# Virtually speaking

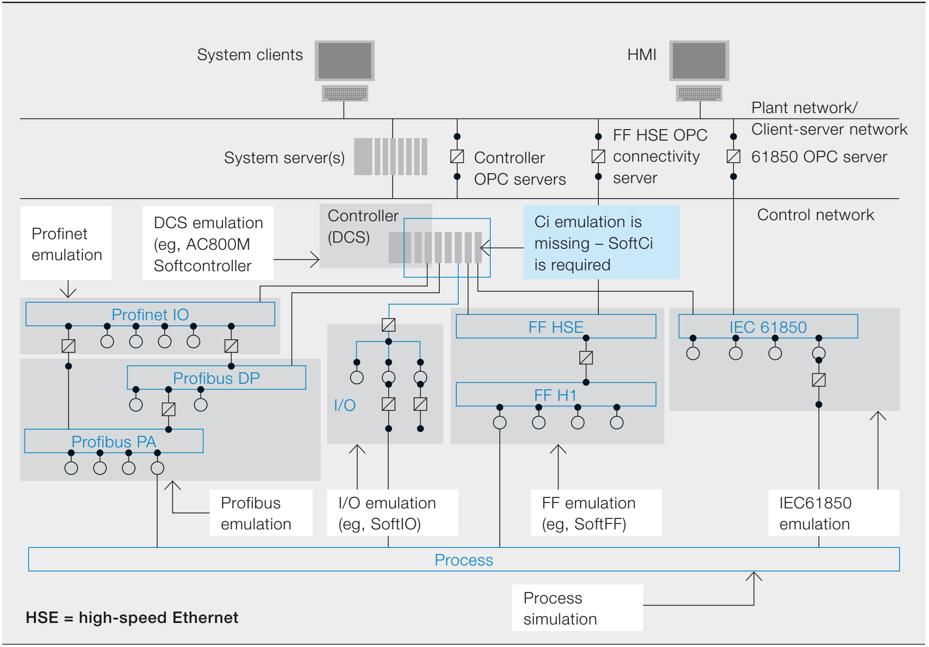
## DCS-to-subsystem interface emulation using SoftCI

MARIO HOERNICKE, TRYGVE HARVEI – A modern industrial plant is a thing of bewildering sophistication. Hundreds, or even thousands, of sensors, meters and various intelligent field devices send data to, and receive data from, automation controllers in a vast, precisely orchestrated torrent of bytes. Building a full hardware test infrastructure in the factory for such a complex system is wholly infeasible, so software emulators are used to imitate its constituent subsystems. Over the years, these have become very refined. However, missing from such emulation environments has been a satisfactory way to emulate the interfaces between the subsystems and the distributed control systems they serve. A research initiative launched to solve this problem resulted in SoftCI. SoftCI is meant to replace AC800M controller communication interfaces during integration and factory acceptance tests.

**Title picture**
Emulating the control environment of a complex industrial plant, such as this liquified natural gas plant in Norway, in software and hardware is a difficult job. A new emulator that ably imitates subsystem interfaces is now making the job a lot easier.

Today, each system has to be tested separately in its own context without regard for system or communication aspects.

1 Emulators for process control system hardware – situation today. SoftCI is needed for communication between the components.



Obviously, a complete process plant cannot be built in the lab to test new automation systems. So, the plant's functions are emulated in hardware and software. Whereas much of the plant's functionality can now be successfully imitated, the communication between the distributed control system (DCS) emulator and subsystem emulator is usually not. Due to the increasing usage of fieldbuses and Ethernet in plants, and the criticality of error-free communication, this is a major problem

Since the communication functions of different CIs are often similar, one generic communications concept could cover many CI types.

The Next Generation Factory Acceptance Test (NGFAT) initiative aims to solve this problem [1]. Included in the goals of the initiative is the development of a generic controller communication interface (CI) emulation – SoftCI. As a result, a software development kit (SDK) has been implemented that allows the user to integrate CI emulation into existing subsystem emulators or execute it in standalone mode.

