ISBN: 978-3-95450-197-7 doi:10.18429/JACoW-RUPAC2018-THPSC12

UPGRADE OF APPLICATION-LEVEL SOFTWARE OF VEPP-5 INJECTION COMPLEX

F. A. Emanov*, D.Yu. Bolkhovityanov, V.V. Balakin, D.E. Berkaev, Yu.A. Rogovsky
The Budker institute of nuclear physics, 630090 Novosibirsk, Russia

Abstract

VEPP-4 and VEPP-2000 experimental facilities are fed with e+e- beams from VEPP-5 injection complex(IC). Application-level software of IC control system is based on CXv4 framework. This software includes a set of engineering and debugging tools for all the hardware, a database with high-level information and configuration tools, machine mode manipulation system, automatic control and data analysis programs. The software architecture and implementation is described.

INTRODUCTION

VEPP-5 injection complex (IC) [1–3] is an electron and positron beam source for BINP colliders VEPP-4 [4] and VEPP-2000 [5]. Beams are transferred to colliders by K-500 channel. Layout of injection complex with colliders is shown on Fig. 1.

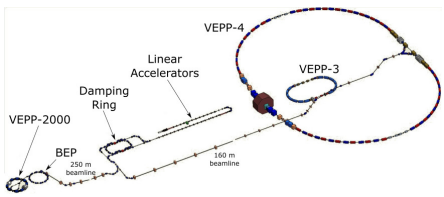

Figure 1: Injection complex and colliders layout.

Injection complex had had a long period of serving as test facility until 2015. During this time injection complex control system software was based on CX and EPICS frameworks and application level software mostly consisted of engineering GUI programs. This software set allowed us to run injection complex as test facility but was not acceptable for routine operation with beam users. It was required to create centralized configuration and operation tools and automatize frequent processes. Since two frameworks were applied it was needed to support both of them in our software. Some of required software was implemented earlier [6]. At the end of 2017/2018 experimental season we decided to fully migrate base control system software to CX since this framework served more than 90% of injection complex control hardware. By the time most of migration work has been completed. Current state of injection complex software is described bellow.

SOFTWARE DEVELOPMENT TOOLS

Software development is carried out in C/C++ and Python. In order to simplify application-level software development we created fast Cython bindings to CX client libraries called pycx4. Currently pycx4 can operate in CX event-loop or Qt4 or Qt5 ones. We use PyQt for joined CX and EPICS programs and for GUI software development. We use cothread.catools for EPICS devices access. In order to be able to use cothread in modern environment we added PyQt5 support and Qt version auto selection to cothread.

A set of PyQt4/5 widgets with embedded CX access was developed to simplify GUI programs creation. Since we are using Qt Designer for rapid software prototyping, designer plugins for CX widgets also implemented. We are currently considering possibility to bind CX to Taurus SCADA.

CONFIGURATION AND ACCELERATOR MODE DATABASES

Large experimental facilities like injection complex require lots of configuration information. In general this configuration information is structured data about machine. Therefore it can be presented in form of graph of some accelerator elements and their relations. Several attempts of implementation configuration tools in a control system agnostic way were made [7, 8]. But to our consideration these tools have not been developed to production-level. So we decided to develop our own configuration tools without solving too general problems. Databases are an appropriate engine to solve problems of configuration and storage for accelerator data like operation logs and modes. We have been using Postgresql since it is the most advanced open source database and it is suitable for all our needs. Currently we migrated to Postgresql 10 since native table partitioning was introduced in this version.

We designed databases and configuration tools in order to reach the following goals:

- to create configuration source for accelerator mode management system and operation logs.
- to create configuration generation scripts for CX.
- to minimize database deployment efforts.
- to create effective storage for accelerator mode data.
- to make mode data tolerant to software and hardware changes.

In order to store control system structure we designed "line" of objects: namespace, device, devtype, channel. Namespace, device and channel are significant parts, and devtype serves for reduction amounts of object relations. We believe logical accelerator systems can be formed to tree structure. Device can be a part of few different systems. Sometimes there is some information about device which is not common to all devices. In this case we can store

* f.a.emanov@inp.nsk.su

ISBN: 978-3-95450-197-7 doi:10.18429/JACoW-RUPAC2018-THPSC12

it as JSON metadata. Developed structure of objects and relations is shown on Fig. 2

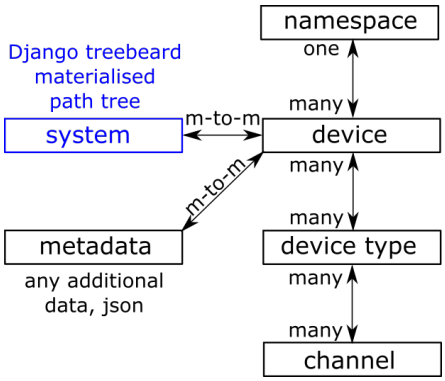

Figure 2: Configuration database objects.

In order to edit data or structure of database along with ordinary database tools we use Django [9] applications. In some cases we use Django ORM for GUI programs. First, self-implemented adjacency list tree was used. But then we switched to materialized path tree implementation from Django treebeard [10].

Data of accelerator modes is stored in separate database. Information about control system channels generated from configuration database is added to fullchan table. So we have some data redundancy here but it simpifies mode selection and increase performance. Mode is stored in mode table, channels information is stored to modedata. Mode can have labels which are used for automatic control and stored to modemark table. Since modedata table can grow to a big size we used native postgresql partitioning with Partman extension to simplify management. Mode database structure is shown on Fig. 3. This database keeps only data required to save or load accelerator modes, but complex mode management tools may require some data from configuration database.

