Software Technologies For A Prototype ATLAS DAQ

D. Burckhart, R. Jones, L. Mapelli [1], M. Michelotto, A. Patel [2], M. Skiadelli [3], I. Soloviev [4]

CERN, Geneva, Switzerland

P-Y. Duval, A. Le Van Suu, R. Nacasch, Z. Qian, F. Touchard

Centre de Physique des Particules de Marseille, IN2P3, France

S. Kolos

Petersburg Nuclear Physics Institute, St. Petersburg, Russia

K. Nurdan

Bogazici Univ., Istanbul, Turkey

S. Wheeler

Sheffield Univ., Sheffield, UK

M. Caprini

Institute of Atomic Physics, Bucharest, Romania

The ATLAS collaboration has defined a set of user requirements for the back-end software subsystem within the context of the data acquisition and event filter prototype "-1" project. Based on these requirements, a number of evaluations have been performed on candidate technologies and techniques in the areas of configuration data storage (Objectivity ODBMS; Rogue Wave Tools.h++ for C++ object persistence), inter-process communication (Corba; MPI), dynamic object behaviour (Harel StateChart generator), graphical user interfaces (cross- platform GUI builder; Java AWT) and software integration (ACE operating-system interface). This paper describes the important requirements which lead to the selection of these technologies, the results obtained from the evaluations and how we intend to apply them to the design and implementation phases of the project.

Preprint submitted to Elsevier Preprint *27 May 1997*

1 Introduction

The goal of the ATLAS DAQ and Event Filter prototype "-1" project [1] is to produce a prototype system representing a "full slice" of a DAQ suitable for evaluating candidate technologies and architectures for the final ATLAS DAQ system on the LHC accelerator at CERN. Within the prototype project, the back-end software is responsible for configuration, control, and monitoring of the DAQ but excludes the management, processing and transportation of physics data. A set of user requirements for the back-end software [2] has been defined that identifies a number of components (e.g. run control, message reporting system, etc.). It became clear that these components will all require a mixture of facilities for data storage, inter-process communication in a LAN network of processors, graphical user interfaces, complex logic-handling and general operating system services. To avoid unnecessary duplication, we decided to use the same facilities across all components and surveyed freeware and commercial software packages to select suitable technologies. Such facilities must be portable (the prototype DAQ will include processors running Solaris, HPUX and WNT). In particular they must be available on the LynxOS [3] real-time UNIX operating system selected for use on the front-end processors. In parallel to these evaluations, investigations have been carried out into the suitability of the StP CASE tool as a software development environment [4]. Part of the work involved the development of scripts to generate definition language directly from diagrams for a number of the facilities described in this paper.

2 Configuration data storage

The group has experience with relational and object databases as well as in-memory systems[5], but since no single commercial or shareware system satisfied all the documented user requirements for a persistent data manager it was decided to adopt a two-tier architecture consisting of a light-weight in-memory persistent object manager to support the real-time requirements and a full ODBMS as a back-up and for long-term data management.

---
[1] Spokesperson.
[2] On leave from School of Computing and Information Systems, University of Sunderland, UK
[3] Now at SYBASE Relational Technology, Athens, Greece
[4] On leave from the Petersburg Nuclear Physics Institute, St. Petersburg, Russia

2.1 *Tools.h++ C++ class library persistence*

Rogue Wave Tools.h++[6] is a C++ foundation class library available on many operating systems (Unix, MS Windows, WNT, OS/2) that has become an industrial standard and is distributed by a wide variety of compiler vendors. It has proven to be a robust and portable library that can be used for DAQ programming since it supports many useful classes and can be used in a multi-threaded environment. We have purchased the sources for the library (at a reasonable price) and ported it to LynxOS.
Tools.h++ persistence is sufficient to support a light-weight in-memory object manager for the DAQ configuration databases and, as an example, the RD13 DAQ [7] Run Control and Detector Parameters Database [8] has been reimplemented using Tools.h++ persistence. However, several requirements for the DAQ configuration databases are not met by Tools.h++ alone: schema evolution is not supported and there is no graphical means to manipulate (i.e. create, browse, update) persistent data. To overcome these limitations, we have developed a package on top of the Tools.h++ library called OKS[9]. OKS is based on an object model that supports objects, classes, associations, methods, data abstraction, inheritance, polymorphism, object identifiers, composite objects, integrity constraints, data and schema versioning and active notification. The OKS system stores database schema and data in portable ASCII files and allows several schema and data files to be merged into a single database. It includes Motif based GUI applications to design database schema and to manipulate OKS objects (Figure 1.).