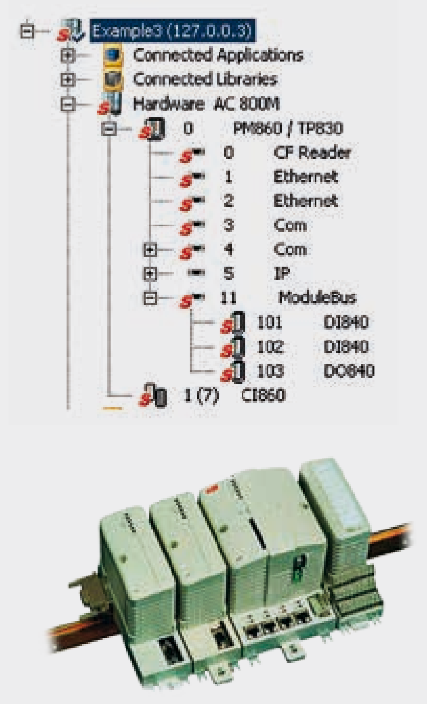
SoftFF [2], a Foundation Fieldbus simulator, can be used to verify the integrity of the emulation.

## SoftCI environment

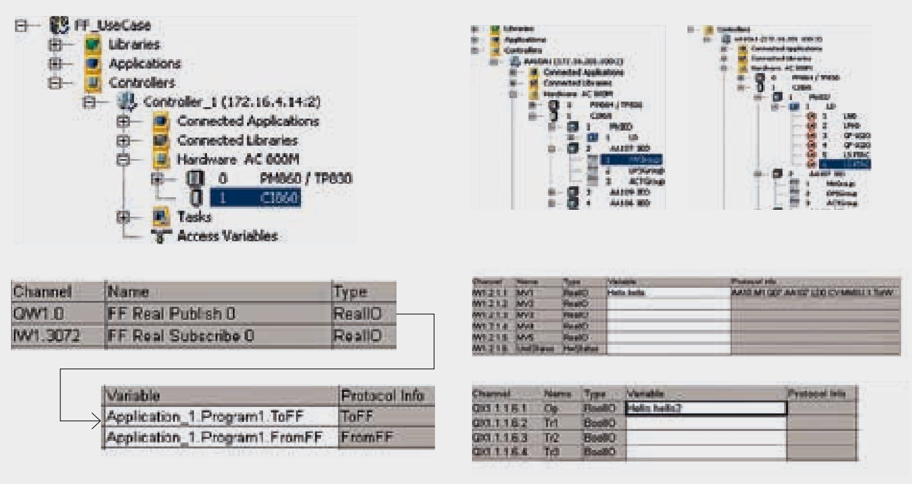
During an FAT, the entire system must be comprehensively emulated in order to test the consistency of the engineering effort. Everything that is not performed on a standard PC is executed on an emulator. Different emulators represent different subsystems and fieldbuses → 1. There is, for instance, a DCS/controller emulation available that focuses on IEC61131-3 code [3]. Today, each system has to be tested separately in its own context without regard for system or communication aspects.

It is against this background that SoftCI was developed. SoftCI handles vertical communication between the DCS and subsystem emulations. A plant will have many subsystems performing many different tasks, but usually only one DCS type. In order to show that the SoftCI concept works, attention has been focused on the ABB AC800M controller.

2 Topology of AC800M. Here, the usage of an FF coupler.



3 Characteristics of Foundation Fieldbus CI860 and IEC61850 CI868, showing the mapping done inside the CIs in Control Builder M (CBM).



