DEVELOPING AND VALIDATING OPC-UA BASED INDUSTRIAL CONTROLS FOR POWER SUPPLIES AT CERN

Michael Ludwig, Marc Bengulescu, Ben Farnham, Jonas Arroyo Garcia, Pablo Gonzalez Jimenez, Fernando Varela, CERN, Geneva, Switzerland

Abstract

The industrial control systems of CERN's experiments are undergoing major renovation since 2017 and well into CERN's second Long Shutdown (LS2) until the end of 2019. Each detector power-supply control system runs several hundred software instances consisting of many different components in parallel on a large scale, broadly distinguishable as servers and clients. Our accumulated experience during LHC runs proves that some complex control issues are impossible to detect using stand-alone components on a small scale only. Furthermore, new components must be developed well before the electronics becomes available, without impact on operations. Moreover, during LS2, the improved and now widely established Open Protocol Communication Unified Architecture (OPC-UA) replaces OPC-DA as middleware protocol. For these reasons, we developed a simulation environment to emulate the real, and valuable, CAEN power-supply electronics underneath the OPC-UA servers. This distributed simulation is configurable to mimic and exceed the nominal conditions during production and provides a repeatable setup for validation. This paper discusses the functionality and use of this simulation service.

WHY A SIMULATION?

The control system for the CERN experiments can be segmented into three layers. Firstly, the electronics layer for power supplies and many other components, which includes commercially available modules connected to the detector sub-parts. Some CERN experiments need many thousands of power channels laid out in “electronic trees” spanning a wide variety of characteristics and behaviour, generally using networked modules from different commercial vendors running black box firmware.

The second layer provides the OPC-UA [1,2] interfaces to the different kinds of power supply modules. OPC-UA provides standardized abstraction and communication, and replaces its now obsolete predecessor OPC-DA. Several tens of server instances connect to the power channels for a typical experiment, each server with its specific electronic tree.

Lastly, the client layer provides classical supervisory control and data acquisition (SCADA) services including logging, error handling and analysis, history data storage and functional abstractions like finite state machines and interfaces for specific detector functionality. The two uppermost layers rely entirely on the accurate and robust translation of the physical electronic trees into the software representation with the intended functionality. A commercial object-orientated SCADA system, WinCC-OA [3], is used to realize much of the upmost layer. This now mature control system has to undergo updates and functional fixes, which all have impact on mission criticality in some way, but where their deployment should not require any dedicated production time.

The investigation of issues and the actual deployment of updates requires very careful coordination and is, generally, a delicate process for the two following reasons. Firstly, still affordable laboratory tests can never approach the same complexity, scale and diversity as a real running system with inputs and errors. Any new versions of control components, concerning also new versions of the electronics, need testing and validation through all control layers. Problems related to scaling, which were reproducible in the experiments in the past but not in laboratory tests, and could not be confirmed by the vendor either, need to be investigated on appropriate scale and complexity, to then be followed up adequately.

Secondly, a system with delicate high voltage detector-hardware must not be used for testing critical and safety related software. Even if one would take the risk, it would be very hard to identify and re-produce any complex failure scenario for further debugging. A satisfying validation for a working control system of that scale could be obtained by performing global centralized “dry runs” using real electronics, but disconnected high voltage output stages, yet such runs are deemed impractical and too expensive. For these reasons a scalable low-level platform which provides **ven**dor **u**nified **s**imulation (VENUS) for power supplies together with a realistic behaviour is needed as an efficient and flexible solution which can also support training and development prototypes.

REQUIREMENTS

Scale and Validation

Both issues, scale and validation, can be efficiently addressed with a simulation environment which emulates the electronics of the first layer. Large scale “developer tests” are possible, being much more independent on whether the electronics is available, in production, or even only a planned purchase. Replaying complex and demanding scenarios should push various controls software components in the upper layer to their respective limits, thus providing useful stress tests. If scale and complexity of the production systems can be significantly exceeded in the simulation, we assume with good confidence that our systems are adequately validated.

Performance and Complexity

Changing parameters in the electronic tree create events with associated data payload, which are pushed upwards to the subscribing server. Event rates of several kHz are

common for real electronic trees of some hundred channels. Saturation typically occurs from about 10kHz onwards, then some buffering and filtering strategies are applied from the server on and upwards. The simulation should be able to deliver event rates significantly exceeding the real electronics. It should also potentially scale better than the most complex real electronics tree and it should be configurable and manageable in compatible ways. Evidently, it must integrate seamlessly to emulate the behaviour of the first layer, and it should be capable of producing and repeating scenarios beyond nominal conditions to prompt associated error behaviour.