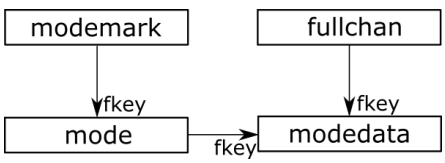

Figure 3: Mode database structure.

MACHINE MODE MANAGEMENT

Injection complex operation requires rapid manipulations with states of accelerator logical subsystems. Such manipulations are involved in storage-extraction loop, switching particles or beam users. Centralized mode management by special daemon was implemented in order to solve these problems.

Manager daemon continuously reads all the control system channels therefore mode can be instantly saved. If mode is used by automatic applications it's labeled with special marks and pre-loaded to manager cache from database. In this case mode selection time becomes neglectable.

We created GUI client program which can manage modes in a very flexible way. Another clients of mode loader is automatic software which implements routine operations of supplying beam users as shown on Fig. 4.

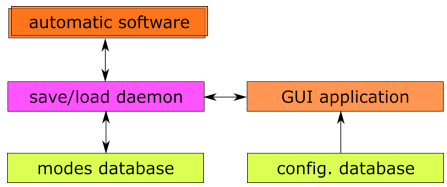

Figure 4: Mode management software structure.

AUTOMATIC SOFTWARE

Injection Complex has 0.2 - 2 Hz storage-extraction loop duration and sometimes it is required to switch particles every 30-60 seconds. Storage-extraction loop and switching between particles and users are implemented in two programs. In 2017/2018 season these programs were having simple connection through CX postbox channels to synchronize selected particles. With this solution operator's attention in a regular run required few times per hour. In order to make injection complex operation more convenient modes used by automatic programs were color-encoded. GUI program for particle and user switching is shown on Fig. 5.

Figure 5: Particle and user switching GUI program.

There are two more obvious needs for automatic programs:

- machine parameter stabilization,
- abnormal situation and failure analysis and automatic recovery.

We have implemented abnormal situations and failures analysis and automatic recovery service in general way and applied it to vacuum and magnetic systems. And we are going to extend this software to all other systems and connect it to automatic machine operation programs. Machine parameter stabilization has not been implemented yet. We are going to try it in 2018/2019 season.

DATA PROCESSING

There are a lot of data processing tasks which can help in machine studies and operation like:

- devices stability analysis,
- extra parameter calculations,

ISBN: 978-3-95450-197-7 doi:10.18429/JACoW-RUPAC2018-THPSC12

- signal amplitude and time change determination,
- BPM data preprocessing,
- image data preprocessing.

We believe the best way is to create microservice for each particular processing problem. Service performs necessary calculations and publishes results to CX server, then any other program can use the results. Example of frontend program which shows processing results is shown on Fig. 6 We are constantly deploy such microservices according to current needs.

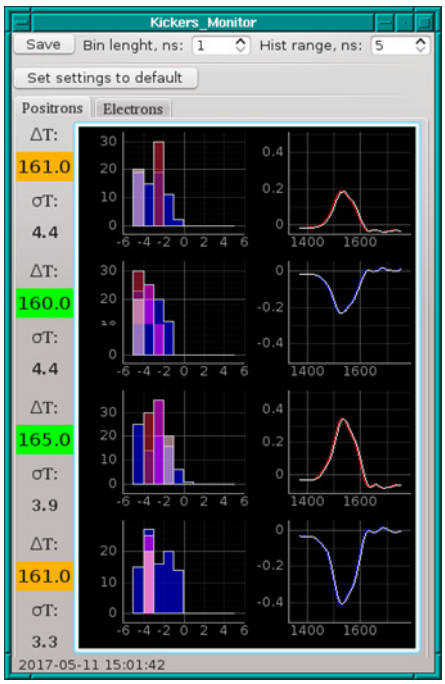

Figure 6: Frontend for kicker signal analysis.

CONCLUSION

During 2017/2018 automatic software development dramatically reduced number of required operator's actions. Implemented development tools significantly simplified application-level software development and increased interest to it. Database structure and function upgrade significantly increased performance of database applications.

REFERENCES

[1] D. Berkaev et al., "VEPP-5 Injection Complex: two colliders operation experience", in Proc. IPAC'17, Copenhagen, Denmark, May 2017, paper WEP1K026.

[2] F.A. Emanov et al., Feeding binp colliders with the new VEPP-5 injection complex, Proceedings of RuPAC2016, St. Petersburg, Russia, WEXMH01.

[3] K.V. Astrelina et al., Production of intence positron beams at vepp-5 injection complex, JETP 2008, vol. 106, issue 1, pp 77-93.

[4] P. A. Piminov "Status of the Electron-Positron Collider VEPP-4", Proceedings of IPAC'17, Copenhagen, Denmark

[5] Yu. Shatunov et al., "Project of a New ElectronPositron Collider VEPP-2000," EPAC'2000, Vienna, Austria, p.439

[6] F. Emanov et al., "Present status of VEPP-5 injection complex control system", Proceedings of RuPAC2016, St. Petersburg, Russia, paper THPSC085

[7] A. Makeev et al., "Centralized Software and Hardware Configuration Tool for Large and Small Experimental Physics Facilities", Proc. ICALEPCS2013, San Francisco, CA, USA, October 2013, paper TUPPC022, http://accelconf.web.cern.ch/AccelConf/ICALEPCS2013/papers/tuppc022.pdf

[8] M.A. Ilina, P.B. Cheblakov, "Applying Ontological Approach to Storing Configuration Data", Proc. ICALEPCS2017, Barcelona, Spain, October 2017, paper THMPL05, http://accelconf.web.cern.ch/AccelConf/icalepcs2017/papers/thmpl05.pdf

[9] Django web framework https://www.djangoproject.com/

[10] Tree implementation for Django https://django-treebeard.readthedocs.io