| Channel | Name | Type |
|---|---|---|
| QW1.0 | FF Real Publish 0 | RealIO |
| IW1.3072 | FF Real Subscribe 0 | RealIO |

| Variable | Protocol Info |
|---|---|
| Application_1.Program1.ToFF | ToFF |
| Application_1.Program1.FromFF | FromFF |

| Channel | Name | Type | Variable | Protocol info |
|---|---|---|---|---|
| IW1.2.1.1 | MV1 | RealIO | Hello hello | [illegible] |
| IW1.2.1.2 | MV2 | RealIO | | |
| IW1.2.1.3 | MV3 | RealIO | | |
| IW1.2.1.4 | MV4 | RealIO | | |
| IW1.2.1.5 | MV5 | RealIO | | |
| IW1.2.1.6 | [illegible] | [illegible] | | |

| Channel | Name | Type | Variable | Protocol Info |
|---|---|---|---|---|
| QX1.1.1.6.1 | Op | BoolIO | Hello hello2 | |
| QX1.1.1.6.2 | Tr1 | BoolIO | | |
| QX1.1.1.6.3 | Tr2 | BoolIO | | |
| QX1.1.1.6.4 | Tr3 | BoolIO | | |

**Characteristics**

| Foundation Fieldbus (CI860) | IEC61850 (CI868) |
|---|---|
| – One CI for input and output variables<br>– FF Signals are copied to 1131 variables (and vice versa)<br>– Mapping is configured in a single table for each CI | – One CI for input and output variables<br>– IEC61850 signals are copied to 1131 variables (and vice versa)<br>– For each IED, a separate mapping table is engineered<br>– IEDs from the field are always inputs<br>– AC800M is treated as being an IED<br>– AC800M IED is always an output |

## In focus: AC800M

The AC800M controller contains input and output (I/O) modules, but also fieldbus and subsystem couplers – the CIs. A coupler is a specific component used to connect the physical fieldbus on the device level ➔ 2.

A CI860, for instance, transfers data between the AC800M and a Foundation Fieldbus (FF) device. It is engineered using a table to map IEC61131 control application variables to FF signals. While IEC61131 variables are connected to the control application, FF provides the counterpart within the corresponding engineering tool, FieldBus Builder FF (FBB FF). The FF signals are attached to the FF function blocks using FBB FF. By connecting the signal to the function blocks, the IEC61131 variable to the control application and mapping the signal and the variable in the mapping table, the engineer can establish a value exchange between AC800M and FF.

Another example of a CI is the CI868 for IEC61850 [4] networks. For IEC61850 the structure below the CI looks different. The CI contains a separate mapping table for each intelligent electronic device (IED) that is attached to the controller. In this table, the signals coming from the subsystem can be mapped to the control variables, as for FF.

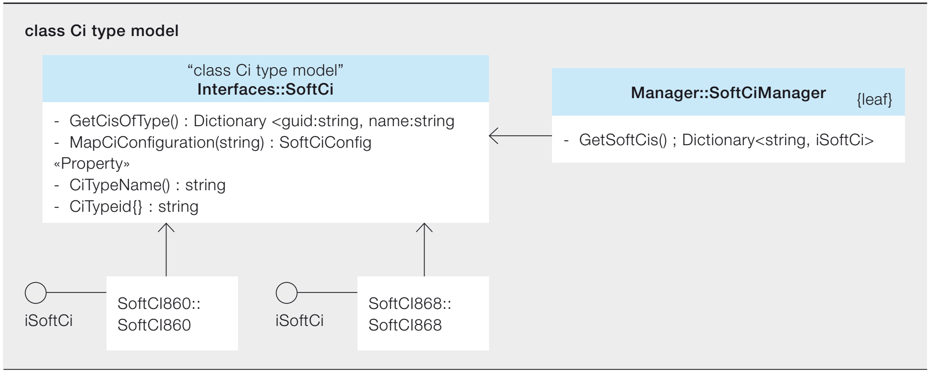
Except for the difference that FF uses a single table and IEC61850 uses several, the CI scheme is similar ➔ 3.

Since the communication functions of different CIs are often similar, one generic communications concept could cover many CI types. For example, most CIs are used to simply exchange values from IEC61131 variables to subsystem signals and do not provide supplemental functionality. Thus, a configured CI could be modeled as a mapping table.

