DEVELOPMENT OF THE MALFUNCTIONS DETECTION SYSTEM AT VEPP-2000

O. S. Shubina†, A. I. Senchenko
BINP SB RAS and Novosibirsk State University, Novosibirsk, Russia

Abstract

In 2007, the creation of the electron-positron collider VEPP-2000 (Fig. 1) was completed at the Institute of Nuclear Physics of the SB RAS. The VEPP-2000 collider facility consists of various subsystems, and a failure of any subsystem can lead to the incorrect operation of the complex for several hours or even days. Thus, there is a need to create software that will warn about possible malfunctions. To accomplish the task, software was developed consisting of three modules. The first performs automatic verification of compliance data obtained from the accelerator complex and rules describing the correct subsystems operation. The second module is a user-friendly web interface that displays information about the state of the complex in a convenient way. The third module acts as some intermediary between the first and the second. It processes messages arriving at the message queue and redirects them to all subscribed clients via the web socket. This article is devoted to the development of test software, that is currently running on the VEPP-2000 control panel.

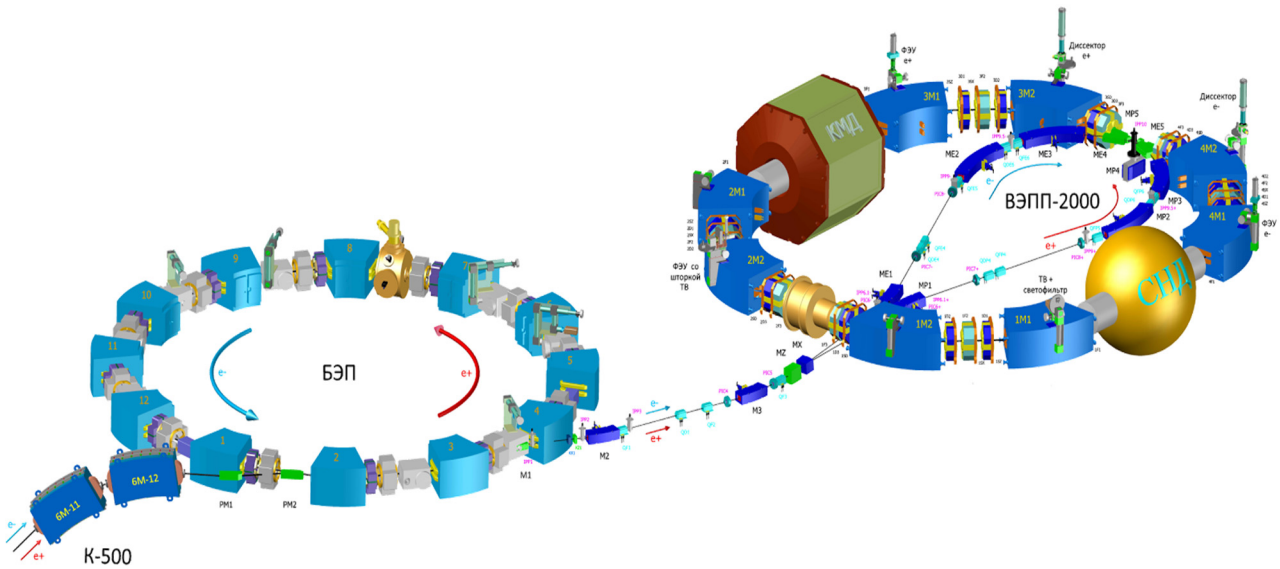

Figure 1: VEPP-2000.

INTRODUCTION

The VEPP 2000 is an electron-positron collider, that was commissioned at the Budker Institute of Nuclear Physics. The VEPP-2000 acceleration complex consists of a few main subsystems: BEP booster ring and VEPP-2000 collider ring. Beam diagnostics is based on 16 optical CCD cameras that register the synchrotron light from either end of the bending magnets and give the full information about beam positions, intensities and profiles. In addition to optical BPMs [1], there are also 4 pick-up stations in the technical straight sections and one current transformer as an absolute current meter. The VEPP-2000 control system [2] has a complex structure and consists of about 4000 channels for monitoring and control. This complicates the timely detection of incorrect operation of the complex or any malfunctions. To solve this problem, software was developed consisting of various parts. The core of the software is a troubleshooting module, that performs automatic validation of rules stored in special configuration files. The following component is responsible for delivering information about the status of the complex to subscribers via the web sockets. Also for the timely notification of new subscribers about the state of the complex uses caching of the state snapshot in the Redis database. Finally, the graphical interface provides a convenient view of all the necessary information.