2.2 *Objectivity database system*

Objectivity/DB is a commercial Object Oriented Database management system introduced to CERN by the RD45 project[10]. We evaluated [11] the basic DBMS features (schema evolution, access control, versioning, back-up/restore facilities etc.) and the C++ programming interface as well as the Classify/DB CASE tool. In order to evaluate the various features, we ported the Tools.h++ Run Control and Detector Parameters Database mentioned above to Objectivity. We found the beta-release of Classify/DB, while interesting for schema design and reverseengineering, not to be very robust and hence of limited use. Objectivity satisfied most of our requirements but we decided to wait for version 4.0 (which includes more sophisticated schema evolution and data migration facilities) became available before integrating it fully into our system.
An issue not yet addressed, is the movement of data between Objectivity and the OKS in-memory system but we are confident that an automated solution can be found.

3 Inter-process communication

Message passing in our distributed environment is a topic of major importance since it is the communication backbone between the many processes running on the different machines of the DAQ system. Reliability and error recovery are very strong requisites as well as the ability to work in an event driven environment (such as X11). Many components of the back-end software will require a transparent means of communicating between objects independent of their location (i.e. between objects inside the same process, in different processes or on different machines). It is hoped that a single package can form the basis of all such communication.

3.1 ILU

We chose to evaluate the Object Management Group's [12] Common Object Request Broker Architecture (CORBA) standard. ILU [13] is a freeware implementation of CORBA by Parc Xerox. The object interfaces provided by ILU hide implementation distinctions between different languages, between different address spaces, and between operating system types. ILU can be used to build multi-lingual object-oriented libraries with wellspecified language-independent interfaces. It can also be used to implement distributed systems and to define and document interfaces between the modules of non-distributed programs. ILU interfaces can be specified either in the OMG's CORBA Interface Definition Language (OMG IDL) or ILU's Interface Specification Language (ISL).
We have developed client-server examples [14] with characteristics similar to those required for the back-end components. To test integration issues, we added an ILU based communication link between a Motif GUI and the Tools.h++ implementation of the Run Control and Detector Parameters Database mentioned above. ILU has proved to be very flexible and easy-to-use. The run-time libraries have been ported to LynxOS.

3.2 MPI

As an alternative communication system, we chose to evaluate MPI [15]. It provides point-to-point as well as collective communications both in a C and Fortran 77 environment. It is supported by the academic community as well as by major constructors (HP-Convex, Cray, IBM, Intel, etc.).
We tested [16] two freeware implementations: MPICH [17] and LAM [18]. MPICH is built on two layers. The upper layer provides the application program interface while the lower layer can be tailored to exploit the hardware

architecture, thus optimizing performance. LAM is used to build a cluster of heterogeneous workstations communicating via a UNIX daemon running on every machine. We found MPICH to be significantly faster than LAM. However, error handling is not sufficient and there is no support for multi-threaded applications in either implementation. Dynamic process management, as implemented by LAM, works satisfactorily but, to our knowledge, no function is available to deal with X11 events while waiting for messages.
Although a new version of the standard [19] is in preparation no thread-safe policy is defined and expected realtime extensions are still at the level of discussions.We therefore prefer to use CORBA for our applications but will continue to monitor the evolution of the MPI standard.