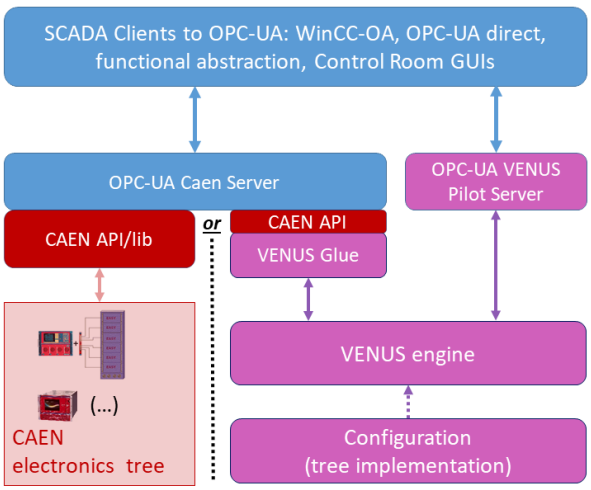

Figure 1: VENUS architecture in the case of CAEN. Colors represent types of communication: vendor specific (red), Ømq-protocol (violet) or OPC-UA (blue).

Furthermore, the simulation should replace power supplies of all major commercial systems used at CERN [4], in terms of software. All of the following discussion refers to the CAEN power supply systems, since they are a widely used system at CERN, provide deep functional complexity including radiation hard sub-trees and feature specific powering options. Simulation of the remaining vendors follows the same principles adapted for their systems.

ARCHITECTURE

The VENUS architecture is shown in Fig. 1. The SCADA layer interfaces to OPC-UA CAEN servers, which communicate with either the real electronics through the CAEN API, or through the replacement API for simulation. A configured VENUS Engine provides the same underlying behaviour by software and is managed by a separate OPC-UA Pilot Server. The simulation replaces the first control layer entirely, where the API for each type of real power supply defines the behaviour of the simulation in each case. The VENUS glue establishes isolated communication to the VENUS engine for the specific CAEN API. Each instance of a VENUS engine is configured to reflect the specific electronics tree to be played.

Configuration

The configuration defines the instances of each type of module including any electronic subtrees for the radiation hard versions. The firmware of each module determines its real API, which evolves over many years and must be reproduced in all cases. Configurations can be discovered from existing electronic trees, designed to produce large scale stress test scenarios with subtrees and mixed module types, define consistency tests and also model yet-to-be-purchased electronics.

Pilot Interface

Further functionality beyond the scope of the vendor API is needed to provide accurate power supply behaviour. A dedicated OPC-UA pilot server communicates directly to the VENUS engine in order to perform:

- channel load management; a load model needed to draw currents,
- ramp switching; an external signal switching between two parameter sets for pre-ramping and ramping,
- current-limit tripping; an excess current switches the channel off in a well-defined manner,
- artificial noise; in order to mimic randomly occurring behaviour like current spikes and degrading loads,
- bulk-channel operations; like tripping selected channels by decreasing their loads to draw currents exceeding their nominal limits,
- event clock tuning to produce higher or lower event rates for use in stress test,
- status and health monitoring of the engine.

The specific functionality in the SCADA layer relating to the pilot is kept carefully isolated from all other clients.

DESIGN AND INTEGRATION

Building Standards

The building, integration and execution of VENUS as a scaling service is shown in Fig. 2. A framework for rapid OPC-UA server development, quasar [5], is used to generate server skeleton code for a configured quasar project. This quasar project configuration defines the layout and characteristics of the OPC-UA address space; interface to the SCADA clients. Quasar supports two toolkits, Unified Automation (commercial) [6] and Open62541 (open source) [7], and guarantees compliance to OPC-UA. The artefacts, VENUS engine and OPC-UA servers, are built from specific source code together with the generated protocol buffer message formats [8] and the quasar generated codes. All artefacts are continuously built using Jenkins-CI to guarantee integrity.

API Serialisation

Each function call of the vendor API, which is implemented in plain C for CAEN, is independent of every other call to a large extent and no state information is kept in the API. Therefore communication between the CAEN API and the object-orientated engine can be realized in transactional pairs of request-reply messages in the VE-

NUS glue. Each message pair maps to a specific API function call and its results. Message pairs need to be executed as transactions to protect against re-entrant calls and parallel requests stemming from multi-threaded servers. The initial calls for hardware discovery are issued serially from the server and the simulation engine replies back with the corresponding part of its electronic tree.

