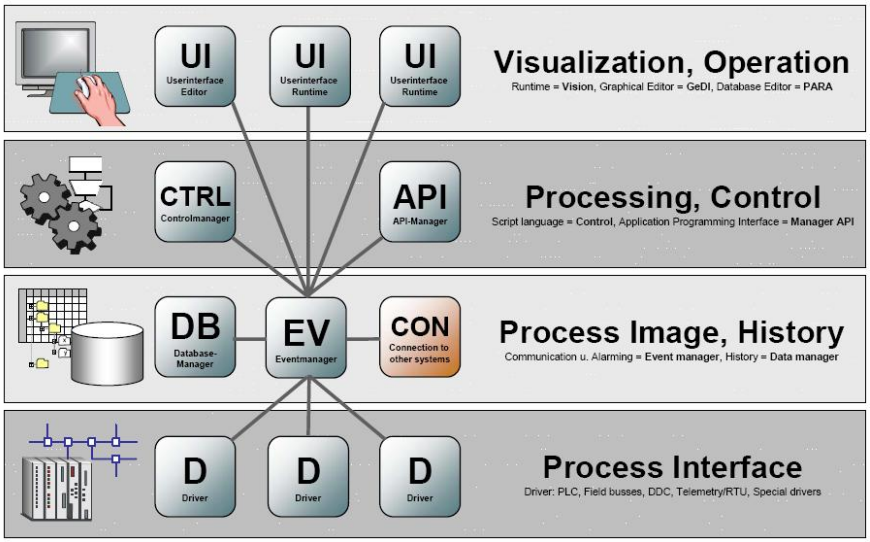


Figure 3.7: Example of a simple configuration of a PVSS-II system showing the core managers. Picture taken from [63].

The process interface modules are all those Drivers(D) that connect PVSS-II with external software or hardware to be supervised. For instance, common drivers that are provided with PVSS-II are OPC, ProfiBus, CANbus, Modbus TCP/IP and Applicom.

The central processing unit in PVSS-II is called *Event Manager* (EV). The EM is responsible for all internal communications, it receives data from drivers, and sends it to the *DataBase Manager* (DB) which provides the interface to the run-time data base. The EV maintains the process image in memory (i.e. the current value of all the data) and also ensures the distribution of data to all managers which have subscribed to it.

The openness of PVSS-II is another feature well-appreciated which is mainly available by means of *Application Programming Interfaces* (APIs). These are implemented as a C++ class libraries and let the developer implement custom functions as additional self-contained managers (forecasting system, simulation, design tools, proprietary databases, etc.). This is the most powerful form in which to add extra functions to PVSS-II.

At the higher level of abstraction, the *User Interface* managers (UI) form the interface with the user. These include a graphical editor, a database editor and the general user interface of the application. In the UI, values are displayed, commands issues or alerts tracked. Trends and reports are also normally included.

In essence, a PVSS-II system is an application containing one DB manager and one EV and any number of drivers, user interfaces, etc. PVSS-II can provide for very large applications, in which case one PVSS-II system would not be enough. In this case, a PVSS-II distributed system, which is a confederation of communicating systems, can be used.

A PVSS-II distributed system is built by adding a distribution manager to each system and connecting them together. In a hierarchical structure the best solution is that the top-level system acts as a server for all other systems, the bottom-level systems act as clients to all other systems and the medium-level systems act as servers for all systems below and clients to all system above. This has two advantages: (a) less entries are needed in the configurations file, and (b) when adding new lower-level systems, it is not needed to change the configuration file at higher-levels (and therefore one do not have to restart the higher-level systems). The ATLAS BE-DCS will be constituted of hundreds of these systems. Table 3.4 shows the slots of distributed systems numbers assigned to the different sub-detector systems.

TRT	1-19	MDT	90-109
SCT	20-29	RPC	110-129
PIX	30-39	TGC	130-149
IDE	40-49	CSC	150-159
LAR	50-69	CIC	200-209
TIL	70-89		

Table 3.4: Each PVSS-II system is required to have a unique system number. The range of system numbers attributed to each sub-detector depends of course on their necessities. 220 and higher are reserved for overall issues like the top level interface systems.

3.6.2 JCOP Framework

The JCOP Framework[64] is meant to provide a set of components to ease the development of the controls of four LHC experiments. The project main aim is to reduce effort in development and maintenance by re-using common components. These components intend to hide the complexity

of the underlying tools and equipment giving a high-level of abstraction and an interface for non-experts through PVSS-II. In the middleware, the Framework uses DIM and OPC as the main communication protocols and, although there are not many developments within the front-end layer, the Framework makes sure that everything down to the real equipment can be integrated coherently. Figure 3.8 shows an schematic view of the Framework architecture.

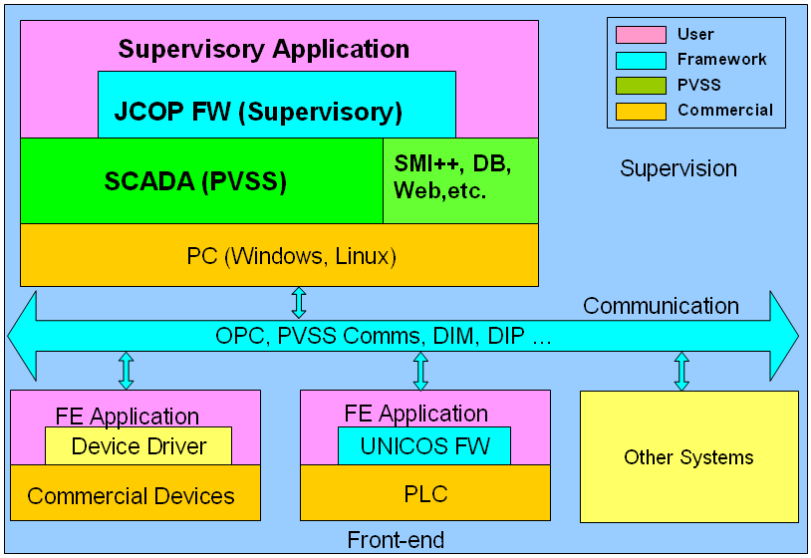


Figure 3.8: Architecture of the JCOP Framework. The supervision layer is based on the commercial SCADA tool PVSS. This is complemented with other packages (e.g. database access, Finite State Machines). The Framework customizes these general tools providing components of interest for the HEP environment. Picture taken from [64].

The Framework can model a front-end device directly as a piece of equipment (e.g. a high voltage crate) or as a logical group of devices (e.g. a group of channels). In the supervision layer, a device consists of one or more PVSS-II Data Points (DP) types, a library that encapsulates the knowledge to manage the device instances, and a set of displays to provide an interface for the user. Therefore, the developer can make the front-end layer to be specific for each application and, the only constraint at this layer, is to have an interface to one of the protocols supported within the Framework (preference is given to protocols coming directly from the maker of the hardware, e.g. the OPC server of CAEN).

The main interface to the Framework is called the Device Editor and Navigator (DEN). This tool allow to configure and operate at a low level all the Framework devices of the DCS. Moreover, simple device hierarchies can be built to give a logical or functional structure to the DCS. System management actions like installation or login can be also performed from the DEN. Following, some of the Framework devices that are currently at the disposal of the users, and are handled through the DEN, are listed:

- *Generic Analog-Digital Devices*: This provides basic functionality to deal with inputs and outputs that can reside in a PLC, a fieldbus module, an I/O card, etc.
- *CAEN Power Supplies*: The power systems of CAEN are interfaced through OPC into PVSS-II. The Framework component provides provides full functionality to model and control the equipment of this maker.

- *Wiener Power Supplies*: Similarly to CAEN this equipment is supported. In this case, the crates have a CANbus interface. All the models that support the standard Wiener protocol are included.

- *Wiener Fan Tray*: They follow a similar interface to the Wiener power supply.

- *ELMB*: Through CANbus and CANopen the ELMB is interfaced into PVSS-II. This Framework component allows to easily read-write the values and operate the device.

- *Logical Node*: This device is a container for other device instances. Once grouped together, the set of instances offers the same interface as a single instance.

Besides of the DEN, the JCOP Framework provides the developer with other tools for trending, access control, interfacing with the data bases, etc. Regardless of their importance, from all of them, one must be underlined in the context of this thesis, this is the JCOP Finite State Machine (FSM) (later expanded in Chapter 4). This is the main component for the organization and automation of the DCS BE. Basically, it provides high-level control by modeling the the different parts of the experiment as software objects that behaves following a finite state machine logic. These objects, as shown in Figure 3.9, are then deployed vertically at the different levels of abstraction of the experiment (i.e. ranging from the front-end equipment to the overall control).

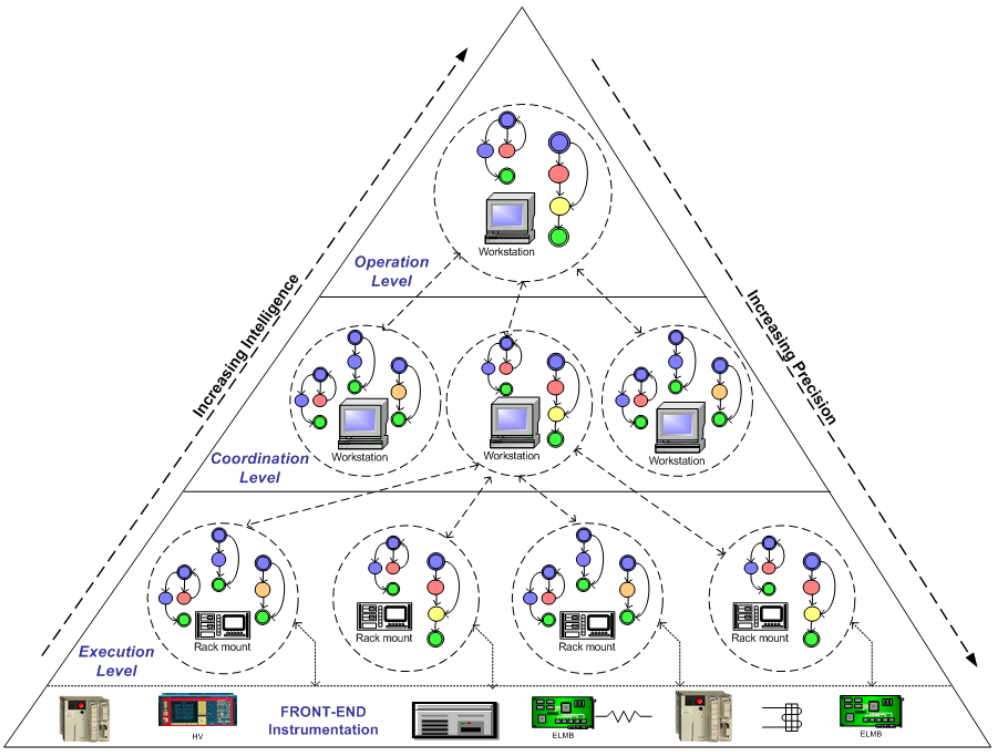


Figure 3.9: A hierarchy of objects behaving as FSM are hierarchically controlled by other FSM. These are distributed over different PVSS-II systems automating the overall DCS.

3.7 BE Controls Outside the HEP World

As it has discussed over this chapter, there is a clear tendency at CERN of using commercial COTS products in combination with suited solutions (e.g. the Framework software engineering platform) for the implementation of controls at the LHC experiments. Hence, due to the wide usage of these commercial products, this section focus the discussion on the industrial automation market. Major makers are listed and, in order to give an analogy between BE controls outside and inside the HEP world, some commercial umbrella software platforms are discussed briefly.

One the one hand, the JCOP uses the homecourt advantage to implement the different tools that compose the back-end control system. The knowledge acquired running previous HEP experiments and the expertise building specialized hardware and software needed by the experiments are combined with the latest COTS products available in the market.

On the other hand, big controls providers produce and sell high reliability fully-integrated control systems that are composed of software and hardware commercially available (i.e. from proprietary instrumentation to proprietary human-machines interfaces). Figure 3.10 shows a compilation of some of these major control companies with their principal mergers and their different umbrella software architectures.

Concerning now the BE system, historically, controls outside the HEP world have been slightly different mainly because of the data produced, which was normally, several orders of magnitude smaller. However, large distributed control systems are incrementally being implemented and, to face this, controls providers have defined two main areas of interest for their business: the need of more scalability and the need of more comprehensive systems[65]. Consequently, at the present level of technology, a key feature pursued by the industry is the vertical integration of information (i.e. ranging from retrieving real-time data to an overall management of the factory resources)[66][67]. The integration means at least to keep associations between resources (e.g. sensors and actuators) and their assets (e.g. role within the production process, production planning, maintenance schedules or faults) [68]. This vertical integration of information is enabled by software technologies which are presented to the industry in the form of proprietary software umbrellas. Next, some of these software umbrellas are discussed giving a special empathize to the fashion they distribute intelligence and operations throughout a distributed system, which is in fact, the main topic of this work thesis. This section, finally, will lead us to the discussion presented in next Chapter 4.

ABB's Automation [70] strategy is based on the concept of *Industrial IT* which is based in the completion of assets. This umbrella compiles information about production assets and the ways these assets are used during their life-cycle. With that tool, assets are ordered and configured, their real-time behavior and performance is modeled and the maintenance schedule defined. The key component that allows the integration of information is called *Aspect Integrator Platform* which provides a user interface to structure the links using the so called object-aspect paradigm. This platform can be seen as an on-line object-oriented modeling tool where class diagrams show the associations and relations between objects. For example, a certain DCS system can be linked to its control program, to actuators, sensors, human-machine interfaces, alarms systems, CAD drawings, etc. These links in the analogy to the JCOP DEN are organized similarly allowing multiple views of the plant. For example one can work with an horizontal view of a control system (control logic hierarchy), a vertical view of all objects at a location (physical location hierarchy), etc. That allows an operator to find information effectively.

The **Honeywell**'s strategy is based on unified system for process, business and asset management called *Experion PKS* (Process Knowledlge System)[68]. The system has fully distributed

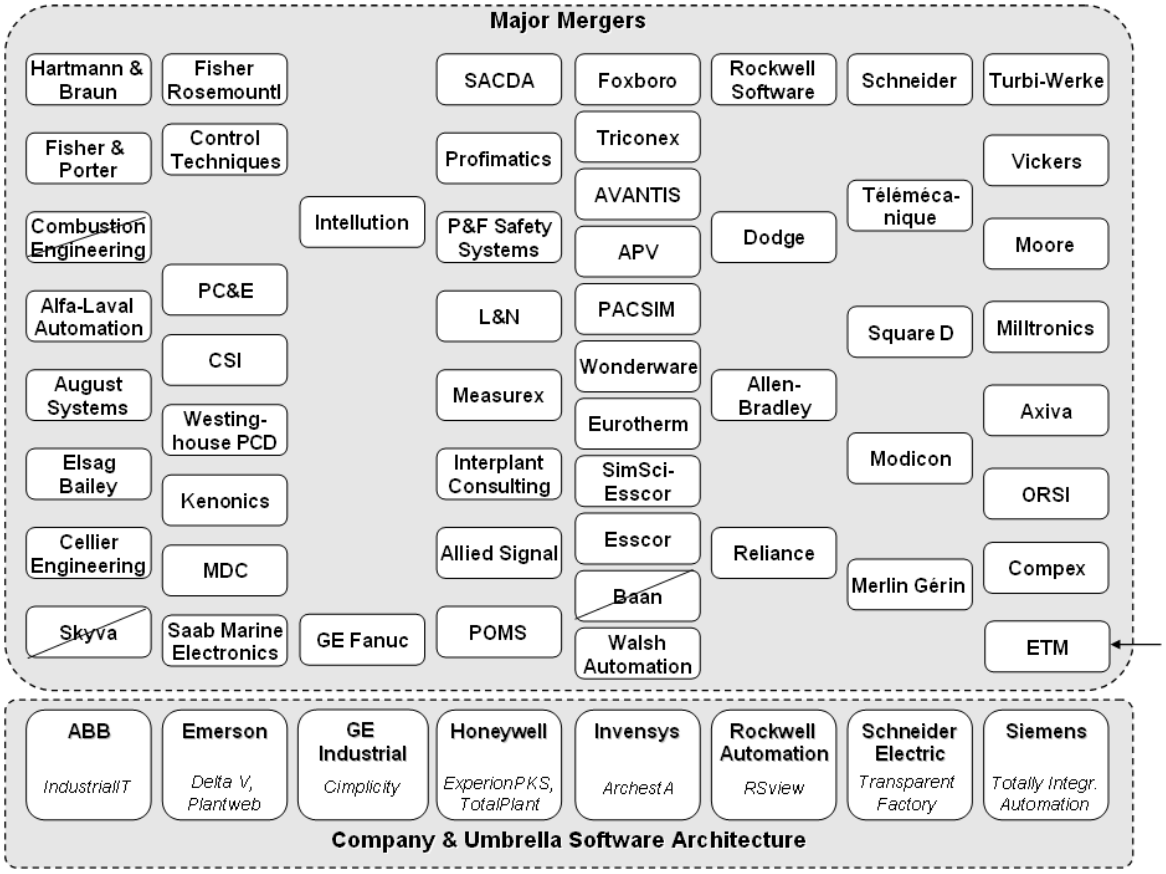


Figure 3.10: The 'Big Eight' business is estimated in about 80 Mia euro per year growing 5% annually. The companies use an umbrella software architecture and had a group of major mergers to provide full-integrated control systems. Picture inspired in [69].

architecture, with high-level redundancy, safety and optimization features. Fault-tolerant ethernet, alert management, integrated safety system, a predictive identification of abnormal situations, integrated model-based control, integrated wireless and digital video, are all features very useful within safety critical applications. The relevance of this software umbrella is the use of a knowledge repository where information is analyzed, classified and distributed for finance, scheduling, maintenance or field operations. Consequently, this knowledge repository enables the implementation of 'knowledge-driven' applications.

Rockwell Automation's [71][68] strategy is based on the concept of e-Manufacturing which major benefit is the distributed information processing at plant level. The main features of such architecture are transparency, seamless access to data, and integration of management services. Rockwell Automation is highlighting the value of four strategically selected points: design (how to swiftly deploy and reconfigure applications), operate (how to optimize process yield and keep consistency through the enterprise), maintain (recognizes the importance of maintaining assets: materials, processes and employees) and synchronize (the importance of a tight coupling among manufacturing operations into the greater supply chain, both upstream and downstream). True to its philosophy, Rockwell Automation has focused on two essential features, much appreciated by users:

- Data-sharing strategy. *FactoryTalk* is a common language that enables manufacturing to integrate seamlessly the enterprise, providing information transparency across all levels of the manufacturing activity.
- Data analysis and modeling strategy. *Arena Factory Analyser* performances improvements and long-term capacity analysis. Examples are: calculating realistic throughput estimates, identifying excess capacity, performing 'what if' simulations.

Schneider Automation[72][68] strategy focusses on the development and production of new agent-based automation structures for intelligent production systems, control and scheduling algorithms, and tools that allow the intelligent control of complex distributed production systems. One of the main aims of Schneider Electric is the transparency of the software as a customer benefit. This lead to the name the umbrella *Transparent Factory*. This transparency has benefits such as instant access to traceable data, historical data, alarm management, direct access to key process indicators, remote diagnosis and maintenance, link with enterprise resources or production system.

The **Siemens**'s strategy called *Totally Integrated Automation*(TIA)[73][68] focuses on the collaboration diagrams. The Siemens solution has a suit concept called Simatic IT Suite which uses a modeling environment to define execution logic by rules. This level is called Simatic IT Framework that allow to model the production process via a graphical interface in terms of plant layout (physical and logical objects) and production operations. The Framework includes in its turn the modeling tools for production, business, operation and services. The use of modeling tools and formal rules is a strong and very usable configuration feature since it can configure a system and it can ensure a correct execution substantially faster and better than just using simple parametrization.

Other providers like Yokogawa, with *VigilantPlant* platform, or Invensys, with *Archesta*, also make use of a software platform to enable the integration of large-distributed control systems. In this context, next chapter introduces the JCOP FSM which is the main component of the JCOP Framework for the integration and automation of control processes within the four LHC experiments.

Chapter 4

The JCOP Finite State Machine (FSM)

Information technology is continuously adding functionality that facilitates the implementation of large-distributed control systems. In that context, the JCOP FSM tool is the main component for the integration, sequencing and automation of control processes at the LHC experiments. This tool forms part of the JCOP Framework and it uses the functionality provided by both, the PVSS-II SCADA system and the State Management Interface (SMI++). Using the JCOP FSM, a large control system is modeled by means of many co-operative software objects that, hierarchically controlled, follow a finite state machine logic.

This chapter starts discussing briefly the finite state machine concept. Then it follows by describing in detail the SMI++ toolkit and its evolution towards the JCOP FSM. Finally, this chapter ends with a discussion that intend to motivate the usage of the FSM concept as a control mechanism for large distributed software system. This last part, firstly explains the actual trend followed in the industry in the field of 'software agents' and secondly, it introduces antecedents of this technology in the HEP world.

4.1 The Finite State Machine concept

The concept of Finite State Machines (FSM) is used to model the behavior of a system by means of a limited number of states, transitions between these states, actions and events. The state reflects the present conditions of the system being a product of the past circumstances. Transitions are movements from one state to another and are described by conditions or rules which must be met to allow a state transition. Actions are activities to be performed at a certain moment. Finally, events, which are either externally or internally generated, trigger actions or rules and can lead to state transitions.

This concept of FSM concept can be applied to a wide range of applications in both hardware and software. In a hardware, the implementation of the FSM concept requires a register (to store state variables) and a combinational logic (to determines the state transition and the output). Logic gates, flip-flops, relays, or more sophisticated elements like PLCs are examples where the FSM concept can be applied. In software, countless applications can be found due to the major flexibility of the medium. This can range from games to HEP control systems. Furthermore, the FSM logic can be used to model many reactive system in many other areas of study such as biology, philosophy, linguistics or mathematics.

The graphical representation of this concept is achieved by means of state diagrams (also know state transition diagrams). There are different kinds of state diagrams that differ slightly and have

different semantic. Moore[74] and Mealy[75] machines are the two classical methods and mixed Moore-Mealy models can be also used. The main difference between both mechanism is that whilst in Mealy an output is based on its current state and an input, in Moore the output is determined by only the current state not depending on the input.

Now in the context of the object-oriented software, state diagrams are represented by means of standardized notations, from which, the most relevant today is the Unified Modeling Language (UML)[76][77]. In the development of a large back-end system, the UML state diagram (see Figure 4.1) is normally combined with other types of UML diagrams (e.g. class diagram, use case diagram, etc). The UML notation represents objects of a single class and track the different states of its objects through the system. UML is discussed again in Appendix B where it is used to illustrate the ATLAS control hierarchies.

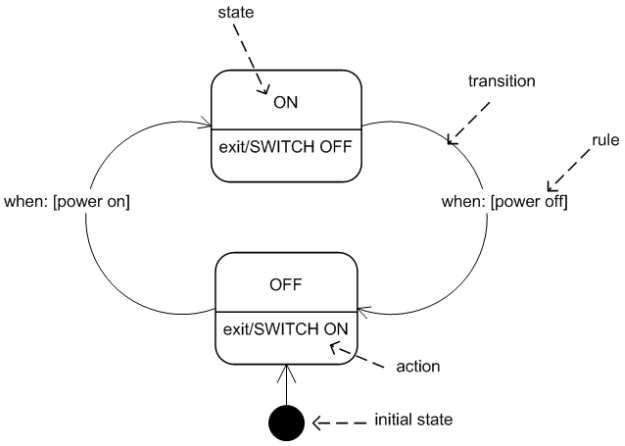


Figure 4.1: UML state diagram of an off-on sequence. The filled circle represents the initial state. Rounded rectangles denote the state, the top part contains the name and below the activity that can be an entry or output activity. The arrow denotes the transition where in brackets appears the rule or guard expression. This must be true to take place the transition.

Finally, in order to apply the FSM concept into a large system, the natural way is divide information in a large number of FSMs. Hence, each FSM defines a particular task and like that the whole can remain understandable. In ATLAS, and in most of the cases, this large amount of state machines are arranged forming hierarchical FSM with interconnections that carefully designed ensure the coherence of the entire system. Next, the technology used for the application of the FSM concept in the ATLAS DCS is described.

4.2 State Management Interface (SMI++)

SMI++ is based on the original State Manager concept[78], which was developed by the DELPHI experiment[60] in collaboration with the CERN Computing Division. The description of SMI++ given in this section is mainly based in the PhD thesis[61].

With the SMI++ framework the ATLAS control system is described as a collection of objects behaving as finite state machines which are associated with an actual piece of hardware or a real software task. Each of these objects interacts with the concrete entity it represents, through a so-called proxy process. The proxy process provides a bridge between the 'real' and the SMI++

worlds. In that way, the real process data is available into the SMI++ machinery, that at the same time, is able to command the processes.

The main attribute of an SMI++ object is its *state*. In each state, it can accept commands that trigger actions. An abstract object, while executing an action, can send commands to other objects, interrogate the states of other objects, and eventually change its own state. It can also spontaneously respond to state changes of other objects.

In order to reduce complexity of large systems, logically related objects are grouped into SMI++ domains. In each domain, the objects are organized in a hierarchical structure, and form a sub-system control. These basic concepts are graphically summarized in Figure 4.2.

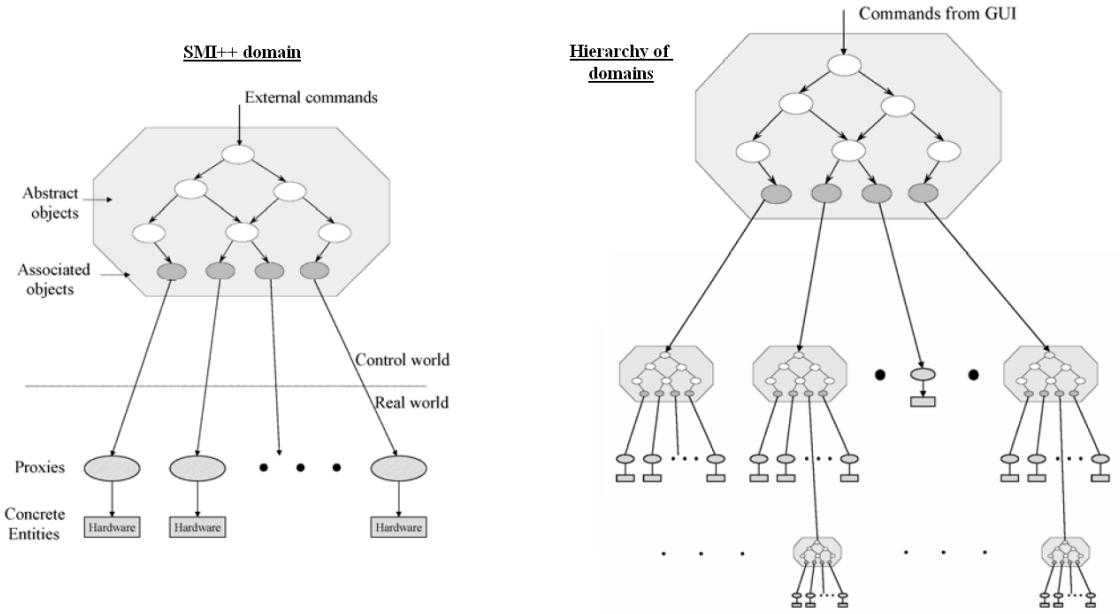


Figure 4.2: Left: One SMI++ domain constituted by proxies and abstract and concrete objects. Right: A final control system constructed as a hierarchy of SMI++ domains. Usually only one object (the top-level one) in each domain is accessed by upper domains. Picture taken from [79].

All issues related to distribution and heterogeneity of platforms are transparently handled by the underlying communication system DIM on top of TCP/IP (see Section 3.5).

The SMI++ framework consists of a set of tools. A special language called State Manager Language (SML) is used for the object description. The SML description is then interpreted by a logic engine called State Manager (SM) coded in C++ that drives the control system. Figure 4.3 shows both, an example of SML code and the main classes of the SM logical engine. Following the SML and the SM logical engine are introduced more in detail.

4.2.1 State Manager Language (SML)

This language allows for detailed specification of the objects, such as their states, actions, and associated conditions allowing the specification of classes of objects. The main characteristics of this language are the following:

- *Finite-State Logic*: Objects are described as FSMs. The main attribute of an object is its state. Commands sent to an object trigger object actions that can change its state.

- *Sequencing*: An action performed by an abstract object is specified as a sequence of instructions which mainly consist of commands sent to other objects.
- *Asynchronous Behavior*: All actions are proceed in parallel. A command sent by object A to object B does not suspend the instruction sequence of object A (i.e. object A does not wait for completion of the command sent to object B before it continues with its instruction sequence).
- *Rule Based System*: Each object can specify logical conditions based on states of other objects. These, when satisfied, will trigger an execution of the action specified in the condition. This provides the mechanism for an object to respond to unsolicited state changes of other objects in the system.

An example of SML code is shown in Figure 4.3. Two objects are declared, RUN CONTROL (abstract object) and CONTROLLER (concrete one). For both objects the list of possible states and the list of possible actions in each state are specified. For instance, in object RUN CONTROL the action START RUN is possible when it is in state READY, this action consists on sending command MOUNT to object TAPE and checking if the TAPE object reaches state MOUNTED. The action START RUN will eventually terminate by setting the state of RUN CONTROL to either RUN IN PROGRESS or ERROR. In state RUN IN PROGRESS a set of WHEN conditions is specified, these conditions are valid during the time the object is in this state and indicate that whenever one of them becomes true the corresponding action will be executed.

4.2.2 State Manager (SM)

This is the key tool of the SMI++ framework. It is a program which, at its start-up, uses the SML code for a particular domain, and becomes the SM of that domain (see Figure 4.2). In the complete operating control system there is, therefore, one such process per domain (called smiSM process). When the smiSM process is running, it takes full control of the hardware components assigned to the domain, sequences and synchronizes their activities, and responds to spontaneous changes in their behavior. It does this by following the instructions in the SML code, and by sending the necessary commands to proxies through their associated objects. In a given domain, it is possible to reference objects in other domains. These are then locally treated as associated objects, with their relevant proxies being the other SMs. In this way, full cooperation among SMs in the control system is achieved.

SM is coded in C++ and its main classes (see Figure 4.3) are grouped into two class categories:

1. **SML Classes**. These classes represent all the elements defined in the language, such as states, actions, instructions, etc. They are all contained within the *SMIObject* class (representing SMI++ objects). At the startup of the process, they are instantiated from the SML code.
2. **Logic Engine Classes**. Based on external events, these classes 'drive' the instantiations of the language classes. These classes are:
   (a) *CommHandler* is the communication handling class responsible of all the communication issues based on DIM. Commands from external objects or user interfaces are sent to the ExternalActionQ. New states from external objects are sent to the StateQ. It also publish new states of internal objects and sends commands to external objects whenever the SMIObject class requests it.

(b) *Scheduler* implements a sort of 'main loop', it sleeps until a new event happens and then operates on the SMIObject instantiations in such a way that each local object executes its own thread.

(c) *ExternalActionQ* queues the actions coming from an external object.

(d) *StateQ* queues new states of objects either coming from outside (CommHandler) or from internal objects (SMIObjects) are stored.

(e) *ExecutableObjQ* queues objects ready for execution waiting for their turn.

(f) *WhenHandler* evaluates 'when' conditions when new states arrive, from which might result a new object ready for execution.

(g) *IfHandler* evaluates 'if' conditions for SMIObjects pending on state busy. The result is communicated directly to the relevant object to continue execution.

```
object : RUN-CONTROL
  state : READY
    action : START-RUN
      do INITIALISE DATA-LOGGER
      if ( DATA-LOGGER not_in_state READY) then
        move_to ERROR
      endif
      do START CONTROLLER
      if (CONTROLLER in_state RUNNING) then
        move_to RUN-IN-PROGRESS
      endif
      ...
  state : RUN-IN-PROGRESS
    when (DATA-LOGGER in_state FILE-FULL) do PAUSE
    when (CONTROLLER in_state ERROR) do ABORT
    action : ABORT
      ...
    action : PAUSE
      do PAUSE CONTROLLER
      do INITIALIZE DATA-LOGGER
        ...
      move_to PAUSED

object : CONTROLLER / associated
  state : READY
    action : START
      ...
  state : RUNNING
    action : PAUSE
    action : ABORT
```

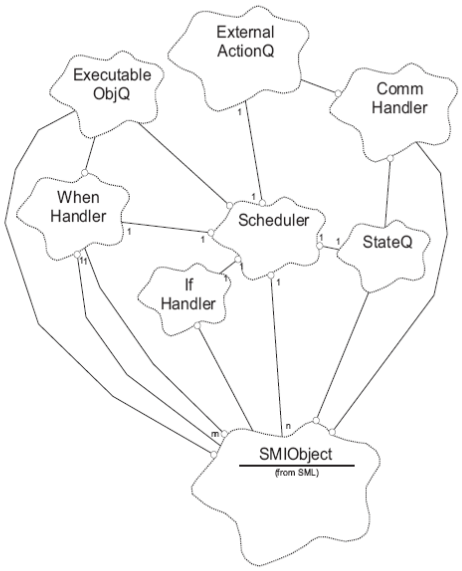


Figure 4.3: Left: Example of SML program. Two objects are declared, RUN CONTROL (abstract object) and CONTROLLER (concrete one). For both objects the list of possible states and the list of possible actions in each state are specified. Right: SM Logic Engine main classes. The execution of a Logic Engine has two phases, first the creation and instantiation of all SMIObject objects according to their description in SML and then starting the Scheduler. The Scheduler (when idle) is triggered by outside events i.e. by DIM encapsulated by the CommHandler class so it is completely asynchronous, no pooling is ever involved[79].

Lastly, the functionality of SMI++ technology (available from the DELPHI era) has evolved for its usage in the control of the LHC experiments. The new resultant tool is discussed next.

4.3 The JCOP FSM, result of the SCADA - SMI++ Integration

Since the SMI++ toolkit is a collection of tools developed in C++, it was possible to integrate the toolkit on top of SCADA taking profit of the openness provided by the API functionality of

PVSS-II. The result of this integration is called the JCOP FSM. Hence, for the LHC experiments controls, it will be the first time that the SMI++ functionality has been mixed with SCADA. This combination of functionalities brought several advantages to both, the JCOP Framework and the SMI++:

1. The SMI++ provides behavior modeling to the JCOP Framework. Abstract and concrete parts of the experiment having different behaviors can be modeled with SMI++.
2. SCADA's provides a database to store the SMI++ description and configuration, i.e. the same database will contain device description and behavior.
3. The SCADA archiving tools can be used to store state changes.
4. SCADA provides user interface building capabilities to SMI++.
5. SCADA provides an integrated graphic editor to be used by the SMI++ developer.
6. SCADA provides device access (e.g. OPC, Profibus, drivers).
7. SCADA provides a scripting language to derive a state out of monitored data and to implement actions on the devices.
8. SMI++ extends SCADA's functionality to control non-supported platforms.

4.3.1 Types of JCOP FSM objects

This section mainly describes the different software objects available for the developer within the object-oriented JCOP FSM toolkit. As mentioned before, the control tree is composed of concrete and abstract segments of the experiment represented by objects. Whilst concrete segments normally model and control directly hardware devices (e.g. a pump), the abstract segments group these devices and control them from a more abstract hierarchical level(e.g. a sub-detector). In order to represents these different segments of the experiment, the JCOP FSM provides the developer with three types of software objects that can be placed at different levels of the control hierarchy (see Figure 4.4):

- *Control Unit* (CU). It is an abstract object (e.g. whole ATLAS, LAR sub-detector, TRT HV system, etc) corresponding to one smiSM process capable of containing children of any type (i.e. CUs, LUs or DUs). These objects are written in SML.
- *Logical Unit* (LU). It also represents an abstract object, but in this case, located within a smiSM process. It can contain children, but not of type CU. LUs have restricted functionality compared with CUs but by using LUs the number of smiSM processes is reduced, and at its turn, the performance improved. Hence, using LUs one is able to create hierarchies with larger number of nodes (see more details in Section 6.4). These objects are again written in SML.
- *Device Unit* (DU). It corresponds to a concrete object in PVSS (e.g. a valve, a HV channel, etc) and, in that case, they cannot contain children of any type. These objects are written in the PVSS scripting language (instead of SML code). Consequently, since these objects can be inserted at any level, they can easily add functionality to the control tree by using the power of the SCADA system. An API manager called PVSS00smi is in charge of the communications with the SMI processes.

