A Proposal for Converting the ATLAS DAQ Back-end Sub-system into an Open Source Project

Bob Jones
CERN EP/atd
(Robert.Jones@cern.ch)

1.0 Introduction

This note presents a proposal for converting the existing ATLAS DAQ Back-end sub-system into an open source project. It presents an overview and the status of the Back-end software, the motivation for moving to an open source project then identifies some issues involved and suggests a structure for the organization of the project and a plan for how to proceed.

2.0 Overview of the Back-end sub-system

This section gives a brief overview of the ATLAS DAQ Back-end sub-system. For further information see the project's web pages (http://atddoc.cern.ch/Atlas/) and most recent conference papers [2].

2.1 Purpose and goal

The goal of the ATLAS data acquisition (DAQ) and Event Filter (EF) 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 sub-system encompasses the software for configuring, controlling and monitoring the DAQ but specifically excludes the management, processing or transportation of physics data. The Back-end software is essentially the "glue" that holds the sub-systems together. It does not contain any elements that are detector specific as it will be used by many configurations of the DAQ and detector instrumentation. The Back-end software is but one sub-system of the whole DAQ system. It must co-exist and co-operate with the other sub-systems. In particular, interfaces are required to the triggers, processor farm, accelerator, event builder, detector read-out create controllers and Detector Control System (DCS).

2.2 The software component model

The user requirements gathered for the Back-end sub-system have been divided into groups related to activities providing similar functionality. The groups have been further developed into components with well defined purposes and boundaries. The components have interfaces with other components and external systems, specific functionality and their own architecture. The components have been grouped into two sets: core components and trigger/DAQ/detector integration components.

2.2.1 Core components

The core components of the Back-end sub-system constitute it's essential functionality and they had priority in terms of time-scale for development in order to have a baseline sub-system that can be used for integration with the data-flow sub-system and event filter. The following components are considered to be the core of the Back-end sub-system:

- **Configuration databases** are used to describe a large number of parameters of the DAQ system architecture, hardware and software components, running modes and status. One of the major design issues of Atlas DAQ is to be as flexible as possible, parameterized by the contents of the configuration databases.
- **Message reporting system** (MRS) provides a facility which allows all software components to report messages to other components in the distributed environment. The MRS performs transport, filtering and routing of messages.
- **Information service** (IS) provides an information exchange facility for software components. Information (defined by supplier) from many sources can be categorized and made available to requesting applications asynchronously or on demand.
- **Process manager** (PMG) performs basic job control of software components. It is capable of starting, stopping and monitoring the status (e.g. running, exited) of components independent of the underlying operating system.
- **Run control** (RC) is used to control the data taking activities by coordinating the operations of the DAQ sub-systems, Back-end software and external systems. It has a user interface for the shift operators to control and supervise the data taking sessions. It has software interfaces with other DAQ sub-systems and Back-end components to exchange commands, status and control information.

2.2.2 Trigger/DAQ/detector integration components

The following components are required to integrate the Back-end with other online sub-systems and detectors:

- **Resource manager** (RM) allocates resources (hardware and software resources which can't be freely shared) and allows several groups to work in parallel without interference.
- **Partition manager** extends RM to allow the simultaneous operation of several partitions.
- **Test manager** (TM) organizes individual tests for hardware and software components (the individual tests themselves are not the responsibility of the TM which simply assures their execution and verifies their return status).
- **Diagnostics package** (DS) uses tests held in the test manager to diagnose problems and verify functioning status of separate components or the entire system (DS verification component) and to control the system, diagnose and recover problems during different phases of system functionality in automatic or operator assistance modes (DS supervision component).
- **Integrated graphical user interface** (IGUI) allows the operator to control and monitor the status of the current data taking run in terms of it's main parameters, detector configuration, trigger rate, buffer occupancy and state of the sub-systems.
- **Online bookkeeper** archives information about the data recorded to permanent storage by the DAQ system. It records information on a per-run basis and provides a number of interfaces for retrieving and updating the information.
- **Event dump** samples events from the data-flow to present them to the user in order to verify event integrity and structure.

2.3 Operational environment

It is expected that this environment will be a heterogeneous collection of UNIX workstations, PC running Linux or Windows NT and embedded systems running various flavours of UNIX operating systems with real-time features (e.g. Lynx OS) connected via a local area network.

