Emulation to the rescue

The virtual emulator framework simplifies process control system testing

MARIO HOERNICKE, RIKARD HANSSON – When process control systems are going through their final test phases, a software simulation of the process is often used to test the control system response. However, this simulation is usually focused on the functional part of the process control system – namely, the centralized control applications. Distributed control functions, located within subsystems or field devices, are often neglected. ABB's virtual emulator framework (VEF) integrates and automatically configures subsystem emulators and so enhances functional testing. The networks of emulators that VEF's virtualization-based technology enables the user to configure appear and behave like the actual automation system, subsystems and networks.

To tackle the emulation complexity encountered in the integration and factory acceptance tests, ABB has developed the virtual emulator framework.

Traditional process control systems (PCSs) commonly consisted of a single subsystem – the distributed control system (DCS) – that interacted with sensors and actuators, and displayed process states or alarms and events on an operator control station. In contrast, the modern PCS is a high-performance product that incorporates – alongside the still-dominant DCS – fieldbuses, intelligent devices and other subsystems in order to provide more flexible and intelligent functionality.

The Foundation Fieldbus (FF) is one of the most prevalent fieldbuses found in modern PCSs [1].

Title picture
ABB's virtual emulator framework greatly simplifies process control system testing by including distributed control elements in the overall simulation. Shown is part of an oil and gas plant in Hammerfest, Norway.

The FF distributes control functions for execution in field devices. In some cases, cascaded loops are used where the inner cascade is located in field devices and the outer cascade is located in the DCS controllers. In other words, the FF can embody an entity that is altogether more complex than the DCS.

Another important example of a DCS-connected subsystem is the electrical control system. The IEC 61850 standard, for instance, describes a fieldbus used for substation automation. The intelligent electronic devices (IEDs) with which it deals are comparable to the controllers of a traditional DCS.

However, there is a price to be paid for the increased functionality these subsystems bring: For each subsystem integrated into the PCS, different engineering methods are required and highly complex interfaces have to be created.

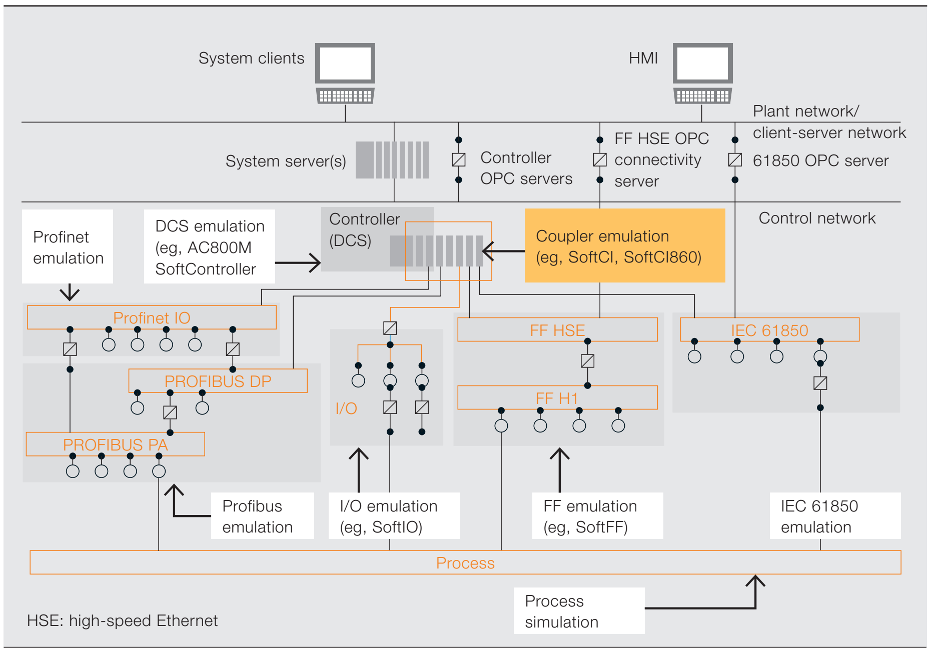
Alongside this increasing sophistication – and, consequently, greater engineering effort – customers also demand a shorter time to market. When the shrinking number of engineers is factored in, it becomes clear that today's PCS engineering world faces a daunting task. Nevertheless, engineers must guarantee acceptable quality, so efficient and complete tests are essential.