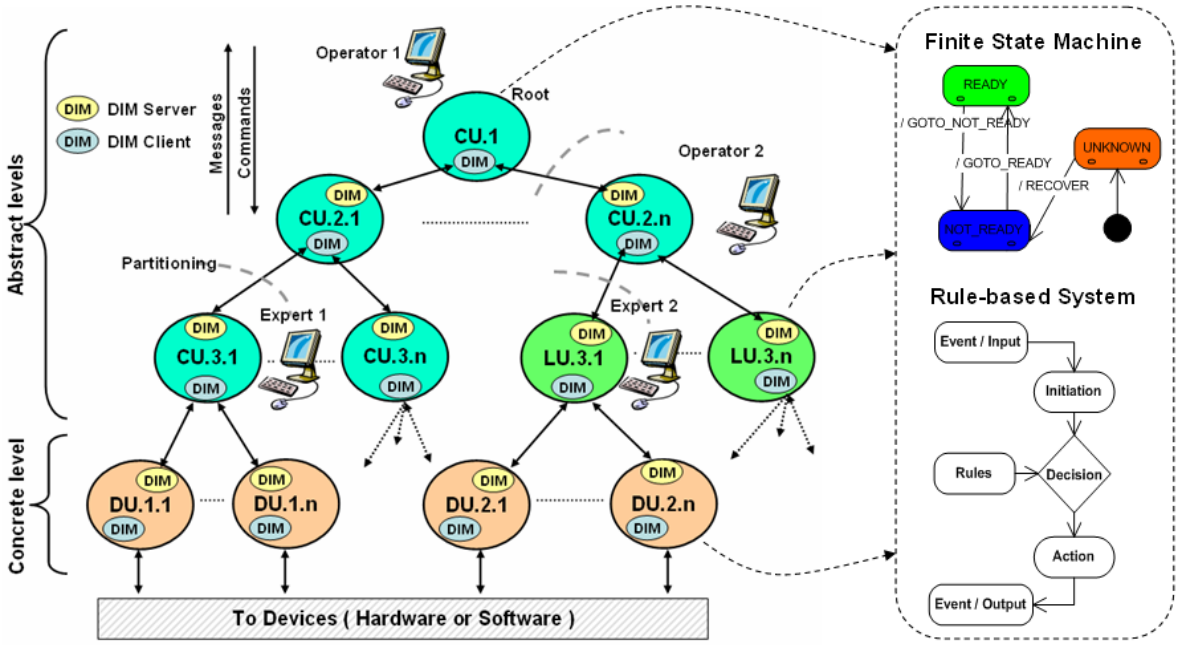


Figure 4.4: Different types of objects with different functionalities are allocated at different levels of the control hierarchy. 'Commands' flow down and 'messages' flow up through the underlying DIM protocol. Each object is controlled hierarchically mixing the functionality of a finite-state machine logic and a rule-based system.

These different kinds of objects will describe the behavior of the many distributed systems that, co-operating, allow the sequencing of control processes in ATLAS. However, in some cases, it is necessary that parts of the detector run in stand-alone mode. *Partitioning* is the feature that allows this and it is discussed next.

4.3.2 Partitioning

The general control of the experiment, composed of different specialized sub-detectors, requires a flexible partitioning concept which allows to operate, debug, commission, calibrate or test different parts of the experiment simultaneously. This feature, which is offered by the JCOP FSM tool, is very appreciated in the HEP domain where large number of independent teams and very different operation modes coexist.

Partitioning implies also the concept of *ownership*. In order to send commands to the different components an operator can reserve the whole tree or a sub-tree in which case he/she becomes the 'owner'(each component has normally one owner at any time).

Concerning the different partitioning modes, Figure 4.5 shows graphically the modes supported by CUs (SMI++ domains). Each CU can partition 'out' or 'in' its children. Excluding a child from the hierarchy can imply: (a) its state is not taken into account any more by the parent in its deciding process, (b) the parent will not send commands to it, and (c) the owner operator releases ownership so that another operator can work with it (note that only the owner can exclude a component from the hierarchy).

Based on Figure 4.4, next it is shown an example of the usage of partitioning during operation. Several generations of delegation are carried out by operators and experts in order to intervene on

different parts of the experiment simultaneously:

1. The Operator 1 becomes the owner of the system;
2. the Operator 1 delegates the CU.2.n system to Operator 2:
3. the Operator 1 delegates the CU.3.1 system to Expert 1 for a new calibration;
4. the Operator 2 disables LU.3.1 in order to be debugged by Expert 2.

Following with the example, the hierarchy now has four partitions that work independently and concurrently: the Operator 1 still manages the majority of the experiment, Operator 2 would manage a certain sub-detector (i.e. CU.2.n), Expert 1 would calibrate for example a HV system (i.e. CU.3.1) and Expert 2 would debug a problem arisen in for instance the cooling of another sub-detector (i.e. LU.3.1).

Concluding, the partitioning concept offered by the JCOP FSM tool permits to pass from a central control (e.g. for central diagnostics or start-up) to a distributed local control (e.g. for debugging or commissioning). This software partitioning feature, which is essential for the operation of any modern HEP experiment, in contrast seems to be not widely applied to the control of existing industrial plants forcing in some cases a choice between a centralized or a decentralized control topology (see [80]).

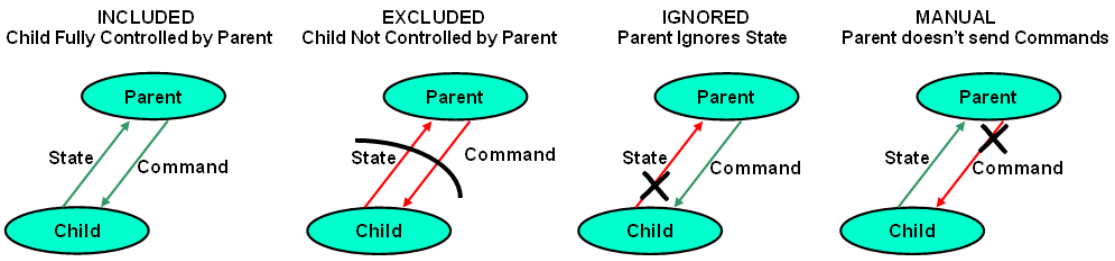


Figure 4.5: Partitioning offers the capability of operating parts of the system independently and concurrently.

4.4 Distributed Software Agents for Controls

The JCOP FSM component can be understood in literature as a software agent technology since the different objects that shape the control tree fulfil the main requirements of what is called 'intelligent software agents': autonomy, goal-oriented, social ability, asynchronous and reactivity[81]. This section is divided in two parts which describe the usage of the software agent technology into different fields of application. The first part is basically a summary extracted from different papers and discusses briefly the application of software agents into the industrial control market today. The second part discusses the usage of this technology, and more in particular SMI++, for the controls of previous HEP experiments.

4.4.1 Implantation of Software Agents in the Industry

As discussed in the previous chapter, makers of industrial controls make use of relatively new technologies like high-speed networks, software agents and the object-oriented paradigm to increase the functionality of many existing industrial segments. Industrial informatics, with its inherent flexibility, is able to act vertically at all levels of the production process, ranging from the overall planning of operations down to device integration. Therefore, information technology is able to add continuously value to existing industrial architectures. However, as yet, few commercial software solutions in the full plant automation or manufacturing are available. The reason of this fact would not be directly related to the information technology as such, but more to the nature of this market[68].

Forrester Research institute, in an analysis of the agent technology market done in 2003, showed that the development of the industrial control market is stopped by the lack of 'killer applications', development tools and multi-vendor critical mass. In a prediction done until 2008+, Radjou[82] distinguished three phases which are shown in Figure 4.6. In the first phase, agents built monitoring applications to sense and interpret data. In the second phase, agents built management applications to coordinate activities with partners. Finally, in the third phase that represents the present moment, agents built decentralized and automation applications. In the Forrester's study, which looks to be correct, automation and plant control are not among the 'killer applications', or 'applications domains' until next year 2008. Historically, there are well-know explanations for such predictions.

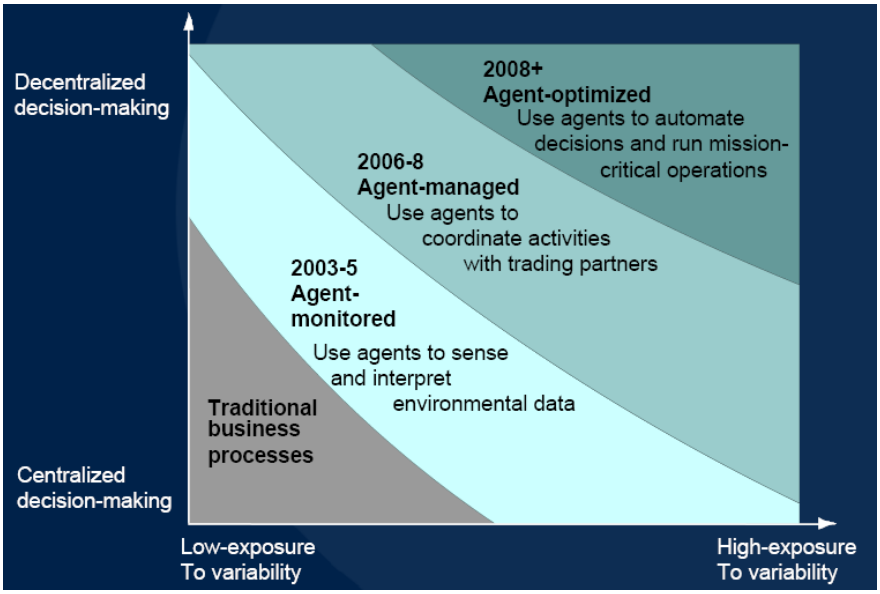


Figure 4.6: Forrester prediction for agents technologies. Automation and plant control are not among the 'killer applications', or 'applications domains' until 2008. Picture taken from [83]

The equipment used for control and automation is normally the result of an incremental development. The first electronic systems had only measurement and the operators' work were to interpret the measured results. Later, hardware controllers replaced the human operators and they passed to a supervisory role. In the next stage, a computer will probably be in charge of the supervisory role and the manual phase moved to planning. As a consequence of this technical evolution and the cost associated to the re-engineering, most existing industrial plants are using a

mix of equipments from different vendors that are required to work together. Operators and experts using the equipment have training, approved working procedures and mental models associated to certain automation process and, all these, make the automation market slower in applying radical new solutions. This may be also the reason why as yet, not many applications of integration of low-level (data acquisition, regulatory control, data reconciliation), mid-level (supervisory control, process monitoring/fault detection and diagnosis) and high-level tasks (planning and scheduling) have been developed for processing plants [84].

4.4.2 Antecedents of SMI++ in the HEP world

Within the HEP world, the author only found one tool offering functionally similar to SMI++, this is the Concurrent, Hierarchical State-Machine (CHSM)[85]. CHSM includes an object-oriented language definition, a translator for converting programs written in the CHSM language into C++ and a runtime library for supporting the generated code. CHSM offers a good method and associated tools that allow the modeling of an application. Although CHSM was tested by the ATLAS data acquisition controls group[86], a lack of flexibility was found in the state machine implementation and difficulties for its apprenticeship. As a result, its usage was rejected.

In previous experiments to the LHC, the DCS was not organized hierarchically and no FSMs synchronized the different stages of the experiments. At that time, the amounts of channels to be controlled by the DCS were much smaller. However, as the detector controls are growing in size and complexity, it has been found very useful to organize the experiment hierarchically by means of FSMs that automate the processes. At present, there exist only two antecedents of the usage of SMI++ in HEP experiments(see Figure 4.7):

1. The DELPHI [60] experiment at LEP decided to take a common approach to the full 'experiment control' in 1997. The full online system was designed and implemented using SMI++. The various areas of DELPHI were mapped into SMI++ domains: sub-detectors, data acquisition system, slow controls, trigger, etc. The full system consisted of about 50 different smiSM domains distributed over 40 computers. During normal operation the so called 'Big Brother' piloted the system with minimal operator intervention.

2. The BaBar detector, which exploits the PEP-II facility at SLAC [87], decided in 1997 to design the Run Control by using the SMI++ framework. The first prototype was installed in the second half of 1998, ready for the subsystem groups to test their equipment. At the top of the hierarchy a smiSM domain, namely 'Master', monitors and controls the BaBar detector hardware such as HV power supplies through the sub-detector domains. The full system consists of about 20 different domains. Under normal running conditions during data-taking, 'Master' monitors, synchronizes, and controls its subsystems fully automatically with again minimal human intervention.

At present, the application of SMI++ in the LHC experiments brings new challenges due mainly to the size and complexity of the new experiments. For the whole ATLAS hierarchy it is envisaged to have thousands of smiSM domains, being the dimensions of each one of the 12 DCS segments in ATLAS, larger than any of the two antecedents (DELPHI and BaBar). This fact, together with the combination of SMI++ with PVSS-II, provides a powerful toolkit but a complex control system to be built. As a consequence, issues such as scalability, homogeneity, error handling, automation, synchronization with DAQ and monitoring had to be studied beforehand. In the following chapters all these issues are tackled commonly for the whole ATLAS DCS in order to achieve an homogeneous, reliable and compressive control system.

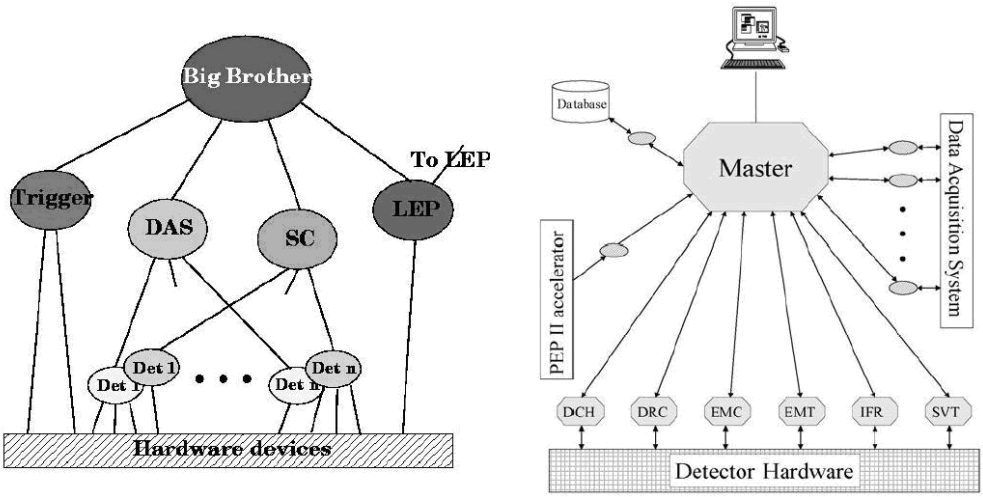


Figure 4.7: The most top node pilots the experiment composed of less than hundred control domains. Left: Delphi's experiment control. Right: BaBar's experiment control. Pictures taken from [79].

Part II

Design Phase

Chapter 5

Motivation

Before to the LHC experiments, known control hierarchies at the HEP world presented a much minor scale. This fact brings us open questions such as scalability, homogeneity, monitoring of operation and handling of errors that need to be solved before the final implementation of this technology in ATLAS.

This chapter presents the first prototype of control tree implemented for the ATLAS DCS. During that work, carried out in spring-summer 2004, key factors for the final design of the ATLAS control hierarchy were found out. The scope of this chapter is to motivate all those implementations discussed later on this thesis.

5.1 DCS Set-up in the building SR1

The building SR1 provides the infrastructure needed for the integration and testing of the components belonging to the ATLAS inner detector (i.e. Pixel, SCT and TRT)(see Figure 5.2). The building is mainly distributed into three different areas as shown in Figure 5.2. These are: a clean room, where the detectors modules are being tested and assembled; a rack area, to allocate off-detector instrumentation (e.g. power system); and a control room, where the three sub-detectors have stations supervising and controlling all the elements. Note that this distribution in different areas of equipment and functionality is similar to the final ATLAS infrastructure presented in Chapter 2. Consequently, the SR1 building became an excellent work bench to test the DCS and the JCOP FSM tool.

The first running project using the JCOP FSM tool at the building SR1 was the Common Infrastructure Controls (CIC). The CIC monitored the environmental conditions within the clean room for ensuring a safe testing and assembling of the silicon tracker and TRT. In addition, the CIC also monitored the electronics racks where the power systems of the sub-detectors were housed.

In order to set-up the CIC, suitable humidity and temperature sensors were selected on the market. These sensors were then read-out through the standard way: these were connected with ELMBs, which via CANbus, furnished the SR1 control room with DCS data. At the same time, one station located within the clean room area accessed the Cooling and Ventilation (CV) information which was published through the DIP protocol. In that way, by checking the regulation of the CV's PLC, environmental disturbances in the building could be predicted in advance.

Besides the CIC project, each sub-detector system in the building counted with several DCS systems, at that time, these were: SCT power system, SCT environment, evaporative cooling, thermal Enclosure, TRT power systems, and TRT mono-phase cooling (for further information

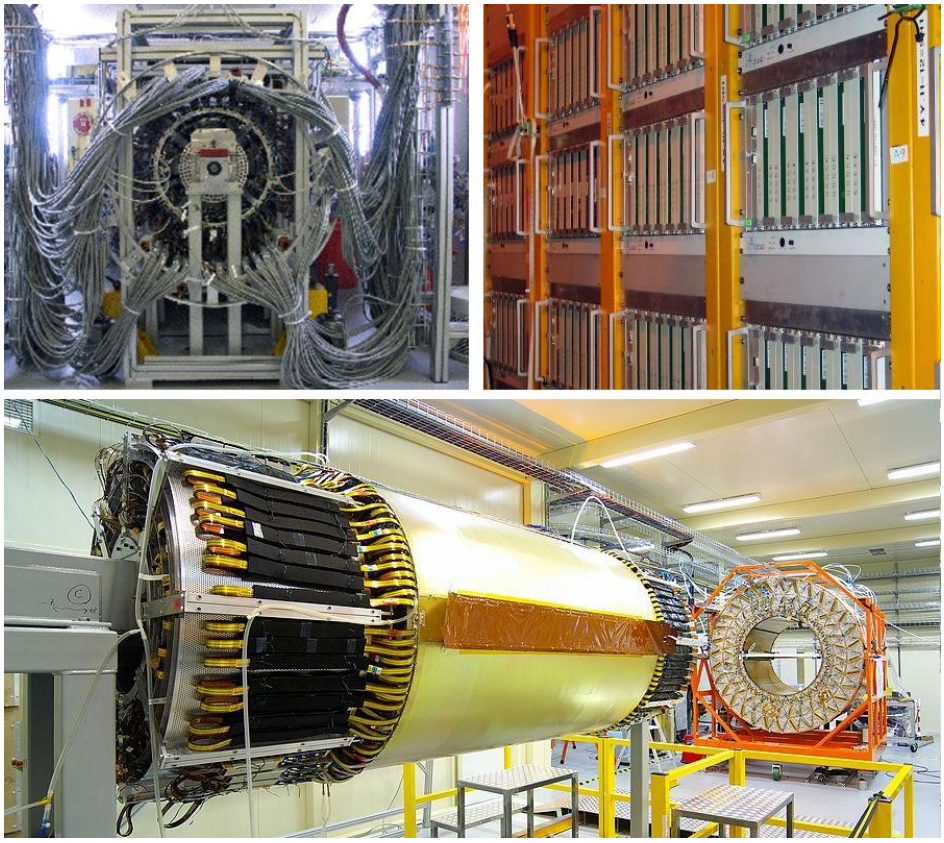


Figure 5.1: Top-left: SCT barrel during acceptance test. Top-right: power system in the rack area. Bottom: In the clean room, preparing the SCT for insertion into the TRT. Pictures taken from [13] and [25]

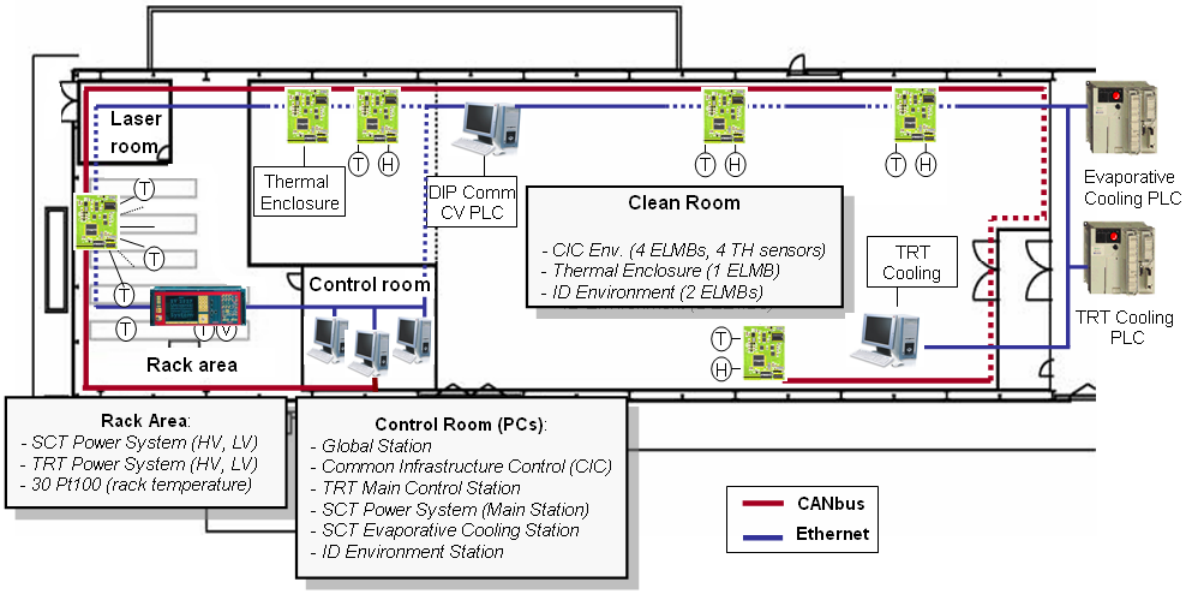


Figure 5.2: Schema of the building SR1. Similarly to final ATLAS most of the DCS instrumentation is separated from the detector (i.e. the power systems sitting in rack area feed the detector being assembled within a clean room). The Common Infrastructure Control (CIC) supervises from the Control Room the environmental conditions of the clean room and racks.

about any of these systems see Section 2.3). At that time, all the different systems worked in stand-alone mode and, the next step was, to establish a PVSS-II distributed system in order to provide CIC data to the rest of systems. Hence, a distributed project with seven PCs and was set up.

These machines then were integrated in a tree-like hierarchy using the JCOP FSM tool (see Figure 5.3). The most top node, located in a global station that had connection with the rest of PCs, mastered the whole control system. Below on the hierarchy, the CIC and two sub-detector main stations, corresponding to the SCT and TRT detectors, were built. At the bottom, the JCOP FSM machinery handled the stations controlling the hardware (i.e. evaporative and monophase cooling projects, environment, thermal enclosure, HV and LV systems).

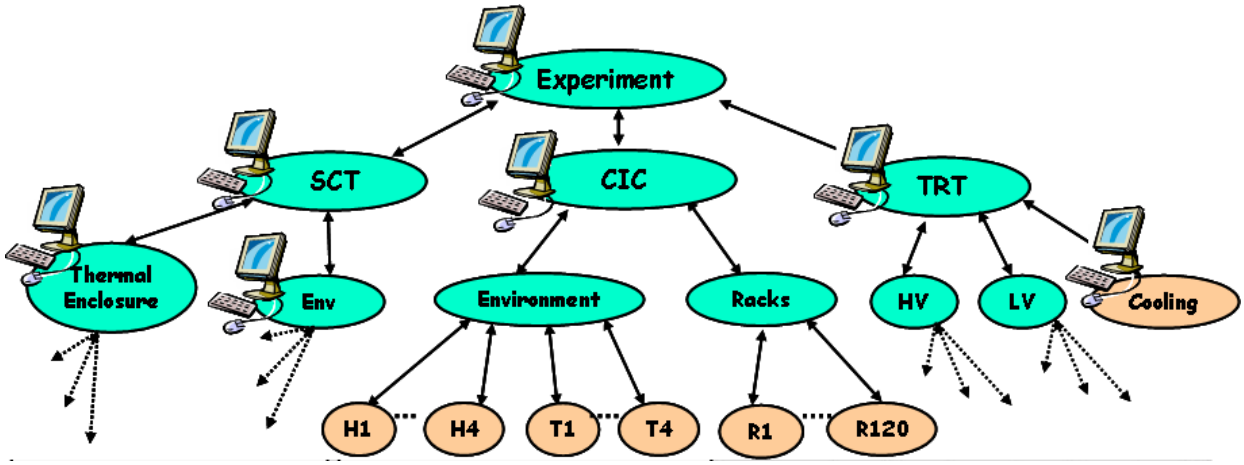


Figure 5.3: Prototype of hierarchy for the building SR1. The tree had about a hundred nodes distributed in 7 PVSS-II systems with a name server running in the CIC PC.

From the work-bench set-up in the building SR1 many things were learnt. The final implementation of the control hierarchy was satisfactory; any change on the operational mode of any of the systems was propagated up through the control-tree being all information available in a single station. This global station, controlling the whole, was able to excluded/included parts of the tree when necessary (e.g. part of the tree not working or being debugged) and, eventually, commands from this station were sent (mainly for resetting CIC ELMBs) with success.

Although the good performance obtained, one main open issue had still to be studied, this was the *Scalability*. Even with only 3 sub-detectors integrated in the control tree, and with only a small part of their equipment included, this prototype lead us to a hierarchy formed by about a hundred nodes (i.e. CUs and DUs), which is the same order of magnitude than DELPHI or BaBar. Hence, based on this exercise, it was envisaged that the final control tree in ATLAS could be up to a thousand times bigger than any antecedent. Two main reasons could explain this huge increase on the amount of nodes:

- The complexity and size of the ATLAS experiment. For instance, the ALEPH experiment[88] at the LEP accelerator counted with around 7000 channels to be supervised by the DCS whilst in ATLAS, the order of magnitude is almost 30 times higher.
- In the past, the granularity defined through SMI++ was relatively coarse; i.e. the bottom level of the control tree was formed by systems such as the data acquisition or a sub-detector.

In contrast, SMI++ is now integrated with PVSS-II, which is a 'device-oriented' SCADA system. As a consequence, the PVSS-II DPs define a finer granularity (e.g. a HV channel or a valve) that increases the accuracy of the modeled architecture but also the amount of nodes to be used.

5.2 Early Conclusions and Further Work

Previously to the LHC experiments, SMI++ was applied in a much minor scale without the usage of a SCADA system. Therefore, taking into account the scale of ATLAS (the DCS will supervise around 200.000 channels) the question now is whether the classical approach is still valid or could be revised. Listed below are the main conclusions extracted from this exercise:

1. Organizational aspects. Physics experiments are designed, built and operated by several different groups of people around the world. Organization styles, distance and even languages are factors to take into account. Hence, in order to build an homogeneous control hierarchy well-defined guidelines will need to be established. These may define issues such as functionality required at the different levels, behavior of each node, number of nodes and levels forming the control tree, look-and-feel, naming conventions, etc.

2. Handling of error states. In all likelihood, transitions to error states will occur often in a control tree formed by thousands of control domains. Figure 5.4 shows the finite state machine logic used in the building SR1 for defining the behavior of a HV node. This logic follows the 'classical approach' where 'error' is a possible state of a node. The question at this point is how to deal with the propagation of errors. The main constraints found following the classical design were:

   (a) Whenever a problem arose deeper in the hierarchy (e.g. a temperature being too high), it was propagated up though the control tree up to the most-top node which implied the lost of the overall state information (e.g. *Ready* was replaced by *Error*). A possible solution to this would be, relying on the PVSS-II alarm screen, to 'hide' small faults and only propagate the important ones. This solution was rejected soon because it could imply the lost of an error state for an uncertain period within a such large control system.

   (b) A state *error* may not give enough information about the severity of a problem. More accuracy could be necessary defining the behavior of a node. This could be solved by increasing the amount of states, however, again, difficulties on handling these states on higher abstraction levels would appear.

3. The sequencing and automation of the systems was very well defined by SMI++ requiring minimum human intervention. However, whenever an intervention from an expert was needed it was very difficult to handle the large amount of displays generated by the JCOP FSM toolkit. Each node in the hierarchy had a display associated (see Figure 5.4) and, even for the very first test, where the amount of nodes was reduced, it was a difficult task to navigate and find the problem to act upon. Consequently, further research and development in operation tools was a mandatory issue.

4. The JCOP FSM is the main tool for the sequencing and automation of the DCS. However, the DCS control tree still needed to interface other DCS tools, external systems, etc. In that sense, a way to integrate within the tree tools like the configuration database or systems like TDAQ had to be studied.

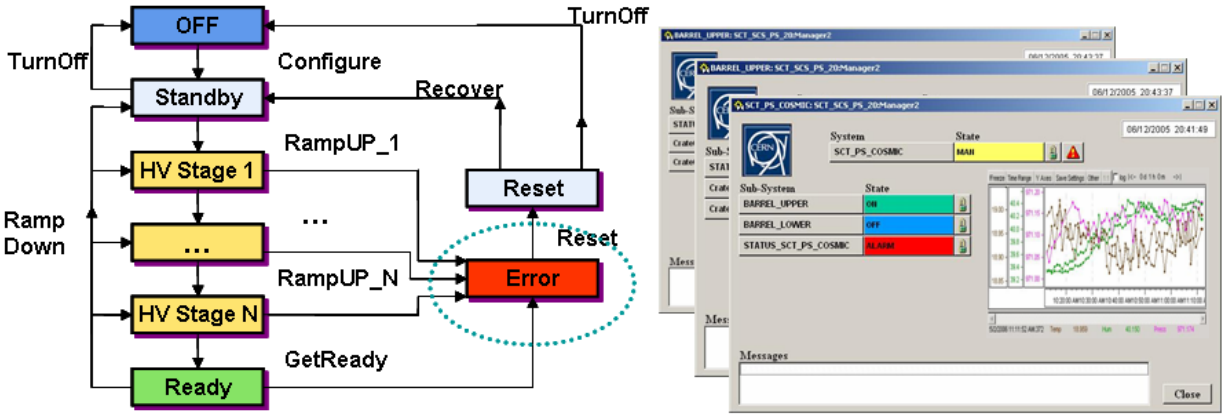


Figure 5.4: Left: classical FSM approach defining the behavior of a HV node where *Error* is a possible state. Right: series of operator interfaces, each node is referred to a display.

Above they are listed coarsely the issues tackled in the following chapters in order to build the ATLAS DCS control hierarchy. In Chapter 6 the design philosophy is discussed. Chapter 7 gives two practical examples of the integration of the sub-detectors within the overall control tree and the interaction between the DCS control hierarchy and the sequencing of the DAQ processes. Finally, Chapter 8 presents the top level interface, which based on the control hierarchy, represents the main facility for driving the overall ATLAS DCS operations.

Chapter 6

ATLAS DCS Back-End Organization

Distributed control modeling involves matching the control model more closely with the physical system. As discussed in previous chapters, this distribution of control will be achieved in ATLAS by the emergent behavior of many simple, autonomous and co-operative software objects. However, in order to apply such a control model in a reliable and homogeneous manner to ATLAS, a common approach need to be followed by the many developers composing the team. This approach has been designed with stress on the detection, diagnose and monitoring of fault that eventually could disturb the operation of the experiment.

This chapter describes first the overall philosophy used to represent, at different functional layers, the sub-detectors, sub-systems and hardware components that constitute the hierarchical experiment control system. Finally, a validation test of the proposal is shown.

6.1 Model of the Back-End Organization

During the operation of the ATLAS detector many complex systems have to collaborate, and to face this complexity, the adopted solution is modularity. We build models because we cannot comprehend the system in its entirety. Consequence, a rigorous modeling is essential for the success of the project. This modularization is carried out by means of SMI++ objects that allow to break down a complex system into discrete pieces - which will then only communicate with one another through *standardized interfaces* within a *standardized architecture*[89].

The different systems that form a detector, such as high-voltage, low-voltage, cooling, etc, are normally produced by different teams using highly specific equipment, and therefore, the result is often heterogeneous. One way to manage this complexity is to reduce the number of distinct elements. In ATLAS, this is done in two different ways:

- Firstly, the software elements use the JCOP framework as much as possible avoiding duplication of work.
- Secondly, the distinct elements in ATLAS are reduced by grouping elements into a smaller number of sub-systems. The dynamic behavior of this grouping is described by means of the JCOP FSM.

During the design phase, the ATLAS DCS will be decomposed into modules. This division into modules involves a partition of information into what we call *visible design rules* and *hidden design parameters*[1]. The visible design rules consist of three parts:

- **Architecture**. It specifies the different constituent parts and their functions. Next Section 6.2 presents the ATLAS DCS standard architecture.

- **Interfaces**. They define how modules interact with each other and externally with the person in charge of the operation of the experiment. Next Chapter 7 presents the DAQ-DCS interface (crucial for the experiment coordination) and Chapter 8 the top level human-machine interface, in charge of the overall operation of the ATLAS DCS.

- **Standards**. They offer ATLAS guidelines for the implementation of the control hierarchy (e.g. common finite state machines with common states, naming conventions, etc). A common framework called '*fwFsmAtlas*' has been built in order to facilitate the integration while following the standards (see Appendix A). Section 6.3 presents the proposed philosophy to build the whole ATLAS control tree which seeks the integration of a fault-detection mechanism within the control hierarchy.

These visible design rules must be chosen such that they do not have a negative impact on functionality or performance. They need to be widely shared throughout the ATLAS community and also must not constrain the evolution of the ATLAS DCS during the long lifetime of the detector (up to 20 years). Hence, these rules must be flexible and continue to be applicable in the case that the BE or FE systems evolve.

In contrast, the hidden design parameters, which are the encapsulation of specific information of a certain module, do not need to be communicated beyond the boundaries of the module.

In the literature regarding modular systems the set of architecture, interfaces and standards is known as modularization[90]. In the next section the architecture is described specifying its different functional levels. Later, it is discussed the fault-detection mechanism, present at all levels of the architecture, and core of the ATLAS mechanism for building the control hierarchy.

6.2 Architecture

The BE system is organized in three functional horizontal layers and in all of them, JCOP FSM engines run to ensure a coherent operation of the whole[32]. The motivation and structure is that tasks and operations are ordered from a more abstract, less time crucial levels to more specific, data intensive and time crucial levels.

The BE hierarchy will allow the dynamic splitting of the experiment into independent partitions which can be operated in stand-alone or integrated mode. The coordination of the different partitions will be performed by means of commands and messages. Remote access to a well-defined set of actions to be performed by the two upper levels of the BE hierarchy will also be provided. Figure 6.1 shows the standard architecture of the ATLAS DCS and next, the three functional layers of this architecture are introduced.

6.2.1 Global Control Stations (GCS)

In the top layer, there will be the Global Control Stations (GCS) which are in charge of the overall operation of the detector. The GCSs will be able to access all stations in the hierarchy. They provide high level monitoring and control of all sub-detectors, while data processing and command execution are handled at the lower levels. They assemble all status information available and present it to the operator in a hierarchical fashion.

All anomalies of operation like warnings, alarms, etc. are collected and displayed. Actions from the operator like acknowledgement can be requested. The GCSs may trigger actions themselves or ask the operator to do so. Any parameter in the system can be inspected in real-time and its history can be plotted. Pre-defined commands can be given to the sub-detectors and to their sub-systems. Settings can be changed for selected parameters.

The GCSs will be available in the ATLAS control room SCX1 and, to some extend, externally via the internet. The different human-machine interfaces constituting the GCS are described in Chapter 8.

6.2.2 Sub-detector Control Stations (SCS)

The Sub-detector Control Stations (SCSs) form the middle level of the hierarchy. Each sub-detector has its own station and three additional ones will exist to handle the CIC, the IDE common infrastructure, and the TDQ rack control.

At this level, the DCS interfaces with external systems like the LHC accelerator, the CERN infrastructure, the magnet system, and the DSS which information is made available to the sub-detectors.