The Back-end software has been developed in C++ and ported to several compilers on the Solaris, Linux, HP-UX, Windows NT and Lynx operating systems.

2.4 Software technologies

The various components described above 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, the same facilities are used across all components. Such facilities must be portable across all the platforms used in the DAQ, in particular they must be available on the LynxOS real-time UNIX operating system selected for use on the VME processors. Candidate open-source and commercial software packages were evaluated to find the most suitable product for each technology. C++ is the primary programming language supported by a general purpose library (Rogue Wave's Tools.h++). The Objectivity/DB object-oriented database and a custom-made in-memory object manager (OKS) is used for data persistence. Objectivity/DB is not necessary to run the core of the Back-end software only to use the Online Bookkeeper component. Corba (ILU) is used for communication and a custom-made package (IPC) as a higher-level of abstraction. Finite state machines are used to implement object behaviour (CHSM) while Motif and Java are used to implement graphical user interfaces.

2.5 Software process

The development has been divided into a number of sequential phases intended to help pace and organise the work. Each phase has been defined to produce an obvious deliverable (i.e. document and/or code) which is reviewed by the whole Back-end group before progressing to the next phase. The phases are: collect requirements; identify and evaluate candidate technologies and techniques capable of addressing the common issues identified from the requirements; produce a design for each component covering the most important aspects; refine the design to add more detail; implement and unit test according to the design; integrate with other components.

The people involved in the Back-end sub-system come from many institutes and have not, in general, been able to work full-time on the project. Faced with this situation, we have tried to organise the work along the component structure. Typically, a single institute has taken responsibility for developing a component there by simplifying communication and reducing travel. Such component groups are small (up to a maximum of 5 individuals). The same individuals have tended to follow a single component through the various phases and hence ensured the continuity of the work.

2.6 Software development environment

Within the Back-end sub-system, elements of the software development environment have been assembled to cover the phases of the software life-cycle described above. Such elements have been selected to be compatible with similar work in the off-line software groups and include: an object-oriented method (OMT/Booch) for analysis and design supported by a CASE tool (Software Thru Pictures tool) whose facilities have been extended to import requirements from FrameMaker documents and generate code for various languages; a configuration management system using the ATLAS Software Release Tools [5]; testing tools to produce static software metrics (Logiscope) and code coverage measurements (Insure++) have been integrated with the SRT. Informal tutorials and official CERN training sessions have been organised for all of the above elements of the software development environment.

2.7 Status

Implementations exist for all the components listed above with the exception of the event dump. Full documentation (requirements, high-level design, users' guide, test plan and test results) exists for all core components and is currently being completed for the trigger/DAQ integration components.

Unit tests for individual components have been made [3] and integration tests involving most of the components in various configurations including multiple workstations, PCs and VME processors [2].

Tests have been made with the data-flow sub-system of the project in which the Back-end system has been used to control and configure a prototype DAQ system including VME based Read-Out Buffer (ROB) crates, the event builder and interfaces to the event filter processor farm have been completed.

3.0 Open Source Projects

This section gives a very brief overview of open source projects and is paraphrased from the web pages of the OpenSource Organization (www.opensource.org):

The basic idea behind open source is very simple. When programmers on the Internet can read, redistribute, and modify the source for a piece of software, it evolves. People improve it, people adapt it, people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing.

The open-source community has learned that this rapid evolutionary process produces better software than the traditional closed model, in which only a very few programmers can see source and everybody else must blindly use an opaque block of bits.

Please refer to the Open Source web pages for further information.

4.0 Motivation

There is a tendency in the HEP community to duplicate work performed in other experiments by producing similar solutions for the same problems. This is particularly true in experimental online groups due to their reliance on the very latest and hence non-standard hardware technology. This is inefficient and also makes it difficult to capitalise on the knowledge and experience gained by physicists and engineers as they move from one experiment to another or to industry. Commodity hardware and software has started to have an effect on such online environments. The advances of commodity systems leads to more standardisation and hence creates more opportunities for sharing software.

In order to support the day-to-day, around the clock operation of the experiment, online groups are often more centralised and focussed (i.e. participants work on fewer projects concurrently) than their offline counterparts. But this situation is changing with the latest generation of experiments. For example, there are more than 50 institutes from 17 countries involved in the trigger/DAQ system of ATLAS. Add to this the fact that host laboratories continue to reduce their staff compliments and one can see a growing need for sharing online software between experiments.