Challenges in testing process control systems

The more complex the automation, the better testing has to be. As is the case in most software projects, testing is usually done throughout the development phase of a PCS. But there are two test stages at the end of the engineering effort considerably more important than the rest: The factory acceptance test (FAT), where the PCS engineers test critical parts in cooperation with, and in the presence of, the customer, in order to validate engi-

The modern PCS is a high-performance product that incorporates fieldbuses, intelligent devices and other subsystems in order to provide more flexible and intelligent functionality.

1 The virtual "hardware-in-the-loop" test bed can replace large parts of the automation hardware.



neering results ("Is it the right product?"); and the integration test, which checks the entire PCS functionality, including, for example, controller parameters or subsystem interfaces. The integration test is performed before the FAT to verify engineering results according to the specification ("Is the product right?"). Both of these stages require the control system hardware to be staged on the shop floor – so servers, field devices, etc. need to be present, configured and administrated. This is a complex and time-consuming task.

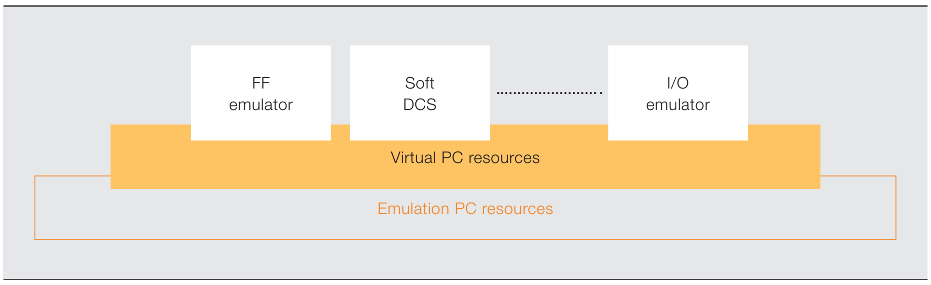
The VEF is able to plan and deploy entire emulation networks for the automation system in question with just a few mouse clicks.

An important aspect of the integration test is the testing of the connections between control functions located in the DCS and the subsystems (eg, FF or IEC 61850). Since the field components are regularly sent directly to the site in order to save shipping and staging costs, this testing is often not possible [2]. Recent initiatives have developed imitations of the automation system hardware to replace the hardware missing at integration test and FAT time → 1.

To stimulate the automation system emulators and to provoke reactions to process states, a process simulation is often used. For the integration test and FAT, lightweight simulation models are usually accurate enough, but in case operator training simulation or virtual commissioning is required for the plant, high-fidelity models can be used for these two tests. The connection between the PCS and the process simulation model, as well as the orchestration of the simulation environment and higher-level simulation functionality, can be performed using software such as the Extended Automation System 800xA Simulator [3].

Although the plant can be completely imitated and tested using process simulation combined with automation emulation and subsystem emulation, emulation is still not well established in the integration test and FAT. Investigations showed that this is due to the immense configuration effort required for the separate tools, combined with the high administration effort for the emulation PCs and the impracticalities involved in engineers mastering a large variety of emulation tools.

2 Virtual-machine template with installed emulators



3 Parameters for emulator integration

Parameter	Usage
Number of Ethernet interfaces	Each emulator requires a defined number of Ethernet interfaces for each instance in which it is executed. The VEF uses this to configure the virtual hardware.
Number of simultaneously executable emulator instances	Depending on the implementation of the emulator, it is capable of executing a defined number of instances on a single PC or VM. This number is used to calculate how many VMs are required.
Number of simultaneously executable subsystem instances per emulator instance	Some emulators can emulate a number of subsystem instances within a single instance of the emulator. VEM needs this number, as well, to create the instances of the emulators and assign a subsystem instance to each.
RAM required per emulator instance	Indicates whether an emulator instance can be executed within a given PC environment and is used to configure the RAM of the VMs.
Object type of the emulated subsystem	Each subsystem (or each emulated object) has an object type. The object type is used to identify the required emulation tool.

