THE NEW ZEUS RUN CONTROL SYSTEM:
AN EXERCISE IN MODULARITY

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*

After the first two years of data taking the ZEUS run control (RC) has been rewritten from scratch. The new system consists of loosely coupled, ZMP-interconnected processes which obey a minimum set of common protocols and standards, may migrate from one machine to another and may be easily exchanged for new modules. The new system proved to be very reliable, much more efficient, simpler, and easier to maintain.

1 Motivation

After the first two years of ZEUS data taking it became clear that the original run control (RC) system¹ had little chance to hold out till the end of the experiment. Its operation was just sufficient to take data, but apart from that it suffered from numerous problems:

- Run sequencing assumed unnecessary dependencies between participating processes, so that global state transitions were very slow and blocked in the presence of any difficulties.
- The central RC program was dealing with almost everything: run bookkeeping, producing fat run summaries, and, worst of all, collecting all kinds of status information from the components and serving it in turn to many other processes.
- The underlying code was strongly dependent on FORTRAN, VMS and DECnet, with added-on UNIX and TCP/IP patches.

All this made the system over-complicated, highly centralised, non-portable and at the end almost unmaintainable.

To overcome those difficulties, the ZEUS Central Data Acquisition Group have implemented a brand new ZEUS run control system during the 94/95 winter shutdown.

3w 9608

2 Prerequisites

The new RC facility had to observe the general line of the ZEUS online environment evolution, founded, among others, on the following principles:

- *Interconnect uniformly the heterogeneous component subsystems running on different types of machines.*
In ZEUS terms it means passing XDR-encoded data over the ZMP message-passing system². This is the minimum requirement on which higher-level protocols are built.
- *Apply clear distinctions of competences among the central and peripheral online tasks.*
For instance, the new central RC facility deals with the run control and only with the run control. Such issues as safety control or data quality monitoring are put totally outside the scope of the RC system, although RC provides its own portion of information for those facilities whose competences do cover safety or data quality monitoring.

- *Every piece of information should be available at source.*
This is another variation of the clear competences rule, this time applied to the ZEUS public data services. The central RC task provides only 'genuine' RC data like the run number, trigger type, participating components, etc. If someone is interested e.g. in the components state, he should address the components.

(More about the ZEUS online software philosophy in another paper in this volume³).

One important factor which has always been present behind the scenes is the diversity of ZEUS component systems. For historical reasons, they were developed independently so that any central task of the ZEUS online has to assume very little about their similarities. Also in case of the central RC the common denominator of component message formats and even of their state transition logic is pretty small. The ways the new run control system can cope with this heterogeneity will be pointed out when necessary.

3 System overview

The architecture of the new ZEUS run control system is shown in Fig. 1. Shaded items belong to the proper RC system, the remaining ones are selected cooperating processes.

3.1 The manager task (RCM)

The Run Control Manager is the centre of communication of the RC system. It is also responsible for

- global state transitions and run sequencing,
- processing of the trigger configuration files,
- bookkeeping of the run number, shift crew, etc.

It is worth mentioning that the new manager task evaluates the global state of the system and also performs the global state transitions in *a totally different way*. Nevertheless, the new logic describes the system consistently, as ZEUS is still taking data!

Figure 1: The ZEUS RC system architecture

Not only taking data, but doing it remarkably better than before. The average global transition time dropped down by a factor of 3 to 4, depending on trigger type and participating components. The total turnaround of the sequence "end the run" + "setup for the next run" + "start data taking" went down from 8-10 minutes with the old RC system to 2-3 minutes with the new one.

The improvement is due mainly to:

- Better run sequencing. RCM performs global transitions in *parallel steps* instead of the previous strict sequence.
- Efficient communication of RCM with the components.
- Improvements on the components side, motivated by the central RC upgrade.

3.2 Processing of the trigger files

The ZEUS 'trigger files' define the sequences of messages which are to be sent to individual components during setting up for the next run.

Due to the diversity of the ZEUS components the trigger files actually contain arbitrary strings, which are passed to the component systems in the prescribed sequence, but without any additional interpretation by the central RC program. The new RC system uses the *C pre-processor* in order to take advantage of the similarities between the existing ca. 70 trigger files. The possibility to `#include` common fragments of trigger definitions, to parametrise them by `#defined` constants, and to introduce conditional definitions on `#if-#then-#else` basis turned out to be surprisingly flexible.

3.3 The operator's task (RCO)

The operator's interface to the run control system works as a separate program (RCO) connected to RCM via ZMP. The ZMP name server assures the presence of only one instance of RCO running at a given time. Of course, RCO normally runs on one of the workstations in the ZEUS control room.

3.4 The viewer tasks

They use the same protocol as RCO for obtaining the continuous 'screen update' data from RCM. However, in contrast to RCO,

- The ZMP name server allows an arbitrary number of viewers to be active,
- The viewer has no possibility to issue the run control commands,
- Spying on the situation in the ZEUS control room from remote locations is easier when the viewer provides a terminal-type interface instead of an X interface.