Development of online software is a very demanding activity. If there is a fault in its design or implementation expensive accelerator beam-time may be wasted and precious event data lost *forever*. There is no online equivalent of being able to re-run an analysis program. These aspects have tended to make online developers more cautious in their choice of software packages and caused them to seek as much control as possible over the origin of the software they have used in order to ensure its reliability. The current move from proprietary to open source operating systems (e.g. Linux) and application software (e.g. the Apache web server) has shown that such projects can produce high-quality and reliable products.

In addition to the general trends outlined above, there are a number of points specific to the Back-end sub-system that justify proposing it as the basis for an open source project:

- The existing Back-end sub-system is a distributed collaborative project spanning the whole of Europe. We have needed to adopt many if not all of the features of an open source project to permit colleagues in the various institutes to collaborate on the project. To organise the Back-end sub-system and bring it to fruitition, it has been necessary to adopt a well defined structure, architecture, software process and development environment. For example, it has been necessary to make extensive use of collaborative tools such as email lists for communication, a central repository for the source code, official releases, nightly builds and a web of information on all aspects of the project. This structure and organisation has lead to a development culture which is very similar to that found in larger, well established open source projects.
- Many collaborating institutes have expressed the wish to use the software they have developed for the Back-end sub-system on other projects and experiments.
- Due to the Back-end's architecture and exceptional level of documentation, developers can quickly become familiar with its operation and internal organisation. For example, CERN summer students assigned to the project (physicists with little programming experience) were productive after two weeks of work.
- The extra effort required to permit the software to be used in other projects and experiments is relatively small compared to that being put into the Back-end today.
- There is a natural reticence in other development groups, even within the ATLAS experiment, to commit to the use of the Back-end sub-system without having some guarantee for the long term survival of the software. By giving access to the source code, such groups have a safety-net on which they can rely in case the Back-end group should fall into disarray.
- By providing all the features of an open source project including access to the source code, it will enable those people not currently involved in the development of the software but who rely on its functionality (such as those people working in detector groups to provide detector specific read-out software) to participate more easily in the project.
- Participation can be in many forms starting with problem reporting. Providing access to the specifications, design and code of the software will make it easier for people to precisely pin-point the cause of the problem. Access will also permit them to suggest improvements and solutions of their own. The very act of reading the documents and source code will mean they are participating in the inspection and review of the software and hence helping to improve its quality. Formal inspection of software, as currently performed in the Back-end sub-system [4], is a necessary but labour intensive task and so any technique that can distribute the load must be considered seriously.
- Participation can also be in the form of documentation. If an area of the software is insufficiently documented or documented from another point of view, developers could provide their own documentation which would become an artifact of the project.
- Participation can also be in the form of new components or porting existing software to new platforms.

5.0 Issues

In order to turn the Back-end sub-system into an open source project, a number of issues that are identified in this section need to be addressed.

5.1 Use of third party software

The Back-end software relies on a number of open-source, public domain and commercial third party software packages to provide specific functionality. The packages used are:

- ILU from Xerox Parc

This package provides the CORBA communication implementation. It carries the following license information:

Unlimited use, reproduction, modification, and distribution of this software and modified versions thereof is permitted. Permission is granted to make derivative works from this software or a modified version thereof. Any copy of this software, a modified version thereof, or a derivative work must include both the above copyright notice of Xerox Corporation and this paragraph. Any distribution of this software, a modified version thereof, or a derivative work must comply with all applicable United States export control laws.

- Tools.h++ from Rogue Wave

This is a commercial general purpose C++ library similar to STL. It was selected for use in the project before STL became widely available. CERN bought source code licenses for this product since it was also used in initial releases of the RD44 (GEANT4) project software. We are not allowed to distribute the source code for this package. We intend to replace Tools.h++ with STL.

- ACE by Doug Schmidt of Washington University

ACE provides C++ wrappers for operating system interfaces. It carries the following license information:

ACE(TM) and TAO(TM) are copyrighted by Douglas C. Schmidt and his research group at Washington University, Copyright (c) 1993-1999, all rights reserved. Since ACE and TAO are open source,free software, you are free to use, modify, and distribute the ACE and TAO source code and object code produced from the source, as long as you include this copyright statement along with code built using ACE and TAO.

We are currently phasing-out the use of this package since it has become redundant.

- CHSM by Paul J. Lucas.