4 Dynamic object behaviour

Many applications within the ATLAS DAQ prototype have complicated dynamic behaviour which can be successfully modelled in terms of states and transitions between them. Previously, state diagrams, implemented as finite state machines, have been used[20] which, although effective, become ungainly as system size increases. Harel statecharts address this problem by implementing additional features such as hierarchy and concurrency.
CHSM [21] is an object-oriented language system which implements Harel statecharts as Concurrent, Hierarchical, finite State Machines supporting many statechart concepts as illustrated in the abstract example shown in Fig- ure 2. (a):

- Hierarchy: states f and e are child-states of parent state q.
- Clusters (logical-exclusive-or state groups): states a, b and c are in cluster x. To be in x is to be in a, b or c. The transition to d is taken regardless of x's child-state x.
- History: when cluster x is re-entered the previous child-state is entered.
- Sets (logical-and state groups): cluster p and cluster q are child states of set s. To be in set s is to be in both child-states p and q.
- Concurrency: if event a occurs while the statechart is in child-states a and f then it will simultaneously make transitions to states e and c.
- Guard conditions: the transition from b to c only occurs if $v < 4$.
- Broadcasting: event e is broadcast when the transition from f to e is made.
- Actions: function f() is executed when the transition from x to d is made, in addition actions can be executed on entering or exiting a state (not shown).

CHSMs are described by means of a CHSM description text file and Figure 2. (b) shows the CHSM description corresponding to the statechart shown in Figure 2. (a). The CHSM compiler converts the description to C++, which may be integrated with user defined code.

We have evaluated the CHSM language system [22] and have shown it to be suitable for describing the dynamic behaviour of typical DAQ applications. For example, a prototype version of the DAQ run control CHSM has been created. Default actions and guard conditions in addition to default enter and exit functions have been defined for each transition and state respectively. This provides a satisfactory framework within which developers of the sub-components of the run control system can implement their required actions and conditions by overriding the relevant default functions using C++ inheritance. We have also found CHSM code to be portable to platforms foreseen for use in the DAQ.

5 Graphical user interfaces

Modern data acquisition systems are large and complex distributed systems that require sophisticated user interfaces to monitor and control them. X11 and Motif are the dominant technologies on UNIX workstations but the advent of WNT has forced us to reconsider this choice.

5.1 Java

Java is a simple object-oriented, platform-independent, multi-threaded, general-purpose programming environment. The aim of our evaluation [23] was to understand if Java could be used to implement the status display component of the DAQ. A demonstration application was developed to investigate such topics as, creating standard GUI components, client-server structure, use of native methods, specific widgets for status displays and remote objects. The performance was compared to X11 based alternatives. The demo contains three essential parts: servers, simulators and applets. Servers realise the binding with remote objects, simulators create simulation data and update remote objects (to mimic the DAQ), applets put simulation data to remote objects or get them from remote objects and display them. The appearance of the Java implementation of the status display is shown in Figure 3.
The entire application is written in Java (JDK1.0), communication is realised by Java IDL (alpha2). The servers and simulators run on a SUN (Solaris2.5) and the applets can be loaded from any machine using a Java compatible browser. Work is continuing on the integration of the demo status display with the ILU communication system described above.

X-Designer [24] is an interactive tool for building graphical user interfaces (GUIs) using widgets of the standard OSF/Motif toolkit as building blocks that has been used extensively in the RD13 project [25]. It is capable of generating C, C++ or Java code to implement the GUI. We investigated [26] the construction of GUIs with a Model- View-Controller (MVC) architecture [27] using the C++ code generation capabilities of the tool.
A number of widget hierarchies were created which correspond to commonly used patterns in GUIs (e.g. data entry field with a label). These also correspond to views or controller-view pairs in the MVC architecture. By making each hierarchy a C++ class it could be added to the X-Designer palette and used in subsequent designs. A GUI for the Run Control Parameter database was successfully built using these definitions.
X-Designer also provides a facility called XD/Replay which can be used to record or playback any Xt based application. In record mode it writes a high-level description of actions performed on the application in a script. On replaying the script, the recorded actions are executed on the application. This facility will be useful for automating the testing of any DAQ application with a Motif GUI and for producing automated demonstrations of applications.
Work will continue to investigate if the MVC approach and Java code generation facilities of X-Designer can be used to develop the DAQ graphical user interfaces, including the status display application described above.