The user is able to monitor and debug the configuration of the selected subsystems and the automation system. The network behaves like – and appears to be – the original automation hardware and subsystem hardware.

To tackle this complexity, ABB has developed the virtual emulator framework (VEF).

The virtual emulator framework

The VEF is able to plan and deploy entire emulation networks for the automation system in question with just a few mouse clicks. Since it uses virtualization to automatically create the emulation networks, the virtual hardware can also be automatically created and configured, according to the automation system and PCS topology.

Integration of emulators into the VEF

As a prerequisite for the automatic generation of emulation networks, the emulation tools need to be tightly integrated into the VEF. Conceptually, the VEF uses virtual machine (VM) infrastructure to integrate emulators and to automatically generate the virtual appliances and the virtual networks for the emulation. The emulators are installed on a VM template in the same way as on a physical PC → 2.

The advantage of the virtual infrastructure is that the template can be easily duplicated – allowing many instances to be created without user interaction and without staging new physical PCs.

Since the emulators have different capabilities, depending on their implementation and the emulated subsystem type, a few parameters have to be provided to the VEF for each emulator type. The VEF uses these parameters to plan the virtual network topology and virtual hardware for each emulator type → 3.

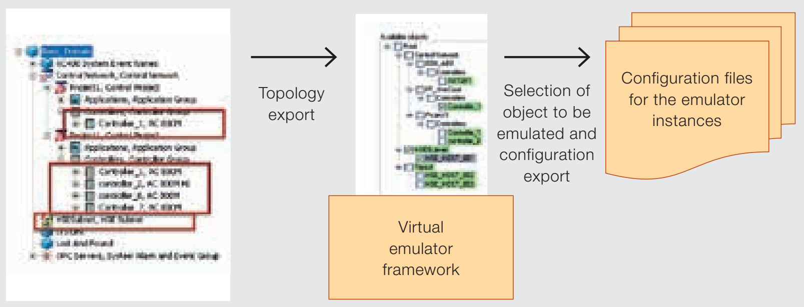
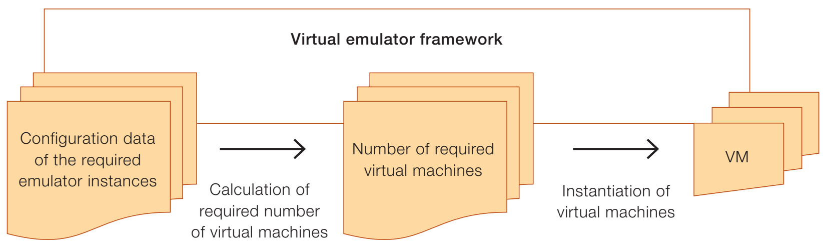
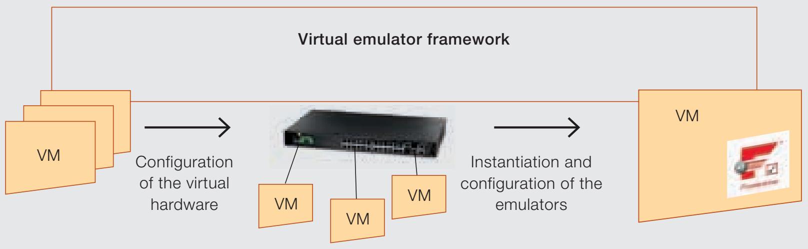
Besides the emulator-type-specific parameters – that have to be specified just once for each emulator type – the IP addresses for each object to be emulated are required. This parameter is instance-specific and therefore needs to be separately configured for each instance of an emulatable object.

Finally, the VEF uses this information to automatically configure the virtual networks and automatically establish communication between the emulator instances and the plant network.