MODULE FOR CHECKING THE STATE OF THE COMPLEX

The hardware part of the VEPP-2000 accelerator complex has a complicated architecture; therefore, a group system has been developed for the convenience of troubleshooting. Each group is a collection of several channels or

† olgashubina2011@gmail.com.

subsystems of the complex, united according to some principle. Thus, during processing, the system operates directly with these groups, rather than with individual channels. Information about groups is stored in a special configuration file. After initialization of all groups and obtaining the necessary data (point 2), the process of troubleshooting is started in accordance with the rules specified in the configuration file. First of all, the system creates a set of objects corresponding to each individual group and storing the information about the rule that is applied to this group, critical and boundary values and a reference to the corresponding data buffer. Each object is an instance of a specific class in which these or other troubleshooting rules are implemented. Then, with a periodicity, a cyclical group check is performed for the presence of malfunction, as well as for the activity of the data source. After the check, a report on the complex status is generated and sent to the message broker RabbitMQ.

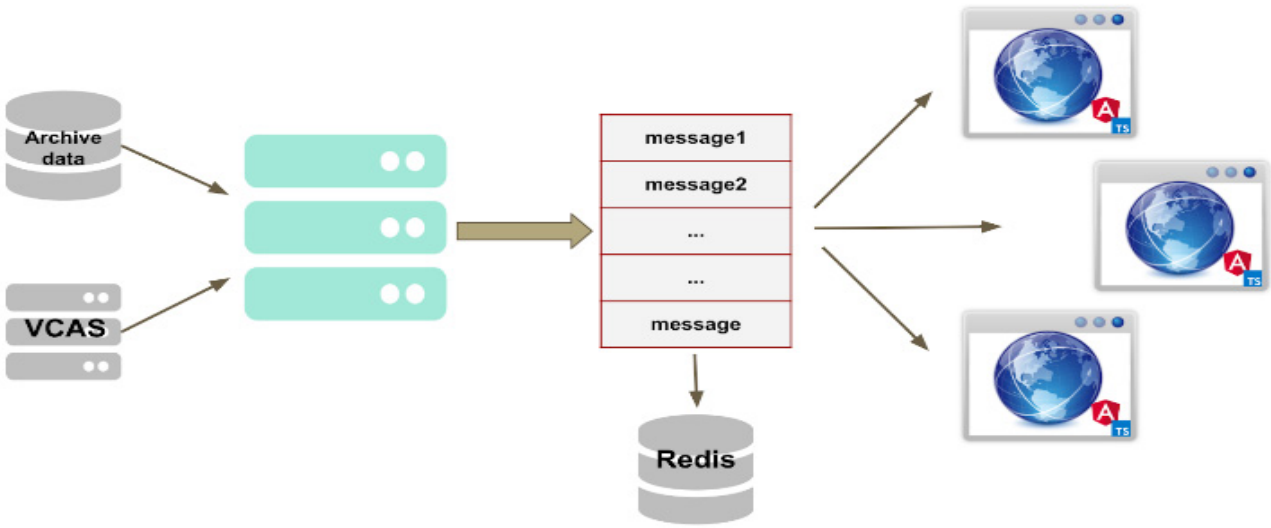

Figure 2: General scheme.

CREATE AND UPDATE DATA BUFFER

Before starting the troubleshooting process, a data buffer is created in the system. Data enters the buffer from the database through a specially developed API for interacting with the accelerator complex archive data. The buffer is filled with data on all the necessary channels for a certain period of time, that is specified in the configuration file groups.

After creating the buffer, it starts the automatic update process. The system subscribes to the VCAS server to obtain online data. Data storage is organized in the form of a ring buffer, i.e. each time, when new data is received, they are saved to the buffer, and the data that is outdated is deleted from the buffer.

The main problem that arose when designing this software is the use of memory. There was a need to create a common memory buffer with which two processes will work in parallel: automatic buffer update and troubleshooting process. At the same time, the solution had to have good performance and be easily incorporated into the developed software and the management system as a whole. This problem was solved by queuing in distributed memory from the multiprocessing library of Python.

MESSAGE BROKER AND COMPLEX STATUS SNAPSHOT

For the distribution of information between customers, a special service, developed on the basis of the message broker RabbitMQ [3], is responsible. It is a web socket server. Any user or software can subscribe this server and receive information on the state of the complex with a certain periodicity.