6 Software integration

ATLAS DAQ applications that deal with distributed control, command and supervision functions, in both the back-end and front-end of the DAQ system, make intensive use of operating system (OS) services to manage tasks and processes, handle data and to provide inter-task communication and synchronization. An efficient way to deal with such functionality on multiple operating systems is to use a platform independent virtual OS layer between the native OS and the applications.
The ACE (Adaptive Communication Environment) OO toolkit [28] provides such a layer implemented as a C++ object-oriented interface called the OS wrapper (Figure 4.). This freeware product was developed for high-speed communication systems and is designed to be efficient from a real-time computing point of view. It includes a valuable set of high-level frameworks and application skeletons to build distributed services.
We have developed applications [29] to test ACE components which use shared memory, threads, synchronization schemes and manage system events. We have integrated ACE with ILU and the X-Window system in a single appli-

cation. A basic distributed process management system has been made and a communication server integrating ILU transactions and ACE pipes was successfully achieved.
We concluded that ACE is a useful product for developing portable applications which need to integrate several software packages and use low-level operating system resources.

7 Summary and future work

We have described the evaluation of software technologies in the areas of data storage, inter-process communication, dynamic object behaviour, graphical user interfaces and software integration and their application to data acquisition systems.
The knowledge gained from the evaluations described in this paper is being applied to the design of the components of the back-end software components of the Atlas DAQ prototype -1 project and the selected technologies will be used in the implementation of these designs.

8 References

[1] ATLAS DAQ and Event Filter prototype "-1" project, L. Mapelli et al. Published in these proceedings.

[2] ATLAS DAQ Back-end software User Requirements Document. 30 May 1996. http://atddoc.cern.ch/Atlas/ DaqSoft/document/draft_1.html

[3] Lynx RTS, LynxOS 2.3.1 for PowerPC, Development Operating System, DOC-0080

[4] Use of a CASE tool for developing ATLAS DAQ software, A.Patel et al. Published in these proceedings

[5] Software Engineering Techniques and CASE Tools in RD13, G. Ambrosini et al. Third International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics (AIHEP'93) conference, Oberammergau in October 1993. ISBN 981-02-1699-8.

[6] Tools.h++ Introduction and Reference Manual Version 6. Thomas Keffer, Rogue Wave Software Inc. 1994.

[7] L.Mapelli et al., A scalable Data Taking System at a Test Beam for LHC, CERN/DRDC 90-64, CERN/DRDC 91-23, CERN/DRDC 92-1, CERN/DRDC 93-25, CERN/LHCC 95-47. See also the World Wide Web at URL http://rd13doc.cern.ch/.

[8] Run Parameters Data Base. G. Mornacchi. RD13 Technical Note number 81, http://rd13doc.cern.ch/public/ doc/Note81/Note81.html

[9] OKS User Guide. OKS Tools Manual. OKS Reference Manual. Igor Soloviev. http://atddoc.cern.ch/Atlas/ Notes/033/Note033-1.html

[10] LCRB Status Report / RD45. LCRB 96-15. CERN.

[11] Evaluation of Objectivity OO database for storing ATLAS DAQ configuration Information. M. Skiadelli. ATLAS DAQ Prototype -1 Technical Note 9. http://atddoc.cern.ch/Atlas/Notes/009/Note009-1.html.

[12] Object Management Group. http://www.omg.org/

[13] ILU documentation. Copyright (c) 1991-1996 Xerox Corporation. ftp://ftp.parc.xerox.com/pub/ilu/ilu.html.

[14] Inter-component communication in the ATLAS DAQ back-end software (evaluation of the ILU multilanguage object interface system). S. kolos. ATLAS DAQ Prototype -1 Technical Note 3. http://atddoc.cern.ch/ Atlas/Notes/003/Note003-1.html

[15] MPI: A Message-Passing Interface Standard http://www.mcs.anl.gov/mpi/mpi-report-1.1/mpi-report.html

[16] Evaluation of the Message Passing Interface Standard for the Atlas Backend Software. F. Touchard. ATLAS DAQ Prototype -1 Technical Note 11. http://atddoc.cern.ch/Atlas/Notes/011/Note011-1.html.

