PRESENT STATUS OF VEPP-5 CONTROL SYSTEM

D.Yu.Bolkhovityanov*, A.Yu.Antonov, R.E.Kuskov
The Budker Institute of Nuclear Physics, Novosibirsk, Russia

Abstract

As VEPP-5 moves to commissioning, its control system — CX — becomes more mature. CX is a distributed, networked control system based on a 3-layer "standard model". It has been used for VEPP-5 control since 2000; most hardware is CAMAC and CAN-bus. Currently most control programs have switched to modular plugin-based architecture, which significantly eases development of applications and enhances the whole control system integration. Large-data-size control hardware (such as digital oscilloscopes and CCD-cameras) is fully supported by CX now. E-logbook is currently being deployed, both as a web application and with direct support in control programs. GIS technology is being introduced to the control system, which opens many interesting possibilities.

VEPP-5 INJECTION COMPLEX

VEPP-5 injection complex [1] is destined to provide electrons and positrons to all BINP electron-positron colliders — existing VEPP-3/VEPP-4M complex and VEPP-2000, which is currently being put into operation. VEPP-5 is able to produce more than 10^{10} positrons per second, which exceeds requirements of all users.

VEPP-5 (see Fig.1) consists of 300 MeV electron linac, conversion system, 510 MeV positron linac, and damping ring with beam injection and extraction channels.

It includes a number of separate systems, requiring computer automation:

- Vacuum control.
- Linac magnetic system.
- Damping ring magnetic system.
- RF-synchronization.
- Beam diagnostic system.

and some more.

VEPP-5 CONTROL SYSTEM

Hardware

VEPP-5 uses mainly CAMAC and CAN-bus control hardware, most of which is designed and produced inhouse.

Historically CAMAC crates were driven by "intelligent" BINP-designed "Odrenok" controllers (for complex tasks) and by "dumb" serial controllers (for simple/slow tasks). For CAN-bus initially PCI interface cards were used.

Currently most of these are replaced by PowerPC-based controllers[2]. CAMAC and CAN controllers are similar, differing mainly in interface. Having 50MHz PowerPC 852 CPU, 32M RAM and 100Mbit Ethernet onboard, they run Linux, thus providing a rich and familiar environment.

Control room computers are 4 office-class PCs with Pentium-III/800MHz and 1G RAM, with 4 Xinerama-joined monitors each. There are 2 more PCs dealing with hardware, differing only with no displays. All of these run Linux. So, there's a unified environment across all 3 layers of control system, which significantly simplifies development and support.

Software

VEPP-5 control system software was created inhouse — mainly due to absence of any reasonable choices in due time.

This software, named CX[3], uses networked 3-layer standard model. It is written entirely in C and uses single-threaded approach which ensures simplicity and reliability. Drivers are loaded dynamically at run-time, thus making the system flexible.

In recent years unified support for arbitrary-size channels (up to 4Mb) was added. So, now all required hardware is supported by CX directly,

Besides VEPP-5, CX was used to automate several small-scale experiments, and had proven to be adequate for such tasks.

UNIFIED MODULAR GUI

Usually most control applications can be fit into one of 2 classes:

- "Simple" applications, which just present a number of "screen instruments", directly mapped to hardware channels. Such applications are often implemented with help of so called "display managers" and are in fact just some descriptions of control screens.
- "Intelligent" applications — those that can't fit into 1st category. This can be conditioned by some non-trivial computations or specific data processing, feedback, or a need to display data in some unusual way. Such applications must be coded individually.

In 2006 the standard libraries, which effectively constitute CX display manager (called `chlclient`), were extended to support "user-supplied", "plug-in" display knobs in addition to standard ones.

So, now all CX control programs can be implemented as "simple" applications. This modular approach significantly decreases cost of application develompent and allows to