A SCS allows the full local operation of the sub-detector by means of dedicated graphical interfaces which are integrated into de GCSs. At the same time, each sub-detector also has a dedicated control room located next to the main SCX1. The SCSs translate global commands, such as 'run', into more detailed commands, e.g. for the individual power supplies, and send these to the LCS layer below for execution.

Finally, at this level in the hierarchy, the sub-detectors establish the connection with the TDAQ system in order to ensure that detector operation and physics data taking are synchronized. The interaction between DCS and TDAQ is described later in Chapter 7.

6.2.3 Local Control Stations (LCS)

The bottom level of the hierarchy is made up of the Local Control Stations (LCSs), which handle the low level monitoring and control of instrumentation and services belonging to the sub-detector.

The LCSs execute the commands received from the SCS in the layer above, but may also trigger predefined actions autonomously if required.

While creating the hierarchy, the designer must decide its granularity. This granularity defines the boundaries of the hierarchy and the information located below is encapsulated and not visible from the JCOP FSM. Hence, at this point, one has to find out the structure of encapsulation which will yield the best system decomposition.

The organization of this level for a given sub-detector is according to both, a geographical and a functional criteria. The former follows the topological decomposition of the sub-detector (e.g. wheels, disks, etc.) whereas the latter is referred to functions or services (e.g. cooling, high-voltage, gas, etc). These two levels, geographical and functional, can swap during the design phase to give more relevance to one or the other. This depends on considerations regarding control timing, error handling and configuration aspects. For example, it might be useful to group all HV systems of a partition to be able to switch them 'on' or 'off' altogether using a single CU. On the other hand, electrical power distribution might require controlling all systems of a specific geographical part of the sub-detector.

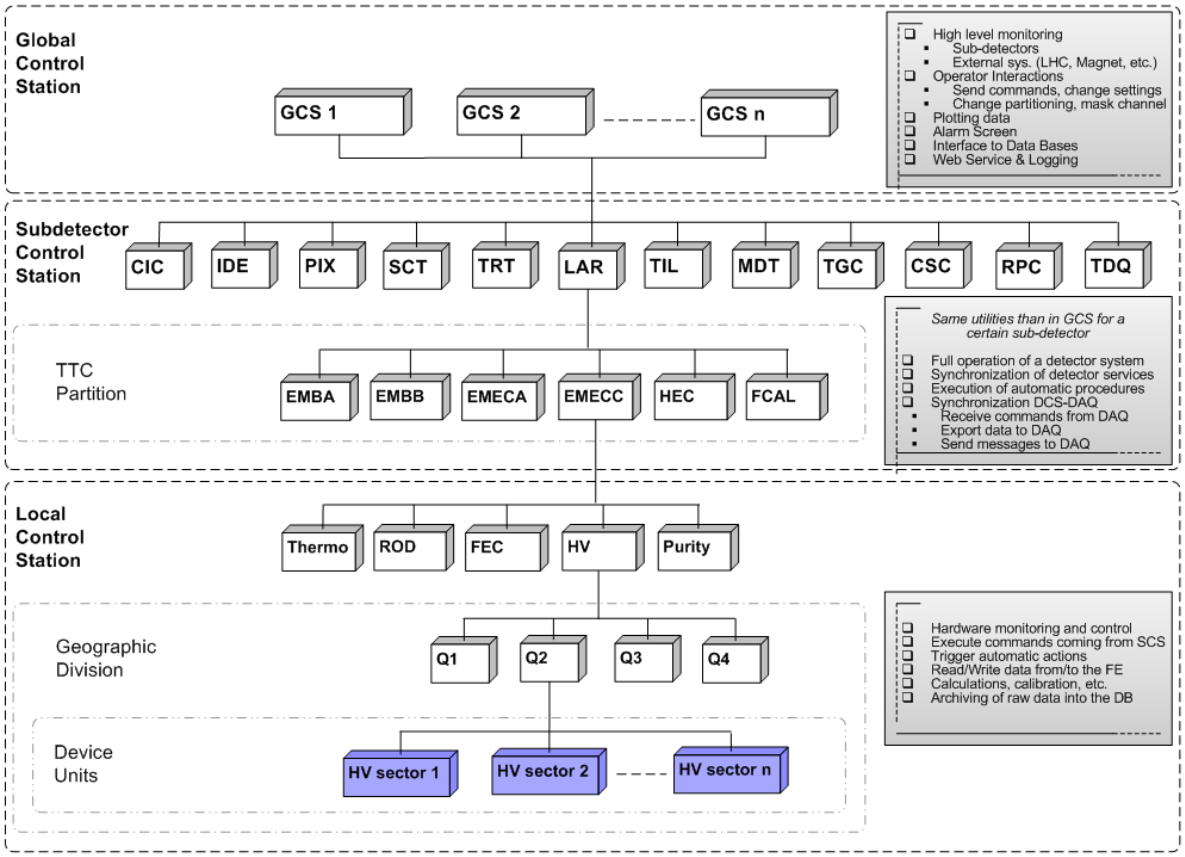


Figure 6.1: The BE is organized in three functional layers. In the most top layer the GCSs, mostly located in the control room, are in charge of the overall operation of the detector. In the middle layer the SCSs permit the full local operation of the sub-detectors and also assemble the overall status of the sub-detector passing it on to the GCS and to TDAQ. In the bottom layer the LCSs control the instrumentation and services that belong to sub-detectors setting like that the granularity of the control tree.

6.3 Fault-Handling Mechanism within the Control Hierarchy

In this section it is discussed the applied strategy for building the large control hierarchy of ATLAS experiment. This approach differs widely from previous implementations and the rest of LHC experiments.

The classical approach used in Delphi and BaBar had one communication path between the nodes of the tree. Through this path, changes in the states of the equipment were propagated up from one node to another in a level above. At the same time, commands were sent down by operators or automatically by a node to levels below. The method, for these applications, fulfill the requirements being the operator able to understand the full process. However, when dealing with the size of ATLAS, where the hierarchy will be certainly hundreds of times larger, several problems arise mainly due to eventual faults that can occur on the equipment (these are listed in previous Chapter 5).

Consequently, the strategy designed for the ATLAS experiment emphasizes the early detection, monitoring and diagnosis of faults based on dynamic fault data that is handled through SMI++. It is defined as a dynamic fault detection and diagnosis mechanism because the fault detection and diagnosis objects detect and diagnose the faults in their initial phase (i.e. the device unit), which are then propagated though the hierarchy.

Fault detection, monitoring and diagnosis are essential parts of the operation of any HEP experiment or industrial plant and an integral part for the coordination of tasks. Physics data taking involving the use of some damaged part would be disrupted, and operations from the supervisory control level could be no longer valid or dangerous. Moreover, when errors occur within a large system, it becomes harder to distinguish the root cause of a problem once the faults propagates through the large control tree and, therefore, the presence of faults can disturb the functional hierarchy structure making difficult the integration of tasks. Hence, early detection, monitoring and diagnosis of faults is crucial during the experiment run; otherwise, damage and lost of physics data can result before a fault present in the system is detected.

The approach used in ATLAS to handle with faults shows various advantages without the need of excessive computing resources by re-using existing SMI++ functionality.

6.3.1 The State-Status Concept. Double Communication Path

The adopted solution in ATLAS handles the information contained in the control tree by means of two different types of SMI++ objects namely, *State* and *Status* objects. These objects work in parallel and provide all the necessary information about the behavior of any node at any level in the hierarchy. These objects are:

- The **State** object defines the 'operational mode of the system' (e.g. the whole experiment, or a HV crate, is running or stopped). These objects can be CUs, LUs or DUs depending on their location in the tree.
- The **Status** object gives more details about 'how well the system is working' (i.e. it warns about the presence of faults). This objects are always of type LU, which means they are located within an smiSM process and do not require of important computing resources.

Hence, faults in the system are managed by the dedicated *Status* objects through which faults are passed on. Figure 6.2 shows and schematic view of the approach. In practice, it is implemented in such a way that the *State* objects are normally SMI++ domains where *Status* objects

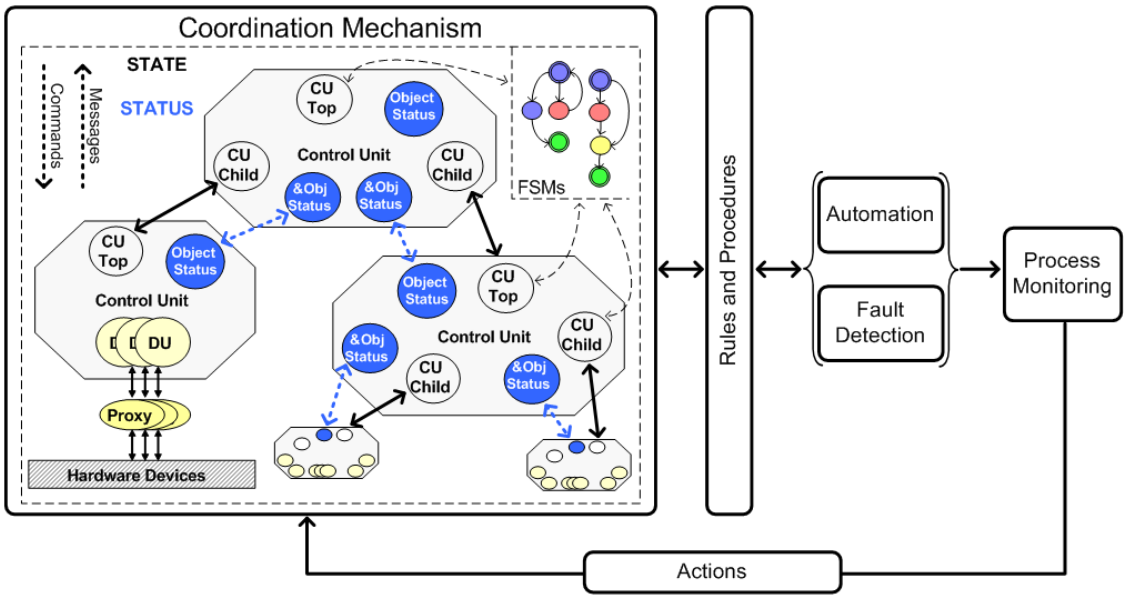


Figure 6.2: Fault-detection mechanism used when building the control hierarchy in ATLAS. Fault-detection and diagnosis SMI++ objects (*Status*) are added to detect and diagnose the faults in their initial phase (i.e. the device unit). These faults are then propagated through the hierarchy of status objects. Furthermore, rules and procedures are established within the SMI++ objects in order to set accurately the behavior of the nodes, automate the process and handle the faults. All these processes will be monitored allowing the operator to act quickly on a malfunctioning system.

are allocated or, in other words, each node has two qualifiers: state (master) and status (slave). Both kinds of objects have, of course, SMI++ functionality which implies to have two parallel sets of FSMs (bringing different kind of information) which are hierarchically controlled by other FSMs. Thus, rules and procedures can be established within the SMI++ objects in order to set accurately the behavior of a certain node under its specific conditions. This mechanism is used for both, sequencing of operations and detection of faults. Summarizing, the inherent advantages of the adopted approach are the following:

1. **Clear and complete definition of the node behavior**. The two aspects define more accurately the behavior for each node in the hierarchy. Thus, complex systems can be described more in detail. For example, Figure 6.5 shows how an error may be treated differently depending on the operational mode of the system. Depending on whether the *State* is *On* or *Off*, different severities can be attributed or different actions triggered. Moreover, defining a common set of *States* and *Status* will give consistency and facilitate the understanding of the node conditions.

2. **No loss of the operational conditions**. Information about the operational mode of a complex system or a group of systems is not lost when an error triggers. For instance, a HV system is in *Ramping_Up* state and this process may take several minutes to finish. If in the meantime an error occurs it can be propagated up by means of the *Status* objects while keeping the same *State*. This will help the operator to understand the operational conditions of the experiment (even for the smaller DU) even with the presence of errors.

3. **Further facilities for monitoring tasks**. The *Status* objects set severities to faults and summarize them hierarchically. As it can be seen in Chapter 8, this will also provide a fast fault-monitoring mechanism. In most of the alarm screens provided by SCADA vendors today, faults are filtered depending mainly on time and severity based on a flat organization of the alarms. Instead, the hierarchical approach facilitates the organization and handling of the faults viewing them from different levels of abstraction.

In order to homogenize the BE control hierarchy of ATLAS a set of rules and conventions have been defined by both State and Status objects. For consistency, the common control domains such as a SCS, HV system or an ELMB have the same (or similar) states, status, transitions and actions. Hence, a common finite-state machine logic and a short list of standard states and 'statuses' have been made available. This leads us to a more clear and common definition of the experiment conditions. Concerning the look-and-feel, similarly to telephone and social security numbers, operators will find natural to remember combinations of seven, plus or minus two, items[91]. Thus, rules have been also applied to the colors associated to states and statuses. Therefore, users should also make an implicit assumption that all things colored the same are associated in some way.

6.3.2 STATE Convention - Operation Mode

Two generic state machines has been proposed in order to homogenize the different control domains. The first state machine is shown in Figure 6.3 and corresponds to those control domains that represent abstract entities such as a SCS, a TDAQ partitions, a geographic partition (i.e. a quadrant, a wheel, etc), or an environmental system.

The abstract domains have two mandatory end-point states which determine the scope of the physics data taking sequence, these are: *Ready* and *Not_Ready*. These establish whether it is possible or not physics data taking. In between these states, the sub-detectors are free of defining their own states depending on the requirements. The sequence *Not_Ready* - *Ready* passes normally through all intermediate states. Two additional states out of the normal sequence are *Unknown* (i.e. lost of communication) and *Calibration*. These states are reachable from any other state.

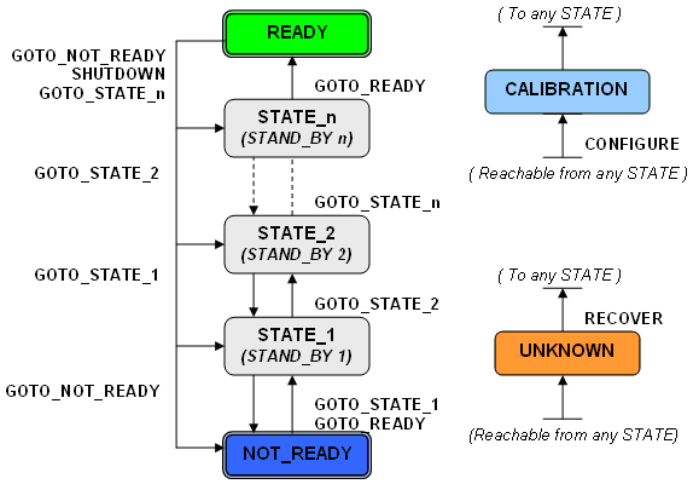


Figure 6.3: Left: state machine for objects representing abstract entities like a SCS, a geographical partition or, and environmental system. Right: Optional states.

The second generic state machine, which is shown in Figure 6.4, corresponds to those control domains that represent concrete device entities like a HV sector, a LV module, or a gas or cooling station. These control domains have again two mandatory end-points *On* and *Off*. In between these states, there could be transient states for those systems with slow response (i.e. *Ramping_Up*, *Ramping_Down*) and as many intermediate states as needed (i.e. *On25*, *On50*, *Standby*, etc).

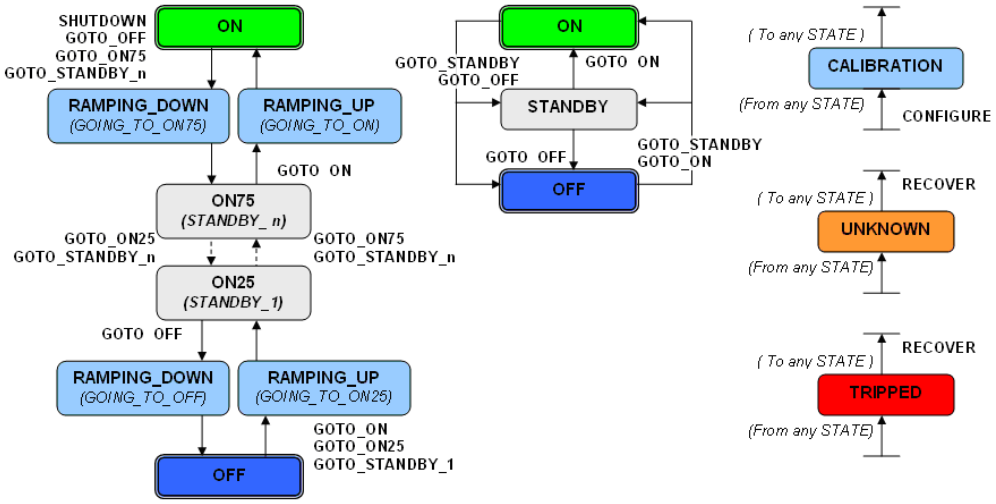


Figure 6.4: State machine for objects representing concrete entities, e.g. an HV system. Left: Example with transient states and multiple intermediate states between ON and OFF. Middle: Simple version lacking transient states and only one intermediate state. Right: Optional states.

Appendix A extends the standards of ATLAS to be used for the final integration of all systems forming the ATLAS DCS. It also describes other issues related to the hierarchical control of ATLAS such as installation of the 'fwFsmAtlas', naming convention or the procedure to follow in all those non JCOP Framework projects.

6.3.3 STATUS Convention - Fault Detection

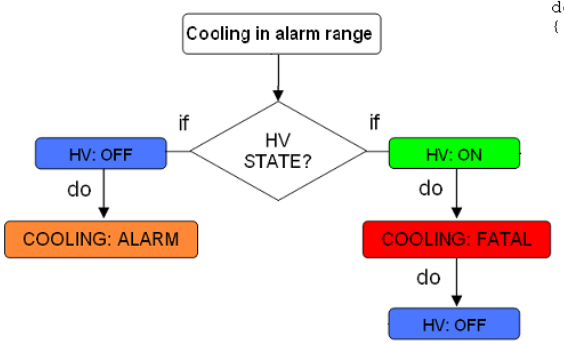
The *Status* names are fixed to a certain fault severity and are applicable to any DCS system in ATLAS. Table 6.1 shows the different status qualifiers that can be assigned to the presence of a fault and their meaning. The colors fulfil the look-and-feel convention.

In order to add *Status* objects to the control tree formed by *State* objects, functions had been made available within the 'fwFsmAtlas' (see Appendix A). This functions, together with guidelines, should facilitate the construction of the 'parallel' control hierarchy by the sub-detectors.

Status	
OK	System working fine.
WARNING	Low severity. The system can go on working. To fix in the following working hours.
ALARM	High severity. Serious error for the functioning of the system. To be fixed ASAP.
FATAL	Very high severity. The system cannot work. Run away!!!

Table 6.1: Possible status qualifiers. Different severities are assigned depending on the kind of fault and context. A fixed name and color is assigned to each kind of severity.

Finally, in order to illustrate the State-Status concept with an example, Figure 6.5 shows a basic implementation of this mechanism for a certain inter-process constraint. This example wants to show how depending on whether the state of a HV device is *On* or *Off*, different severities can be attributed to a problem related with, for instance, a cooling system. The cooling process, that is monitored all the time, could simultaneously act on the HV to switch it off.




```
STATUS_TRT_COOLING_valueChanged( string domain, string
device, float value, string &fwState )
{
     string stateHv;
     if (value == 1)
          fwState = "OK";
     else if (value == 2)
          fwState = "WARNING";
     else if (value == 3)
     {
          fwCU_getState("TRT_ECA_HV", stateHv);
          if (stateHv == "ON")
          {
               fwState = "FATAL";
               fwCU_sendCommand("TRT_ECA_HV", "GOTO_OFF")
          }
          else
               fwState = "ALARM";
     }
```

Figure 6.5: In some cases the severity of one fault can depend directly on the operational mode of a different system. Left: Diagram of activities to be followed after a cooling failure within alarm range. Right: code describing the behavior of the status object for the cooling system.

6.4 Validation Test of the Proposal

The performance of the JCOP FSM tool in the proposed organization, in terms of number of modules and levels of the hierarchy, has been investigated. In order to study the behavior of a real set-up, a work-bench simulating a SCS was developed in 2005. Below the SCS PC, twelve LCSs running on six PCs (i.e. 2 LCS per PC), were integrated. In each LCS, a three-level control tree was implemented, where each node had three children (i.e. 13 CUs per LCS, 26 CUs per PC). On the bottom level, different quantities of modules defining hardware components were used in order to run several tests. The largest set-up contained more than 10.000 modules (DUs), which is a factor three more than expected for the largest sub-detector. Several tests were carried out, such as forcing changes in all the modules at the same time with a certain rate, or checking the propagation time from the bottom to the SCS level after a change of state. The results of the test proved that SMI++ machinery worked properly when imposing the ATLAS approach presented above.

However, even if the first test were satisfactory, the number of SMI++ objects that can run in a PC is not unlimited. In order to provide recommendations to the ATLAS community we had first to distinguish (from the point of view of performance) between the different kinds of JCOP FSM objects (i.e. CUs, LUs and DUs). From the three, the main object adding overload to the system is the CU. The CU is the key tool of the SMI++ framework, it is a program which, at its start-up, uses the SML code for a particular domain, and becomes the SM of that domain (see Figure 4.2). In the complete control tree each CU is therefore, one such a process. The CU specifies a domain where the LUs and CUs are located. When the process is running, the CU can take full control of the hardware components assigned to the domain, sequences and synchronizes their activities, and responds to spontaneous changes in their behavior. This of course takes computing resources.

Figure 6.6 shows the results obtained when increasing the amount of CUs in a two-levels control tree using both approaches (with and without *Status* objects). In the test we used a Pentium 4 with 2,8GHz CPU and 768 RAM. The results show that the proposed strategy implies in average a 13,4% overload respect to the 'classical' approach. A hierarchy with 10 CUs consumed in total about 30 MB while a hierarchy with 100 CUs consumed about 245 MB. Light CUs with no children consumed about 3MB each whilst, a heavy one, with 100 children, consumed 63 MB. In other words, when increasing the number of children CUs, the parent CU increased the memory usage by ∼0,6MB per children. Thus, depending on the shape of hierarchy (i.e. how many levels and children had each parent), the memory consumption varies widely.

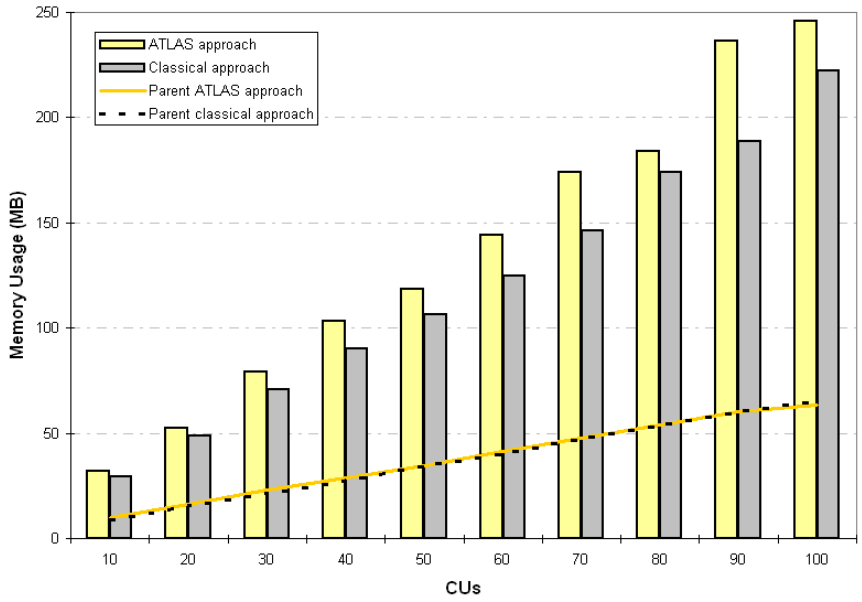


Figure 6.6: Comparison of the memory usage between the ATLAS and the 'classical' approach using a hierarchy of two levels. The parent CU counts from 10 up to 100 CU children. The columns represent the total amount of memory usage (i.e. parent plus children) whilst the dashed lines represent only the memory used by the parent CU. The application of the state-status concept in ATLAS implies in average about a 13% overload compared with the classical approach. In contrast, the parents CUs consume a similar amount of memory increasing ∼0,6MB per children CU. Around 100 CUs per PC looks to be a limit in terms of performance in both approaches.

On the other hand, while increasing the number of children, operations, such as to start/stop the machinery, or to send a command, demanded of more processor time. Figure 6.7 shows the results for three hierarchies of 30, 50 and 100 CUs each. Once the full machinery is started, we can observe that operations like changing the ownership mode (e.g. an operator 'take' or 'release' the control tree) vary from a few seconds (3-5 sec) with 30 CUs to almost 30 sec in a 100 CU hierarchy.

Concerning LUs and DUs both are light objects running within a CU. The CU itself does not suffer an important overload when a large number of LUs or DUs are 'attached' to its control domain. Figure 6.8 shows the low increase in memory usage suffered by a smiSM domain after including, indistinctively, up to 2000 new LUs or DUs. However, the CPU load can vary widely depending on the complexity defined within the PVSS script that defines the DU behavior. This is entirely in hands of the developer. At the same time, increasing the amount of DUs (connections

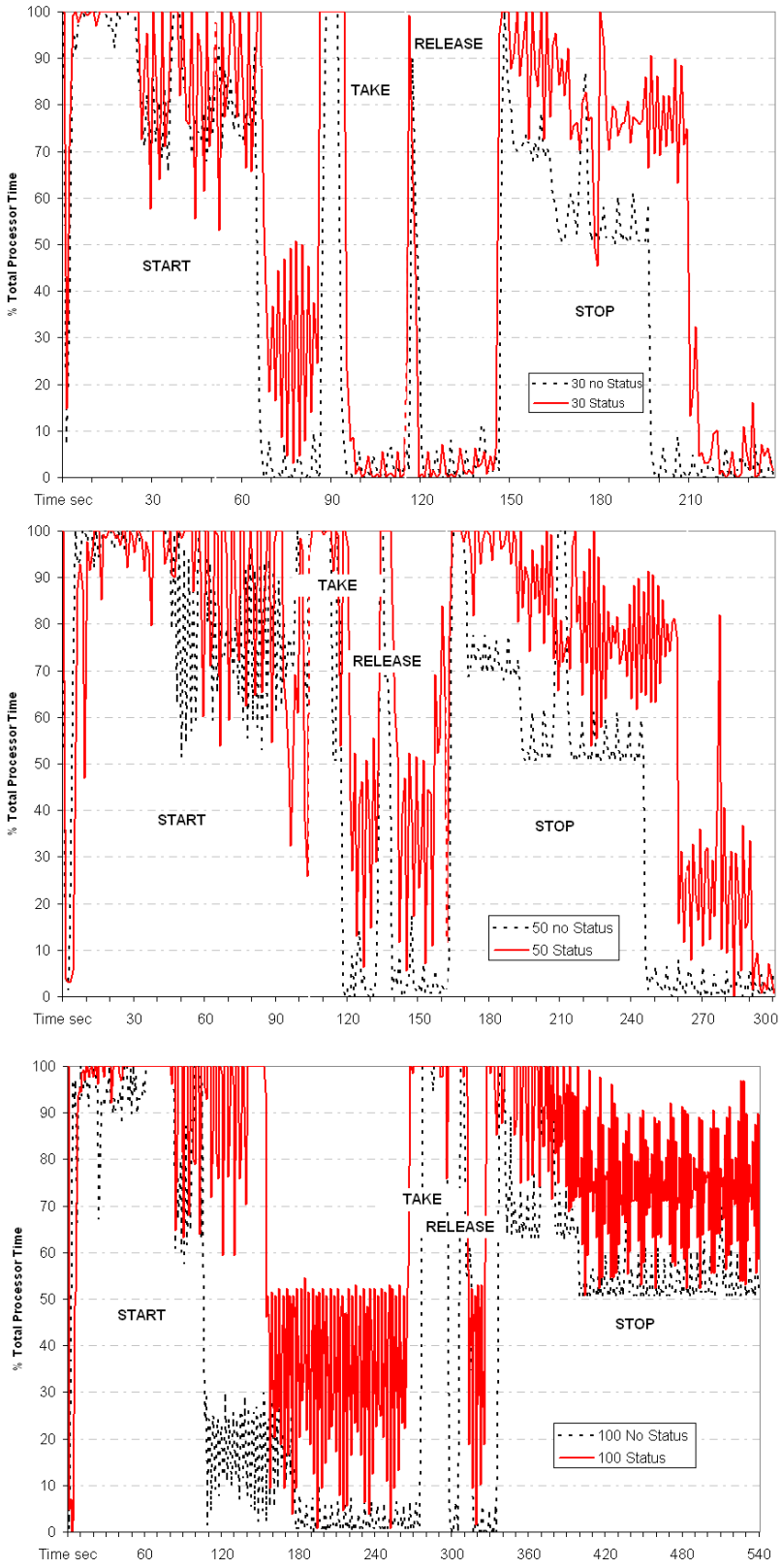


Figure 6.7: Dynamic response using a two levels hierarchy. The parent CU counts with 30, 50 and 100 children CUs. The solid lines represent the ATLAS approach and the dashed ones represent the 'classical' one. The start-up and the stop of all process are the most time consuming actions but also the less usual (normally only once). The dynamic response varies from a few seconds (3-5 sec) for 30 CU to almost half a minute for 100 CUs. Around 50 CUs per PC looks to be a limit in performance for both approaches.

with PVSS) makes the reaction time to new commands longer. Making a test with a very simple type of DU (only two states defining a boolean) shown us that the reaction time to set a command into different numbers of DUs ranged from 2s for 50 DUs up to 9s for 1000 DUs.

The numbers presented above are recommendations that can vary from the many different shapes and complexities the designer can attribute to the control hierarchy and, under certain circumstances, can be exceeded. Guidelines to the sub-detectors were given in order to understand these safes ranges where one can work safely (these are presented in Appendix A).

The second scope when defining common limits to be follow by the different sub-detectors is to reach at the end an isomorphic hierarchy. Since two control hierarchies defining the same system could have very different shapes and amount of objects the number of CU/LU layers between the partition and any device layer should be kept as low as possible, following the standard architecture presented at the beginning of this chapter, and never exceed five. On the lowest level, device units should be grouped such that one CU or LU controls devices of the same type. In addition, the number of devices per parent unit should be kept below $\sim$100.

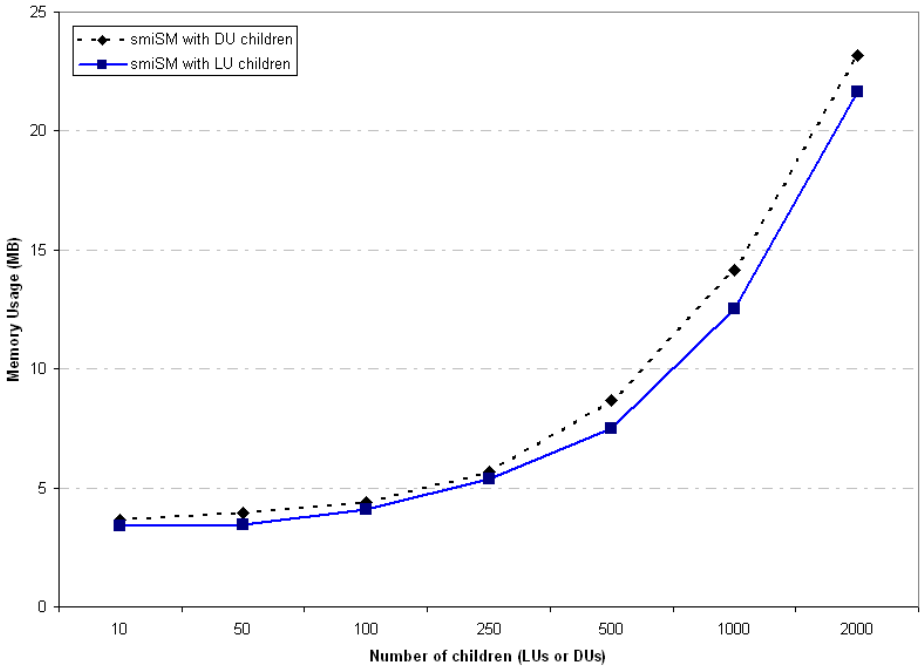


Figure 6.8: Memory requirements for LUs and DUs. Both are light objects (running within a CU or smiSM process) that present similar requirements in terms of memory. Up to 2000 of these objects can be included within a smiSM process increasing the memory used only by 25 MB. In contrast, the CPU load can vary widely. This is entirely in hands of the developer when writing the PVSS scripts that define the DU behavior. In order to reach a certain homogeneity between sub-detectors while keeping a proper performance 500 LUs and 1000 DUs are the advices limits for these objects.

6.5 Conclusions

The organization of the supervisory level of the ATLAS DCS has been presented. Due to the complexity and size of the detector, a hierarchical organization which follows the natural segmentation of the detector into smaller sub-systems, has been chosen. This hierarchy, as well as the behavior of its different components are modeled using a JCOP FSM toolkit. In order to scale the control tree to the dimensions of the ATLAS DCS it has been found appropriate to split the information in two different but cooperative trees. The new strategy focuses in a fault-detection mechanism associated to each node. The granularity of the hierarchy and the new approach used in ATLAS have been investigated with the aim to optimize the overall system performance. The results of these investigations led to the definition of design rules to be followed during the design and development of the sub-detector's control hierarchies.

Part III

Implementation Phase

Chapter 7

Integration Phase: Process Modularization and Data Reconciliation

The integration of the ATLAS control systems into a control tree involves several tasks including the process model identification and synchronization with external systems. This integration is done mainly by means of the JCOP FSM tool using the ATLAS guidelines.

The first part of this chapter shows two examples of sub-detectors process model identification integrated in a control tree. Both, follow the standard architecture discussed in the previous Chapter 6. Concerning the synchronization with external systems, the interaction between TDAQ and DCS has naturally been limited, but becomes increasingly important as the detector nears completion and is controlled through the DCS. The second part of this chapter focuses more in the data reconciliation of these two systems.

7.1 Building the Control Hierarchy.

The SMI++ platform stress the distribution, autonomy, communication, coordination, and organization of individual nodes within a control tree. Taking profit of all these features, the approach to be used for the modeling of the overall ATLAS control hierarchy was defined in the previous chapter. The aim of this section is to show practical examples of sub-detectors' integration.

Each sub-detector has very specific instrumentation and requirements for operation that have to be taking into account when designing the control tree. The best way found to design and implement hierarchical FSMs is from the lower levels upwards, after an initial top-down planning phase has anticipated the overall idea of the structure. In other words, we propose to use a combination of top-down and bottom-up approaches for the process modeling identification. The top-down approach deals with high level abstractions and conceptual tools, which facilitate capturing and modeling the structure and the behavior of the system being developed. This is what we have done in previously defining a standard architecture. Bottom-up modeling refers to developing scenarios that show in detail how the systems should interact with users and other external environments and this is what we show in this section.

Next, two examples of bottom-up process modeling identification are shown. The first refers to the HV system of the LAR sub-detector which was the first prototype of the final control hierarchy developed in spring 2005. The second example shows the organization of the Pixel and SCT sub-detectors. They both share common infrastructure that has been organized into the IDE. One

of these common system is the evaporative cooling which is a crucial driving system for the well functioning of the two ATLAS silicon detectors.

In order to represent formally the design of the different control hierarchies in ATLAS, the UML (Unified Modeling Language) notation has been chosen. Appendix B explains shortly this notation and shows the present compilation of the control trees of ATLAS. To find the best system decomposition close interaction with sub-detectors' experts was of prime importance. At this point, it must be underlined that the implementation of the two examples presented next and the description shown in Appendix B are the result of this collaboration.

7.1.1 Example 1: LAR HV sub-detector

The implementation of any sub-detector hierarchy goes across three functional layers of stations as shown in Figure 6.1. While creating the hierarchy, the designer has to decide its granularity. This granularity defines the boundaries of the hierarchy. The information located below these boundaries is encapsulated and not visible from the JCOP FSM. At this point, one has to find out the structure of encapsulation which will yield the best system decomposition. The goal is to find the modularization that minimizes interdependencies and most cleanly decomposes the system. The example discussed here refers to the High Voltage (HV) system of the Liquid Argon (LAr) Calorimeter which design is shown in Figure 7.1.