[17] MPICH. Argonne National Laboratories, Illinois. http://www.mcs.anl.gov/mpi/mpich

[18] LAM. Ohio Supercomputer Center, Ohio State Univ. http://www.osc.edu/lam.html

[19] MPI-2 http://www.mcs.anl.gov/Projects/mpi/mpi2/mpi2.html

[20] Building distributed run-control in UNIX. G. Ambrosini et al. Computing for High-Energy Physics Conference 1992, Annecy, France ISBN 92-9083-049-2

[21] CHSM: A Language System Extending C++ for Implementing Reactive Systems. P. J. Lucas, F. Riccardi. http://www.best.com/ pjl/software.html

[22] Evaluation of the CHSM language system. P-Y. Duval, R. Jones, K. Nurdan, A. Patel, S. Wheeler. ATLAS DAQ Prototype -1 Technical Note 12. http://atddoc.cern.ch/Atlas/Notes/012/Note012-1.html

[23] Java language evaluation for DAQ status display. M. Caprini, Z. Qian. ATLAS DAQ Prototype -1 Technical Note 10. http://atddoc.cern.ch/Atlas/Notes/010/Note010-1.html.

[24] X-Designer Release 4.5. User's Guide, Imperial Software Technology Limited, 1996.

[25] Use of Motif in a HEP DAQ. G. Ambrosini et al. Second International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics (AIHEP'92) conference, La Londe-les-Maures, France in January 1992. ISBN 981-02-1122-8.

[26] Object-Oriented Programming with X-Designer. S. Wheeler. ATLAS DAQ Prototype -1 Technical Note 4. http://atddoc.cern.ch/Atlas/Notes/004/Note004-1.html

[27] A cookbook for using the model view controller user interface paradigm in Smalltalk-80. Glenn E. Krasner and Stephen T. Pope, Journal of Object-Orientated Programming, 1(3):26-49, August/September 1988.

[28] ACE. Douglas C. Schmidt. http://www.cs.wustl.edu/ schmidt/

[29] Evaluation of ACE Object-Oriented OS wrapper for ATLAS DAQ. P-Y. Duval, R. Nacasch. ATLAS DAQ Prototype -1 Technical Note 8. http://atddoc.cern.ch/Atlas/Notes/008/Note008-1.html.