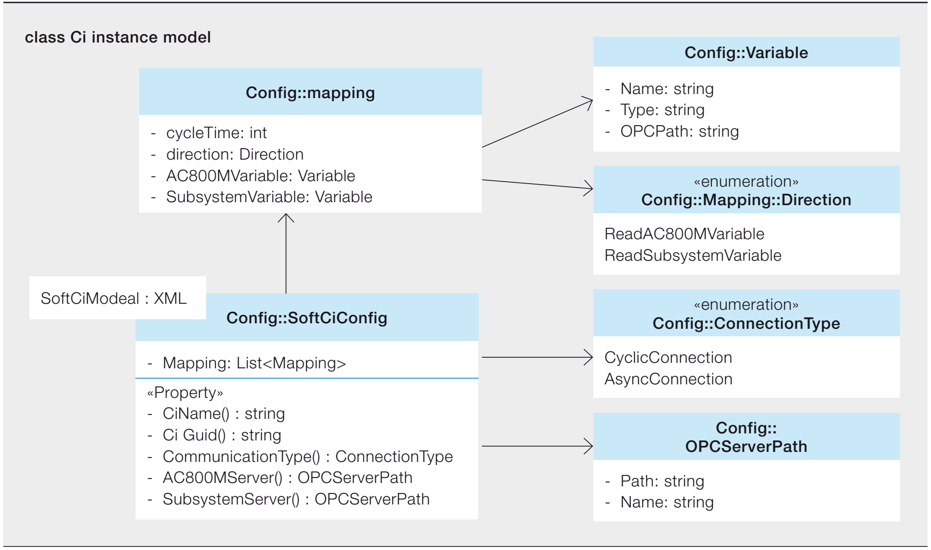
The model of the CI is not the only part that needs to be developed. During runtime, a generic communication method has to exchange values with the AC800M. This is required for every type of CI since the CIs are always exchanging data with the AC800M. On the field level, so to speak, a communication to the subsystem emulator is required. The subsystem emulation differs depending on the subsystem type. The emulation can be ABB-owned, open-source or third party and can be characterized as being either open or locked.

SoftCI handles vertical communication between the DCS and subsystem emulations.

4 CI model for a specific type



5 CI model of a specific instance



Open emulators allow further functionality to be included: The SoftCI could be integrated into the subsystem emulator and directly exchange the variables from the AC800M with the subsystem emulator's variables. The CI model is executed within the subsystem emulation.

Locked emulators do not allow expanded functionality. These tools, however, should still be usable with SoftCI and, therefore, a different vertical standard communication method needs to be evaluated. This implies that SoftCI should be able to execute CI models in standalone mode and that a CI model might be developed manually and must, therefore, be stored in a human-interpretable form.

## Generic CI emulation

The scene has been set, then, for the development of generic functionality to emulate different CI types.