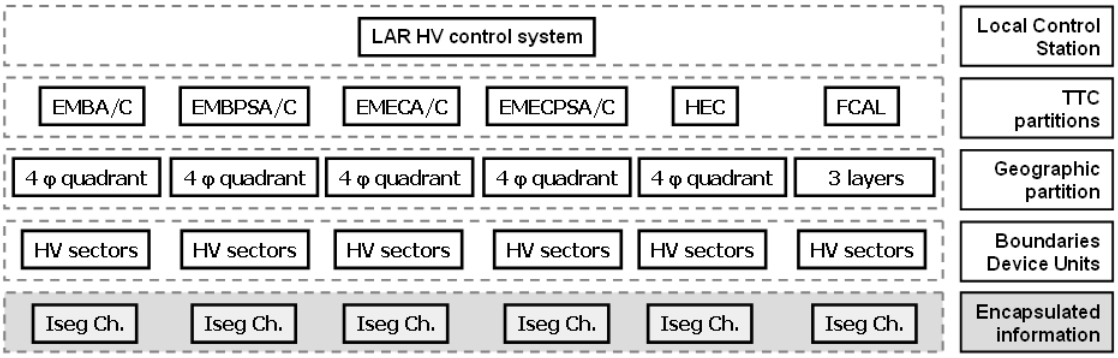


Figure 7.1: HV organization for the LAR calorimiter. The HV system is split on the different DAQ partitions. Moreover, a geographical division common to also other equipment of the LAR has been introduced. The granularity of the system is defined by the HV sectors which match with hardware modules and encapsulate a group of channels.

The full set-up for the HV system in LAr is made up of more than 5000 channels. The channel level is not modeled in the control tree since the selected granularity has been a HV sector, which is a physical part of the detector using a group of channels. However, detailed information about the channels will be accessible from a user interface for experts. The reasons to select this boundary has been:

1. Too fine a granularity increases the connections between the JCOP FSM and the SCADA system, which may overload the processors.

2. Too coarse a granularity would accumulate too much information in a single entity, making it difficult to define its functioning state.

3. The HV sectors are the smallest entities where commands have to be sent from levels above.

4. The behavior of a HV sector is well defined based on the state of a group of channels.

The next level up divides the detector geographically into quadrants. The idea here is to divide geographically LAr in a common way for all the systems (e.g. HV, LV) that form a certain region (e.g. a barrel) of the calorimeter. Hence, all LAr systems will also use quadrants to organize their FSMs.

During operation DCS will receive commands from DAQ, and for this reason a DAQ layer is introduced. The HV system is divided such that it maps to the partitions of DAQ (the interaction at this level is later discussed in Section 7.2). The final result is a control hierarchy formed by about 1900 nodes where 58 are CUs and the rest DUs.

7.1.2 Example 2: Silicon Tracker - Evaporative Cooling

Semiconductor detectors are very precise, compact in size and can have very fast response times. However, this kind of detectors have a very high power density across the active area which implies the need of a sophisticated cooling system to low temperatures before they can be operated. For this reason, an evaporative cooling system is being installed in the ATLAS pit (see Section 2.3). This system is formed by a unique cooling plant that is multiplexed into 204 independent cooling circuits which allow the different silicon structures to operate at $\sim$-6°C. Because SCT covers more area with silicon modules it counts with 116 cooling circuits while Pixel has 88. This partitioning of resources makes the two silicon detectors of ATLAS dependent of a unique, and crucial, cooling system while are still independent of each other during operation. Hence, due to the topology and functionality of the evaporative cooling system, it has been found a good exercise to illustrate the common decomposition of this part of the experiment. Figure 7.2 shows the organization of the IDE, Pixel and SCT focused on the evaporative cooling system.

The first control hierarchy modeled is related to the evaporative cooling which is part of the IDE SCS. The cooling most top node has a DU monitoring the overall conditions of the plant regulation (i.e. running, recovering, stand-by, off, etc). Then, the hierarchy is naturally divided into sub-detectors parts (i.e. Pixel, SCT barrel and end-caps) and finally in cooling circuits and heaters being both at the same bottom level. These define the granularity of the system, they match with physical parts of the detector and are the smallest entities where commands have to be sent. These commands are in principle two (temperature set-point and switching on/off a circuit) and are only accepted if they remain in the range of values compatible with the cooling system operation and safety. The heaters are only active when the detector is in operation or in stand-by mode monitoring temperatures, voltages and currents. In total, the evaporative cooling control tree is formed of about 420 nodes, being most of them DUs.

On the other hand, we found the control hierarchies modeled for the sub-detectors, Pixel and SCT. Being aware that the cooling can suppose a constraint for the rest of systems, it must take up a relevant position within the control tree of both sub-detectors. Thus, as a design decision, the cooling circuits have been located just below the DAQ partitions. The cooling circuits cover the whole geometry of both silicon detectors, and by means of cooling loops, geographical divisions are molded. At this level then, in order to get the conditions of the cooling system, both sub-detectors have references to the correspondent cooling loops objects (i.e. *Ide_Cooling_Loop*). The next levels down will group mainly the HV and LV systems located through a certain cooling section. This design approach had mainly two advantages:

- Both sub-detectors present an isomorphic control hierarchy (see Figure 7.2). This clarifies the design and facilitates to the future operator to get a clear view of the process in mind.
- Having into account that the silicon sub-detectors cannot be operated before reaching low temperatures, this organization avoid the automatic or manual starting of the power system before the cooling system is ready. At the same time, corrective automatic actions can be established easily.

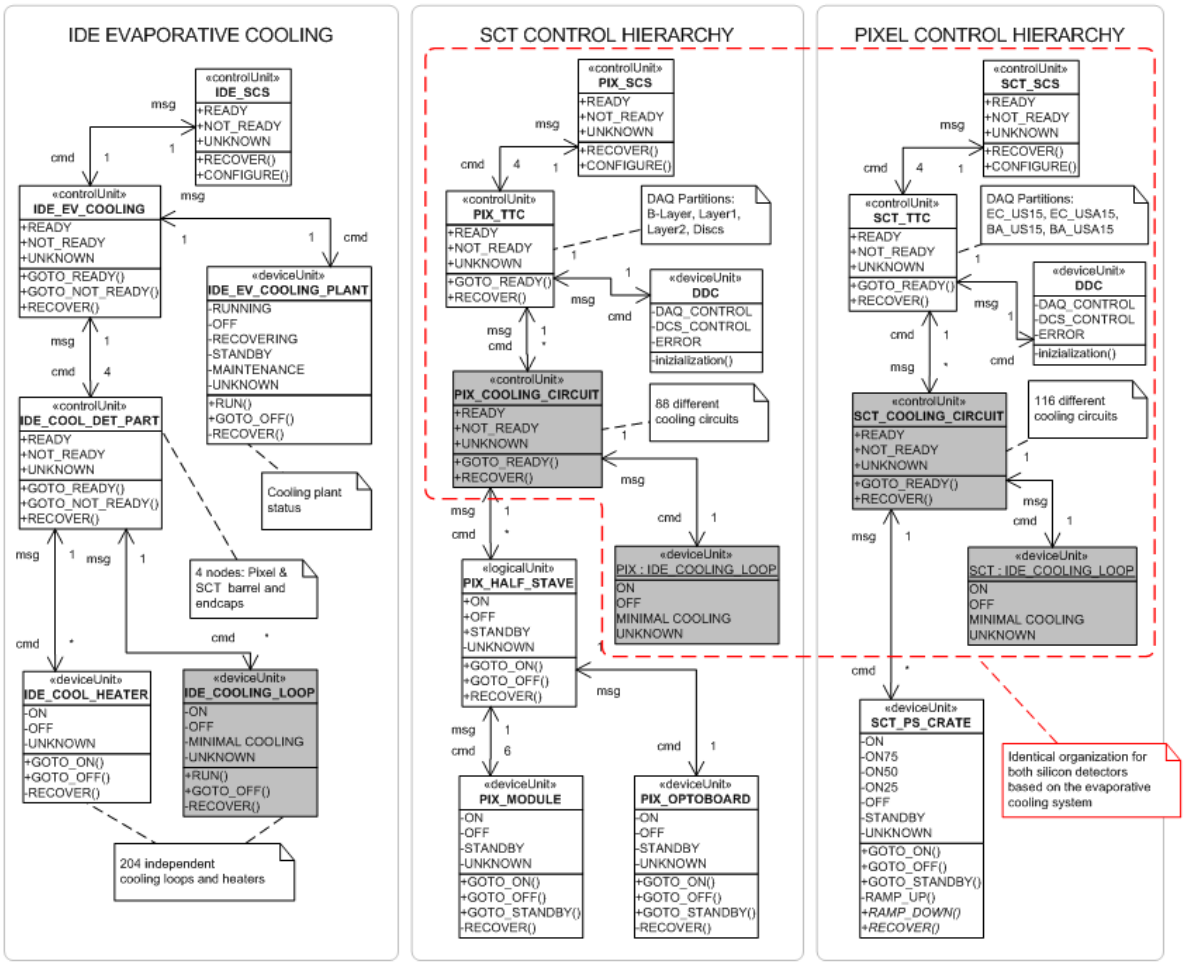


Figure 7.2: UML diagram of the IDE, Pixel and SCT control hierarchies. The figure focuses in the functional relationship existing between the evaporative cooling system handled by the IDE SCS and the PIX and SCT SCSs. In red: both silicon detector present an isomorphic hierarchy based on the cooling system.

7.2 Interaction DAQ-DCS

As already mentioned in Chapters 2 and 3, an intense interaction between the two major systems of ATLAS, TDAQ and DCS, is of prime importance for the coherent overall operation of the experiment. The data from the detector is read out through the Trigger and Data AcQusition (TDAQ) system which is a large and heterogeneous system and contains a wide variety of components to

be controlled. These items are typically clustered according to the detector topology. They range from readout modules connected with the detectors to nodes in the computer farms used for data selection. The Run Control (RC) for TDAQ is in charge of controlling the hardware and software elements involved in the data taking process. The RC is also built in a hierarchical and distributed manner. The interaction between TDAQ and DCS is handled by the DAQ-DCS Communication (DDC) software package [92].

This section first discusses and compares the TDAQ control with the DCS control, both using the same finite state machine concept. Secondly, the coordination between TDAQ and DCS is discussed with stress on the mapping of DDC package within the control tree. Finally, the interaction of both systems during an envisaged joint operation is described.

7.2.1 Comparison of the DAQ-DCS Finite State Machine Approaches

The two major systems for governing the ATLAS detector, TDAQ RC and DCS, are based on the same concept of FSM. This, as already discussed, allows for the sequencing and automation of operations. Even though both systems are based on the same concept, they have different requirements and use different technologies for implementing it:

- The TDAQ control is governed by *a single global FSM* for all components in its control tree. This guarantees that the same sequencing is followed by all its different components during the read out process.
- In contrast, the DCS is composed of many different systems with different behaviors that need to be automated, for example High Voltage (HV), Low Voltage (LV), cooling, etc. As a result, the DCS control is composed of *many FSMs* arranged in a hierarchy with rules of the parent-child interaction defined.
- Some parts of the detector should operate continuously since any interruption could be very costly in time and money or may even be detrimental to the performance of the detector. Hence supervision by the DCS is needed at all times. The TDAQ system in contrast is required only while physics data are being taken or during specific monitoring, calibration, or testing runs. Therefore the DCS system must be able to run completely independent of the TDAQ.
- An essential requirement of both, DCS and TDAQ systems, which is particularly important in the commissioning and installation phases, is the ability to partition the system into several independent, but fully-functional subsets. It must be possible for several detectors and/or several parts of a given detector to be triggered and to take data in parallel and independently of each other. Thus, to allow partitioning while keeping interaction between the two systems, at a certain level, both hierarchies are a mirror of each other (see Figure 7.3). This simplifies the communication between the systems as the TDAQ RC can always interact with a corresponding part of the DCS.

7.2.2 The TDAQ Run Control Hierarchy

The TDAQ RC system is responsible for initialization and supervision of the full TDAQ system. This encompasses starting/stopping of processes, the distribution of commands given by the operator and monitoring and handling of any errors and faults that may occur. The RC system implementation is based on the CLIPS programming language [93] with heavy use of C++ extensions.

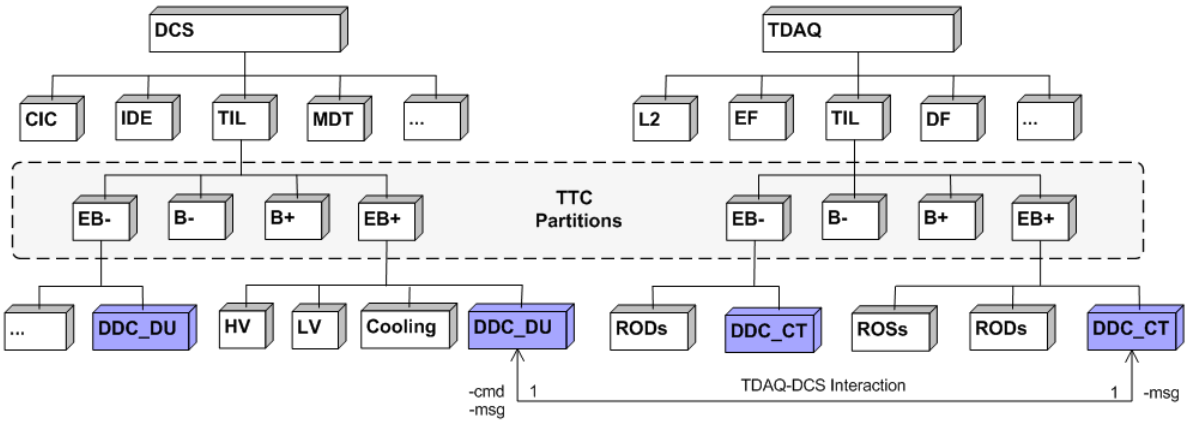


Figure 7.3: Because TDAQ is the master while physics data taking the DCS has introduced a layer into the control hierarchy which matches with the TDAQ organization. Using the DDC package, TDAQ and DCS are synchronized allowing to run different parts of the detector in parallel. DDC objects exist in both control hierarchies.

CLIPS was chosen due to its flexibility and previous positive experience with the language. More information on the evaluation and choice of technologies can be found in [94].

The TDAQ system is naturally divided into sub-systems (sub-detectors, event building farms, etc). The system configuration is therefore divided into segments typically representing a sub-detector, crates of readout modules or similar. A segment contains a set of applications, and possibly, other segments. The top-level segment is called a partition and may or may not represent the entire TDAQ system. The RC builds a tree of controllers where each controller corresponds to a segment.

The controllers are then in charge of all the applications contained by the segment (as specified in the configuration database) and possibly other sub-segments. Each controller is responsible for the initialization and the shutdown of software and hardware components in its domain. Similarly to the DCS control hierarchy, the TDAQ controllers are responsible for passing commands to child controllers (controllers under its control) and for signalling their overall state to their parent. During the operation phases, in case of a malfunction of a detector, for example, the controller can start corrective actions and/or signal the malfunctioning by issuing error messages. Severe malfunctions that are beyond the capabilities of a controller can be signalled by a state change to its parent (e.g. moving from a *Running* state to a *Paused* state). It is then the role of the parent controller to take further actions depending on the circumstances of malfunction. Figure 7.4 shows an example of a simple control tree.

To keep a status of all the applications in the control tree and to ensure a coherent execution of commands a FSM has been implemented for the RC system. The applications are subdivided into two groups:

1. *State aware applications*. These follow the state machine and receive and confirm commands from their controller. Examples: ReadOut System (ROS), Event building applications, etc.

2. *Stateless applications*. These are the applications that run independent of the state machine, but do still belong to a controller and report any errors to it. Example: monitoring applications, etc.

All controllers and state-aware applications follow the same global FSM. This is necessary to ensure that certain systems are started and stopped in a well-defined way and that dependencies across the system are taken into account. To allow for a more flexible structure, the possibility of hidden sub-states for any controller (and thus its sub-tree) has been implemented in the system. Figure 7.4 shows the state machine currently used for the experiment.

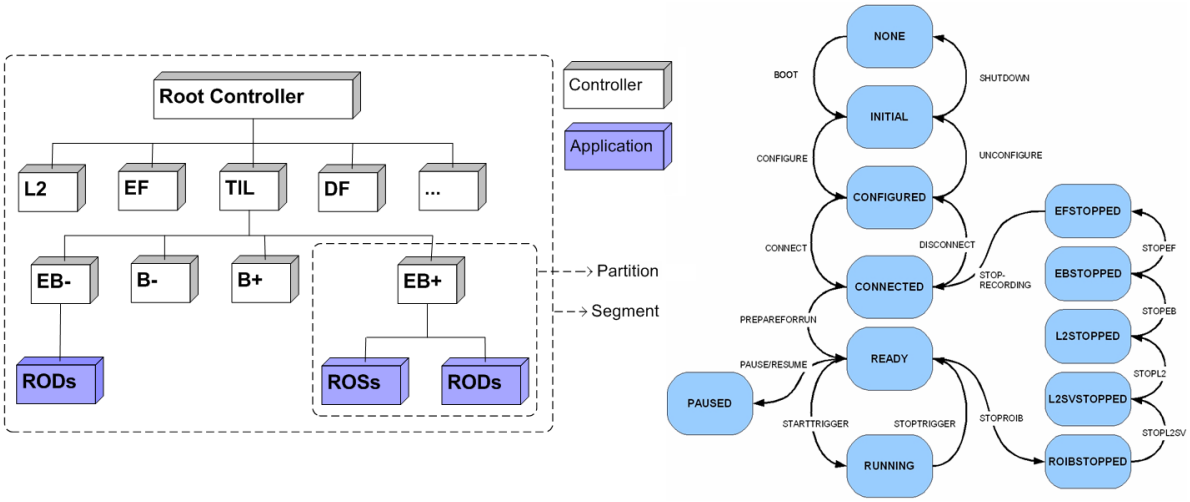


Figure 7.4: Left: TDAQ control tree. The RC builds a tree of controllers where each controller corresponds to a segment. The controllers are then in charge of all the applications contained by the segment (as specified in the configuration database) and possibly other sub-segments. Right: TDAQ FSM. All the components in the control tree are synchronized during read-out following the same global FSM.

Whenever a controller receives a transition command it sends it to all the applications and controllers in its segment. Any child controllers propagate the command and so on throughout the system. A controller confirms the transition only when all the children have confirmed the new state, so any errors while performing a state change are effectively propagated to the root controller.

7.2.3 Implementation of the DAQ-DCS Synchronization

The interaction between DAQ RC and DCS is performed at the supervisory level enabling the coherent operation of the experiment as a whole (i.e. there is no direct interaction between TDAQ and the DCS front-end). The control interface between DCS and TDAQ is arranged with the assumption that the latter is the master when the experiment is taking data. Thus, the TDAQ control applications are able to drive the DCS by sending commands and getting feedback about the result. Furthermore, the DCS informs asynchronously TDAQ about any failure on the detector preventing incorrect data taking.

The interaction is done by means of the command transfer application DDC-CT of the DAQ-DCS Communication (DDC) software package [92], after called the DDC controller. Like any other leaf controller, the DDC controller is responsible for receiving commands from the parent TDAQ controller and transferring these commands to the domain under its control, which is a DCS subsystem. It also receives error signals from the corresponding DCS and passes them to the TDAQ. The mapping between the TDAQ transition commands and the FSM commands on DCS is defined by the DDC controller configuration.

The number of DDC controllers running for a sub-detector corresponds one-to-one to the number of Trigger and Timing Control (TTC) partitions [95] of that sub-detector. A TTC partition is the minimal part of a detector that can be controlled autonomously for data taking. By running one DDC controller for each partition we achieve the finest common possible granularity to both systems. On the DCS control hierarchy each DDC controller corresponds to a DDC Device Unit (DDC_DU) (see Figure 7.3) [4]. The DDC_DU uses the JCOP FSM functionality to execute the commands sent by TDAQ and inform about a change of state. The main duties of this Device Unit are:

1. **Reporting the DCS state**: Asynchronously it reports to TDAQ any occurrence of conditions preventing the data taking. The granularity is the TTC partition. Thus, the DDC Device Unit checks the state of its TTC domain and sets a flag (i.e. 'notDataTaking') that reports the state of the detector for a certain TTC partition. The standard FSM states for a TTC partition are:

   (a) *Ready* : TTC partition ready for data taking

   (b) *Not_Ready*: TTC partition not ready for data taking. The designer must be aware, when propagating the states upwards to the TTC level, that *Not_Ready* means the whole TTC partition disabled for physics data taking

2. **TDAQ Command Execution**: TDAQ can operate the DCS executing transition commands by means of the JCOP FSM. The set of FSM commands to be issued by TDAQ are meant to be general (i.e. *Prepare_for_Run*). When a command is triggered (i.e. 'trigger' flag) from the TDAQ, the DDC_DU reads a set of parameters and it sends the selected command to any specific node. The parameters (i.e. 'fsmParameters') are filled by TDAQ with the following three-field format:

   (a) '*FSM domain*' specifies the node to be commanded.

   (b) '*FSM command*' specifies the command.

   (c) '*Timeout*' specifies the command execution timeout. The response given by DCS to TDAQ (i.e. 'response') depends on the timeout and the transition caused by the command execution. If the timeout value is 0, the response is set to a good state automatically. In case of an existing timeout, the response is set within the timeout interval to either a good or bad state.

In order to illustrate this synchronization with an example, Figure 7.5 shows a basic interaction sequence. The first sequence shows a failure (e.g. severe HV trip) that could damage the quality of the data taking. Then, DCS at the TTC level in its hierarchy informs TDAQ through the DDC mechanism. The second sequence represent the Tilecal TDAQ operator sending the command 'prepare for run'. It implies a set of actions to be performed at the DCS side by means of the JCOP FSM (i.e. *Goto_Ready* means ramp up HV, switch on LV, etc). The assignment of DAQ commands to the relevant DCS commands is done through the DDC controllers; the DDC_DU applies these commands within the DCS hierarchy of FSMs.

From the TDAQ system point of view, a leaf DDC controller accepts all transition commands and follows the global TDAQ FSM. If a DDC controller enters an error-state (for example due to a problem within the DCS) this state is propagated up the TDAQ control tree.

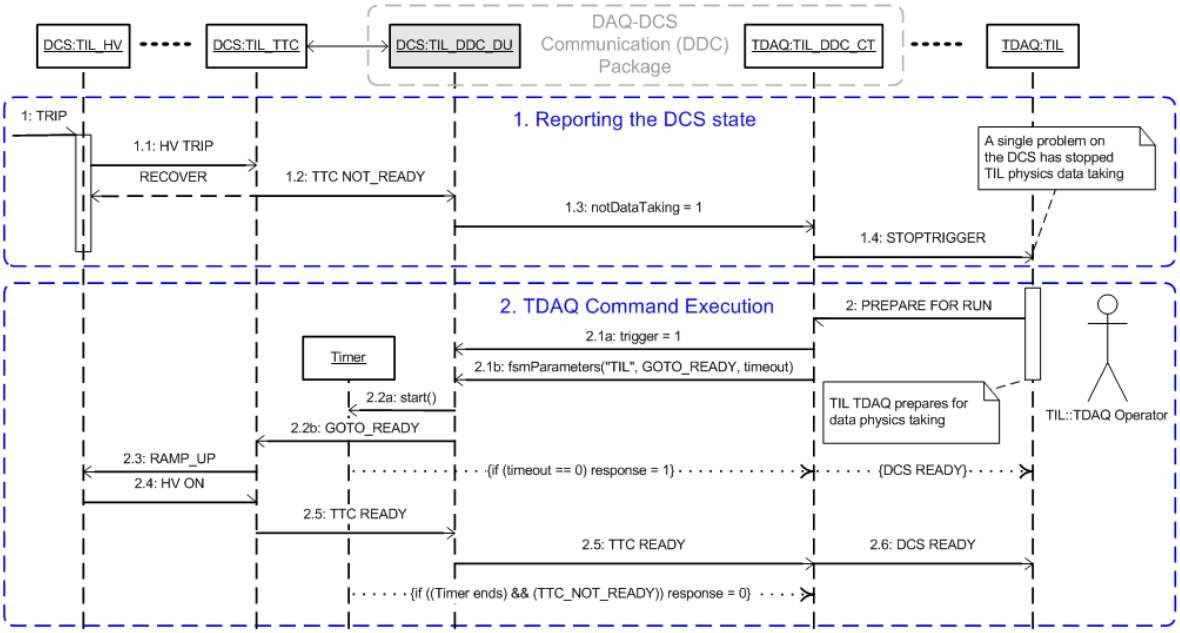


Figure 7.5: UML Sequence Diagram of the interaction TDAQ-DCS though DDC. Sequence 1 reports a Tilecal DCS failure to the TDAQ preventing incorrect physics data taking. Sequence 2 shows the Tilecal TDAQ operator mastering the whole TDAQ-DCS. The command 'prepare for run' is sent to the DCS who give back the response about the success of the command execution.

7.2.4 Operational Data-Taking Phases

The TDAQ states shown in Figure 7.4 are traversed when the TDAQ system is run through initialization, preparation, data-taking, and shutdown phases. In this scenario it is assumed that the LHC accelerator is running with stable beam. During the operational phases, systems and subsystems perform a number of operations independent of each other. During the state transitions, actions specific to the sub-system like initializing software and hardware elements, loading software and parameters to hardware modules and configuring them, or starting processing tasks, are performed. The operational phases described in [32] are used to illustrate an envisaged joint operation of both TDAQ and DCS:

1. **Initialization**. TDAQ starts from booted but idle CPUs. The operator selects a partition that is described in the configuration database or defines a new one. The process management and information distribution infrastructure (i.e. process managers, Information Services, Error Message system) is started, verified and initialized. Once the TDAQ infrastructure is in place, the controllers and the application processes are started. Having successfully finished this transition, the TDAQ system is in the Initial state.

   The DDC communication software is started and the communication between the systems is initialized. The data describing the selected partition are transmitted to DCS establishing a first control communication with DCS.

2. **Preparation**. This involves synchronizing the configuring of all software applications and hardware elements that participate in the data-taking process, for example, the establishment of communications channels between TDAQ elements, or the setting of hardware or software parameters.

The preparation of the sub-detector equipment for data taking involves the issuing of commands to DCS and the subsequent execution of defined control and initialization procedures. These commands can be associated to state transitions of the TDAQ controllers, or be asynchronous commands issued directly by the TDAQ operator or by stand-alone applications. The actions are detector-specific and are defined in the configuration database.

Commands are sent from TDAQ to DCS specifying the DCS target node, command and timeout. The DCS, before performing the action, validates and cross-check with the states of the external systems and of the common infrastructure (all of them being part of the control hierarchy) to guarantee the integrity of the equipment. For example, the evaporative cooling of the IDE is checked before turning on the Pixel or SCT HV systems. The readiness of the sub-detectors for data taking is reported to the run control by the DCS, which will then be in the *Ready* state for all the TTC partitions selected by the TDAQ operator.

The readiness of the TDAQ system to take data is defined as the moment when the top-level run controller is in the Ready state. This implies that all the data-acquisition and trigger systems have been initialized, connected and configured correctly and they are ready to receive event data.

3. **Data taking**. When the run is started by the operator, the LVL1 trigger is enabled and event data-taking operations commence. If necessary, a run can be paused and resumed in an orderly manner with a minimum time overhead simply by inhibiting and enabling the LVL1 trigger. Partitions comprising one or more TTC zones can be removed from the main data-taking partition, for example in case of problems with a given detector element. The removed detector element can then be configured for stand-alone operations to allow the faulty element to be identified, tested, and repaired. Once the element is functional again, it can be re-attached to the main partition.

4. **Stopping** When the operator stops the run, the LVL1 trigger is inhibited and data taking is stopped. The control and application processes involved remain active. No changes on the DCS side are foreseen, the sub-detectors remain in the *Ready* state.

5. **Shutdown** On receipt of the shutdown command all applications and their controllers are stopped. Finally the TDAQ software infrastructure is closed down in an orderly manner, leaving the system available for a new data-taking session. If no further data taking is foreseen the *Goto not ready* command is given to DCS in order to bring the detector to a safe state by ramping down and switching off the low and high voltage applied to the sub-detectors.

7.3 Fusion of Other External Systems

As already mentioned in Chapter 2 and 3 external elements to the DCS such as the electricity and the magnet system, and specially the LHC conditions, are interfaced through the DIP mechanism(see sub-section 3.5.4). DIP allows to recollect heterogeneous data from different systems making it available in PVSS-II and consequently in the DCS control hierarchy.

During the period thesis the author was also involved in the early design stage to interface the conditions database with the control hierarchy[96]. Further developments and its real implementation were carried out internally in the JCOP group. The FSM transitions will require the re-configuration of parts of the DCS. All configuration parameters of the devices integrating the

control systems are stored in an external database called Configuration DB. The main task of the FSM-Configuration DB interface is to ensure the availability of the configuration data in PVSS, used during the FSM transitions, for a given type of run mode, e.g. *Physics*, *Cosmics*, *Calibration*, etc. This is achieved by synchronizing the contents of the Configuration DB and of the PVSS internal caches used by the FSM at the beginning of the run.

7.4 Conclusions

When looking for a functional or geographical view on the ATLAS DCS facilities, one notices the strong relationships existing between the elements. Sub-detectors are structured in different geographical divisions (e.g. end-caps, barrel, etc), which themselves contain a set of sub-systems (e.g. HV, LV, Cooling, etc). These sub-systems themselves are also built from mechanical assemblies and these are built from devices such as a crate, a valve or an ELMB. The ATLAS guidelines for the JCOP FSM facilitate the standard mapping of these mechanical structures into the control hierarchy structure. As a consequence, uniformity is reached among devices, sub-systems, systems and sub-detectors. On the other hand, all the logic for the automation and sequencing of the experiment will reside into CUs, LUs and DUs. The DUs, which are in between the JCOP FSM and PVSS-II, contain the scripts that supervise and command a certain device. This uniformity among sub-detectors and the grouping of logic and PVSS scripts into the JCOP FSM will help the engineer to understand the control program and will facilitate the integration and future maintenance of the DCS back-end.

This chapter has also shown the organization, implementation and coordination of two of the main control systems of the ATLAS detector, namely TDAQ Run Control and DCS. We have looked closely at the use and implementation of the FSM concept in both systems and pointed out similarities and differences between the two. The DDC package is mapped in both control hierarchies at the TTC level where the synchronization is established.

Chapter 8

The DCS Top Level Human-Machine Interface

The top level human-interface will be able to access all the DCS information in the final set up. This information, coming from about 160 station, will be available in both: the ATLAS control rooms and, to some extend, externally via the internet. A systematic approach has been developed with the aim of facilitating the integration of the many displays required for the operation and maintenance of the experiment. This approach utilize heavily the abstraction hierarchy provided by the JCOP FSM in order to organize and display the DCS information.

This chapter first shows the envisaged operational schema of the ATLAS control room, the different consoles for operators and tasks are listed. Then the chapter follows by describing the strategy used to make available the information contained within control hierarchy at the different consoles. Later, the tools made for the human operation and supervision are described briefly with emphasize on the DCS operator interface. The main interface requirements and the design phase are discussed.

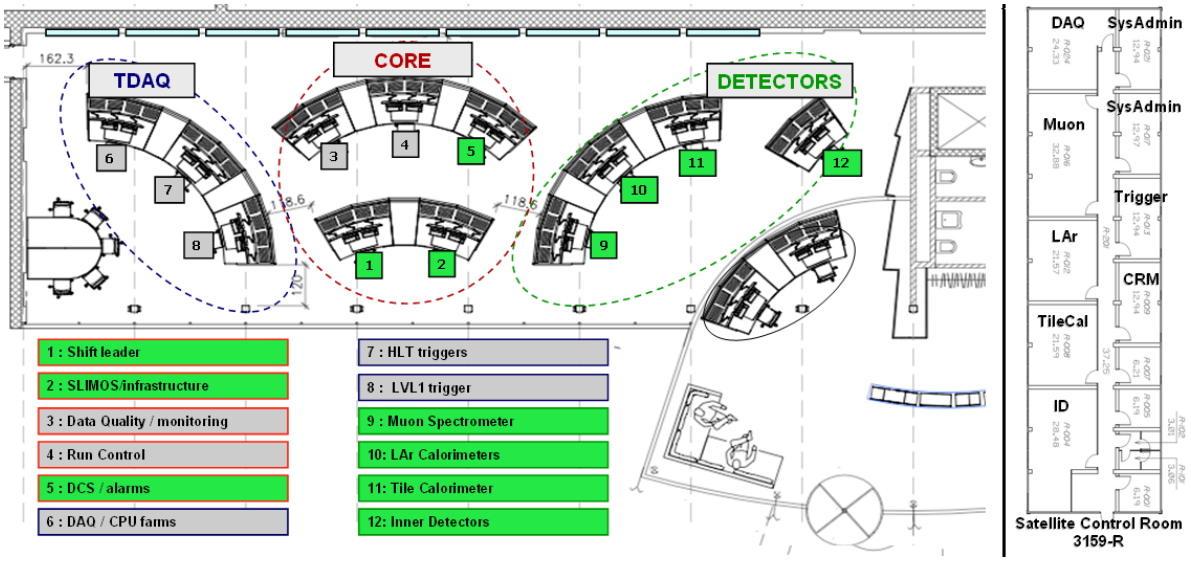


Figure 8.1: Left: ATLAS Control Room (ACR). It is marked in green all those stations able to access the DCS human-computer interfaces discussed later in this chapter. Right: Satellite Control Room (SCR) located next the the ACR.

8.1 Operational Schema of the ATLAS Control Room

By discussing briefly the draft of the ACR operational schema [97], this section intends to put in context the DCS top level human-machine interface.

The ATLAS Control Room (ACR) in the surface will be managed centrally by a run coordinator with a team of shifters. In addition, detector experts will sit in Satellite Control Rooms (SCR), nearby the ACR mainly on-call basis. In total, 38 shifters are foreseen at any time and 114 shifters/day will be necessary to operate the experiment. Figure 8.1 shows the distribution of these shifters in the ACR which are next discussed briefly.

The core area is formed by five stations/shifters (with 3-4 screens in each) that have different roles for the operations. The shift leader (1), who is responsible for the interaction with the LHC and access to cavern (UX15), is informed by the rest of shifters and he/she is able to monitor any function of the ACR. The Shift Leader In Matter Of Safety (SLIMOS) (2) is responsible for access control, safety system, cryogenics, magnet and common infrastructure. The data quality shifter (3) is responsible for the physics event display and event data monitoring. The run control shifter (4) drives the data acquisition system (run configuration, DBs monitoring, luminosity monitoring and main e-logbook). The main DCS shifter (5) is responsible of the overall monitoring of the experiment conditions and alarms and has all privileges to run the equipment.

The TDAQ area in the control room counts with three stations, DAQ & SysAdmin (6), high-level triggers (7) and LVL1 trigger (8). They carry out TDAQ specific operations on for instance the read-out system, the data flow networks and the triggers.

Finally, there is an area with four detector shifters (9-12) which monitor the DCS, sub-detector infrastructure and detector's performance with links to SCRs.

A large number of displays and interfaces will be available in the control room. Generally, on the one hand, the TDAQ interface gives access to the RC, but also interfaces with a variety of other systems necessary for the extraction of physics events. The DCS, on the other hand, gives an in-depth view of the detector hardware and provides tools and functionality to monitor and operate the detector and also locate, identify and fix errors that may occur.

8.2 Requirements of the Top Level Human-Machine Interface

By the experience acquired in the building SR1 (see Chapter 5), it was concluded that a new strategy was needed to interface the large amount of data inherent to the DCS into the ACR. Hence, before starting the design and implementation of the DCS top level interface, the requirements that may accomplish the different the tools have been studied. Below we intend to classify these requirements by studying the operation from different points of view:

Operational Scenarios

The lifetime of the ATLAS project could be divided into the following operational scenarios: (1) design, the simulation phase; (2) device-by-device operation; (3) start-up (or, shut down) operation; (4) adjustment operation (including calibration, commissioning, machine studies, etc.); (5) physics data-taking operation; (6) diagnosis of ATLAS problems; and (7) operation for maintenance. These scenarios need of different tools for operation and are normally repeated cyclically. For example, in a typical year of LEP operation the accelerator time was divided into about 70% on routine operation, 20% on machine development and 10% on fixing problems [98]. Similar behavior is

expected from ATLAS and, consequently, the different tools to be developed for the operation of the DCS should be able to adapt to all these different operational scenarios.