This package implements the state machines used by the run control and process manager components. It is distributed as source code and carries the GNU General Public License.

- CLIPS from NASA

This is a C based expert system tool used as a basis for the diagnostics package. It carries the following license information:

Copies of CLIPS executables, source code, and documentation obtained from the CLIPS download site can be freely used and redistributed without restrictions.

CLIPS is now maintained as public domain software by the main program authors who no longer work for NASA.

- CommandLine by Brad Appleton

This is the C++ package used to manage command line switches and parameters for all programs developed within the Back-end. It carries the following license information:

Permission is hereby granted to freely copy and redistribute this software, provided that the author is clearly credited in all copies and derivations. Neither the names of the authors nor that of their employers may be used to endorse or promote products derived from this software without specific written permission.

- Motif toolkit from the The Open Group

The Motif graphical toolkit is covered by The Open Group Master Software License and distinguishes between different agreements such as source code, object code and run-time copy licenses. LessTif is a freely available Motif toolkit clone included in many Linux distributions such as RedHat and covered by the GNU Library General Public License (LGPL).

A number of freely available graphical widgets based on the Motif (or LessTif) toolkit are used in the OKS interfaces. The XBaeMatrix widget carries the following license information:

Permission to use, copy, modify and distribute this material for any purpose and without fee is hereby granted, provided that the above copyright notices and this permission notice appear in all copies, and that the name of any author not be used in advertising or publicity pertaining to this material without the specific, prior written permission of an authorized representative of Bellcore and current maintainer.

The XmHTML widget carries a GNU Library General Public License and the XmTree widget is in the public domain.

It has been necessary to make small modifications and add additional files to all the packages in order be build them with the SRT release management tool and port them to other platforms, notably LynxOS. Hence they are not exactly the same as the original versions available from the authors. The authors normally do not want to incorporate such changes into their own versions because each has their own (often incompatible) build and configuration system.

Objectivity/DB is a commercial object database system used by ATLAS and other experiments offline software. It is currently required for the Online Bookkeeper Back-end component but it is not installed in SRT or distributed on the CD-ROM.

The SRT configuration management tool is based on autoconf and GNU make. CVS is used as a repository. The current policy of storing third-party packages in the project's repository may need to be reviewed to simplify tracking updates in such packages.

5.2 Selection of a suitable license

It is necessary to apply some form of license to open source software to ensure that the identity of the developers is associated with the source code (since they will not be directly rewarded financially for their contributions) and guarantee that it remains available for everyone to use and improve. The Open Source Definition summarizes the qualities of a license that open source developers have found necessary from a practical standpoint to maintain project viability in an open source context.

There are several popular open source licenses circulating on the Internet which are similar but have subtle differences affecting how the software can be distributed and used in other projects and commercial products. Such licenses include:

- BSD license
- GNU General Public License
- GNU Library General Public License
- Artistic License (created by Larry Wall for use with Perl but is legally less precise)
- Mozilla Public License and the related Netscape Public License

Several HEP projects, for example ROOT and GEANT4, have created their own licensing policy.

It will be necessary to adopt a suitable licensing policy, possibly one of the above, and include it in the source code and as part of the official releases.

5.3 Support

Even though access will be provided to all the source code of the software, we will continue to distribute official releases in binary format, packaged for easy installation by end-users with quality assurance testing.

An appropriate level of support will be maintained for official releases to be determined between the participants of the project and their respective institutes and experiments. Modified versions, or versions built on alternative platforms will not be supported directly but participants will provide assistance on a *best effort* basis.

Participating experiments will still need developers in their online groups to be responsible for areas covered by the Back-end software. The Back-end software will not cover all the needs of the experiment. Rather it should be seen as a third-party set of packages, to be integrated with the experiment's own software. The experimental online groups remain responsible for ensuring the Back-end software works *for them* in *their* experiment. The advantage of making the Back-end software an open source project is that the support load can be shared between developers of many

experiments. Also experiments will, via participation, have a greater possibility to influence the content and development of the software.

5.4 Organisation

From the experience gained in developing the existing Back-end software and by taking into account the structure of other projects in HEP (e.g. GEANT4) and world-wide open source projects (e.g. Mozilla) the following organisation is seen as being the most appropriate.