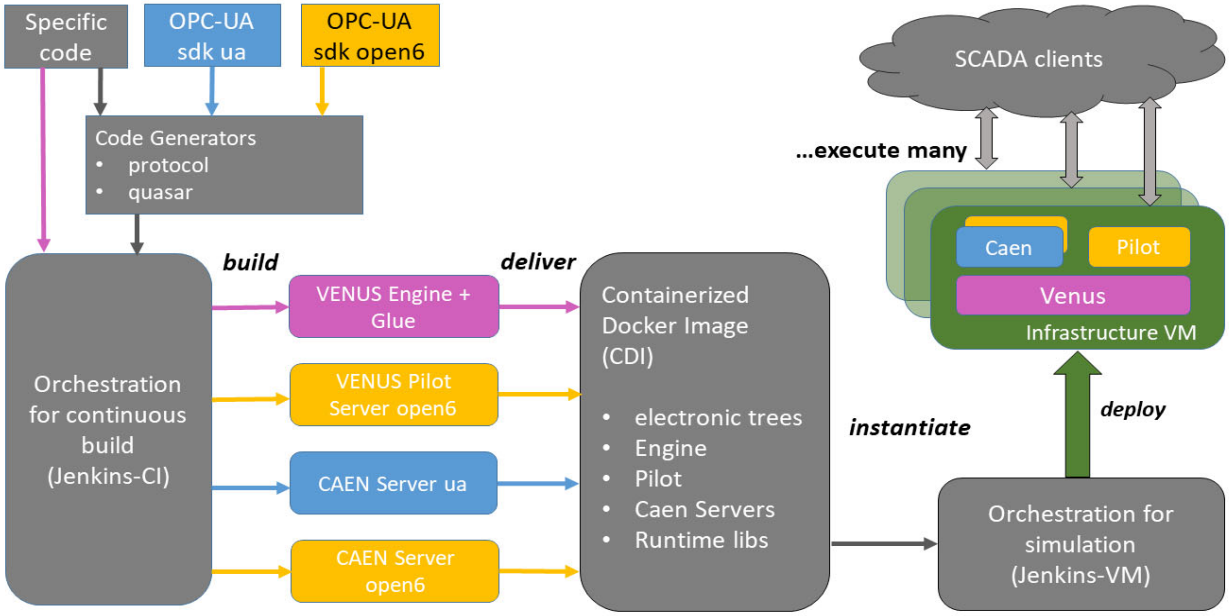

Figure 2: Building, integration and deployment as a service in a cluster of virtual machines. The workflow stages source, build, deliver, instantiate, deploy and execute are shown. Colour and shape coding: standards (pink, yellow, blue), tools (grey, round corners), infrastructure service (green), source codes (sharp corners), complex clients (cloud).

Deployment and Scalability

A set of all servers and the engine, together with their runtime dependencies and previously validated configurations for the electronic trees is delivered into a containerized docker image (CDI) [9] as shown in Figure 2. Both OPC-UA sdk-flavours for the servers may be used and compared, but the open6-sdk is preferred for the pilot due to licensing restrictions. The CDI can also be dissociated into separate units which communicate over network for further distributed stress testing. The orchestration for simulation, which is optimized for continuously running services, selects one of the valid configurations for deployment and runs it in a virtual machine (VM). Many VMs with identical or different configurations are used to realize simple or complex scenarios for the SCADA client layer. Up to one hundred VMs can be managed using the available resources, where each VM offers some ten to several thousand channels.

CONCLUSION AND OUTLOOK

The need for a fast, flexible and highly scalable simulation service arises from the ongoing control system renovation at CERN for LS2. As long as the vendor API in question is easily serializable, a clean solution based on distributed networking with protected message pairs can be naturally found. A seamless integration, without any impact on the very complex SCADA layer, is provided by replacing the vendor's implementation of the API and adding a pilot interface for further management. Setups can be run all-in-one on single computers or widely distributed. Since the VENUS implementation uses C and C++ it can cover event rates well above and below the characteristics of the real electronics. The construction of challenging stress tests becomes thus possible, leading to a systematic and repeated validation of a large, complex and mature system, as used in the experiments. Further work is ongoing to extend VENUS for the other vendors.

REFERENCES

[1] W. Mahnke, S.-H. Leitner and M. Damm, *OPC Unified Architecture*. Springer, Berlin Heidelberg, Germany, 2009.

[2] The OPC Foundation, "OPC Unified Architecture", http://opcfoundation.org/opc-ua/

[3] WinCC-OA, http://www.etm.at

[4] CAEN, http://www.caen.it;
ISEG, http://www.iseg-hv.com;
WIENER, http://www.wiener-d.com

[5] P. P. Nikiel, B. Farnham, S. Schlenker, C.-V. Soare, V. Filimonov and D. Abalo Miron, "Quasar – A generic framework for rapid development of OPC UA servers", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, paper WEB3O02.

[6] Unified Automation GmbH, "C++ based UA Server SDK".

[7] Open62541, http://open62541.org

[8] Google Protocol Buffers, https://developers.google.com/protocol-buffers/

[9] Docker container technology, http://www.docker.com