DD

ZEUS ONLINE ENVIRONMENT ON ITS WAY TO OPEN, DISTRIBUTED COMPUTING

JAROSLAW MILEWSKI (*)

CHRISTOPHER YOUNGMAN (**)

(*) *II. Institute of Experimental Physics, Hamburg University,
Luruper Chaussee 149, 22761 Hamburg, Germany.
milewski@desy.de*

(**) *Deutsches Elektronen-Synchrotron DESY,
Notkestrasse 85, 22603 Hamburg, Germany.
youngman@desy.de*

We point out some of the difficulties encountered by the ZEUS online software in the initial phase of its operation and then propose a remedy to those problems. The general guidelines of the present ZEUS online evolution are summarised and examples of software development following those guidelines are given.

sw9608

1 Introduction

When we speak about Computing for the Next Millennium, we often get fascinated with GigaFlops of computing power, GigaBauds of network throughput or TeraBytes of storage capacity. What is equally important is how to make all those technological wonders work together. Among others, the key issue is to make the systems

- more open, so that the heterogeneity of computing resources does not prevent us from gaining the best of each of them,
- more distributed, so that the cooperating resources do not disturb each other in the work on the common goal.

The ZEUS online environment is an interesting example of a complex system evolving in the direction of more open and more distributed computing. After the first few years of ZEUS operation (including 2 years of 'real' data taking) we have observed several problems in the original design of the ZEUS control software:

- Communication based on central message switching (such as CERN OSP¹) suffered from bottleneck and single-point-of-failure problems.
- Ad hoc solutions based on direct TCP, UDP or DECnet communication did not help; they were leading to a maze of hidden interconnections between tasks.
- Heavy-weight central facilities like the central Slow Control or the central Run Control ended up by serving all possible pieces of status/error/data information to everybody, departing from their genuine control functions.

- The ever growing variety of 32-bit and 64-bit, VMS, UNIX-like and OS-9 based machines caused data exchange and process migration problems.

In this situation the ZEUS Central Data Acquisition Group took several decisions aiming at a radical change of the ZEUS online environment.

2 The guidelines

Let us formulate some of the general rules and guidelines that govern the present evolution of the ZEUS online software.

2.1 *Communication as the basis*

Probably 90% of online software of any HEP experiment is directly connected with moving the data around. Therefore,

Selecting the right communication system is possibly the most important decision in the experiment's online software.

The communication system must, of course, be available on all the machine types used by the experiment and support platform-independent representation of data. In order to reduce the inter-process dependencies we have further decided to use asynchronous and connection-less communication protocols (see 3.1).

2.2 *Portability of code*

Component systems are often attached to dedicated pieces of hardware which can be controlled only by certain hosts. But even then the components should count with moving to other machine type or operating system at any time.

The central facilities are usually less hardware-oriented and should work on any of the major computer types used by the experiment. All the programs discussed in section 3 are capable of running under both VMS and UNIX (in its ULTRIX, OSF/1 and IRIX variants), or at least the expected time to port is no longer than a few days.

2.3 *Distributed responsibility*

ZEUS experience shows that best results are obtained when the functions performed by the programs are well separated and they strictly follow the competences of the people who are maintaining them.

Programs should do only what no other programs around do, and nothing more.

When we apply this rule to the central online tasks, they turn out to be quite simple. For instance, the central Slow Control does not have to maintain the register of all the out-of-limit conditions reported by the components if every component can resend all the warnings on request. The central Run Control cares only about those aspects of component states which are relevant for run sequencing, etc.

2.4 Distributed data sources

Distributed responsibility means also that no program should serve data actually produced by somebody else.

Every piece of information should be available at its very source.

For instance, the momentary luminosity values are to be provided by the ZEUS LUMI component and not by the central Slow Control, the Global First Level Trigger rates should be made public by the GFLT and not by the central Run Control, and so on.

