Hitachi DCS Emulator Design to Support NPP Simulator Implementation

Y. Nakashima[1], K. Ishii[1] and D. Chiba[1]
[1] Hitachi Ltd., Information & Control System Company, Ibaraki-ken, Japan

Abstract

Nuclear Power Plant (NPP) simulators are the main means for operator training and as such are a crucial part of the NPP operation life-cycle. Efficient development and testing of NPP software and design concepts require a robust platform mirroring the design and configuration of the operating plant. DCS Emulator and full-scope simulator (FSS) technologies support both these objectives for the entire NPP life-cycle allowing users and operators to implement, test and use actual control and information software applications and designs. This paper describes Hitachi's latest simulator development and challenges in implementing a DCS emulator to provide a code emulation platform for developing NPP software.

1. Introduction

Hitachi's latest DCS product, the G–HIACS νSAFE (pronounced "nu-safe") DCS is the latest version of Hitachi's successful "HIACS" series of DCS focusing especially on IEC 61508 Functional Safety (SIL 2) and IEC 61131-3 (programming languages) compliance. To support νSAFE application software developers, the product software platform includes the G-HIACS νSAFE DCS Emulator with the fundamental concept of "total emulation by software". The DCS Emulator will support not only the product's internal development strategy but also new NPP FSS development. As a follow-up to Hitachi's 2010 CNS DCS emulator paper [1], this paper describes Hitachi's latest simulator and emulator challenges and development targets.

2. Hitachi's latest simulator experience, Japanese domestic NPPs

Since designing and building its first BWR simulator (the first BWR NPP simulator in Japan) in 1989, Hitachi has built 5 more NPP simulators (4 BWR type and 1 ABWR type) in Japan. Hitachi's new NPP simulator contract is for the turn-key supply of an FSS for ABWR currently under construction. As shown in Figure 1, the new ABWR fuel load is scheduled to begin in December 2013 and commercial operation in November 2014; however, the simulator will be operational in December 2012. Hitachi is also planning a refurbishment for a BWR-5 in 2013 and a new ABWR simulator in 2014. Current plans include the construction of 2 new ABWR NPPs; however, exact effects on the construction schedule of the March 2011 earthquake and tsunami are not yet known. For Hitachi, new simulator construction opportunities are scheduled to be continuously available for the next 10 years.

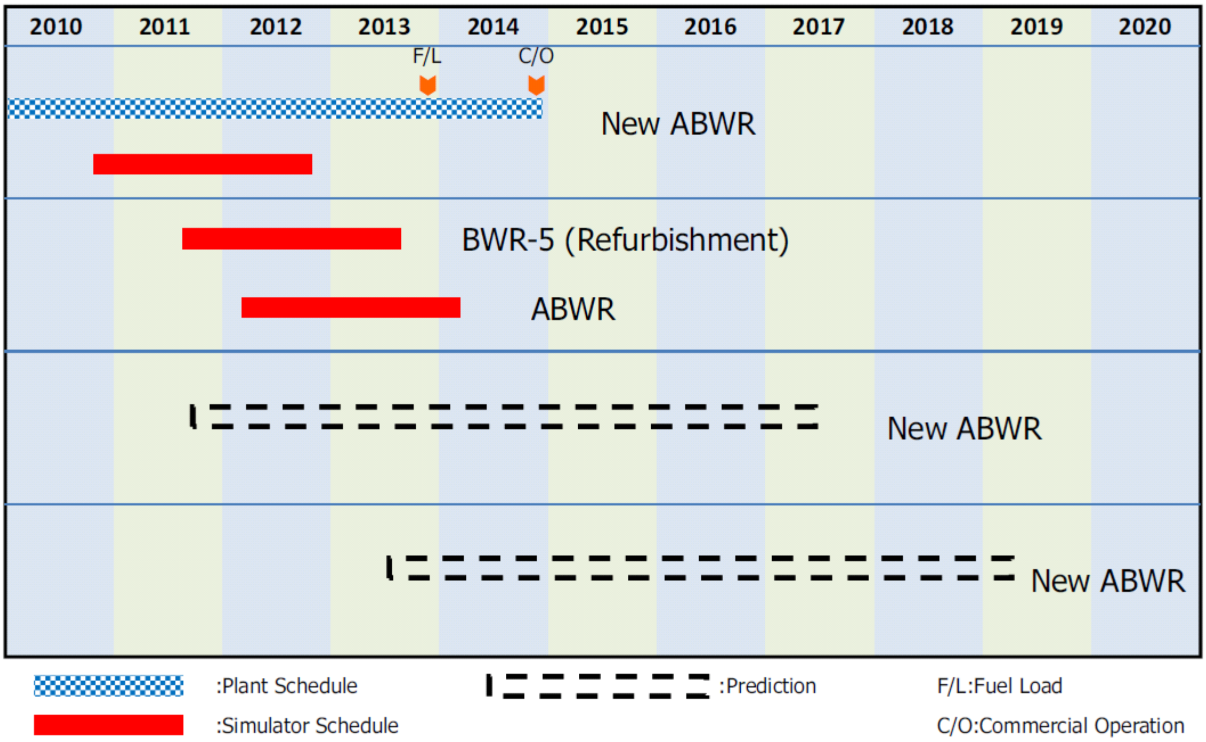

Figure 1 Hitachi's latest simulator schedule

In order to handle these challenges, efficient implementation of plant data is an important and high priority task to support the design work. Hitachi's design work is supported on the software side by our DCS Emulator technology (see Nakashima et al., 2010 [1]). Hitachi has already applied the use of DCS emulator technology to the latest ABWR simulator. On the hardware side, Hitachi is developing an ECWD (Elementary Control Wiring Diagram) converter application. The converter analyzes CAD-based ECWDs and remakes them into FBD-based application programs. Until the latest ABWR, Hitachi used manual techniques to remake ECWDs into FBD based application programs. The converter will be applied to the next generation of simulators. Hitachi believes that these experiences and technology can support any NPP's simulator work.