Algorithm to generate a virtual emulator network

Based on the VM template, the VEF is able to apply a multistage algorithm. With this algorithm, the VEF is able to generate the required VMs, configure the virtual hardware according to the requirements of the emulator instances, configure the network interfaces and execute the emu-

4 Algorithm to create the virtual emulation network

Step No.	Description	Illustration
1	**Export of the automation system topology** In the first step, the topology of the automation system is exported from the PCS engineering system. The engineered topology within the PCS contains the automation system components, eg, controllers. Based on the object type, emulatable components can be identified and highlighted.	 Topology export Selection of object to be emulated and configuration export Configuration files for the emulator instances Virtual emulator framework
2	**Selection of parts that need to be emulated** Usually, only certain parts of the plant are tested at the same time and the user can select these. The configuration files for these instances are then created. Those files are later used to automatically configure the emulator instances.	
3	**Calculation of required number of VMs** Based on the parts selected for emulation, the required number of VMs can be evaluated, whereby some rules have to be adhered to: – The maximum number of Ethernet cards per VM may not be exceeded. – The maximum number of executable instances on one VM may not be exceeded for any emulator. – The maximum number of emulatable objects per emulator instance may not be exceeded. – Virtual RAM may not exceed physical RAM.	 Virtual emulator framework Configuration data of the required emulator instances Calculation of required number of virtual machines Number of required virtual machines Instantiation of virtual machines VM
4	**Instantiation of virtual machines** Based on the calculated number of required VMs, the VM instances are created from the template.	
5	**Configuration of virtual hardware** Based on the parameters of the emulator types, the hardware of the specific VM instances is configured. The required number of Ethernet interfaces and the required RAM for each individual instance are configured.	 Virtual emulator framework VM Configuration of the virtual hardware VM VM VM Instantiation and configuration of the emulators VM
6	**Configuration and execution of emulator instances** In the last step, the VM instances are started, the configuration files created in step 2 are used to configure the emulator instances, and the emulator instances are executed.	

The VEF automatically configures the virtual network and automatically establishes communication between the emulator instances and the plant network.

lator instances with the instance-specific configuration → 4.

When the algorithm has been successfully executed, the user is able to monitor and debug the configuration of the selected subsystems and the automation system. The network behaves like – and appears to be – the original automation hardware and subsystem hardware.

VEF system architecture

To evaluate the developed algorithm, a prototype was developed for the PCS System 800xA. Two emulators, the AC 800M SoftController [4] and SoftFF [2] – including SoftCI [5] – were integrated into the prototype.

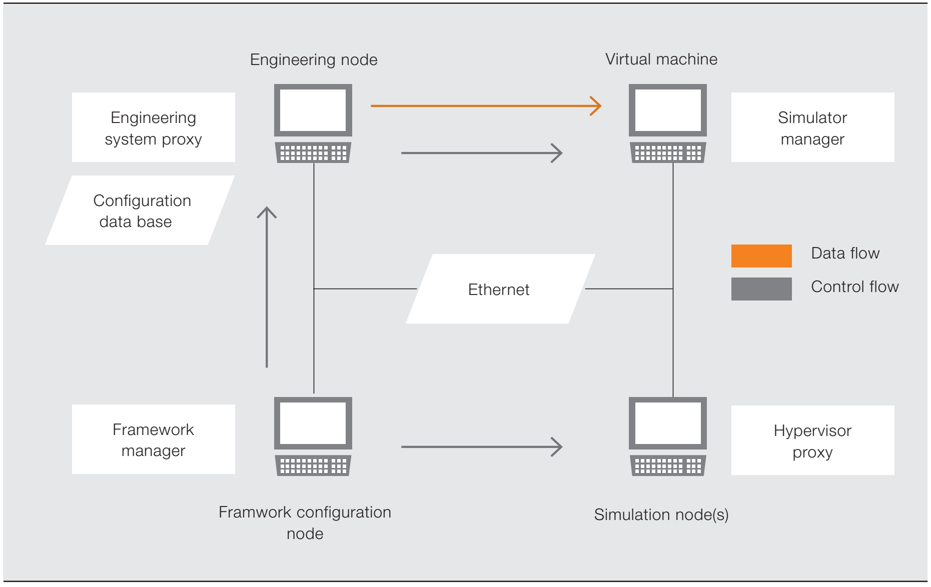
Since System 800xA has a distributed system architecture and the focus has been on large system emulations, the architecture of the VEF is also of a distributed nature. As a base for the communication between the different nodes within the VEF and System 800xA, TCP/IP was chosen.