2.5 Distributed functionality

Instead of having a complex program performing a complex task, split it into a set of simple programs, each of them covering a well-defined subset of the required functionality.

Note that the whole set is not simpler than the original program: the communication layer may even add a new dimension to complexity. The whole set is also not necessarily easier to maintain: the maintenance effort is just more distributed. But what is important for us is that every single piece may be easily changed without affecting the other pieces if only the communication standards are observed.

3 Software developments

This section presents some of the software projects carried out by the ZEUS Central Data Acquisition Group (CDAQ) in the past 1.5 years. All of them follow the guidelines sketched in the previous section.

3.1 Communication

The ZMP message passing system^{2,3} has been developed as a uniform computing platform for all the ZEUS online tasks. It is a fully distributed communication package, which may be seen as an asynchronous alternative to RPC, designed especially for real-time control tasks. ZMP relies on platform-independent data representation standard (XDR) to ease data exchange. Moreover, ZMP uses a dynamic name service which frees the online processes from being attached to definite machines.

3.2 Data servers

We have started to free the central Slow Control and the central Run Control facilities from serving all possible kinds of information to everybody. Instead of that, numerous ZMP-based data servers attached to triggers, data taking components, etc. have emerged as a replacement for overloaded data services of the central facilities. In that way the data are available directly at the place where they are really produced and there are less doubts who is responsible for proving what type of information to the rest of the experiment.

Apart from the ZEUS data servers, there exists a sound server that emits audio messages for the shift crew, gateway servers importing data from other experiments at DESY, etc.

3.3 Cross-platform process monitoring

A preliminary version of a general-purpose process monitoring and restart system has been released. In case of any vital process crash a replacement is started automatically on any suitable machine, in cooperation with the ZMP name service.

The 'restart daemons' running on different types of machines are driven by configuration files which describe the criteria and methods for restarting (and killing, if needed) of the processes under control. What is still missing are more elaborated algorithms taking into account process groups and process relationships.

3.4 Cross-platform symbol definitions

The ZEUS online as a whole maintains implicitly a number of 'environmental variables' like the name of the printer where some summary report has to appear or the number of the last run. Since those variables must be shared by processes running on different machines under different operating systems, a simple dictionary-type server provides the variable bindings to all the interested parties. Due to a destructive read operation on symbols, it can also be used for synchronisation of processes running on different machines.

3.5 The new Run Control system

The central ZEUS Run Control system has been rewritten from scratch during the last winter shutdown, following the new standards of the ZEUS online. It consists now of a set of loosely coupled processes running on different machines, so that e.g. message logging, run sequencing and run bookkeeping are almost totally independent. (The new ZEUS RC is described in more detail in another paper in this volume⁴).

An equally radical upgrade of the Slow Control system is planned for the 95/96 shutdown.

4 The final word

The reader would probably agree that the keywords 'open' and 'distributed' are closely related to modularity of software design. However, our understanding of the word 'modular' is rather specific. We are less interested in the reusability of code but rather in exchangeability of code: every brick must be ready to be thrown away and replaced by something better if needed. The smaller, simpler, and platform-independent the bricks are, the better. The only thing the components of the system really share are the communication standards.

Acknowledgment

The work of J. Milewski is supported by BMBF grant 05-6HH291.

References

1. R. Jones, G. Mornacchi and R. Russell, *OSP user's guide, version 3.0*, CERN DD/OC Group (1990).
2. J. Milewski, C. Youngman and A. Kotanski. *The ZEUS message-passing system*, proc. CHEP'94, San Francisco (1994), pp. 177-181.
3. A. Kotanski, J. Milewski and C. Youngman. *The ZEUS message-passing system (ZMP), release 1.2*, ZEUS note 94-150, DESY Hamburg (1994), 59 pp.
http://zow00.desy.de:8000/ZEUS_MANUALS/ZMP_spec.ps
4. J. Milewski and C. Youngman, *The new ZEUS run control system: an exercise in modularity*, CHEP'95 (in this volume).