Who are the users

The second issue to tackle is who should use/define the human interface. The experience has shown that any HEP control system is used at different times by different personnel[99]. During the construction and commissioning phases, and later on for detailed investigations of faults, many equipment specialists need to run extensive tests of their hardware. During commissioning, and later on in machine development, physicists need a wide variety of utilities to measure and analyze the physics data. Once the machine is up and running, operations need a comprehensive suite of applications software for the long-term exploitation of the experiment. These different groups of people each have very different and often conflicting requirements. Hence, when designing the DCS human-machine interface, we will need to satisfy all kinds of users, giving more specific data to the detector experts and intuitive-general displays to the operators in shift.

Functionality

The third issue to clarify concerns all those functions that the DCS human interfaces should provide during operation (to different kind of users at different operational scenarios). We sort these functions as follows: (1) in-place and remote operation; (2) stand-alone operation of each system; (3) cooperative operation among systems, automation; (4) fault-detection operation; (5) operation support based on log trends for correlation analysis; (6) quick human operation into a selected system; (7) access control operation; and (8) multimedia support operation.

Classification of events in the future ACR

One way to classify the future events that a human interface has to handle is according to their degree of novelty from the perspective of first operators and then designers[98]. Three areas have been identified:

- *Familiar events* are routine in that operators experience frequently. As a result of the experience and training, operators will acquired the skills required to deal with these events.

- *Unfamiliar, but anticipated events* occur infrequently and thus operators will not have a great deal of experience to rely on. However, the events have been anticipated by detector designers, who have built software or hardware solutions to deal with them (e.g. FSM automatic actions and procedures, or hardware interlocks). These anticipated solutions provide operators with the help they need to cope with this class of events.

- *Unfamiliar and unanticipated* are also unfamiliar to operators because they rarely occur. Unlike the previous category, however, the event has not been anticipated by detector designers. Thus, operators cannot rely on a built in solution but must improvise one themselves. In order to tackle these kind of events, the operator interface should provide the necessary functionality to first identify constraints between systems, and second, to access quickly to the malfunctioning system.

Navigability

In the past, when the control systems counted with less channels to be supervised and controlled, it was possible to display the different systems serially. However, already with the experience gained in the SR1 it was found difficult to operate the system in this fashion; research in 'Ecological Interfaces'[100] has confirmed not surprisingly that forcing subjects to search for information across windows significantly impairs the performance. Thus, one important requirement for the design of the final DCS human interface is to enable the operation of the system from a single window making all information available in parallel. It is not desired to search for information across series of windows.

Lifetime and evolution of ATLAS

Experience in previous HEP experiments has confirmed that during the lifetime of the experiment, the control system is improved. In particular, this is true for the operators console that will evolve in order to use the experiment with greater efficiency and convenience (instead of simply maintaining the set of displays designed at the beginning of the operation) [101], [100], [102], [103], [104]. This fact is illustrated in Figure 8.2 that shows the typical evolution of any large software project or HEP experiments. A clear example of the envisaged operator console evolution can be found in the 'comprehensive beam measurement and fixed display system', introduced at LEP in 1994 and now considered essential for routine operation. Thus, the top level interfaces should not restrict the evolution of the experiment, new displays, or upgrades of existing ones, must be integrated easily not supposing the interruption of the rest of operations.

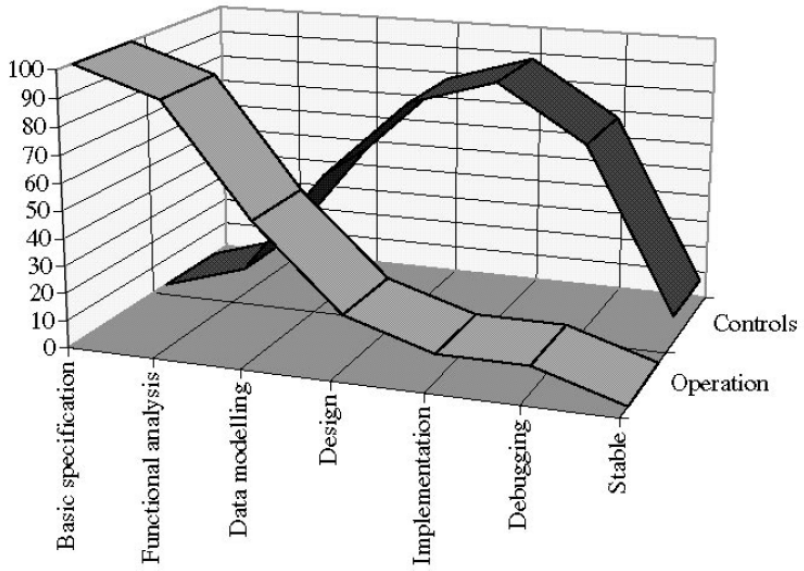


Figure 8.2: Normal evolution of the operation and controls the through the various phases of a large applications software project. Picture taken from [98]

Summary of Operators Needs

Over the lifetime of the experiment, operators will be by far the main users of the system. This implies one question to be answered, what features do operators want?[98]. Leaving aside the

detailed functionality required to run the machine, two things set operators in shift apart from other users of the control system:

- Firstly, the scope of their work necessarily covers the whole machine rather than specialist areas, and in this they prefer a high level of standardisation across different applications or systems to control. This would enable all the essential functions (e.g. start, stop, push button) and some slightly deeper functionality (display handling, commands input) always to be performed in the same way. With this, *navigation* around the system and a large part of the operator's interaction with it should be standard.
- Secondly, machine operation happens 24 hours a day, seven days a week, typically for 30 to 40 weeks each year and the 'comfort' provided by the control system is extremely important. A few characteristics of user-friendly interactive application are: easy-to-use, reliability, stability against change, error reporting, and execution speed. During stable operation not-interactive displays with the general conditions of the experiment contributes greatly to the operator's comfort.

Hence, the DCS top level interface has to be developed with the expectation to cope all listed requirements. The DCS operator interface should be modular and flexible enough in order to be accommodated to new operational scenarios, fulfill necessities of all kind of users, and permit the future evolution of the control system. Next section presents the strategy followed for the design of the ATLAS top level interface.

8.3 Strategy to Interface the Control Hierarchy into the ACR

As shown in Figure 8.1 the DCS can be supervised from different stations within the control room. These stations will be able to access in total about a thousand of displays showing the conditions of the large experiment set-up with different levels of detail. Hence, a common strategy has been again designed with a twofold target: first, facilitate the integration of the many sub-detector displays into a common facility; and second, to provide the future users with an homogeneous environment to work.

This strategy, shown in Figure 8.3, uses the JCOP FSM to extract both summarized and detailed information of the detector conditions. Each node in the control hierarchy contains useful data to be displayed from different abstraction levels, for example, the whole ATLAS, a sub-detector, a HV system, an ELMB, etc. Thus, taking advantage of the modular and hierarchical design provided by the DCS back-end, each node will constitute a workspace that has one display associated (i.e. a PVSS-II panel). Furthermore, in order to facilitate the interface of the control tree functionality into the displays, special graphical widgets have been developed in the context of the fwFsmAtlas. These graphical widgets make use of standard JCOP FSM and fwFsmAtlas functions, which involves a low development cost and easy maintenance. At the same time, these widgets can be re-used in different DCS facilities like the operator interface, fixed DCS status screen or DCS web pages that are discussed later.

The main advantage of this approach is the intrinsic modularity of the DCS top level interface which has been extrapolated from the control hierarchy. It is easy to integrate new displays and upgrade existing ones. Elements like widgets are re-used, and uniformity among sub-detectors displays can be achieved. Moreover, a 3D representation of the experiment can be achieved by extracting the geometry of the physical parts of the control hierarchy that has been defined in a external database.

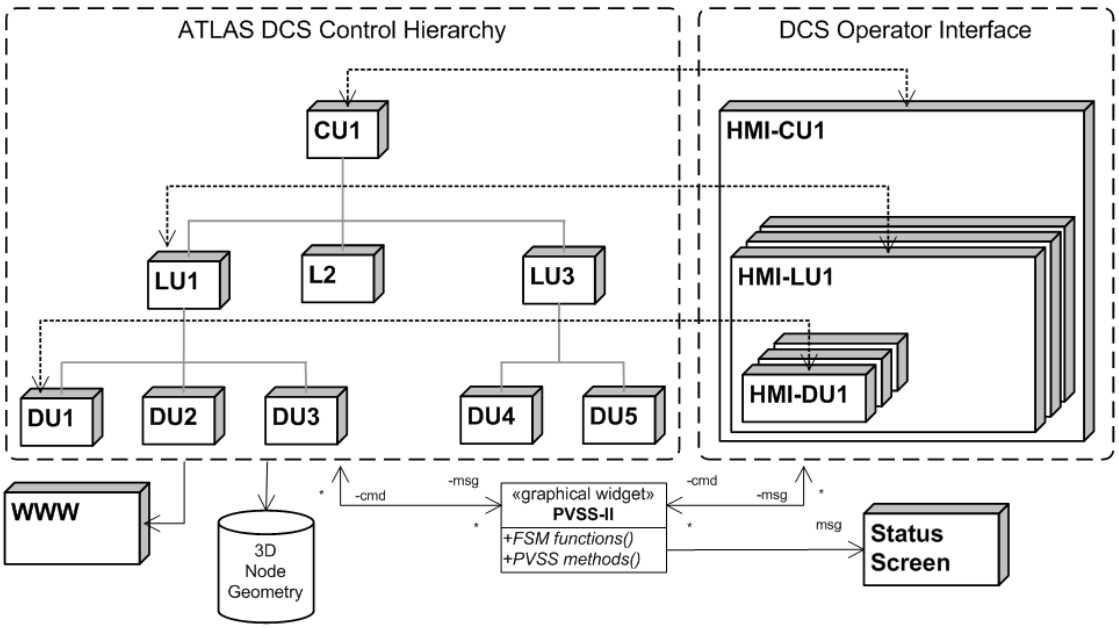


Figure 8.3: Interface of the DCS control tree into the top level interface. Each node in the control hierarchy (i.e. the whole ATLAS, a sub-detector, a HV system, an ELMB, etc) will have a display (PVSS-II panel) associated. Special graphical widgets, with all the JCOP FSM functionality (i.e. display state and status, send commands and change of partitioning mode), have been developed. These widgets make use of standard functions, which involves low development cost and easy maintenance. The geometries of all those physical detector parts mapped on the hierarchy are available in a database which will permit a 3D representation of the experiment. Finally, the data of the control tree is also used to build web pages.

8.4 Constituent parts of the Top Level Interface

The constituent parts of the ATLAS DCS Top Level Interface are discussed below. This set of human-machine interfaces has been designed to accomplish the monitoring, control and analysis of the operation of the ATLAS detector providing both, overall and in-depth views of the experiment conditions. From the point of view of interaction with the operator we divided them into active and passive interfaces [5].

8.4.1 Passive Interfaces

Passive interfaces collect DCS data for monitoring and analysis without means of any interaction, in other words, all kinds of commands are disabled. There are three types of these kind of interfaces that have different roles:

- **Data Viewer**: Basically, this provides a graphical representation of system parameters. It is a classical tool for operations in any industrial plant. In the SPS and LEP a similar tool, also called dataviewer, was used [105]. This common data viewing tool is provided by the JCOP framework and uses PVSS-II. The tool serves to analyze the system behavior during a specified period of time, allowing to correlate concurrent data. Figure 8.4 illustrate an example of its usage in the LAR cooldown tests. The main advantages are: it is a standard tool and thus familiar to the operators; it is powerful with all the required functionality (e.g. save, print, zoom, etc.) and it involves a low development cost.

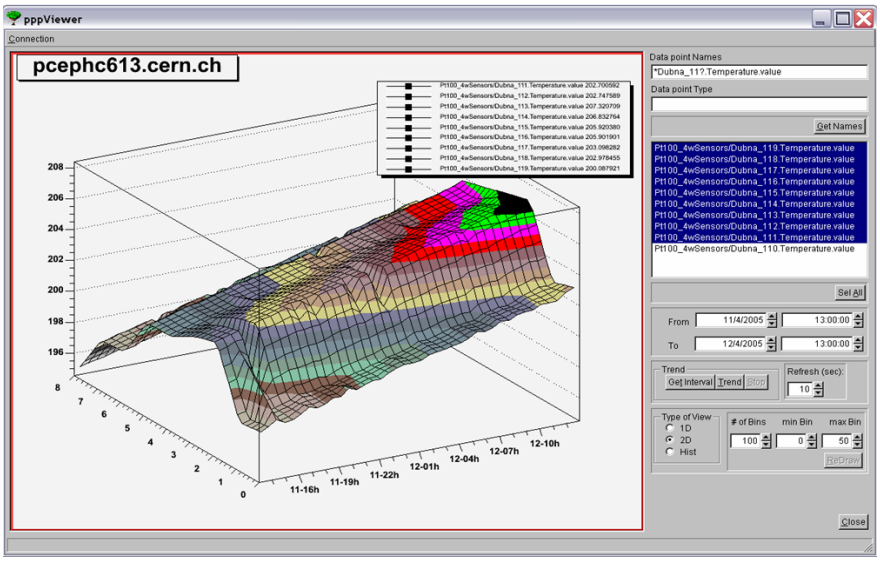


Figure 8.4: Real time data plotted in ROOT from the LAR cooldown test (April 05). Today, PVSS-II functionality on plotting permits its application for the final data viewer screen. Picture provided by V. Filimonov.

- **Status Screen**: Operations is not all interactive, during stable conditions it is foreseen that ATLAS will run with minimum operator intervention. What the operator needs at these times is a good fixed display system, giving him at-a-glance access to a rather large amount of data which tell him either that all is running well or that something has gone wrong. Display

of information of this kind on a permanent basis, with a regular update, contributes greatly to the operator's comfort. Learning from past experiences, for stable LEP operation there were about 15 fixed display screens regarding the essential for operation of the machine. The aim of the *Status Screen* is therefore to give a general view of the detector conditions at the ACR. Basically, as shown in Figure 8.5, it would be a matrix with two axis, the TDAQ partitions and the DCS systems belonging to them. In that way, the conditions of each single detector system are monitored. The matrix will be build re-using graphical widgets created in the context of the DCSOI (see later Section 8.6). Thus, once again, it is a standard screen build with familiar elements to the operators, it involves a low development cost and easy maintenance.

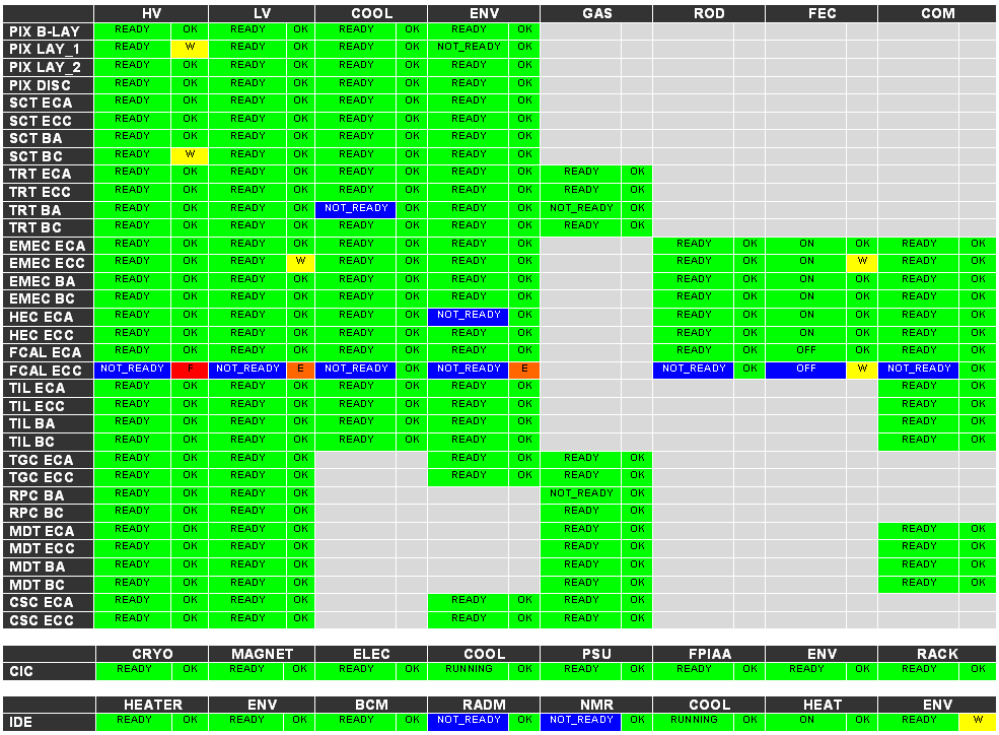


Figure 8.5: Status Screen draft. The displayed data is retrieved from the DCS control tree. The states and status are displayed by means of standard widgets.

- **WWW**: ATLAS is built and operated by a consortium of institutes, and laboratories. Each collaborator is responsible for a complete section of the machine including all subsystems. This responsibility includes design, construction, testing, commissioning, participation in operations, maintenance and repair of faulty components [106]. Hence, regardless where the control center is located, the operation of the experiment will depend on a continuous flow of information to all of the collaborators. Thus, simply because of the distances involved, remote troubleshooting capability is essential. HTML pages incorporating DCS data from PVSS-II for the Internet are already available (see Figure 8.6) [107]. Furthermore, remote PCs connections to any PC in the ATLAS DCS set-up is possible though a Windows Terminal Server. The conclusion is that any facility could also easily be commissioned or tested remotely.

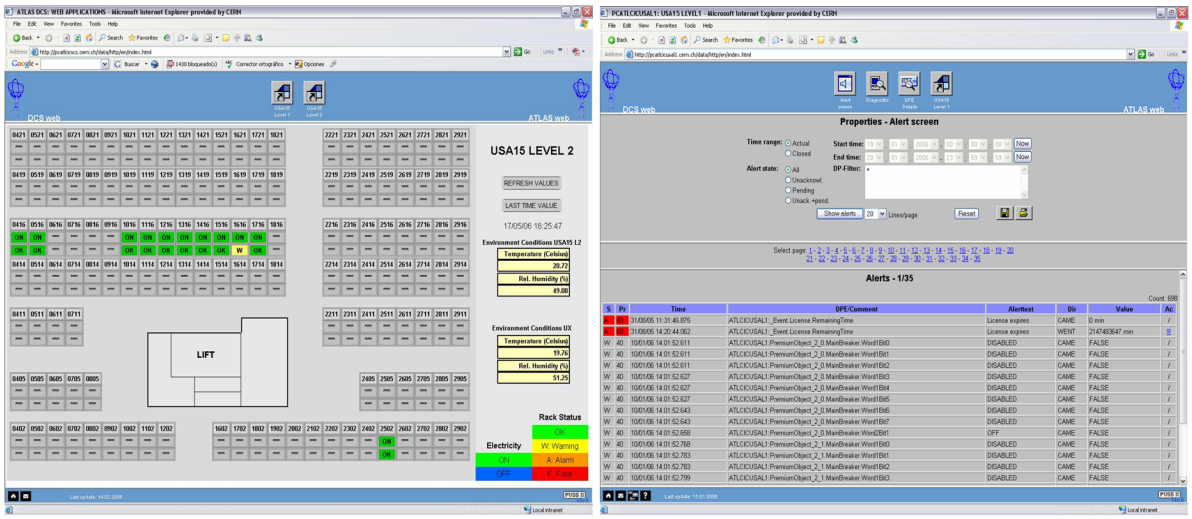


Figure 8.6: First running web pages implemented for the DCS in January 06. Right: A PVSS HHTP server was set in the CIC SCS to interface the control tree data (giving the electronics racks' conditions) and the underground environmental conditions. Right: In addition, the DCS alarm screen was available on the web.

8.4.2 Active Interfaces

Active interfaces are all those which permits the operator to interact directly with the displays in order drive the experiment. For the ATLAS DCS there are two active interfaces, the alarm screen and the DCS Operator Interface (DCSOI). The former allows to the shifter to handle alarms and the later is the main tool that the DCS shifters will use to act on the experiment.

- **Alarm Screen:** It provides specific information about the occurrence of errors in the detector. The main alarm screen will allow filtering depending on the time and severity of the alarms and it is provided as a JCOP framework component based in PVSS-II (see Figure 8.7). The alarm screen is a particular example of a fixed display of great importance. It is foreseen that the DCS shifter will have two alarms screen in the control room like in the previous LEP and SPS control rooms (one shorting all the alarms and the other used to acknowledge). During stable operation, the alarm screen is the first place to look if something abnormal happens before searching interactively. During a startup, on the other hand, the screen can be used as a kind of check list; removal of alarms from the screen implies getting closer to a working machine. In both cases, the integrity of the system is of great importance, an alarm system in which one does not have 100% trust is almost useless.

- **Operator Interface**[5]: This is the principal tool to operate the whole DCS and it is accessible from the ACR, SCRs and remotely via Windows Terminal Server. In steady operation the main DCS shifter in the ACR will have all the privileges and the ownership of the DCS. This means that supervision will be possible from the rest of locations but control, for safety reasons (e.g. avoiding contradictory commands), will be restricted on hands of the main shifter. In turn, the main shifter is able to delegate a control segment of the detector to another shifter or expert permitting in that way the independent and concurrent operation of different parts of the experiment. The design and development of the DCSOI, which is based on the control hierarchy of the experiment is discussed in more detail next.

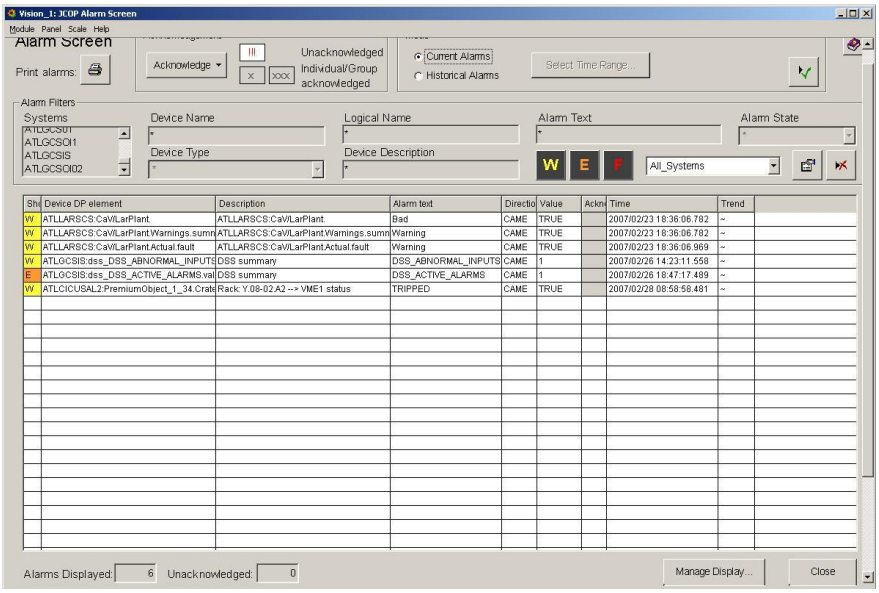


Figure 8.7: Alarm Screen. It is a JCOP framework component based in PVSS-II. It allows an easy handling of alarms by filtering depending on the time, name or severity of the alarms. Picture provided by J. Cook.

8.5 DCSOI Abstraction Hierarchy of Displays

As shown in Figure 8.3, the DCSOI makes use of the structure provided by the control tree in order to organize the large amount of displays required by the DCS. Each node in the hierarchy represents a detector section (or a certain abstract functionality) that constitutes a workspace susceptible to be operated independently. Consequently, in order to reach a certain degree of consistency among the different sub-detectors and displays, the developer must have a clear description of the display's goals and may only implement that functionality relative to their abstraction level. This hierarchical distribution of information permits first to accommodate the same console to different kind of users, and second, to homogenize the information across the many sub-detectors displays. Figure 8.8 shows the distribution of operations at the different displays contained in the abstraction hierarchy of ATLAS. Briefly, six levels of abstraction have to be modeled on the DCSOI:

1. Global operations. The whole DCS is represented by the most top node. Information on experiment goals, outputs of sub-detectors, general infrastructure and LHC, DAQ and magnet system conditions should be present. This should be the workspace used by the DCS shifter during steady operation.

2. Sub-detector Operations. These are displays that allow the independent operation of a certain sub-detector from the rest. In there, the sub-detector goals, infrastructure, outputs of the main systems or the conditions of the sub-detector DAQ partitions must be present. These should be the workspaces used during steady operation by the sub-detector shifters in both the ACR and in the SCRs.

3. Operations with DAQ. These displays show specific information relevant to the data taking process of a certain DAQ segment (e.g. run control state, DDC controller) and the state of the different DCS systems that are contained within the given DAQ partition. These should

be the workspaces used in the DCS for the partial operation of a sub-detector and the low level coordination of the DCS with the DAQ.

4. Generalized operations of systems (e.g. HV or LV). From this level the shifter or expert can supervise and control entirely a system that belongs to a ceratin DAQ partition (i.e. switch on or off the HV). In there, constraints with other systems must be clearly stated. These workspaces, and the rest located below on the abstraction hierarchy, should be used more for commissioning of debugging rather than for operations.

5. Topological operations of a system. The experiment at this level is decomposed in parts and, at higher levels of the abstraction hierarchy, it can be displayed in 3D. These workspaces should be used to investigate in-depth an error allocated or correlated geographically by means of the 3D displays. The displays at this level show detailed information of all the systems contained within the given detector region and provide an intermediate level of decomposition on top of the device.

   These two levels, geographical (5) and functional (4), can swap during the design phase to give more relevance to one or the other as discussed in Section 6.2.

6. Physical device operations. Detailed information on the conditions of a physical device such as a HV crate or a cooling loop are displayed to the shifter or expert.

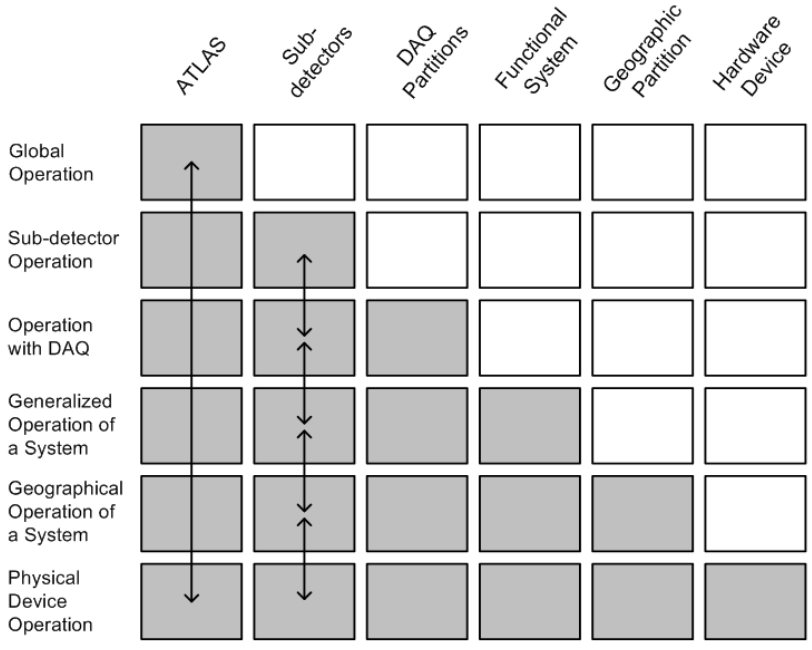


Figure 8.8: Modeled cells of the ATLAS DCS abstract hierarchy (in grey). Six levels of abstraction have been identified. In order to move through the different parts of the DCS, tools to enable an effective navigation need to be implemented.

Finally to mention that this design based on an abstraction hierarchy of displays follows the three general principles of 'Ecological Interfaces'[108],[109] in designing human-machine interfaces for process control:

1. To support knowledge-based behavior. Representing the ATLAS DCS work domains in the form of an abstraction hierarchy serves as an externalized mental model that support problem solving.
2. To support rule-based behavior: Each node in the control hierarchy has a display associated. By means of navigation facilities the displays can also provide a consistent one-to-one mapping between the work domains constraints and the cues provided by the interface.
3. To support skill-based behavior: The interaction via time-space signals allows the operator to act directly on the interface being able to accommodate the console to his/her needs or degree of expertise.

Design Problem

Two questions pertinent to interface design arise when creating the displays for the different workspaces of the abstraction hierarchy presented before: How to describe the domain complexity? and; How to communicate this information to the shifter/expert? This would provide a basis for determining the form, content and structure the information should take. The idea is of course that information should be presented in a form that is compatible with human cognitive and perceptual properties. These two questions define the core of the interface design problem that any developer should have into account when designing a display.

In designing a well-behaved human-machine interface for process control, some guidelines are given in order to solve eventual design problems that can appear such as consistency, intuitive approach or reduce chance for typing errors.[110]. They are listed here:

- The graphics interface must be designed with the operator's view of the process in mind. Taking advantage of the modular design, the system model will contain both, geographical and functional representation of the sub-detectors.
- Functional relationships between sub-systems need to be represented in the display next to each other ('out of sight, out of mind'). Otherwise operators can treat the sub-systems as being independent of each other.
- To prevent confusion, the user interface itself should be as simple as possible. Complex tasks may require complex interfaces, but that is no excuse for complicating simple tasks. The effectiveness can be also lost if the screens distract rather than inform, or in case one needs to navigate into too many screens to get the job done.
- Minimize color using grayscale in most of the displays. The developer should resist the temptation to play with the many great colors available and use the standard ones presented in Appendix A. When the job is to control a physics experiment, the technicolor (wild random use of color), screen furniture and visual toys are very distracting.
- The displays should follow cultural standards (i.e. language, time and standard units).
- Irrelevant information must not be present. Otherwise, operators will have to determine what information to attend to, and what information to ignore. Furthermore, the interface should be designed such that the perceptual saliency of its objects is relative their importance.

Obviously, all these statements are easy to say but more difficult to implement. Appendix C, shows the usage of the DCSOI and discussed guidelines for the partial implementation of the common infrastructure set-up of the ATLAS pit.

8.6 DCSOI Implementation

Once the strategy and required functionality of the DCSOI have been clarified, the layout of the console can be designed. The main constraint in designing a common layout is that we want to be able to display the information contained in this large system by means of a single console. Therefore the standard layout of the DCSOI has five constituent parts that allow a multiple displaying of information and an easy navigation through the DCS (see Figure 8.9). Different privileges or roles can be granted when using the console, these are classified into: observer, control and expert. Following, the attributed functions to the different parts of the layout are discussed:

- **FSM Module**: It gives a general fixed view of the process informing about the behavior of the topmost node in the hierarchy and its direct children. From this module the operator can send general commands to the children and also can include/exclude a certain part of the control hierarchy (i.e. to allow an expert to debug a certain part of the control tree). These functions for partitioning are also available from the main and from the secondary modules.

- **Main Module and Secondary Modules**: Theses are the main interfaces for a selected workspace (i.e. a SCS, a HV system, etc.). The secondary module is normally a summary of the information contained in the main module. The purpose of this double display is to keep the perspective of the detector while studying more in detail a problem in another workspace. Moreover, the shifter/expert can accommodate the console to his/her needs.

- **Alert module**: Its behavior is similar to the alarm screen, faults can be filtered (by severity, time, name) and be acknowledged. However, while the alarm screen treat only with PVSS-II data, the Alert Module is integrated into the control hierarchy by using the fault-detection mechanism discussed in Section 6.3. This mechanism manages faults in the system by dedicated SMI++ objects (*Status*). In case of a fault, different severities (i.e. ok, warning, error and fatal) can be attributed, and rules and procedures, can be established within the SMI++ objects. This information is of course very valuable and consequently it is applied to the Alert Module. The main advantage of this module is that it constitutes a quick fault-detection mechanism integrated into the DCSOI, enabling the operator to access any conflicting workspace within the full control hierarchy with a single mouse-click.

- **Message module**: It shows to the operator important messages about the detector operation with time stamp.

- **Navigation facilities**. In order to reach quickly against a problem, navigation functionality across the control hierarchy has been integrated within the DCSOI. A navigator is available within both the main and secondary modules and it works like a web browser. The navigator keeps a storyboard of the screens, to monitor the operator progress, see improvements, and keep track of the goals. There are four buttons: back, forward, home, and go-up (1 level in the hierarchy). All the navigation functionality is also available in functions available from the fwFsmAtlas to be used when building the displays. Thus, the operator can act directly on the display in order to access a selected workspace.

The frame, the organization of the displays, as well as the navigation functionality, are all based on the back end organization presented in Chapter 6 and the strategy shown in Figure 8.3. Concluding, to remark that the hierarchical organization and the modular design of the JCOP FSM represent an excellent skeleton suited for building the DCSOI. Following the main advantage of the DCSOI are summarized:

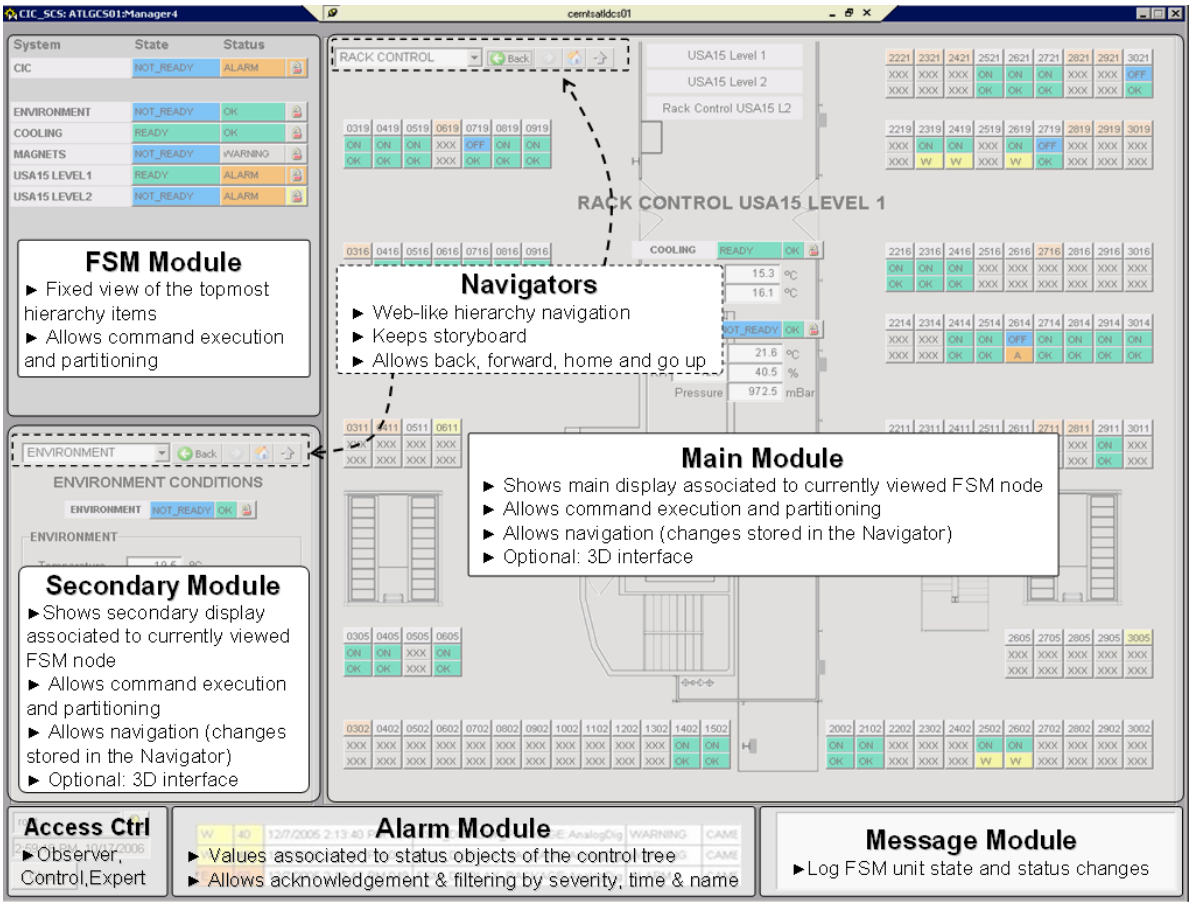


Figure 8.9: Layout of the DCSOI. There are five constituent parts to accommodate the large amount of DCS data into a single console allowing at the same time an effective navigation across the detector parts. 1) The FSM module gives an overall fixed view. 2) and 3) The main and secondary modules present to the operator or expert the different workspaces of the hierarchy. Whilst one of these modules can be kept fixed, the other can investigate one part of the detector more in detail. 4) The message module shows important messages to the operator with time-stamp. 5) The alert module constitute a quick-fault detection mechanism integrated into the DCSOI. By means of a single mouse-click any workspace in the hierarchy can be accessed by the operator or expert.