The basic unit of development is a component or package. For each package, there is a single coordinator who is one of the developers working on the package and often the primary author. The coordinator is responsible for fielding bug reports, enhancement requests, patch submissions, and so on. The coordinator should facilitate good development, as defined by the developer community. This is true for in-house developed packages and external third-party packages as well. In the case of a third-party package, the coordinator is responsible for tracking the development of the package and selecting the most appropriate version for use within the project.

While all feedback and input is welcome for all packages from all developers, it would be useful if participants from a single institute or experiment could work together on the same package. This would simplify communication, reduce travel and build regional centres of expertise.

Experience has shown that distributed development is the most efficient and centralised integration works best. This is currently how the Back-end sub-system is organised. When new packages are added or existing ones undergo major changes, the coordinator performs the integration in close relationship with the software librarian, release organiser and testers on the CERN site. All such additions and modifications must follow a defined software process as described in 2.5.

6.0 How to Proceed

Release 0.0.7 (known as the James Bond release) is available on CD-ROM and via the web. This release includes full documentation (requirements, designs, user guides, test plans and results), header files, libraries and binaries for all supported platforms. It does not include source code. The source code of all in-house components and publicly available third-party packages is visible via the web using the LXR tool. The corresponding UML class diagrams generated with the Together/C++ CASE tool are also available via the web.

Individuals, institutes and experiments who are interested in participating in the proposed open source project are recommended to start by evaluating the James Bond (0.0.7) release of the software to determine if it is suitable for their needs. All feedback is welcome.

Mechanisms (a problem tracking system, FAQ lists, newsgroup etc.) will be put in place to ensure that feedback from participants is handled promptly and impartially, independent of their affiliation. Modifications and additions will be judged on their technical merit and quality as quickly as possible so that they can be included in future releases.

A training session is being organised for developers on how to use the Back-end software. It will be a mixture of presentations and hands-on exercises. The presentations will be recorded (probably using the techniques developed for the Web Lecture Archive Project http://webcast.cern.ch/Projects/WebLectureArchive/index.html) and the exercises will be made available via the web. The training will then be accessible via the web at any time and could also be distributed on CD-ROM. The training session will be scheduled for Spring 2000 and will be based on the forthcoming release 0.0.8 of the software.

The steps described above represent incremental improvements to the organisation of the Back-end group and software suite and hence the effort will not be wasted if the concept of an open source project is not retained. Should the proposal to turn the Back-end sub-system into an open source project attract sufficient interest a new document giving more details of the planning, resources, organisation and issues will be produced. This document could then be used to drive the development of the project and produce regular releases of the software.

7.0 References

1 G. Ambrosini et al., The ATLAS DAQ and Event Filter prototype "-1" project, Computer Physics Communications 110 (1998) 95-102. http://atddoc.cern.ch/Atlas/Conferences/CHEP/ID388/ID388-1.html

2 I. Alexandrov et al., The Performance and Scalability of the Back-end DAQ sub-system, Proceedings of the CHEP2000 conference

3 I.Alexandrov et al., Performance and Scalability of the Back-end sub-system in the ATLAS DAQ/EF Prototype, RT-99 Conference, Santa Fe, June 1999 http://atddoc.cern.ch/Atlas/Conferences/RT99/RT175.ps

4 I.Alexandrov et al., Impact of Software Review and Inspection, Proceedings of the CHEP2000 conference

5 L. Tuura, Overview of ATLAS Software Release Tools, Proceedings of the CHEP'98 conference, Chicago, September 1998. http://www.hep.net/chep98/

8.0 Acknowledgements

Most of the information concerning open source projects included in this document was gathered from reading the OpenSource Organization's web pages (www.opensource.org). The issues and justifications discussed are based on arguments outlined by Frank Hecker in his paper entitled "Setting Up Shop: The Business of Open-Source Software" (http://www.hecker.org/writings/setting-up-shop.html). The organisational aspects take into account those published by the GEANT4 project (http://wwwinfo.cern.ch/asd/geant4/geant4.html) and the Mozilla Organization (http://www.mozilla.org/). Michael K. Johnson at RedHat added a lot of precision to an earlier draft and was a source of good advise on open source topics. Paul Kunz provided information on possible configuration tools for the future. This proposal is only possible because of the exceptional work of everyone involved in the ATLAS DAQ Back-end sub-system, present and past. Special thanks go to the CERN ATLAS DAQ group leader, Livio Mapelli, for having the courage to make this step.