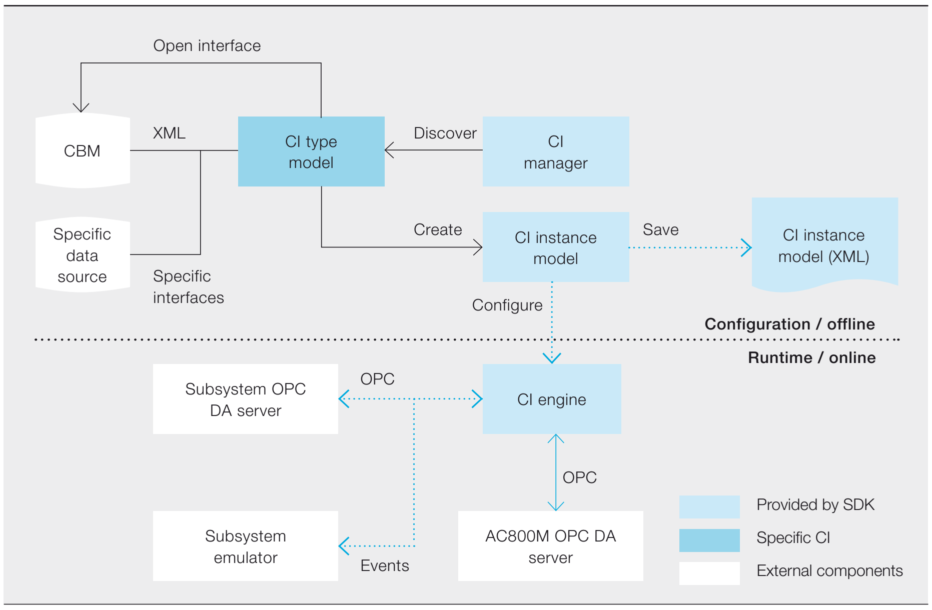
## CI models

A major part of the functionality is the creation of CI models. CI models consist of one part that describes the CI type and another part that describes the CI instance. The model for the CI type is developed once and is the same for each instance, while the instance model is created for every instance separately.

The meta-model for a CI type is delivered as a .NET interface that can be implemented in a class ➔ 4. It consists mainly of the descriptive properties, name and ID of the CI type, and a method to get the instances of the described type from the 800xA engineering workplace. If automatic creation of instance models is desired, a method to create the model can be implemented. If this is done, SoftCI can identify the instances and create the models for the instances without user interaction.

The meta-model for the instances is delivered as .NET classes, but is storable as XML ➔ 5. The instance meta-model does not use methods, but data properties only to describe the specific instance. In general, it is the representation of the mapping table from CBM.

6 SoftCI SDK modules



A CI860, for instance, transfers data between the AC800M and a Foundation Fieldbus device.

In addition to the mapping table from CBM, the name and ID of the CI instance is described. With those, SoftCI can show the user which instances are emulated at present. Additionally, the communication type of the CI instance needs to be described. The communication type might by acyclic or cyclic, depending on the communication method of the subsystem. Mixtures of both are allowed.

## CI communication

As mentioned above, a method to communicate with the AC800M Softcontroller had to be found, so the communication parameters for an instance have to be modeled.

The chosen communication method is OPC Data Access (DA). OPC DA is supported by the AC800M Softcontroller and is usually preconfigured for the actual hardware and the production system. Hence, reconfiguration for the emulation is not required. Since there might be several OPC servers used in the process control system, the path to, and name of, the AC800M OPC server for the specific controller instance has to be described within the instance model.

## SoftCI modules

The SoftCI SDK consists of a number of software modules → 6. Besides the model, a CI manager is supplied that can be used to identify CI type models. It is possible that several types are used within a single process control system for a plant. The manager can identify them and automatically create instance models for each without user interaction.

The modules delivered with the SDK are provided as class libraries that can be used to quickly create CI emulation within subsystem emulators. Additionally, a very small and simple user interface is provided that can be used to execute CI emulation in standalone mode.

## Testing times

The concept described has been successfully tested by implementing CI emulation for FF. The concept has been proven in conjunction with IEC 61850 emulation too. SoftCI868 has been implemented in prototype form and value exchange between IEC 61850 emulation and the AC800M softcontroller functions.

## Towards the virtual plant

SoftCI is an SDK that provides generic emulation functionality for AC800M communication interfaces and, in doing so, closes a white spot in the emulator landscape. Although it might not be usable for every kind of CI, the majority of CI types can be addressed. SoftCI860 has already been im-plemented in SoftFF and is ready for use in integration testing and FATs (a first pilot project is currently underway).

SoftCI is an evolutionary step towards exhaustive virtual commissioning functionality for ABB's Extended Automation System 800xA – ie, another important step towards perfection of the virtual plant.

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

**Trygve Harvei**
ABB Process Automation
Oslo, Norway
trygve.harvei@no.abb.com

**References**

[1] M. Hoernicke, J. Greifeneder, *"Next Generation Factory Acceptance Test," Annual Report ABB Corporate Research Germany,* pp. 83–89, 2011.

[2] M. Hoernicke, P. Weemes, H. Hanking, *"The fieldbus outside the field: Reducing commissioning effort by simulating Foundation Fieldbus with SoftFF," ABB Review,* 1/2012, pp. 47–52.

[3] IEC61131-3: *Programmable controllers – Part 3: Programming languages.* Edition 2.0, 2003.

[4] IEC61850: *Communication networks and systems in substations, 2003.*