- A quick fault-detection mechanism has been developed within the frame of the DCSOI. This allows to identify and quickly display the root of the problem.
- The nodes interfaced into DCSOI normally belong to different sub-detectors and levels of the abstraction hierarchy. These sub-detectors present an isomorphic hierarchy that can serves as an externalized mental model to support problem solving.
- The same DCSOI can be used by different groups of people, each having different and often conflicting requirements, in different operational scenarios. Operators in shift can look at the whole machine displays rather than specialist areas. However, during commissioning or detailed investigations of faults, many equipment specialists need to run extensive tests of their hardware. In that cases, both shifters and detector experts can reach quickly an in-depth view of the experiment by using the same interface.
- Low code cost development. Many graphical widgets that interface the control hierarchy into the displays are provided by the fwFsmAtlas. Giving different shapes to the widgets, they can represent different hardware or logical nodes. These widgets can be latter re-used for new facilities as the Status Screen or web.
- Easy maintenance and upgrade. In case a certain system is removed or a part of the experiment is improved only this part need to be re-designed for the DCSOI while the rest can continue working normally. Moreover, using standard fwFsmAtlas and JCOP FSM functions facilitates maintenance and development.
- Easy Integration. The DCSOI has been built within the context of the JCOP Framework. The PVSS panels, libraries, examples, etc. are installed as a typical JCOP framework. This utility, which is very well known by the DCS developers, facilitates the integration of the different sub-detectors.

8.7 Conclusions

ATLAS can be understood as a very large and complex industrial system which is envisaged to have about a thousand workspaces to monitor and control. These workspace correspond to different nodes at different levels within the control hierarchy, for example, ATLAS, a sub-detector, a HV system or part of it, etc. How to support the shifter or expert in controlling this large system and how to maintain integration across the many displays are the main challenges faced during the design and implementation of the DCSOI. In order to answer these questions a common strategy, based on the design of an abstraction hierarchy of displays, homogenizes and integrates the different sub-detector systems into the same console. Then, by means of an effective navigation any workspace within the control hierarchy is accessible. Faults, that are treated within the control tree, are hierarchically monitored and, their associated workspaces, quickly accessible. The different tools developed for the DCS top level interface provide enough flexibility to allow the future evolution of the control system facilitating the integration and maintenance of the numerous displays that compose the full control hierarchy.

The top level interface has been in use since February 2006 with satisfactory results. During the commissioning period the DCSOI has been installed both, underground in the electronics room next to the equipment, and in the ACR.

Conclusions

The integration of the ATLAS DCS into a control hierarchy has been presented. The control hierarchy is designed to be implemented using the JCOP FSM tool which is based on the SMI++ toolkit. This allows describing the back-end control system as a collection of objects that are organized hierarchically following a finite-state machine logic. These objects have asynchronous behavior, and following AI-like rules, allow the sequencing and automation of all the DCS tasks for operation. Previously to the LHC experiments, SMI++ was successfully applied to the two HEP experiments Delphi and Babar. However, these applications had a much smaller scale and were not combined with any SCADA system.

On the other hand, outside the HEP world, few commercial solutions in what is called 'intelligent distributed agents' for plant automation or manufacturing are available today. The main reason for that may not be related to technological constrains, but more to the nature of the market. Whilst controls at HEP experiments evolve continuously with new operational procedures and radical new solutions, the automation market is historically slower in accepting changes. Operators and experts have a solid training and working procedures associated to a certain stable automation process. However, information technology is adding incrementally value to existing industrial plants leading industrial automation and control research towards 'distributed intelligent' systems that provide a more modular and distributed design.

Taking the scale of the ATLAS DCS into account, requiring the supervision of around 200.000 channels, the organization of a control hierarchy which is in charge of the whole experiment automation becomes a challenge. When looking for a functional or geographical view on the ATLAS DCS facilities, one notices the strong relationships existing between the elements: sub-detectors are structured in different geographical divisions (e.g. end-caps, barrel), which themselves contain a set of sub-systems (e.g. HV, LV, Cooling). These sub-systems themselves are also built from mechanical assemblies and these are built from devices such as a crate, a valve or an ELMB. Consequently, taking advantage of the modular design provided by the JCOP FSM, a standard architecture and standard interfaces have been developed to model the DCS facilities. The standards are described within ATLAS guidelines for the implementation of the control hierarchy. The architecture specifies the different constituent parts and their functions at each level. The interfaces define how objects interact with each other, with external systems, and with the person in charge of the operation of the experiment. These standards are widely shared throughout the ATLAS community enabling the construction of an isomorphic hierarchy.

The State & Status concept used for ATLAS emphasizes the early detection, monitoring and diagnosis of faults based on dedicated SMI++ objects. Fault-detection, monitoring and diagnosis are essential parts of the operation of any HEP experiment or industrial plant and an integral part for the coordination of tasks. For example, physics data taking involving the use of some damaged part would be disrupted, and operations from the supervisory control level could be no longer valid or be even dangerous. In order to avoid this situation, fault-detection and diagnosis SMI++ objects

detect and diagnose the faults in their initial phase (i.e. the device unit) and then propagated them through the hierarchy. Different procedures or actions can be taken at low levels of the control tree depending on both, the severity of the fault and the operational mode of the system involved. This mechanism implies to split the information into two different but cooperative trees which together define the behavior of any node in the control tree. All the logic for the automation and sequencing of the experiment reside in CUs, LUs and DUs. This grouping of logic helps the engineer to understand the control program simplifying the integration and future maintenance of the DCS back-end. Guidelines and functions that facilitate the application of the State & Status concept have also been created. The granularity of the hierarchy, and the new approach used in ATLAS have been investigated with the aim to optimize the overall functionality and system performance.

This thesis has also shown the synchronization of the two main integration systems involved in the operation of the experiment, namely TDAQ and DCS. Data reconciliation between the two systems will be of prime importance in order to obtain good-quality data. We have looked closely at the use and implementation of the FSM concept in both systems and pointed out similarities and differences between the two. A dedicated layer within the DCS control hierarchy has been reserved for this interaction purpose. This functional level at the DCS organization mirrors data acquisition partitions such that synchronization between both systems is achieved using the DDC package. The main target of this synchronization mechanism is twofold, first it allows the TDAQ to master different parts of the DCS during the physics data-taking process, and second, it prevents incorrect data taking when a certain detector part is malfunctioning. Hence, a common mechanism, with tools and guidelines, has been developed ready to be applied to any sub-detector hierarchy.

This thesis work concludes presenting the human-machine interfaces used for the operation of ATLAS DCS. This set of interfaces has been designed to accomplish the monitoring, control and analysis of the operation of the ATLAS detector providing both, overall and in-depth views of the experiment conditions. The main development effort has been done for the frame of the DCS Operator Interface (DCSOI) which is the main tool for the experiment supervision and operation. The hierarchical structure and the modular design of the control hierarchy are an excellent framework to build the DCSOI. The workspaces belonging to different levels of the abstraction hierarchy are accessible by means of an effective navigation. A quick fault-detection mechanism has been developed within the frame of the DCSOI. This mechanism allows to identify and quickly display the origin of the problem. The interface is modular and flexible, can be accommodated to different operational scenarios and fulfil the necessities of all kind of users. The first production system integrated within the DCSOI was the Common Infrastructure Control that has been in use since May 2006 with good results.

Appendix A

Framework FSM ATLAS (*fwFsmAtlas*)

In this appendix, some design considerations, recommendations and functionality provided by the fwFsmAtlas are given along with hints for the actual implementation of the control hierarchy in ATLAS. All necessary steps which have to be performed by the sub-detector experts to create the hierarchy are described. In order to follow these guidelines it is necessary a certain knowledge of PVSS and the JCOP FSM.

State-Status Concept

The 'State' and 'Status' are two aspects that work in parallel and provide all the necessary information about the behavior of any system at any level in the hierarchy. The State defines the 'operational mode of the system' and the Status gives more details about 'how well the system is working' (i.e. it warns about the presence of errors).

- **State definition**. For consistency, the common control domains (i.e. SCS, HV, LV, Cooling, etc) should have the same (or similar) states, status, transitions and actions. As a result, we will make life easier to the future shift operator. Thus, two generic state machines has been proposed in order to homogenize the different control domains (see Chapter 6). Table A.1 shows possible states, commands and transient states whit their colors associated.

- **Status definition**. The Status names and severities are fixed for all sub-detectors and subsystems which must follow these qualifiers. The colors fulfil the Framework look-and-feel convention. The Status convention was shown in Table 6.1.

Concerning the colors standards, Table A.2 shows the color coding used for the implementation of the finite state machine logic. Similar to for instance telephone numbers, shifters will find easier to remember combinations of seven, plus or minus two, items. Thus, this rule is applied to colors and, consequently, users should make an implicit assumption that all things colored the same are associated in some way.

STATE	COMMAND
READY	GOTO READY
NOT READY	SHUTDOWN, GOTO NOT READY
UNKNOWN	RECOVER
STANDBY	GOTO STANDBY
ON, ON25, ON50, ON75...	GOTO ON, GOTO ONxx
OFF	GOTO OFF
TRIPPED	RECOVER
LOCKED	UNLOCK, LOCK
RUNNING	RUN
STARTED	START
STOPPED	STOP
TRANSIENT STATE	COMMAND
RAMPING UP, GOING TO ON,...	GOTO ON, GOTO ON75, GOTO STANDBY n
RAMPING DOWN, GOING TO ON,...	GOTO OFF, GOTO ON25, GOTO STANDBY n...
CALIBRATION	CONFIGURE
GETTING READY	GOTO READY
GETTING NOT READY	GOTO NOT READY
STARTING	START
STOPPING	STOP

Table A.1: List of states, transitions and commands to be used in the ATLAS FSM.

Color Coding for the DCS	
GREEN	Static state. The system reached its final operational stage
BLUE	Static state. The system did not reached yet its final operational stage
LIGHT BLUE	Transient state.Signalizes an ongoing transition between normal states
ORANGE	Error state. Used if state is UNKNOWN (e.g. due to loss of communication)
RED	Severe Error state. For example TRIPPED in case of a power supply trip.

Table A.2: Common color coding used by the DCS

Creating the Hierarchy Step by Step

In ATLAS the control tree is composed of two parallel paths, one for the State (CUs, or LUs at the button levels of the tree) and one for the Status (logical objects within a CU or children of a LU). 'Control Units', 'Logical Units' and 'Logical Objects' are all logical object types. The difference is that the CU runs as a SMI++ domain while the Logical Objects and the LUs run inside the SMI++ domain. Figure A.1 shows the typical distribution of these objects for ATLAS.

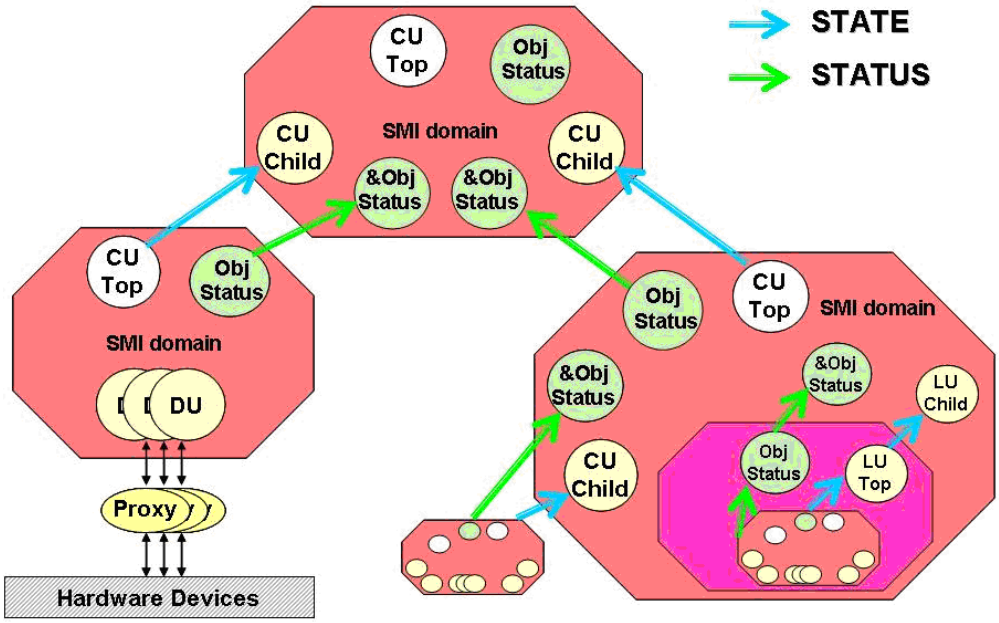


Figure A.1: Schema of a Control Hierarchy formed by two communications paths or parallel control hierarchies.

Each CU and LU must have associated a Status object defining the severity of the problem if it exists. On the other hand, for the DUs the user has the freedom of associating a Status object to it or not.

To implement our hierarchy we have to create different Logical Object Types for the Status and for the State entities taking into account the naming convention (see later Section A). Once the types and the 'base hierarchy' formed by State objects are created, in order to build the hierarchy of Status objects the developer can do it 'manually' or by using the following functions within a PVSS script:

- *fwFsmAtlasAddAllStatusObjects*: It adds all status objects to the hierarchy starting at the most top node.
- *fwFsmAtlasRemoveAllStatusObjects*: It removes all status objects from the hierarchy starting at the most top node.
- *fwFsmAtlasAddStatusObject*: It adds a status objects with its reference to a certain node.
- *fwFsmAtlasRemoveStatusObject*: It removes a status objects and its reference from a certain node.

These functions attach all those Status objects to their pertinent CUs (smiSMI domains), LUs or DUs. At the same time they build the different references between the Status objects in order to ensure communications within the parallel hierarchy formed by the Status objects.

Figure A.2 illustrate an example corresponding to the hierarchy of a CIC prototype at the building SR1 and the resultant DEN screenshot. The CIC is the top CU and STATUS_CIC is not pointed for anyone else in an upper level. The value STATUS_CIC is taken from &STATUS_CLEAN_ROOM and &STATUS_RACKS. The symbol '&' means that the object is a reference. In this case, these are pointing to objects located in a level below (i.e. STATUS_CLEAN_ROOM and STATUS_RACKS) and so on. In that way, the values are propagated upwards.

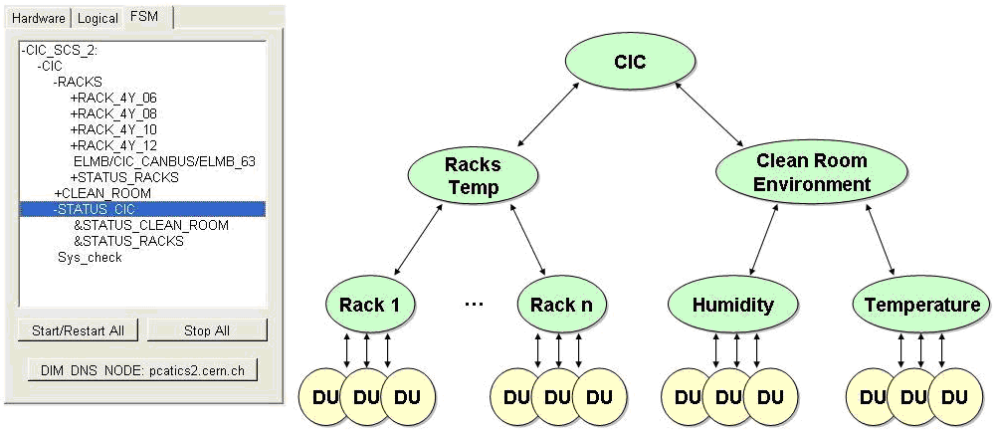


Figure A.2: Left: Framework Device Editor Navigator view of the CIC prototype of control hierarchy. Right: Schematic view of the prototyped hierarchy.

FSM Hierarchy for Non-Framework Projects

The JCOP FSM is a framework component and, in principle, it relies on the framework standards for its utilization (i.e. PVSS DP structure). However, in some cases, it is not possible to use these standards. Then, a procedure has been established in order to give to the developer the maximum flexibility when creating the control hierarchy. This procedure permits the developer to describe their DUs by means of a PVSS scripts rather than relying on the Framework. Consequently, the developer is free of defining the granularity of the tree and the behavior of the lowest level by building home-made DUs through the PVSS scripting language. Figure A.3 illustrates the mechanism that requires of various steps to be followed both in the PVSS and JCOP FSM side.

Work in the PVSS side

The developer must write a PVSS script that corresponds to the DU. This script runs all the time (e.g. by declaring a PVSS control manager to run at start-up) and sets the proper State and Status. The scripts must be quick and simple as possible in order to not became a bottleneck. For getting a better performance in the PVSS script we propose for instance:

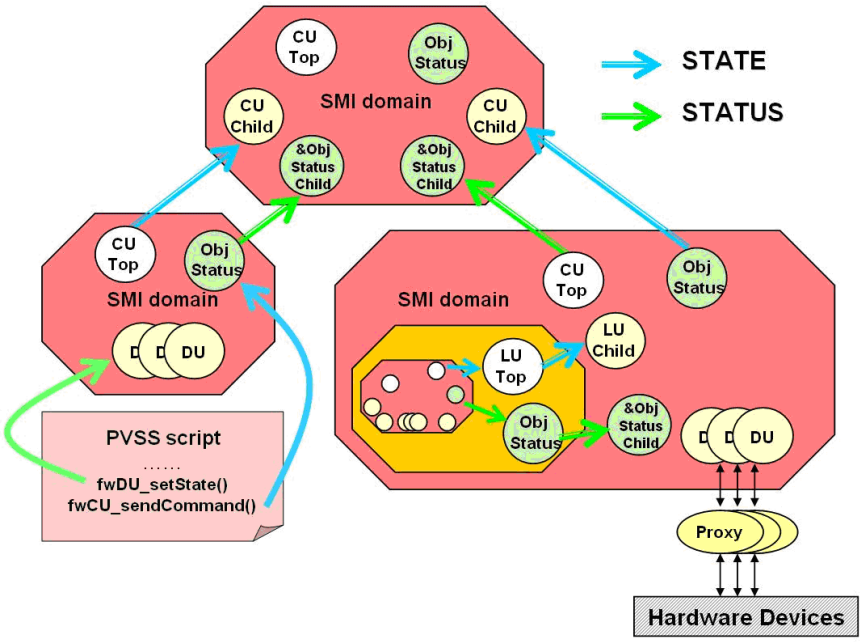


Figure A.3: Creation of a DU by menas of a PVSS script.

- The developer can DP connect to the alarm state (if the alarms thresholds matches with the FSM state) instead of to the value. Like that the callback function associated to the DP connect will be executed only when a change of state arise.

- If dpConnects are used within the script, the developer must keep the callback functions as simple as possible.

- If the device is not time critical one can use a time function instead of dpConnects.

Once the State has been calculated the function *fwDU_setState* is then called in order to set the new state of the DU.

Whenever the Status has changed the function *fwCU_sendCommand* needs to be called. Within this function, it is specified the status object that belongs to the DU and the command calculated on the script that will trigger a change of Status (i.e. GOTO_WARNING). The corresponding Logical Object Type has a set of commands declared (GOTO_OK, GOTO_WARNING, GOTO_FATAL, GOTO_ALARM) that will cause a transition on the Status. Any 'when condition' needs to be declared on the Status Logical Object Type.

The reason for having a Status object instead of another DU for the Status is that we want to group within a unique DP both State and Status. Thus, in the PVSS side, the same DP type *Fw_DUsWithScript* (see Figure A.4) contains the alert handling of the device, plus, the archiving of the State and Status.

Work in the FSM side

The developer must create a Device Unit type where all possible states and commands are declared in the 'State List' and in the 'Action List'. Moreover, within this Device Unit Type:

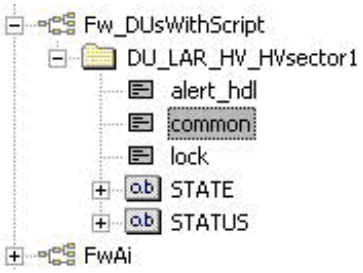


Figure A.4: Fw_DUsWithScript.

- The developer must leave in blank the scripts belonging to 'Configure Device Initialization' and 'Configure Device States'.

  Don't use the 'Simple Config', set your states and commands directly on the State and Action List. Your home-made scripts will set the new states and using the wizard might accidentally overwrite them.

- In case the DU has a list of actions, these must be scripted on the 'Configure Device Actions'.

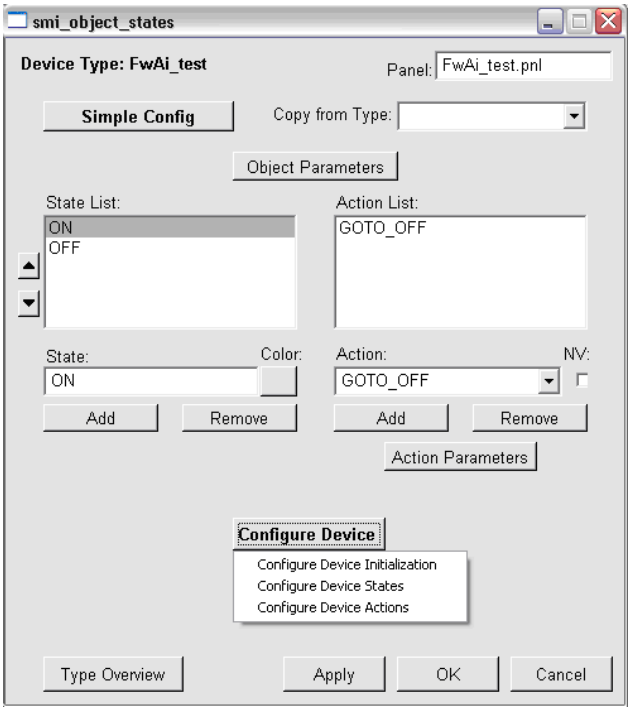


Figure A.5: Device Unit Type.

If the Device created through the script has also an Status this must correspond to a Logical Object Type (see Figure A.6). This Logical Object looks very simple. In the 'State List' there is: OK, WARNING, ALARM and FATAL. In the 'Action List' there are commands that cause the change of transition (e.g. GOTO_WARNING). This commands are sent from the home-made script through the function *fwCU_sendCommand* already described before.

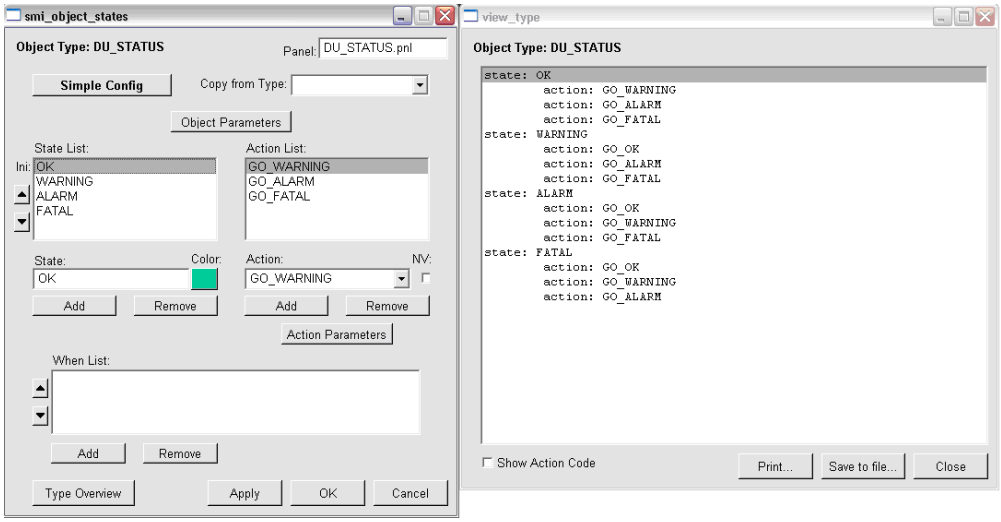


Figure A.6: Status Logical Object Type.

Recommendations

Performance Issues

- CU - Control Unit (Up to 50 CUs per PVSS PC)

  Can be Included, Excluded, etc and Taken in stand-alone mode.

  Corresponds to one smiSM process

- LU - Logical Unit (Up to 500 LUs per PVSS PC)

  To be used in the bottom levels of a tree (just above the DUs).

  Can contain children, but not of type CU

  Can be Enabled/Disabled (can not run in stand-alone).

  Corresponds to an object within a smiSM.

- DU - Device Unit (Up to 1000 DUs per PVSS PC)

  Corresponding to a 'real' device in PVSS.

  Can be Enabled/Disabled (can not run in stand-alone).

  Behavior defined via PVSS scripts (instead of SMI code).

The numbers presented above are recommendations, and, under certain circumstances, they could be exceeded. However, it is foreseen that any system belonging to a 'normal' control hierarchy will need to exceed these quantities.

Alarm Handling

Alarms from the PVSS alert configurations at the DP level will be displayed using the framework alarm screen, and are intended to be used for detailed problem tracking and acknowledgement. It is strongly recommended to have an alert handling configuration at least for each DP that corresponds to a certain DU.

In addition, a simplified alarm handling mechanism is introduced at the level of the FSM units - the Status - representing a scaled down version of the PVSS alerts. The Status allows for context based signalization of problems and error tracking inside the control hierarchy directly on the DCSOI operator interface. Note that consequently the alarms of those DPs which are not considered in the hierarchy are thus only visible on the FW alarm screen.

Command Execution

For the final production systems it is envisaged that DCS users will operate the systems only through the DCSIO and the FW Alarm Screen.

Thus, the developer must implement the commands to be sent by the operators inside the FSM rather than from PVSS-II panels or scripts. This will help to maintain and understand the different DCS projects.

FSM Version Consistency

When integrating different FSM trees belonging to different PVSS systems check that the FSM versions are the same. The most recent FSM production version to be used will be announced on the central ATLAS DCS web page or directly installed from the central DCS repository disk. If you are upgrading your FSM please always check the release notes before:

http://lhcb-online.web.cern.ch/lhcb-online/ecs/fw/FW_FSM.HTML

PVSS Panel Export

It is important to verify that your final operator panels function correctly in different systems since they will be exported to other distributed projects (i.e. GCS or SCS).

Most probably the system name will be different in your development project compared to your production project(s). Check your PVSS panels before exporting them to the GCS.

Simple DU Scripts

When defining the DU types, keep the state configuration scripts as simple as possible. These scripts are called each time there is a change on the value of the DP associated to the DU.

Unique DIM DNS node

During the development process one can run the DNS.exe locally. However, when integrating several FSM running in different computers a unique DNS DIM node or node list must exist. For final ATLAS, there will be a list of redundant DNS DIM nodes to be used by all FSM objects (as of June 2006 one global node runs on PCATLCICSCS).

Useful Functions

fwDU_startTimeout: In each DU with device action execution this function has to be present in the device action configuration script. Thus, a time-out trigger is inserted into a DU in order to limit the time to switch from one state to another. With this function it is possible to program the switch of a DU into a 'UNKNOWN' state in case the target state or any new state change does not occur within a given delay.

```
if (command == "RAMP_UP")
{
        /* fwDU_startTimeout(delay,domain,device,errorState,targetState); */
        fwDU_startTimeout(10,domain,device,"NOT_RESPONDING","ON");
}
```

fwFsmAtlas_openPanel: This function permits to open any panel associated to a certain object in the DCSOI.

fwCU_connectObjState: During the preparation of the operator displays, this is the function to call for displaying the State and Status of a certain node.

fwDU_getAlarmLimits: The Alert handling thresholds of a certain DP can be re-used to define the Device Unit thresholds. This function should be ONLY used for those small numbers of DPs which change the alert handling configuration during operation. The inclusion of this function in the DU script makes the performance worse.

```
FwElmbAi_valueChanged( string domain, string device, float value , string &fwState)
{
        dyn_float limits;
        fwDU_getAlarmLimits(device, "value", limits);
        if      (value < limits[1])
                fwState = "OK";
        else if ( ( limits[1] <= value ) && ( value < limits[2] ) )
                fwState = "WARNING";
        else
                fwState = "ALARM";
}
```

More functions in: http://lhcb-online.web.cern.ch/lhcb-online/ecs/fw/FW_FSM.HTML

Use ONLY the functions listed in the web page. The functions contained in the FSM libraries are not supported for users.

FSM Naming Convention

All FSM names are upper case.

1. **FSM Object Names**

   (a) Sub-detector top-most node: ATL_<sub-detector name> Ex: ATL_LAR

   (b) The children of the sub-detector top node are the sub-detector's TTC partitions. <sub-detector name>_<TTC name> Ex: LAR_EMECC

   (c) The children of the sub-detector can be either a geographical division or a system division. <sub-detector name>_<TTC name>_<geographic name> Ex: LAR_EMECC_Q1 <sub-detector name>_<TTC name>_<system name> Ex: LAR_EMECC_HV

   (d) The next level can again be either a geographical division or a system division. <sub-detector name>_<TTC name>_<geographic name>_<system name> Ex: LAR_EMECC_Q1_HV <sub-detector name>_<TTC name>_<system name>_<geographic name> Ex: LAR_EMECC_HV_Q1

(e) The next children will normally be a DU. However, if it is not the case, one could follow with the same convention: <sub-detector name>_<TTC name>_<geographic name>_<system name>_<part name> Ex: LAR_EMECC_Q1_HV_SECTOR1 <sub-detector name>_<TTC name>_<system name>_<geographic name>_<part name> Ex: LAR_EMECC_HV_Q1_SECTOR1

(f) The Status nodes belonging to a certain node should add the prefix 'STATUS_'. Ex: STATUS_ATL_LAR, STATUS_LAR_EMECC, STATUS_ LAR_EMECC_Q1

2. **FSM Object & Device Type Names**. The fwFsmAtlas component provides different type templates for all FSM types (e.g. ATLAS_CU) which should be used for the top-level FSM nodes. Below the partition level, custom FSM types can be used which should follow the following naming scheme:

   (a) Logical Object Types: <Sub-detector name>[_<partition type>][_<sub-partition type>] Examples: LAR_HEC_LV, CIC_ENV_HUMIDITY

   (b) Device Unit Types [<Device base type>_]<sub-detector name>_<sub-system name> Examples: fwAi_TRT_HVMODULE For the STATUS FSM object, the predefined types ATLAS_STATUS and ATLAS_DU_STATUS should be used for CU/LUs and DUs, respectively. Whenever it is necessary to modify these types, follow the same naming scheme as above and append _STATUS.

3. **PVSS Panel Names**

   Each Control Unit (CU), Logical Unit (LU) or Device Unit (DU) can have a PVSS panel associated to it. The name of this panel should be the same as the FSM object.

   (a) Example: if there is a CU with name 'TRT_SCS', then the panel will be called 'TRT_SCS.pnl'. For each FSM node, it is also foreseen to have an additional secondary panel with the suffix '_info' (see FSM User Interface).

   (b) Example: if the CU panel is called 'TRT_SCS.pnl' the secondary panel will be called 'TRT_SCS_info.pnl' To follow the convention for the panel names is important for a later integration of all the ATLAS DCS distributed systems in the shift operator interface.

Setting Up the DCS Operator Interface

This section explains the basic procedure to be followed in order to integrate all displays within the DCSOI.

Screen Layout

In order to see properly the PVSS panels the screen resolution must be set to 1280x1024. The background of the DCSOI is an empty PVSS-II panel with fixed dimensions 1269x997. On the foreground 5 modules present the behavior of the detector for the different levels of the hierarchy. The description of these modules is the following.

1. FSM Module: The State and Status of a FSM node and its children is displayed providing all FSM functionality. This module has a limited size, and, in case of many FSM children a scroll bar appears.

2. Main Module: Its dimensions are 900x861. This is the main panel for the selected FSM node, this can be a SCS, a HV system, etc. Two dollar parameters are passed from the FSM Module to Main Panel ($node and $obj).

3. Secondary Module: Its dimensions are 310x410. The purpose of this is to keep a main view of a certain sub-detector while studying more in detail a problem that triggers deeper in the hierarchy. Thus, it is needed to create an additional panel with a summary of the information presented for each main panel. Two dollar parameters are passed from the Main Panel to the Secondary Panel ($node and $obj).

   A navigator is available within both, the main and secondary module. The navigator looks as a web browser. It has a combo box with the list of all the panels included bellow in the hierarchy and four buttons: Back, Forward, Home, Up (the operator goes up one level on the FSM hierarchy). Moreover, additional navigation possibilities exist though dedicated functions.

4. Message module: important messages with its time stamp.

5. Alert module: using the Status objects one must be able of filtering the alarms that belongs to that certain part of the tree.

In both Main and Secondary module, the developer has full FSM functionality (i.e. one can send FSM commands, change the partitioning mode, etc). This also means that within a certain workspace (i.e. HV system) information related to any other workspace (i.e. Cooling) could be displayed. Using the navigation functionality the operator can jump from one workspace to another using any of both, main and secondary modules. In order to assist developers in the creation of FSM panels with common functionality, a set of widgets have been created (and new ones will come with time). These widgets permit to display the state and the status, change the partitioning mode, send FSM commands and navigate between different control domains.

Creating the Displays, Graphical Widgets

The look & feel aspect becomes of primary importance when integrating the different sub-detectors. Thus, in order to facilitate the work to the developer and the understanding of the panels by the final operators in shift a set of widgets has been created and are available within the fwFsmAtlas. Figure A.7 shows several examples.

The widgets can be placed in both, the main and secondary panels, and are equipped with navigation facilities. To make the widgets work, the FSM node, object and label are specified as a $parameter. Therefore, these have all the FSM functionality being able to: display state and status, change the partitioning mode and send of commands. In the case of the 'error finder', it does not need any setting.

Organization of Displays

When organizing your FSM panels in your own system and exporting them to other distributed systems the developer must include them in the folders: 'FSMmainPanels' for panels to be displayed

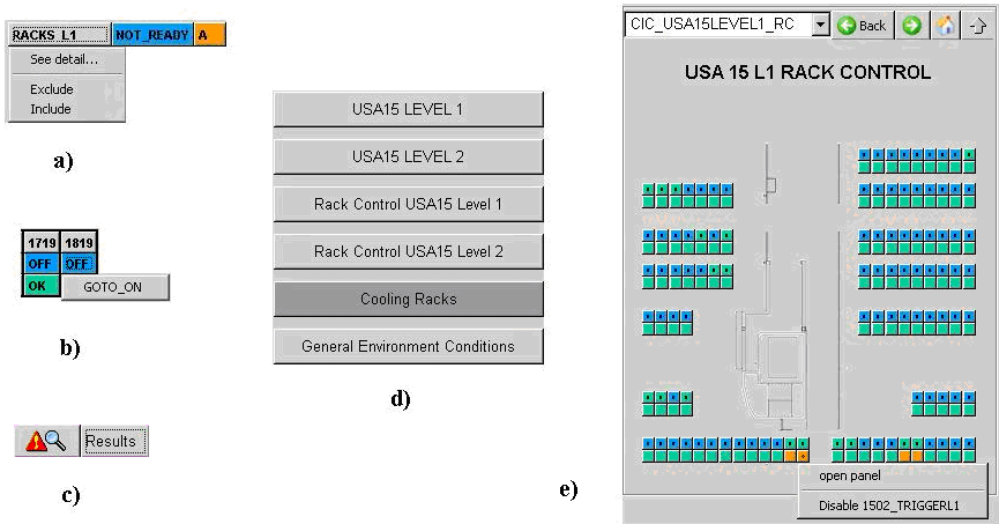


Figure A.7: Widgets for the DCSOI. a) Accessing/partitioning the Rack Control USA15 L1. b) Operating a particular rack. c) Error finder. It finds and access all the errors below in the hierarchy d) Menu with direct access to different CUs e) View of the secondary panel of all the racks in USA15 L1.

in the main module, and 'FSMinfoPanels' for panels to be displayed in the secondary module (see Figure A.8). This process is first done locally on the system for debugging. Later the final production panels will be exported to the same folders into the ATLAS central repository.