* bolkhov@inp.nsk.su

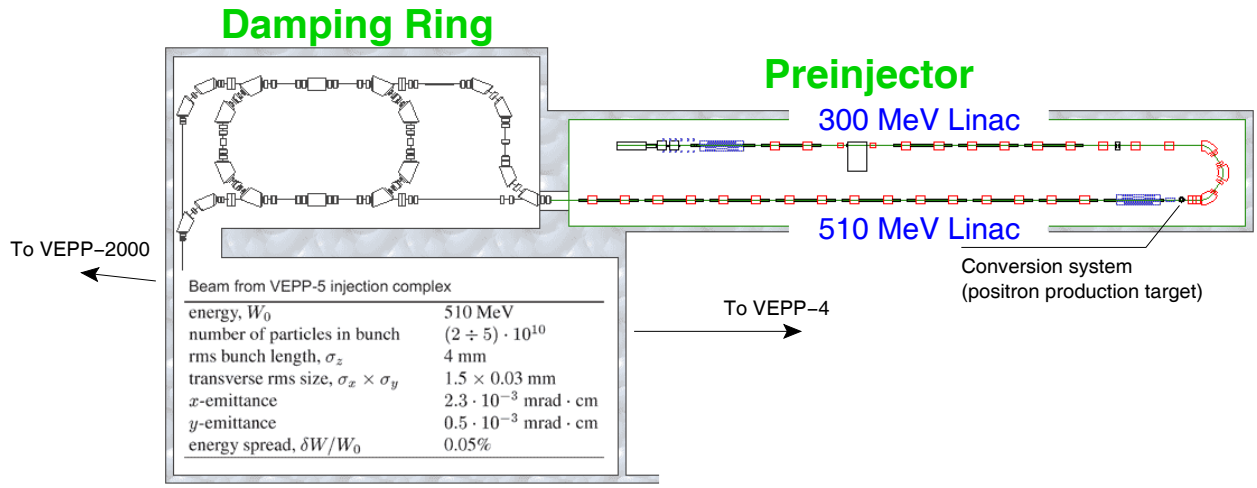

Figure 1: VEPP-5 injection complex layout and parameters.

deal with systems, which previously required "intelligent" applications, in a unified way[4].

E-LOGBOOK

An electronic logbook (e-logbook) becomes a must for large experimental facilities not only during operation, but also at building and commissionning stages (where VEPP-5 is now).

Unfortunately, the "market" of such products is almost nonexistent. So, the choice is narrow: either use some other lab's software (adapting it for local needs) or create your own one from scratch.

We have chosen the former way and picked DOOCS e-logbook from DESY. Main changes concerned localization (since Russian uses cyrillic letters, not latin) and data feeding mechanism (due to different model of logging from applications).

E-logbook is deployed at VEPP-5, and consensus of opinion is that it should have been done "the day before yesterday".

GIS

Geographic Information Systems (GIS) look very promising for use at an accelerator complex. Their application can include equipment inventory system, automatic generation of visual control screens, and so on. VEPP-5 traditionally uses free software, so we made a survey of available free GIS tools and systems. Free GIS tools haven't yet reached the same state of maturity as commercial ones, so the choice is neither unambiguous nor trivial.

For now, we have settled on Mapserver. Preferrable data storage is PostGIS, which provides GIS extensions to PostgreSQL relational database. Server-side PHP software, allowing web-browsing of GIS data, is currently being developed. The next stage would be a standalone "VEPP-5 Explorer" application to provide all control parameters in symbolic circuit view.

Main complexity lies in the fact that AutoCAD drawings, used as source of GIS data, are poorly suitable for this task: they contain disjoint shapes and lots of useless information. So, these source files have to be hand-edited before storing data in GIS. Additionally, logical object information (also lacking in .DWG files) have to be added.

Unfortunately, the principal conclusion is that free GIS software is too immature at present time. There's no one complete solution, and everyone has to do many things (including basic ones) himself.

WEBIFICATION

Besides control software as such, control system requires a number of supporting services.

As web gains ubiquity, and many services are web-oriented or at least have a web-interface, web has become the most reasonable platform for supporting services.

So, we allocated a dedicated Linux PC with Apache, which runs following services:

- Subversion, which stores all source files of control system and related software.
- Shift planning system.
- Hardware configuration database.
- E-Logbook.
- GIS.
- Web-presentation of current status.

This approach also significantly eases and quickens installation of control system for other projects.

FUTURE DEVELOPMENT

Hardware

In 2008 long-serving PIII-800s with RedHat-7.3 will be replaced with modern PCs. Our current OS of choice is CentOS, as most stable yet free.

Another change concerns display system. Modern videocards support very high resolutions, and there are appropriate monitors on the market at affordable prices. So, we plan to replace 4 individual (albeit joined via Xinerama) 1152×864 monitors with a single large one, such as Samsung 305T (30", 2560×1600). Elimination of interscreen gaps will make work more comfortable and enables more rational use of screen space.

Software

As CX is almost 10 years old now, the volume of desired improvements has exceeded some "critical mass". More precisely, cost of correct implementation of all improvements would be higher than re-writing the whole system from the ground up. So, the next, 4th version of CX is currently being written from scratch. This allows to base a new version on different principles.