After passing the next check iteration, a report on the state of the complex is generated and sent to the message broker. Then, the report is processed and sent to all existing subscribers. To reduce traffic and the load on the system, information about the state of the complex is not sent at every iteration of the test, but only when the state changes. Therefore, there may be a problem that the new subscriber does not receive information about the state of the complex for a long time.

To solve this problem, it was decided not only to transmit information online, but also to save the state cast in-memory data structure store Redis. The storage scheme of the state cast has the structure shown in Fig. 2. And now, when adding a new subscription, it will read up-to-date information from the repository, and then receive updates via the web socket when the state changes.

Each author should submit the PDF file and all source files (text and figures) to enable the paper to be reconstructed if there are processing difficulties

MALFUNCTION DETECTION METHODS

Currently, several methods of troubleshooting have been developed to test this system (Fig. 3), in part for current

and vacuum systems. To troubleshoot current systems, three simple rules are used:

- check of the equality of the set current to the specified parameter
- check of the difference between the set and obtained value
- the standard deviation of the normalized value

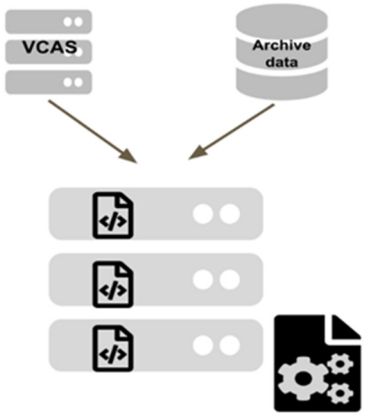

Figure 3: Malfunction detection scheme.

The second fault finding algorithm was developed for those subsystems in which measurements are rarely made to detect a malfunction in time. And in this case, the algorithm performs:

- interpolation of values by the most optimal function
- extrapolating values for current or future time

Templates are provided for recommended software and authors are advised to use them. Please consult the individual conference help pages if questions arise.

WEB GUI

For the convenience of displaying information about the state of the complex, a graphical web interface (Fig. 4) was developed based on the Angular 5 framework [4]. The first consists of colored indicators of each group. Here, red means that an error has occurred on this subsystem, yellow indicates a situation close to critical, green indicates a correct operation. Also, if the data source for any of the subsystems is unavailable, then it is displayed in gray. In addition to the color indication, there is a module in that an informational error message is displayed indicating the time of the error and the subsystem on that it occurred.

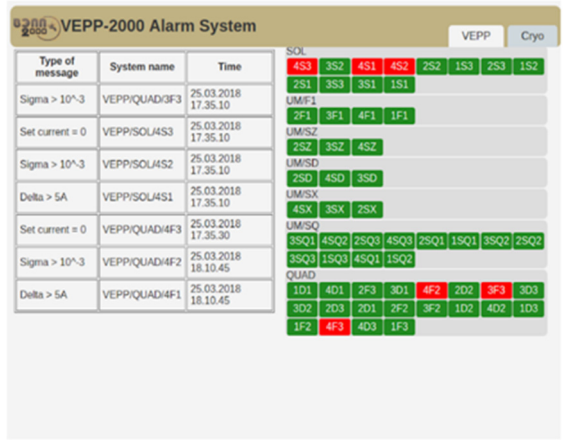

Figure 4: WEB GUI.

CONCLUSION

At present, a prototype of a troubleshooting system at the pulping VEPP-2000 has been implemented and launched in test mode. As part of this work, several algorithms for checking the state of the complex were implemented and connected to an automatic check. In the future we plan to expand the set of rules, as well as review the way they are stored. In addition, the task is to improve the performance of this system, and solve a number of problems associated with distributed memory and timely notification of the state of the complex. It is also planned to expand the scope of their research not only as applied to troubleshooting, but also to their automatic prevention.

REFERENCES

[1] Yu. Shatunov *et al.*, "Project of a New ElectronPositron Collider VEPP-2000", in *Proc. EPAC'00*, Vienna, Austria, p.439.

[2] A. Senchenko *et al.*, "VEPP2000 Collider Control System", in *Proc. PCaPAC'12*, Kolkata, India, Dec. 2012, paper FRCB04.

[3] RabbitMQ official site, `https://www.rabbitmq.com`

[4] Angular5 official site, `https://angular.io`