Including your panels

At this point your main and secondary panels have been created taking into account: the standards panel dimensions, the FSM panels naming convention, and the organization of panels into folders. To include your panels into the DCSOI the following steps have to be done.

1. To attach a main panel to a certain FSM node one must follow the normal procedure. In Editor mode, select a node, right click → Settings → Chose inside the folder 'FSMmainPanels' the panel that corresponds to the selected node (if the naming convention has been followed the name should be the same)(see Figure A.9).

2. The secondary panels are automatically opened by the DCSOI relying on the naming convention. The secondary panels use the same name as the main panels adding the suffix '_info'.

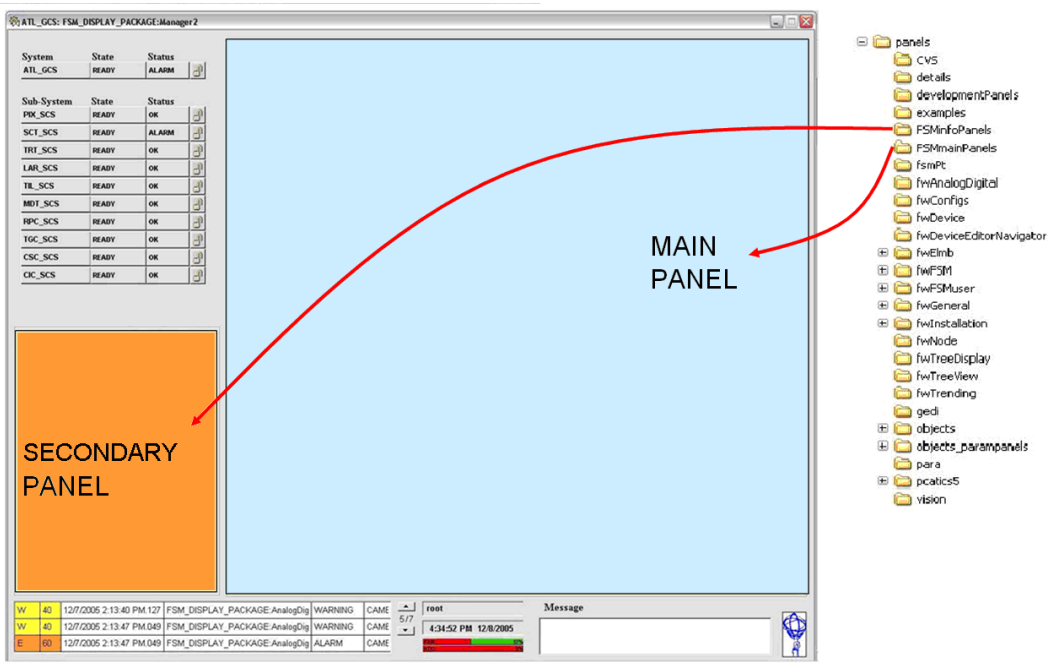


Figure A.8: Panels organization.

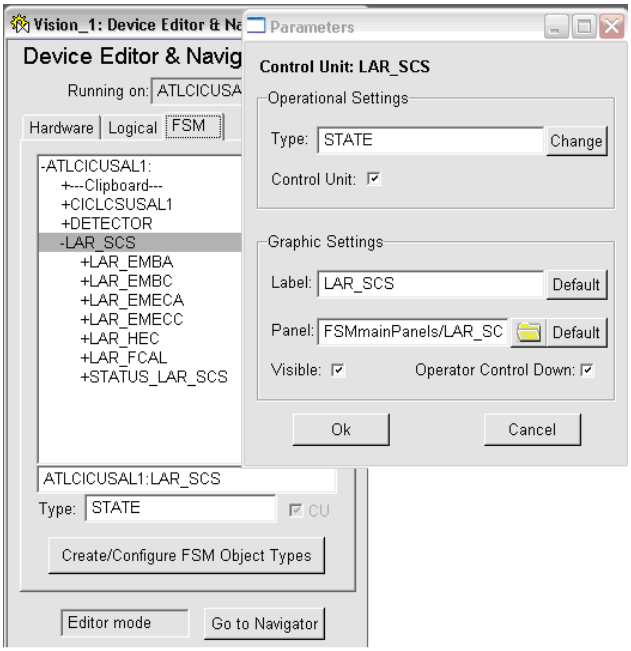


Figure A.9: Setting a main panel to a FSM node.

Interaction with TDAQ Control

It is foreseen that the number of DDC controllers running for a sub-detector should correspond one-to-one to the number of TTC partitions of that sub-detector. Similarly, should exist one FSM domain per TTC partition, and one DDC DU per FSM domain (see Figure A.10). Thus, within each TTC domain there is one Device Unit interacting with a certain DDC controller.

(1 TTC partition) x (1 DDC controller) x (1 TTC FSM domain) x (1 DDC DU)

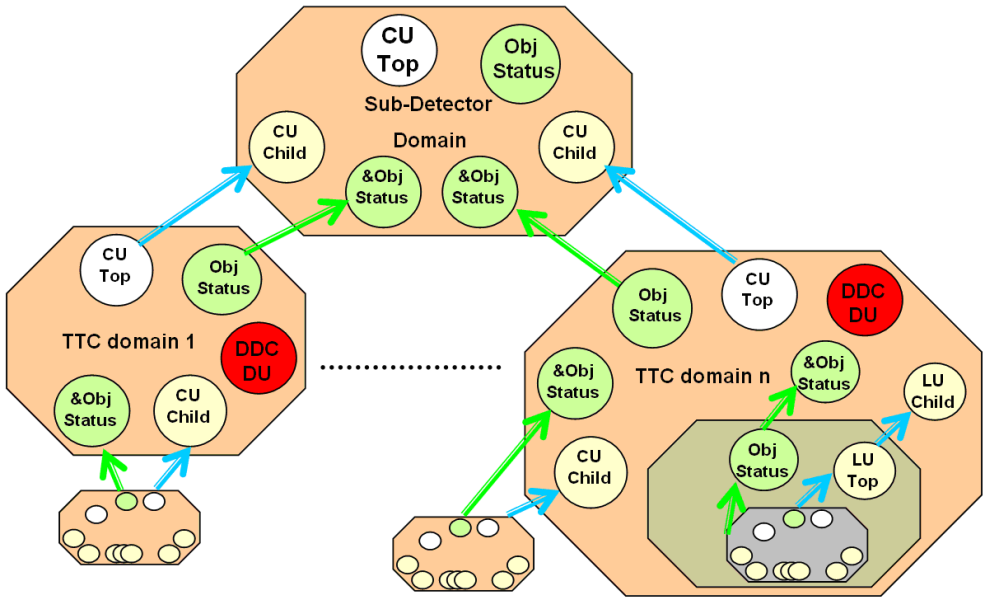


Figure A.10: One DDC DU within each TTC FSM domain.

During the installation of the 'fwFsmAtlas' package, a DP Type as well as a Device Unit Type, both called 'FwFsmAtlasDDC', are created (see Figure A.11). The main duties of this Device Unit are:

1. Reporting the DCS state. Asynchronously it must report to TDAQ any occurrence of conditions preventing the data taking. The granularity is the TTC partition. Thus, the DDC Device Unit checks the state of its TTC domain and sets a flag (dpe: 'notDataTaking') that reports the state of the detector for a certain TTC partition. The FSM states for a TTC partition are: READY (TTC partition ready for data taking) and NOT_READY (TTC partition not ready for data taking).

   Note: Be consistent when propagating the states upwards to the TTC level. If the DCS is NOT_READY, it means that the whole TTC partition is not ready for data taking.

2. TDAQ Command Execution. TDAQ can operate the DCS executing transition commands by means of the FSM. The set of FSM commands to be issued by TDAQ are meant to be general (i.e. 'PREPARE_FOR_RUN'). When a command is triggered (dpe: 'trigger') from the TDAQ, the DDC Device Unit reads a set of parameters and it sends the command to the selected node. The parameters (dpe: 'fsmParametrs') are filled by TDAQ in the format: FSM_domain|FSM_command|time_out. 'FSM_domain' and the 'FSM_command' mandatory fields. In case the command is sent to a single DU the device object must be specified

in the dpe 'parameters'. The response (dpe: 'response') to the command execution depends on the timeout. If the timeout value is 0, the response is set to a good state automatically. In case of an existing timeout, the response is set within the timeout interval to either a good or bad state.

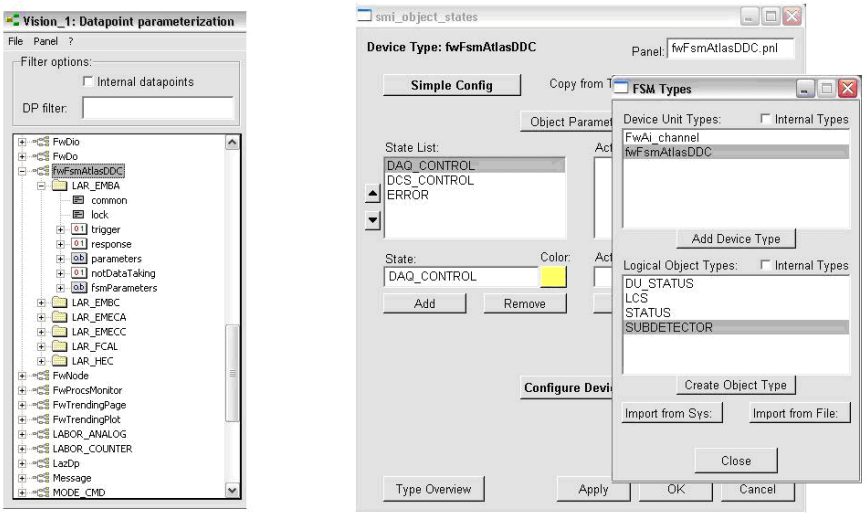


Figure A.11: FwFsmAtlasDDC.

Work in the PVSS side

The developer have to create a DP of type 'FwFsmAtlasDDC' for each TTC partition (DDC controller). This type is available after installing the fwFsmAtlas. The DDC controller, together with the FSM DU, will be on charge of R/W of this DP.

Work in the FSM side

The developer must insert the DU previously created in its corresponding TTC FSM domain. The DU unit type is already edited with the functionality explained above. Thus, if this functionality is enough for the sub-detector no extra work needs to be done(see Figure A.12).

Access and Installation of the fwFsmAtlas Component

The actual skeletons of the FSM panels are assembled within a framework component module 'fwFsmAtlas' which is accessible via CVS. It can be found in the repository 'atlasdcs' of the central CVS server of the CERN IT department.

The 'fwFsmAtlas' is installed using the Framework Installation Tool. Once the fwFsmAtlas is installed, one needs to change the settings of the FSM tree using the DEN (see Figure A.13). In Editor Mode select your system, right click → Settings → Choose the panel 'fwFSMuser/fwUiAtlasFrame.pnl'. This panel provides the standard FSM background with the 5 module. When the new main panel is selected then set the panel size to a static value 1269x997.

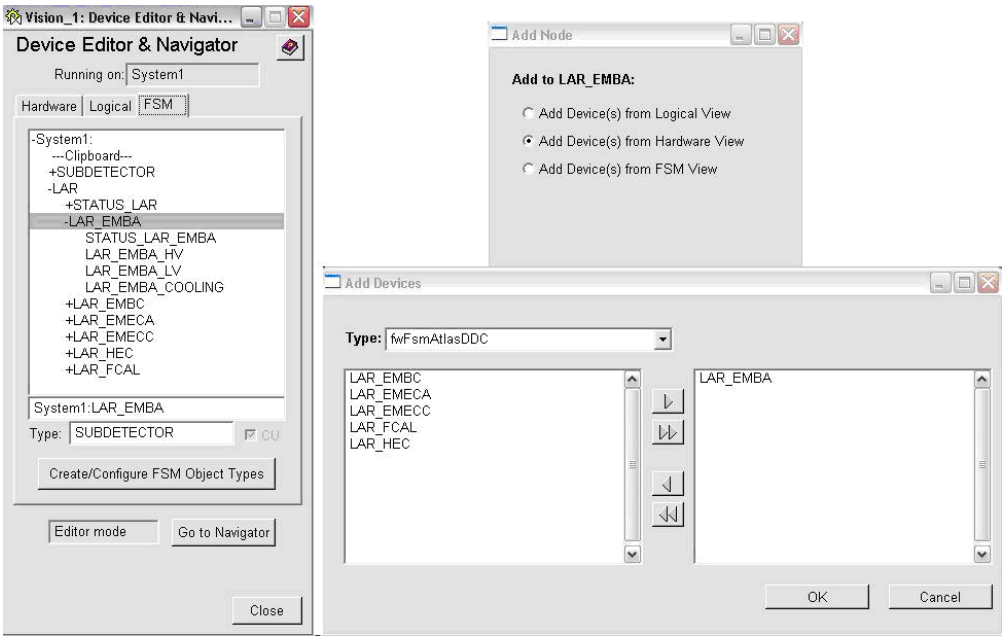


Figure A.12: Adding the DDC device unit.

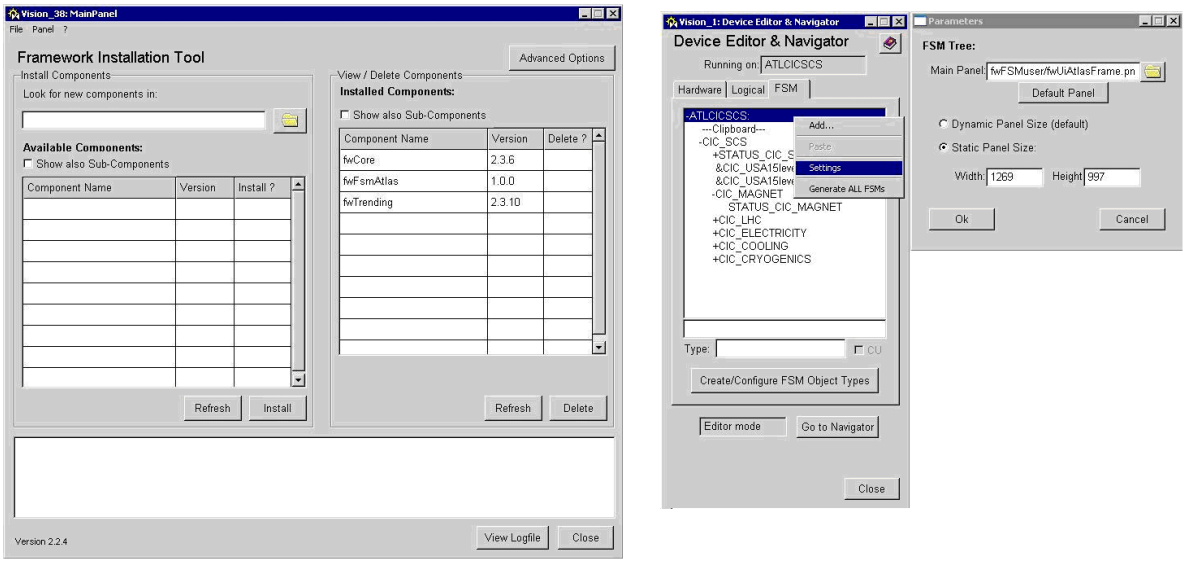


Figure A.13: Left: Framework Installation Tool. Right: FwFsmAtlas, initial settings for the DCSOI.

Appendix B

Control Hierarchies - UML Static Diagrams

UML provides us with a formal method for designing concurrent, distributed, and real-time control applications [111]. For our purposes, a formal method consists of a notation used to unambiguous specify the requirements of a computer system and that support the proof of properties of this specification and proofs of correctness of an eventual implementation with respect to the specification [112].

There are other tools for modeling distributed 'agent-based' control systems following different approaches as shown in Table B.1. However, UML has been chosen for the modeling of the different control hierarchies in this thesis for its simplicity and suited application in the context of object-oriented software projects. This appendix shows the UML Static Diagrams corresponding to the actual hierarchical organization of the different sub-detectors.

UML is an industry standard providing a standardized notation for describing object-oriented models. UML has twelve different diagram types divided into three classes: structural diagrams, behavior diagrams, and model management diagrams. With the combination of the UML Static diagram and UML State diagram the overall ATLAS DCS control hierarchy can be described. On the one hand, the different types of classes and its location within the control hierarchy are specified in the static diagram. On the other hand, the behavior of each object (i.e. states, transitions, actions, and rules) can be described using state diagrams. Furthermore, the new UML 2 has the possibility of defining nested state diagrams, and subsequently, a hierarchy of finite state machines can be described.

Approach	Method
Process algebra	Communicating Sequential Processes (CSC), Calculus of Communicating Systems (CCS), π-Calculus or Input/Output Automata
Model-Oriented	Z, B, Petri Nets or X-Machines (XM)
Logics	Temporal Logic, Real-Time Logic (RTL), BDI Logics, KARO Logic.
Other	Artificial Physics, Software Cost Reduction (SCR), Mathematical Analysis, Game Theory or hybrid approaches

Table B.1: Formal methods for describing distributed 'agent-based' control systems.

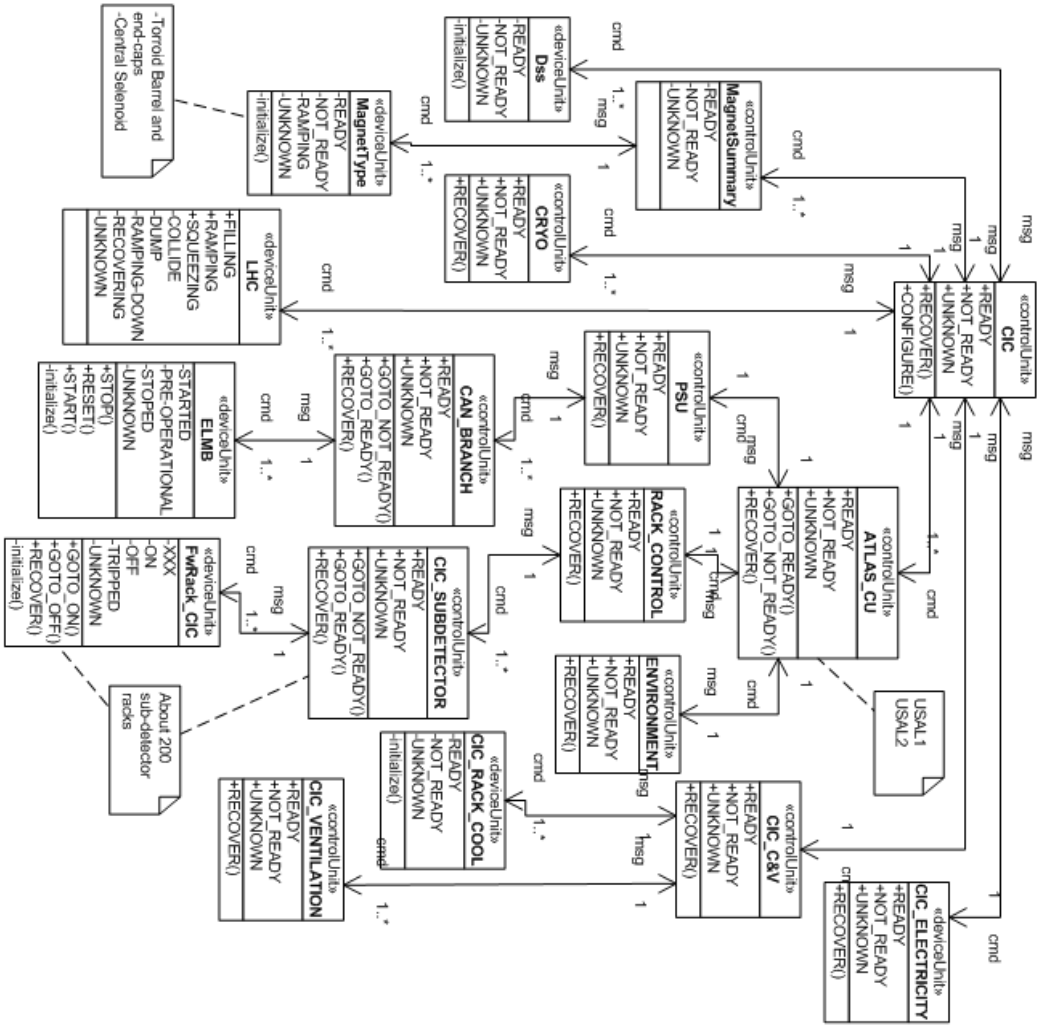


Figure B.1: UML Static Diagram of the CIC control hierarchy.

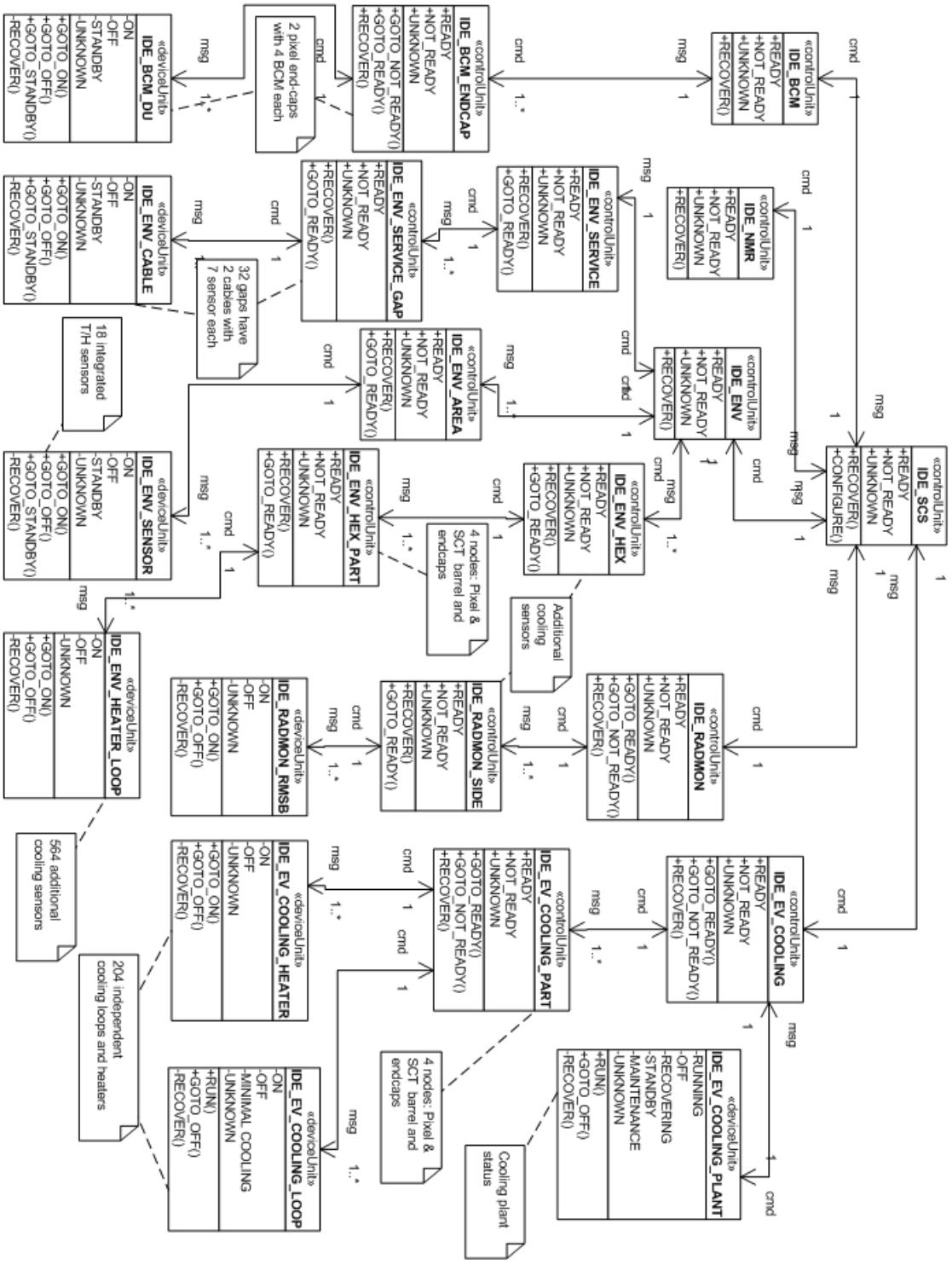


Figure B.2: UML Static Diagram of the IDE control hierarchy.

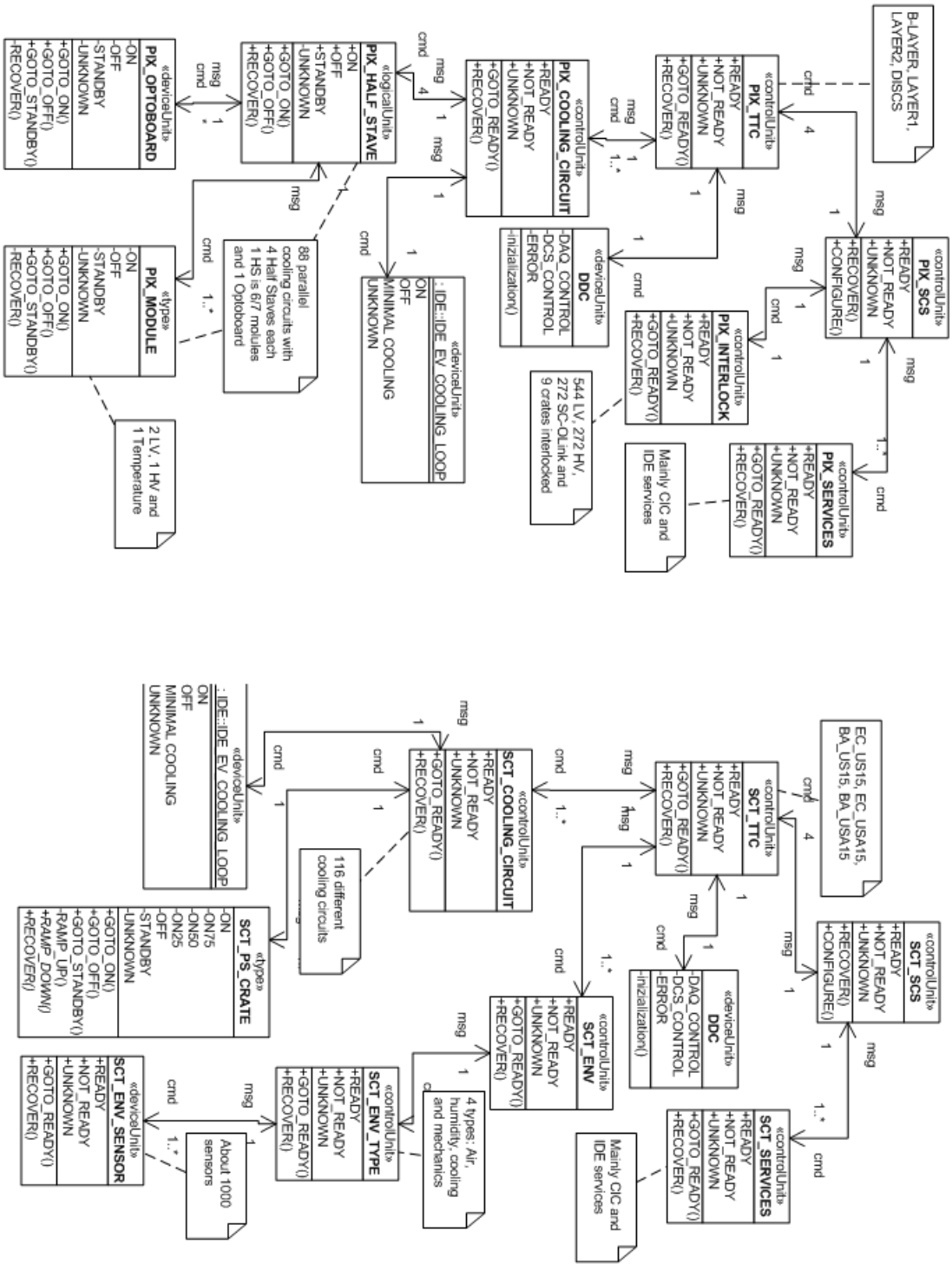


Figure B.3: UML Static Diagram of the PIX and SCT control hierarchies.

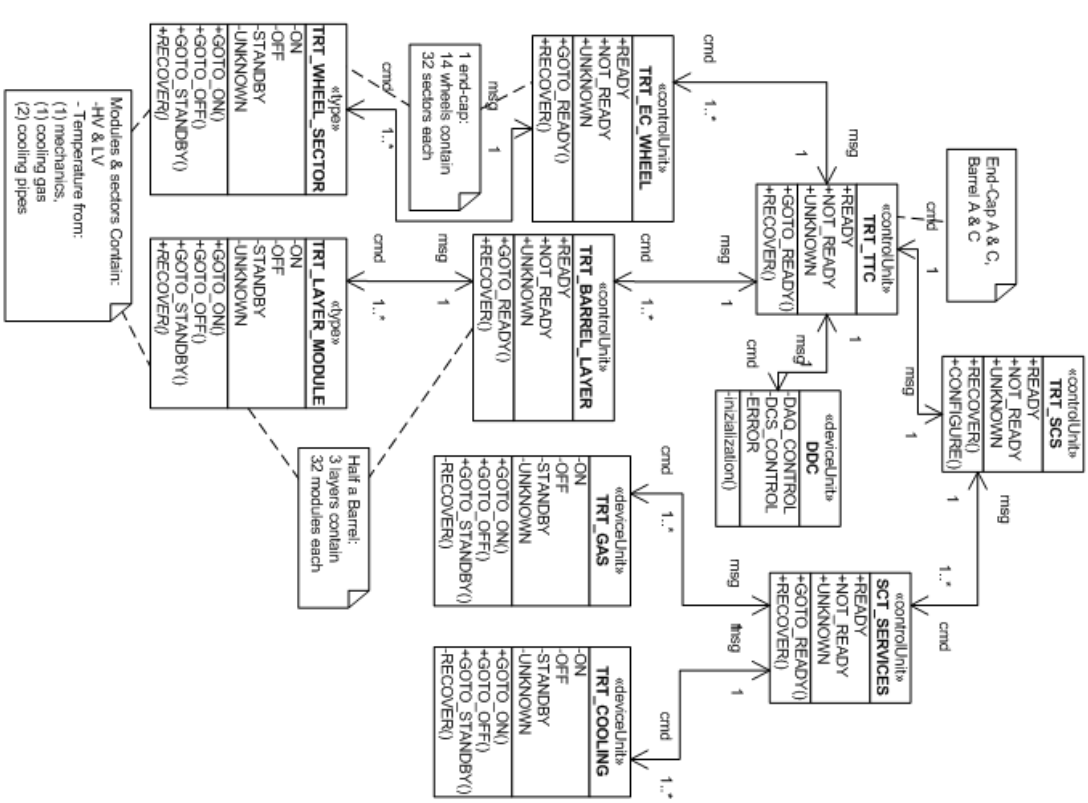


Figure B.4: UML Static Diagram of the TRT control hierarchy.

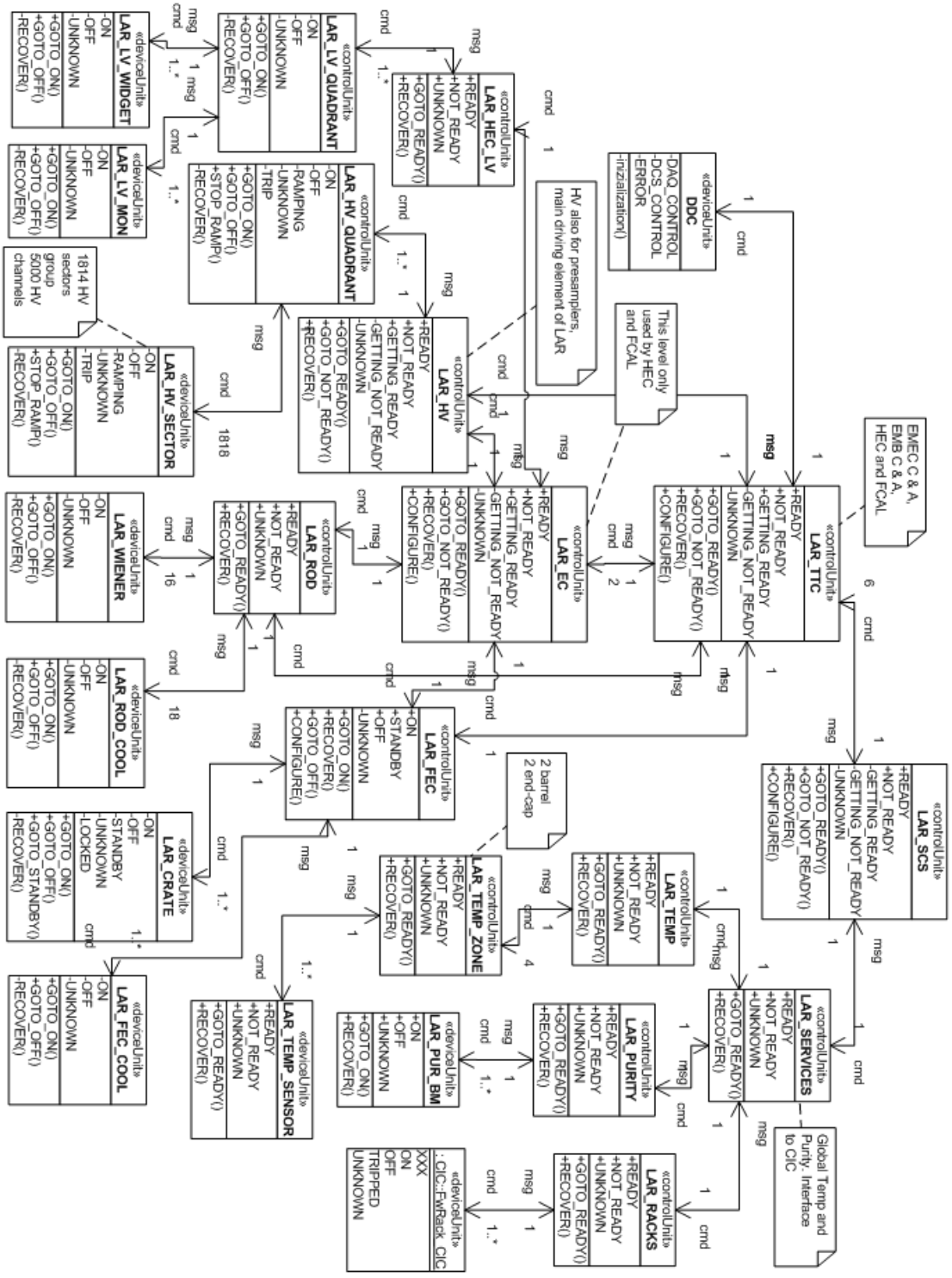


Figure B.5: UML Static Diagram of the LAR control hierarchy.