Control systems, like most other software, are usually created according to more or less fixed requirements. But when requirements (unexpectedly!) change or, even worse, interoperation with a different control system is required, the developer finds himself in a trouble, since (s)he has to touch many places and aspects of the code.

The point of plug-ins technology[5] is replacement of a fixed implementation (or a fixed set of bindings) with a dispatcher and a set of implementatios, which can be easily extended later. This allows to take specific and potentially mutable areas of code out of a basic framework. These parts are placed into separate modules and some dedicated API is used for interaction. More modules can be added later as required, adding unforeseen functionality to otherwise intact system. Plug-ins are often implemented as dynamically loaded libraries (`.so`-files in Unix, DLLs in Windows), so that extension can be performed "on the fly", without a need to recompile the core.

Most control systems already use this approach at one particular place: drivers are usually separated from other code and are developed separately. CX goes one step further and since 2006 also uses plug-ins approach for screen knobs[4].

CX version 4 is based on plug-ins approach at *all* levels (see Fig.2).

Modular implementation of data-server access layer for clients would enable CX clients to easily access hardware, controlled by other control systems (EPICS, LabVIEW, ...). On the other hand, replacement of a single data-access protocol implementation in CX-server with a modular frontends architecture would allow other control systems to obtain CX-controlled data. Thus, use of plug-ins approach makes integration with other control systems an easy task.

Windows Integration

Another improvement is Windows support. Initially CX was postulated to work under Unix/Linux only. But in practice most physicists wish to have access to control system from their Windows desktops. So, CXv4 is being developed to support Windows from the beginning.

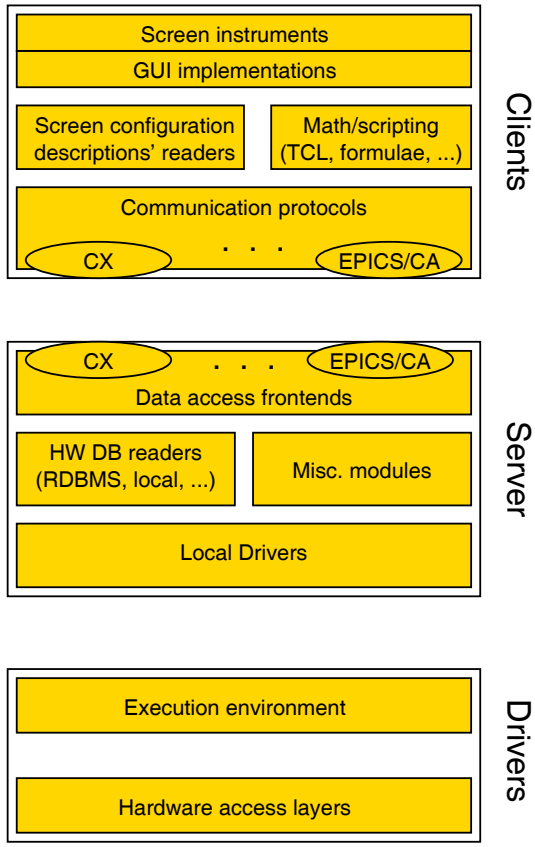

Figure 2: Plugin-based architecture of CX version 4 (shown are pluggable parts).

REFERENCES

[1] P.V.Logatchev et al, "Status of VEPP-5 Injection Complex", Proc. RuPAC-2006
`http://rupac2006.inp.nsk.su/ready/moao10.pdf`

[2] D.Bolkhovityanov et al, "PowerPC-based CAMAC and CAN-bus controllers in VEPP-5 Control System", Proc. PCaPAC'2005 (Hayama, Japan, March 2005)
`http://conference.kek.jp/pcapac2005/↩ paper/WEB4.pdf`

[3] D.Bolkhovityanov, "VEPP-5 Injection Complex Control System Software", 2007, Ph.D. thesis (in russian)
`http://www.inp.nsk.su/~bolkhov/publs/↩ bolkhov_phd_final.pdf`

[4] D.Bolkhovityanov, "UI-oriented Approach for Building Modular Control Programs in VEPP-5 Control System", Proc. PCaPAC-2006 (Newport News, VA USA, October 2006)
`http://www.inp.nsk.su/~bolkhov/publs/↩ pcapac2006_pUI_paper.pdf`
`http://www.inp.nsk.su/~bolkhov/publs/↩ pcapac2006_pUI_poster.pdf`

[5] "Plugin", Wikipedia, the free encyclopedia
`http://en.wikipedia.org/wiki/Plugin`