3.5 The logger/infoserver task (RCX)

RCX is essentially an extension of RCM, which frees the main task from dealing with RC logfiles and serving of RC-related data to external clients.

The main reason for separating the two is that RCM functionality is necessary for data taking, whereas the availability of logfiles or of a public data service is a (very useful) option. If so, any possible error related to those extra functions (e.g. an error writing the logfile, thousands of requests sent to the data server by an undisciplined client) have no influence on the operation of the manager task.

RCX obtains the RCM data stream in exactly the same way as RCO or other viewers. One more thing it provides is the RCM portion of the run summary. This is neither RCM nor RCX, though, that produces the *global* run summary.

3.6 Run summary production

The run summary is a short document which describes the most important characteristics of a completed run, like the run configuration (number and type, trigger configuration, partici-

pating components), the amount, average speed, etc. of data taking, plus any summary information that the individual components find relevant.

The old run control program was producing the entire run summary itself, basing on the information collected from the components during the run. The new facility just merges arbitrary run summary files provided by the components, taking into account only the synchronisation issues (the components produce their pieces at different speeds and at different moments of time after the run completion). The absence of any part, including the RCM part, is signalled but tolerated.

3.7 The communication subsystem

At present, only half of 22 ZEUS RC components use the new communication system (ZMP) and new message formats for direct communication with RCM. The other ones pretend to talk to the old RC with message formats and communication functions (ZIP) unchanged. The reason is that components were not required to move to the new standard during the same winter shutdown period as the RC system emerged, due to time constraints.

In order to tolerate the mixture of old and new style components, ZIP/ZMP and ZMP/ZIP gateways have been placed between the old style components and RCM; the latter uses only ZMP. It is expected that most of the RC components will ultimately move to the new standard during the 95/96 winter shutdown. From the central RC point of view, the existing gateway processes may be taken out of the system and replaced by direct ZMP communication at any time.

4 The development cycle

The full development of the 100% new run control system took 4 months (September-December 1994), including 2 weeks for operator's interface. The tools used included C++ (the central run control program), XDR (all protocol specifications) and Tcl/Tk (graphical user interfaces).

The system went into operation as a fully mature product and runs very stably from the beginning up to now. Only a few bugs were spotted and relatively few improvements were found to be necessary.

The use of ZMP has been one of the key factors in successful testing and maintenance. The ZMP name server frees the run control processes from being assigned to fixed machines, which enables fully transparent migration of all the central RC tasks from one machine to another in case of any hardware or system problems. The logical division of the ZMP name space into separate domains allows 'virtual' yet complete run control environments to coexist with the proper RC system. This helped considerably in the smooth re-integration of the ZEUS components with the new run control and in the further development of new features.

5 Conclusions

Since we promoted the term "modularity" in the title of our paper, let us summarise what kind of modularity we were trying to achieve in the design of the new ZEUS run control system.

Although fully-fledged object-oriented programming was applied in the production of the central RCM program, this is not the modularity at the level of objects building up a program that was discussed here. We were rather interested in modularity at the architectural level, i.e. at the level of tasks building up the complete system.

Now, one way of speaking about the modularity in system architecture is the 'vertical' modularity of software layers, covering each other conceptually and functionally until the level of application is reached. Although the new ZEUS system is also based on layered design, mainly on communication layers built upon ZMP, this is also not the kind of modularity we would like to highlight.

The idea we are trying to promote here is something one could call the 'brick' modularity, i.e. the method of building the system out of very loosely coupled parts which observe a really minimal set of protocols and standards, whereas not the reusability of parts is most important, but their exchangeability.

In case of the ZEUS run control system nothing happens if a component changes the format or contents of its setup or run summary data, and very little happens if the component changes the format or contents of the messages sent to RCM. The central run control program doesn't care about either the location or format of its own logfiles, nor about the functionality of its subsidiary data server and cares very little about the presence of the operator or viewer tasks. The communication of RCM with the components is loose enough to permit the restart of all the central RC programs at any time, even in the middle of data taking.

Certainly, only all the parts together make a fully functional system and normally all of them are up and running. Nevertheless, the principle of minimum interdependency makes the system more robust, easier to understand, to maintain and to develop.

Acknowledgment

The work of J. Milewski is supported by BMBF grant 05-6HH29I.

References

1. I. Park, *Run control system of ZEUS online data acquisition*, in: *ZEUS contributed papers of CHEP'92*, DESY report 92-150, Hamburg (1992), pp. 43-46.
2. J. Milewski, C. Youngman and A. Kotanski, *The ZEUS message-passing system*, proc. CHEP'94, San Francisco (1994), pp. 177-181.
3. J. Milewski and C. Youngman, *ZEUS online environment on its way to open, distributed computing*, CHEP'95 (in this volume).