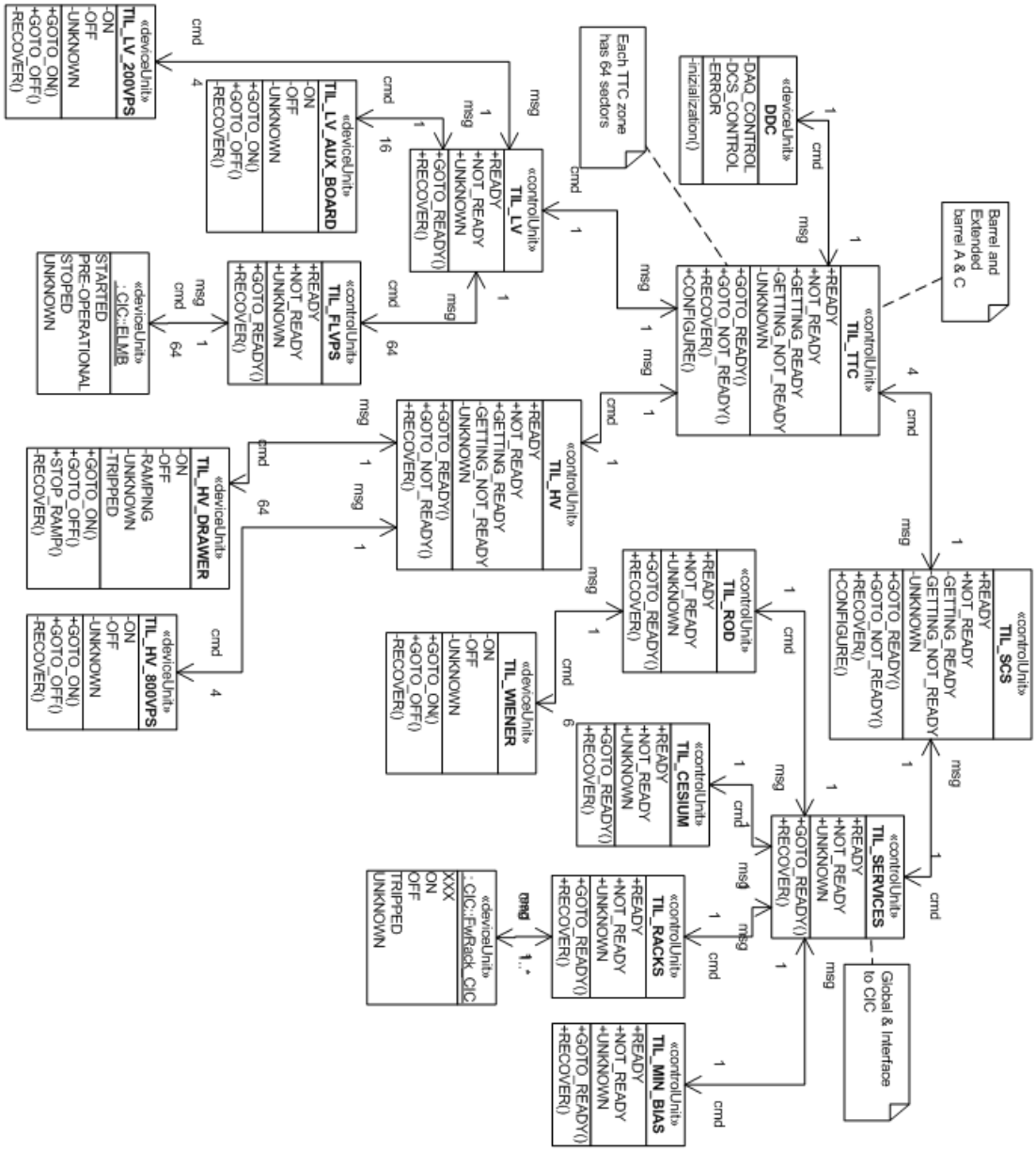


Figure B.6: UML Static Diagram of the TIL control hierarchy.

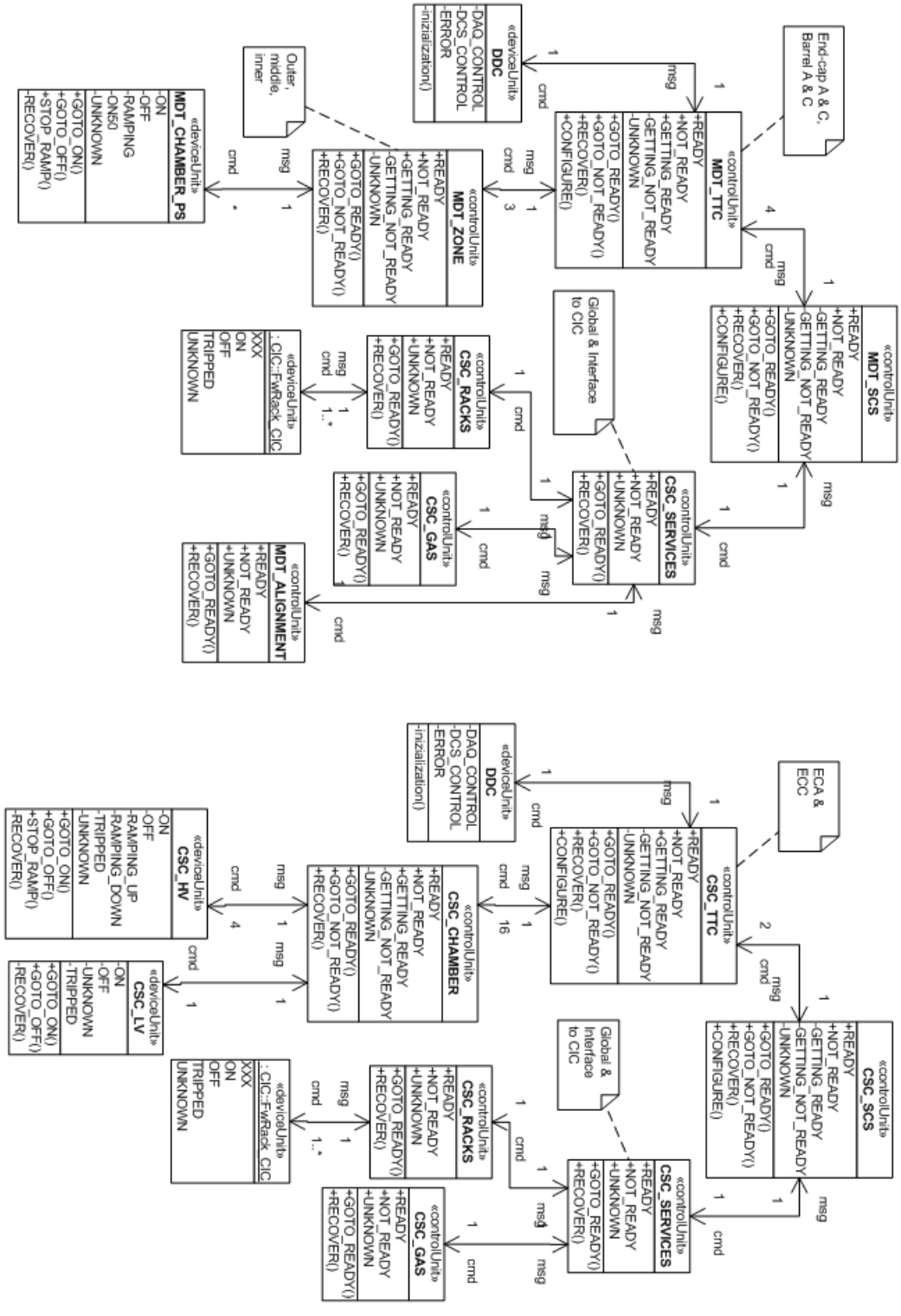


Figure B.7: UML Static Diagram of the CSC and MDT control hierarchies.

Appendix C

Partial DCS Set-up for Operations

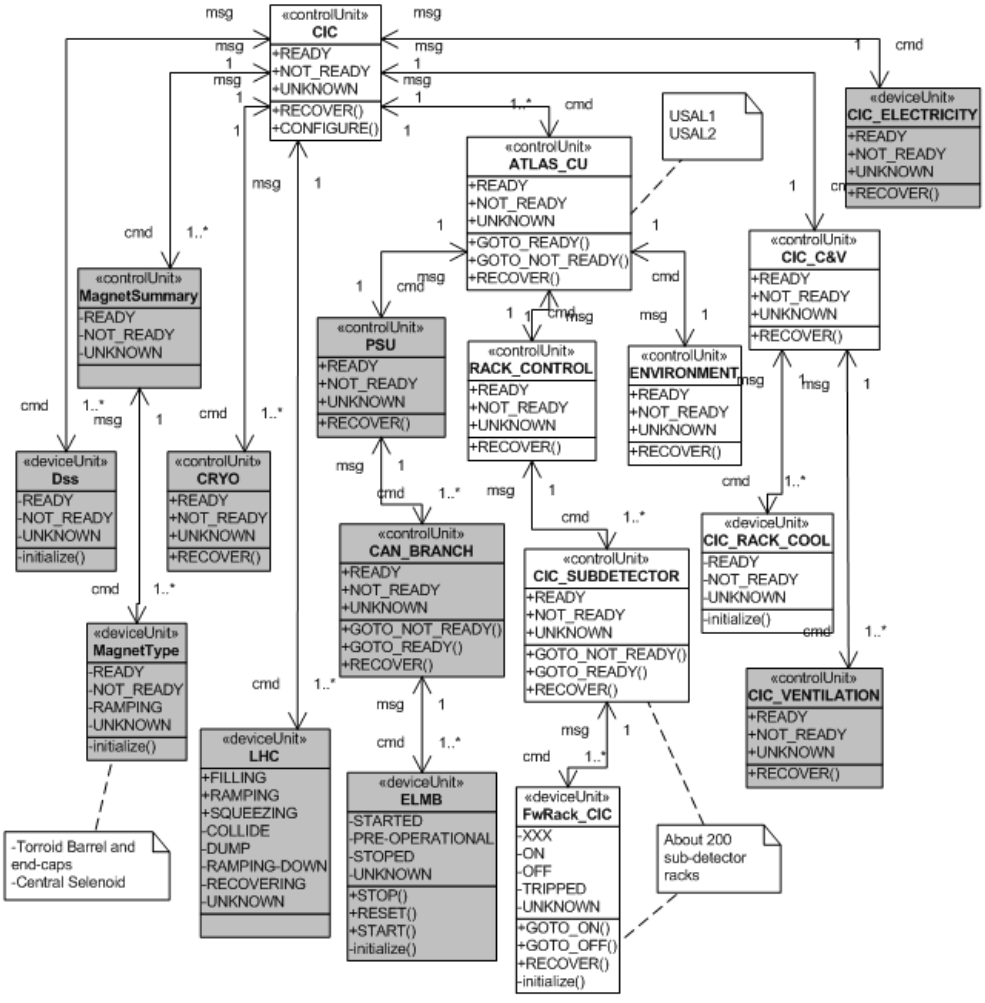


Figure C.1: In blank it is show the control hierarchy belonging to the CIC implemented and included within the DCSOI. At that time the control hierarchies and interfaces from the sub-detectors were being built in parallel.

Figure C.2: Prototype of ACR in operation since April 2006. Various screens where the DCS top level interface is running. At that time however, the control was 'taken' from underground, where much of the work was being carried out.

Figure C.3: Temporal 'control room' at the electronics room USA15L1. In operation from February 2006. From this station, the CIC could be operated partially.

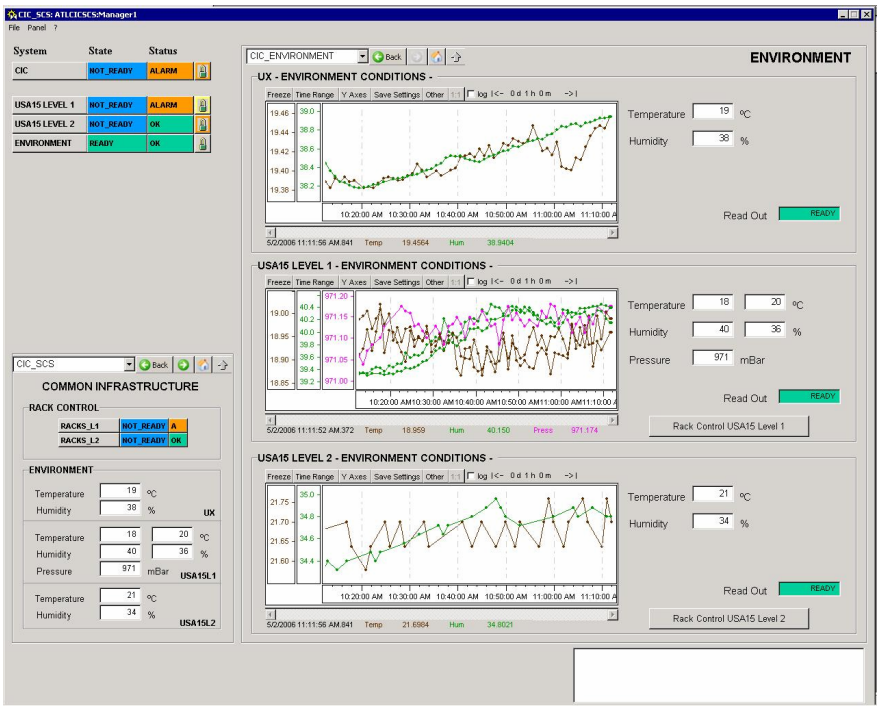


Figure C.4: Main Module: Environmental conditions USAL1, USAL2 and UX cavern. Secondary Module: General conditions. FSM module: fixed general view.

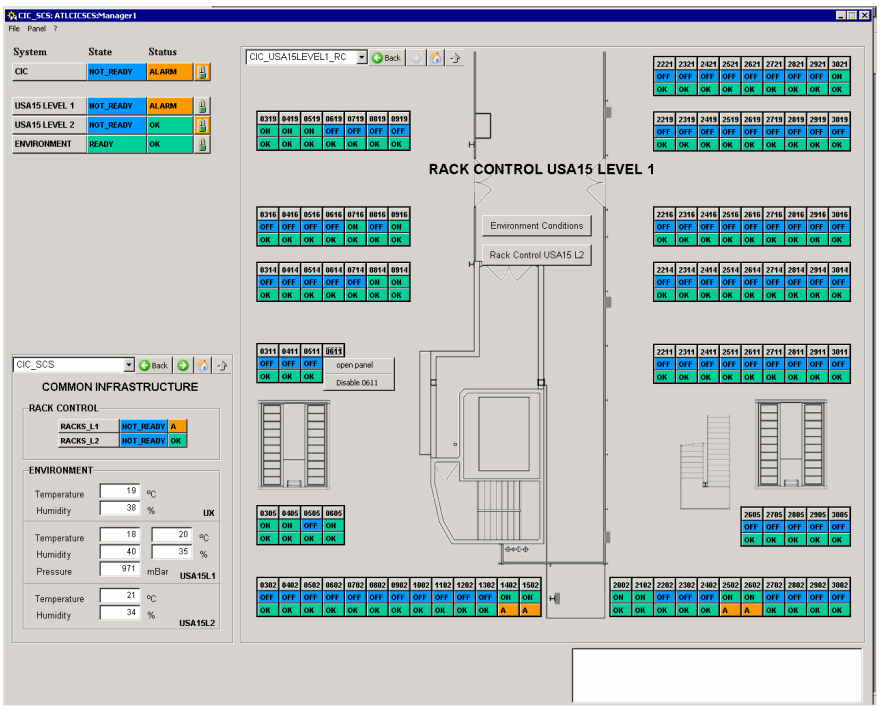


Figure C.5: Main Module: Rack control USA15L1. It is possible to command single racks or groups of racks belonging to a certain sub-detector. Access to relevant workspaces is possible from the same module (i.e. environmental conditions or rack control USA15L2). Secondary Module: General conditions. FSM module: fixed general view.

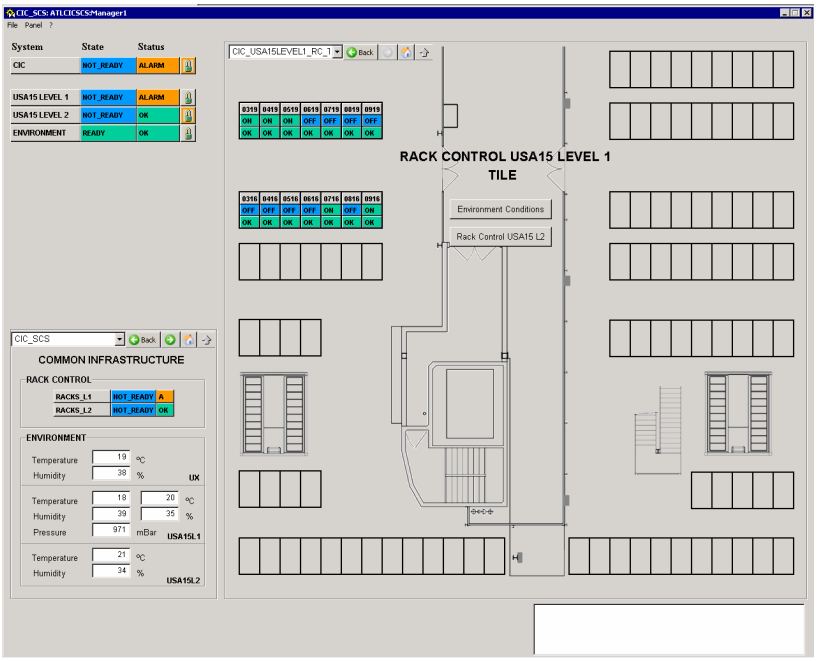


Figure C.6: Main Module: TIL sub-detector Rack control USA15L1. We are one level down now. It is possible to command single racks all together. Access to relevant workspaces is possible from the same module (i.e. environmental conditions or rack control USA15L2). Secondary Module: General conditions. FSM module: fixed general view.

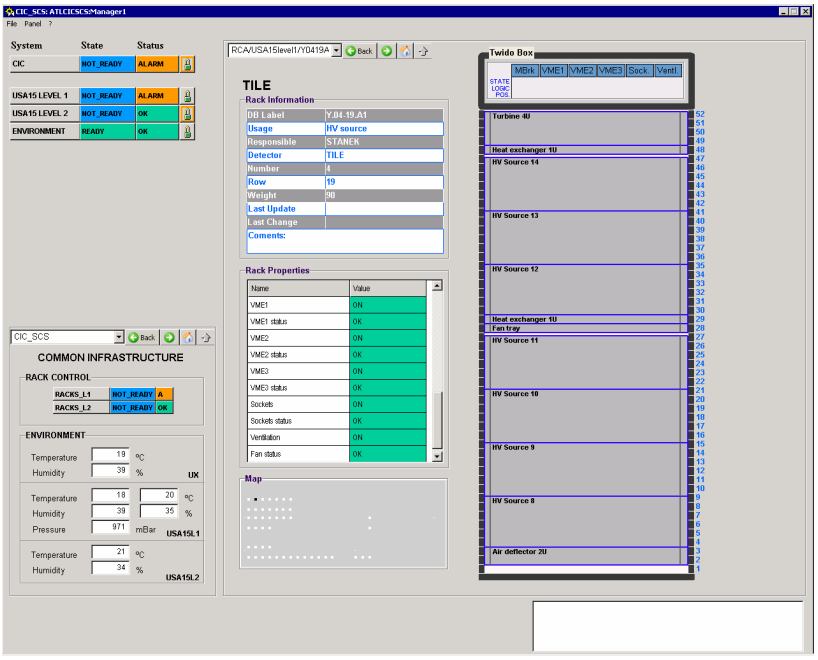


Figure C.7: Main Module: TIL sub-detector rack. We are at the lowest level now. It is possible to command this single racks or see very specific data. Secondary Module: General conditions. FSM module: fixed general view.

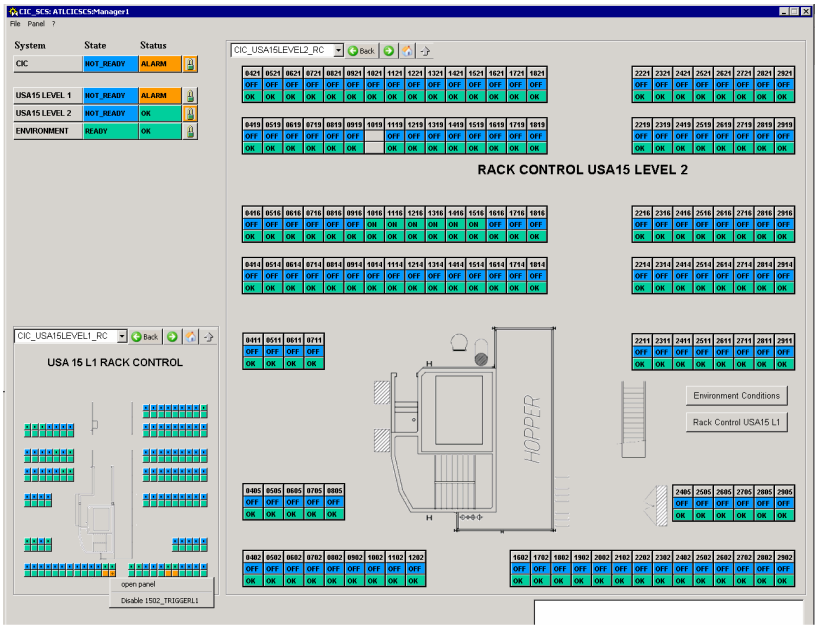


Figure C.8: Main Module: We have move to rack control USA15L2. It is possible to command single racks or groups of racks belonging to a certain sub-detector. Access to relevant workspaces is possible from the same module (i.e. environmental conditions or rack control USA15L1). Secondary Module: We have move to rack control USA15L1. The same operations available in the main module are also here. FSM module: fixed general view.

Bibliography

[1] A. Barriuso-Poy, H. Burckhart, L. Carminati, J. Cook, V. Filimonov, V. Khomutnikov, Y. Ryabov, and F. Varela. Hierarchical Control for the ATLAS Experiment. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 2005.

[2] A. Barriuso-Poy, H. Burckhart, J. Cook, V. Filimonov, V. Khomutnikov, F. Varela, S. Schlenker, and S. Franz. ATLAS DCS Integration Guidelines. Technical Report ATL/DQ/ON/0013, CERN, 2006.

[3] A. Barriuso-Poy, J.E. Sloper, V. Khomutnikov, and G. Lehmann. Operation of the ATLAS Experiment: Organization of the Detector Controls and the Data Acquisition System. *IEEE International Conference in Idustrial Electronics, IECON*, 2006.

[4] A. Barriuso-Poy, V. Khomoutnikov, and H. Burckhart. Subdetector Control Interaction with TDAQ Controls. Technical Report ATL-DQ-EN-0027, CERN, 2006.

[5] A. Barriuso-Poy. The Top Level Human-Machine Interface of the ATLAS Detector Controls. *Seminario Anual de Automática, Electrónica Industrial e Instrumentación, SAAEI*, 2006.

[6] A. Barriuso-Poy. FSM Integration Guidelines. Technical Report ATL/DQ/ON/0010, CERN, 2005.

[7] A. Vergara Fernandez. *Reliability of the Quench Protection System for the LHC Superconducting Elements*. PhD thesis, CERN & Universitat Politècnica de Catalunya, 2003.

[8] LEP accelerator. http://lepewwg.web.cern.ch/LEPEWWG/plots/winter2005/.

[9] ATLAS Collaboration. ATLAS Technical Proposal for a General-Purpose pp Experiment at the Large Hadron Collider at CERN. Technical Report CERN/LHCC/94-43, LHCC/P2, CERN, 1994.

[10] CMS Collaboration. CMS Technical Proposal. Technical Report CERN/LHCC 94-43, CERN, 1994.

[11] ALICE Collaboration. ALICE Technical Proposal. Technical Report CERN/LHCC 95-71, CERN, 1995.

[12] LHCb Collaboration. LHCb Technical Proposal. Technical Report CERN/LHCC 98-4, CERN, 1998.

[13] From CERN Photolibrary. http://www.cern.ch/.

[14] ATLAS Counting Rooms. http://atlas-crm.web.cern.ch/atlas-crm/.

[15] ATLAS Collaboration. ATLAS Technical Co-ordination. Technical Report CERN/LHC /99-01, CERN, 1999.

[16] ATLAS Collaboration. ATLAS Inner Detector Technical Design Report. Technical Report CERN/LHCC 97-16, CERN, 1997.

[17] ATLAS Collaboration. ATLAS Inner Detector Technical Design Report. Technical Report CERN/LHCC 97-17, CERN, 1997.

[18] ATLAS Collaboration. ATLAS Liquid Argon Calorimeter Technical Design Report. Technical Report CERN/LHCC 96-41, CERN, 1996.

[19] ATLAS Collaboration. ATLAS Tile Calorimeter Technical Design Report. Technical Report CERN/LHCC 96-41, CERN, 1996.

[20] ATLAS Collaboration. ATLAS Muon Spectrometer Technical Design Report. Technical Report CERN/LHCC 97-22, CERN, 1997.

[21] ATLAS Collaboration. ATLAS Magnet System Technical Design Report. Technical Report CERN/LHCC 97-18, CERN, 1997.

[22] S.R. Leo. *Techniques for Nuclear and Particle Physics Experiments. A how-to approach*. Springer-Verlag, 1994.

[23] M. Olcese and M. Stodulski. Inner Detector Evaporative Cooling System - Requirements and general design issues. Technical Report ATL-IC-ES-0006, CERN, 2005.

[24] M. Imhauser, T. Henß, S. Kersten, and Joachim Schultes. Design and Implementation of the ATLAS Detector Control System. *Preprint submitted to Elsevier Science*, 2006.

[25] H. Sandaker. *ATLAS SemiConductor Tracker Development and Physics Simulation*. PhD thesis, University of Oslo, 2005.

[26] A. Sfyrla and P. Ferrari et al. The Detector Control System for the ATLAS Semiconductor Tracker Assembly Phase. *IEEE Transactions on Nuclear Science, Vol. 52*, 2005.

[27] Z. Hajduk. TRT Detector Control System - Specifications and Requirements. Technical Report ATL-IT-ES-0009, CERN, 1999.

[28] R.K. Bock and A. Vasilescu. *The Particle Detector BriefBook*. Springer-Verlag, 1998.

[29] C. Joram. Particle Detectors, 2003.

[30] ATLAS Collaboration. ATLAS DAQ, EF, LVL2 and DCS Technical Proposal. Technical Report CERN/LHCC/2000-17, CERN, 2000.

[31] F. Varela, H. Burckhart, J. Cook, V. Filimonov, V. Khomoutnikov, B. Hallgren, and H.Boterenbrood. First experiences with the ATLAS PixelDetector Control System at the Combined Test Beam2004. *IEEE Real Time Conference*, 2003.

[32] ATLAS Collaboration. ATLAS High-Level Trigger Data Acquisition and Controls. Technical Report CERN/LHCC/03-22, CERN, 2003.

[33] P. Burkimsher and H. Milcent. Applying Industrial Solutions to the Control of HEP Experiments. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 1999.

[34] L3 Muon control software. http://itcowww.cern.ch/L3-CTRL/muon.html.

[35] NA48 experiment. http://itcowww.cern.ch/NA48/welcome.html.

[36] D.R. Myers. The LHC Experiments' Joint COntrol Project, JCOP. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 1999.

[37] C. Gaspar and M. Dönszelmann. DIM, A Distributed Information Management System for the DELPHI experiment. *Real Time on Computer Applications in Nuclear, Particle and Plasma Physics*, 1993.

[38] S.A. Lewis. Overview of the Experimental Physics and Industrial Control (EPICS). Technical report, Lawrence Berkeley National Laboratory, 2000.

[39] A. Daneels and W. Salter. Selection and Evaluation of Commercial SCADA systems for the Controls of the CERN LHC Experiments. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 1999.

[40] C. Mazza et al. *Software Engineering Guides*. Prentice-Hall, 1995.

[41] H.J. Burckhart. Detector Control System. *Workshop on Electronics for LHC Experiments*, 1998.

[42] H.J. Burckhart. Communication between Trigger/DAQ and DCS. *International Conference on Computing in High Energy and Nuclear Physics*, 2001.

[43] H.J. Burckhart, J. Cook, F. Varela, B. Varnai, V. Filimonov, V. Khomoutnikov, and Y. Ryabov. The Supervisor of the ATLAS Detector Control System. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 2003.

[44] M. Dentan. ATLAS Policy on radiation tolerant electronics. Technical Report ATC-TE-QA-0001, CERN, 2000.

[45] B. Hallgren, H.Boterenbrood, H. Burckhart, and H.Kvedalen. The Embedded Local Monitor Board (ELMB) in the LHC Front-end I/O Control System. *International Conference on Accelerator and Large Experimental Physics*, 1999.

[46] Kvaser Advanced CAN Solutions. http://www.kvaser.com/.

[47] P. Farthouat. ATLAS Electronics System Overview. Technical report, CERN, 2002.

[48] CAEN, Nuclear Physics. http://www.caen.it/nuclear/index.php.

[49] Wiener, Plein & Baus Ltd. http://www.wiener-d.com/.

[50] Iseg, High Voltage. http://www.iseg-hv.de/start.php/lang.en/.

[51] J. Pedersen. The Use of UPS in the CERN Power Distribution. *CERN Technical Services Internal Report*, 2003.

[52] Fieldbus Foundation. 9390 Research Boulevard, Suite II-250, Austin, Texas 78759, USA.

[53] G. Baribaud et al. Recommendations for the use of fieldbuses at CERN. Technical Report CERN-ECP/96-11, CERN, 1996.

[54] PROFIBUS. e.V. Nutzerorganisation , Haid-und-Neu-Strate 7, D-76131, Karlsuhe, Germany.

[55] WORLDFIP. 3 bis, rue de la Salptrire F-54000, Nancy, France.

[56] CAN bus, CAN in Automation. e.V.International Headquarters Am Weichselgarten 26, D-91058, Erlangen, Germany.

[57] G. Gruhler and B. Dreier. *CANopen Implementation Guidelines*. STA Reutlingen, 1997.

[58] OLE for Process Control. http://www.opcfoundation.org/.

[59] J. Lange. Ten years of OPC: From Data Access to Unified Architecture, 2004.

[60] T. Adye et al. The Slow Controls of the DELPHI Experiment at LEP. *Proceedings International Conference on Computing in High Energy Physics*, 1992.

[61] C. Gaspar. *Methods and Tools for the Design and Implementation of Control Systems for Large Physics Experiemnts*. PhD thesis, Institut National des Sciences Appliquees de Lyon, 1998.

[62] http://itco.web.cern.ch/itco/Projects-Services/JCOP/welcome.html. IT/CO Collaboration.

[63] PVSS-II. http://www.pvss.com.

[64] M. Gonzalez-Berges. The Joint COntrols Project Framework. *Computing in High Energy and Nuclear Physics*, 2003.

[65] By staff. Process Automation 2004. *Control Engineering Europe*, 2004.

[66] E. Svensson and C. Vetter and T. Werner. Data Consistency in a Heterogeneous IT Landscape: A Service Oriented Architecture Approach, 2004.

[67] E. Svensson and C. Vetter and T. Werner. Integrating Heterogeneous IT Systems: A Standards Based Approach using SOAP, 2005.

[68] G.A. Fodor. The Case for Hierarchy-Mobile Agents in Industrial Informatics. *IEEE Industrial Informatics Conference 2004*, 2005.

[69] H. Kirrmann. Lectures in Industrial Automation. Technical report, École Polytechnique Fédérale de Lausanne, 2006.

[70] Aspect-Object Technology from ABB. www.abb.com, 2004.

[71] Industry White Paper. Making Sense of e-Manufacturing: A Road-Map for Manufacturers, 2000.

[72] Schneider Electric. Transparent Technology solutions Ethernet and Web based technologies for the factory floor, http://www.transparent-ready.com/index.htm.

[73] Simatic IT Production Suite, Siemens. www.ad.siemens.de/meta/index_76.htm.

[74] E. F. Moore. Gedanken-experiments on Sequential Machines. *Automata Studies, Annals of Mathematical Studies, no. 34, Princeton University Press*, 1956.

[75] G. H. Mealy. A Method for Synthesizing Sequential Circuits. *Bell System Technology Journal vol 34*, 1955.

[76] Object Management Group - UML. http://www.uml.org/.

[77] H.E. Eriksson and M. Penker. *UML Toolkit*. Wiley Computer Publishing, 1998.

[78] J. Barlow, B. Franek, M. Jonker, T. Nguyen, P. Vande-Vyvre, and A. Vascotto. Run Control in MODEL: The State Manager. *IEEE Trans. Nucl. Sci., vol. 36, pp. 1549-1553*, 1989.

[79] B. Franek and C. Gaspar. *SMI++, Object Oriented Framework for Designing and Implementing Distributed Control Systems*. SLAC, 2006.

[80] J. Papenfort. Centralised vs. Distributed Control. *Control Engineering Europe*, 2005.

[81] A. Barriuso-Poy. Intelligent agents based telecommunication applications. Master's thesis, University of Warwick and University Rovira i Virgili, 2001.

[82] N. Radjou. Software Agents in Business: Still an Experiment. *Forrester Research*, 2003.

[83] N. Radjou. Is It Prime Time For Agents In Business? *Agent Technology Conference: Agents for Commercial Applications*, 2003.

[84] Y. Power and P. Bahri. *Integration techniques in intelligent operational management: a review*. Knowledge-Based Systems. Elsevier, 2005.

[85] P.J. Lucas. *An Object-Oriented Language System for Implementing Concurrent, Hierarchical, Finite State Machines*. PhD thesis, University of Illinois, 1993.

[86] P. Croll., P.Y. Duval, P. Jones, S. Kolos, R.F. Sari, and S. Wheeler. Modelling the Dynamic Behaviour of the ATLAS DAQ Prototype-1 Run Control using Statecharts. *IEEE Real Time Conference*, 1997.

[87] The BABAR detector. Nuclear Instrumentation Methods.

[88] ALEPH Collaboration. *Aleph Handbook, Volume 2, chapter 10*. CERN, 1997.

[89] R. Langlois. *Modularity in Technology and Organization*. Research Papers Network Institutional Theory, 2000.

[90] C. Baldwin and K. Clark. *Managing in an Age of Modularity*. Harvard Business Review, 1997.

[91] R. Rothrock and V. Martz. Good Screen Design. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 1999.

[92] V. Khomoutnikov. ATLAS DAQ-DCS Communication Software: Users Guide. Technical Report ATL-DQ-ON-0008, CERN, 2005.

[93] A tool for building expert systems CLIPS. http://www.ghg.net/clips/CLIPS.htm.

[94] G. Liko et al. Control in the ATLAS TDAQ system. *CHEP*, 2004.

[95] ATLAS Collaboration. ATLAS: Detector and Physics Performance Technical Design Report. Technical Report CERN/LHC/99-14, TDR 14 and 15, CERN, 1999.

[96] F. Varela Rodríguez and F. Calheiros. FSM - Configuration Database Interface, http://itcobe.web.cern.ch/itcobe/Projects/Framework/Download/Components/FSMConfDB/welcome.html.

[97] D. Lissauer, L. Mapelli, M. Nessi, and S. Stapnes. The ATLAS Control Room, Operation Model - draft 1. *ATLAS overview week*, 2006.

[98] R. Bailey. How can operations get the applications software that they want? *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 2005.

[99] E. McCrory, J.F. Ostiguy, and S. Mishra. Who Gets to Specify the Control System? *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 2005.

[100] K.J. Vicente. Operator Adaptation in Process Control: A Three-Year Research Program. *Control Engineering Practice, Vol. 5, No. 3*, 1997.

[101] M. Tanaka. Feedback of Operators' Experiences to Console Programs in the KEK e^-/e^+ Linac. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 1997.

[102] D. J. Ciarlette and R. Gerig. Operational Experience from a Large EPICS-Based Accelerator Facility. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 1993.

[103] J. Cuperus, F. Di Maio, and C.H. Sicard. The Operator Interface to the Equipment of the CERN PS Accelerators. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 1993.

[104] A. Miller and M. Joyce. Accelerator Operators and Software Development. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 2001.

[105] M. Lamont. What constitutes a good control system for operations? *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 1993.

[106] P. Czarapata et al. How to commission, operate and maintain a large future accelerator complex from far remotes sites. *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS*, 2001.

[107] http://atlas-dcs.web.cern.ch/atlas dcs/. ATLAS DCS monitoring web page.

[108] J. Vicente and J. Rasmussen. Ecological Interface Design: Theoretical Foundations. *IEEE Transactions on systems, man, and cybernetics*, 1992.

[109] J. Vicente and J. Rasmussen. Ecological interface design for petrochemical applications: supporting operator adaptation, continuous learning, and distributed, collaborative work. *Computers and Chemical Engineering 25*, 2001.

[110] J. Raskin. *The Humane Interface*. Addison-Wesley, 2000.

[111] H. Gomaa. *Designing Concurrent, Distributed, and Real-Time Applications with UML*. Addison-Wesley, 2000.

[112] C.A. Rouff, M. Hinchey, J. Rash, W. Truszkowski, and D. Gordon-Spears. *Agent Technology from a Formal Perspective*. NASA Monographs in Systems and Software Engineering. Springer, 2006.

List of Figures

List of Tables