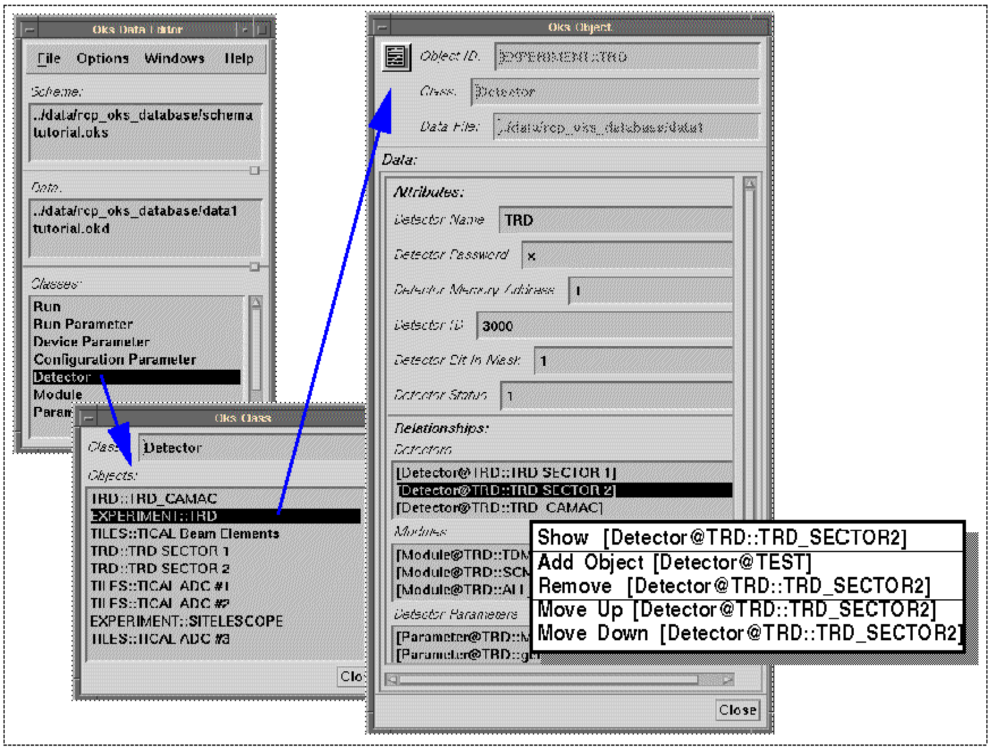

Fig. 1. The OKS data Editor

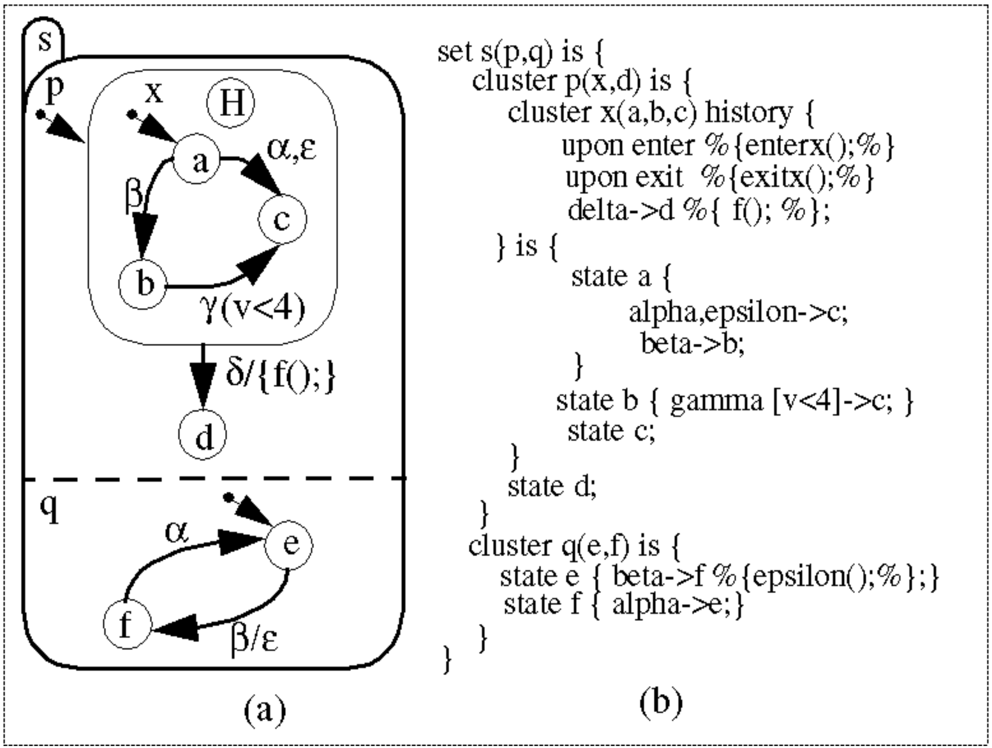

Fig. 2. (a) StateChart (b) CHSM description