The VEF consists of four node types that can be installed on the same physical PC or in VMS or it can be distributed – just as in

5 System architecture: virtual-emulator-framework nodes

Node type name	Usage	Residence
Configuration node	The configuration node is used to select the parts that will be emulated.	A PC inside the System 800xA network; not necessarily a System 800xA node.
Engineering system access node	This node type consists of a single service that is used to interface with the engineering system of System 800xA. It is used to export the automation system topology and the configuration files for the emulator instances.	A System 800xA node that contains the required engineering tools for exporting the configuration and that has access to the aspect directory.
Orchestration node for the hypervisor	Orchestration of the hypervisor controlling the VMs is required as well. This node makes the connection to the hypervisor (eg, VMware vSphere) and to start, stop, create, etc. the VMs.	This node needs to be installed and executed on a PC that has access to the hypervisor (eg, PC with vSphere Client installed). It must be located in the same networks, like the other nodes.
Virtual machine node	The VM node configures the emulator instances and starting and stopping the emulator instances inside the VMs.	This node is installed in the VM template and executed automatically on every VM instance.

6 Virtual-emulator-framework communication structure



System 800xA → 5-6. Except for the configuration node, the VEF consists only of MS Windows services that do not require user interaction.

As a base for the prototype, the VMware platform was chosen. The prototype has been evaluated using VMware Workstation for small emulation networks that can be deployed on a single PC and VMware vSphere Hypervisor for large virtual emulation networks, based on the VMware ESXi platform. Both scenarios have been tested with positive results.

Fast FAT

The VEF was developed to enable an efficient integration test and FAT preparation, with little manual intervention, and it has been successful in this.

The VEF requires only two user interactions to deploy an entire virtual emulation network: The selection of the objects to be emulated and the configuration of the private cloud/virtualization PCs. The purchase, staging and configuration of special emulation PC hardware is no longer necessary.

By using virtualization, the configuration of the hardware interfaces required for the emulator instances also becomes unnecessary. Now, the configuration can be performed automatically in accordance with engineering data exported from the engineering tools.

The prototype demonstrates the feasibility of implementing this solution for a complex PCS. Hence, the VEF is a scalable solution to efficiently configure and deploy heterogeneous emulation networks for process control systems.

Mario Hoernicke
ABB Corporate Research
Ladenburg, Germany
mario.hoernicke@de.abb.com

Rikard Hansson
ABB Process Automation, Simulator Solutions
Oslo, Norway
rikard.hansson@no.abb.com

References
[1] H. Sato, "The Recent Movement of Foundation Fieldbus Engineering," in SICE Annual Conference, Fukui, 2003, pp. 1134–1137.
[2] M. Hoernicke *et al.*, "The fieldbus outside the field: Reducing commissioning effort by simulating Foundation Fieldbus with SoftFF," *ABB Review* 1/2012, pp. 47–52.
[3] System 800xA Simulator – Improve safety and productivity through simulation. (Accessed 2015, January 22). Available: http://www.abb.com/product/seitp334/a5beb9cb235cd859c125734700336e07.aspx
[4] System 800xA system guide summary. (Accessed 2015, January 22). Available: http://search.abb.com/library/Download.aspx?DocumentID=3BSE069079&LanguageCode=en&DocumentPartId=&Action=Launch
[5] M. Hoernicke, T. Harvei, "Virtually speaking: DCS-to-subsystem interface emulation using SoftCI," *ABB Review* 2/2013, pp. 58–63.