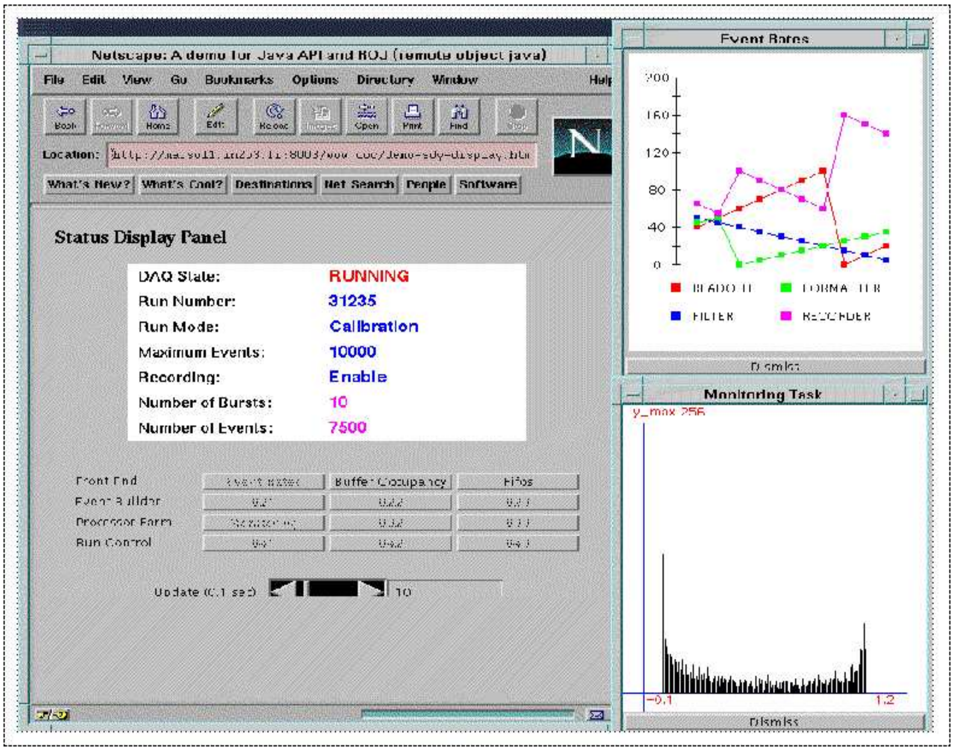

Fig. 3. Elements of a DAQ status display implemented with Java

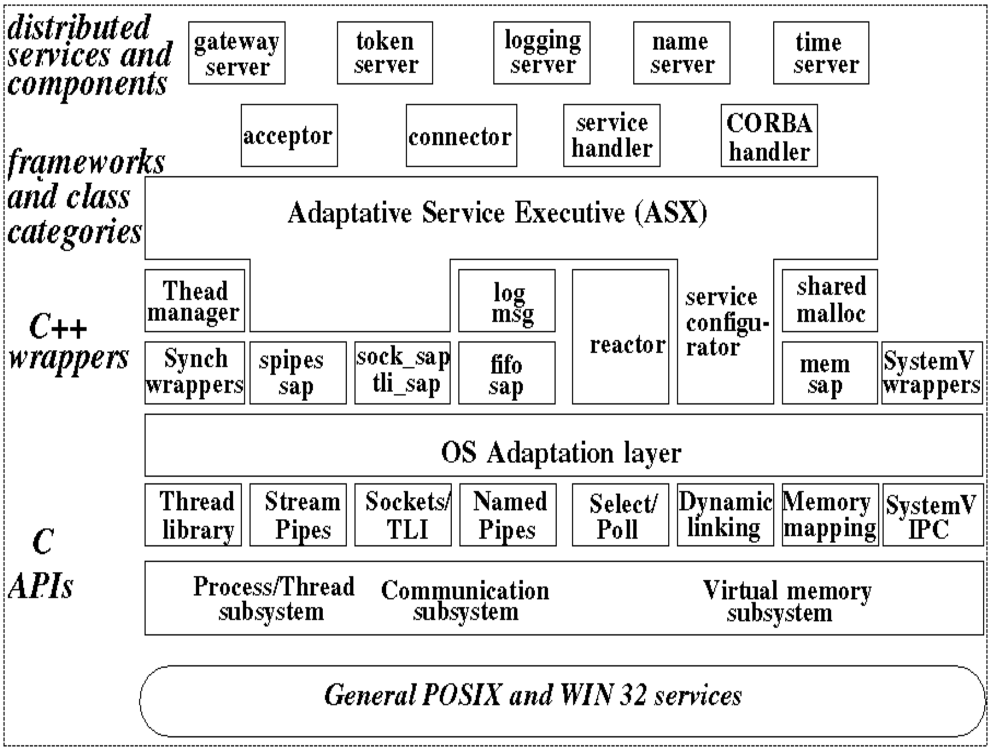

Fig. 4. The ACE object oriented toolkit layers