3. Conceptual requirements for simulator design

A first step for design and validation support would be the development of a full-scale mock-up of the NPP control room; a conceptual architecture is illustrated in Figure 2. With Ethernet as the main communications path, this architecture could flexibly incorporate other Ethernet-compatible stations in addition to those shown. This mock-up can be used as a proving ground for validating design concepts and performing control analysis as the development of NPP DCS application software and system architecture progresses. In order to support this objective, the ability to develop and test DCS system configurations and application software is required. The goal is that as the development of an NPP advances, the DCS emulator will be able to evolve into a major component of its training simulator, executing the DCS code that is used in the physical plant. Furthermore, the design of the DCS emulator accommodates any changes to the plant code that are made after commissioning and throughout the life of the plant. DCS emulator support would also be required for the full scope training simulator, classroom simulator and desktop simulator.

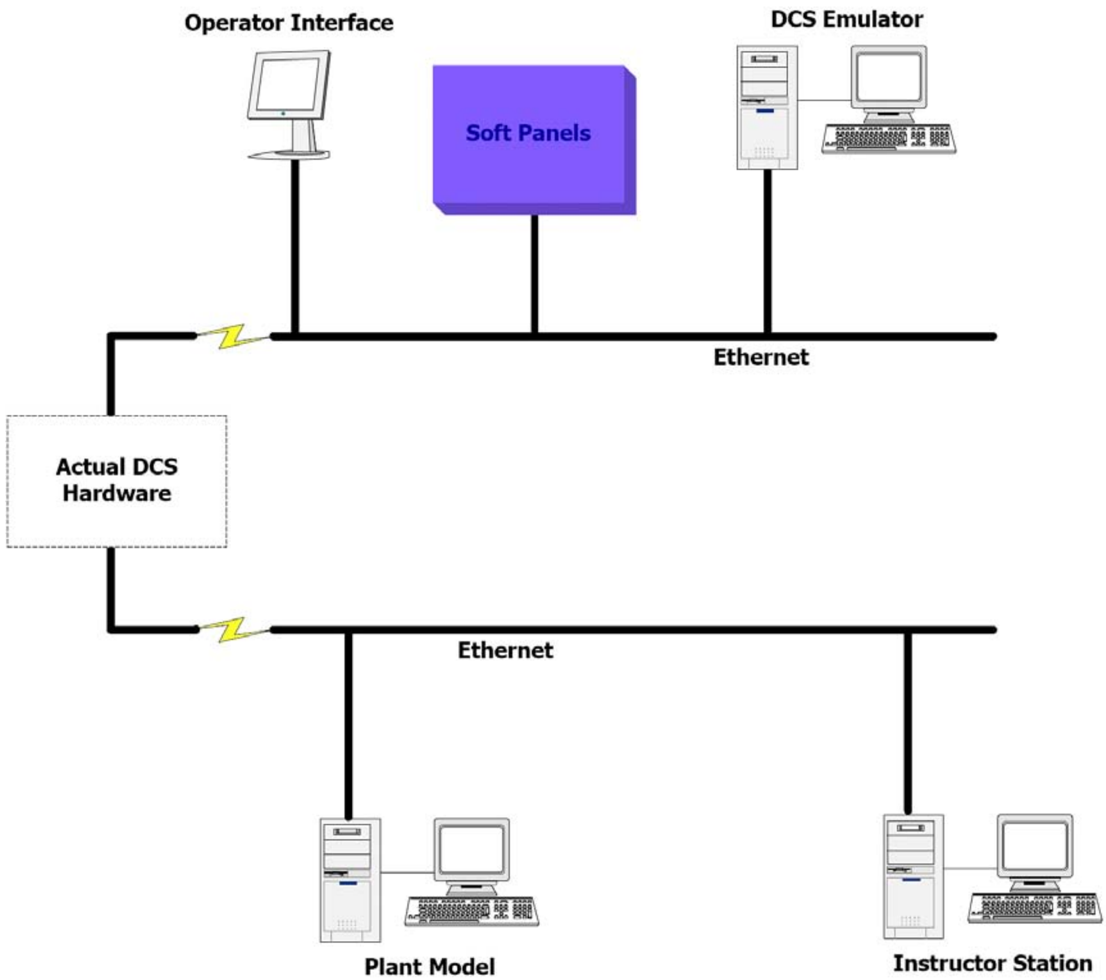

Figure 2 NPP mock-up architecture

Initially, test code is ported to the DCS platform. The DCS emulator in the NPP mock-up architecture will aid in the validation and testing of this code. To support an effective validation platform and simulator function, the capability to execute application logic outside of the native DCS hardware is required. Ideally, the DCS emulator needs to be able to execute actual DCS code, and the execution of DCS application software must be as close as possible in behaviour and timing to the actual DCS hardware. In order to preserve the fidelity of the application logic and minimize the effort required to prepare the plant software for the DCS emulator, the emulator loads the original application software without modification.

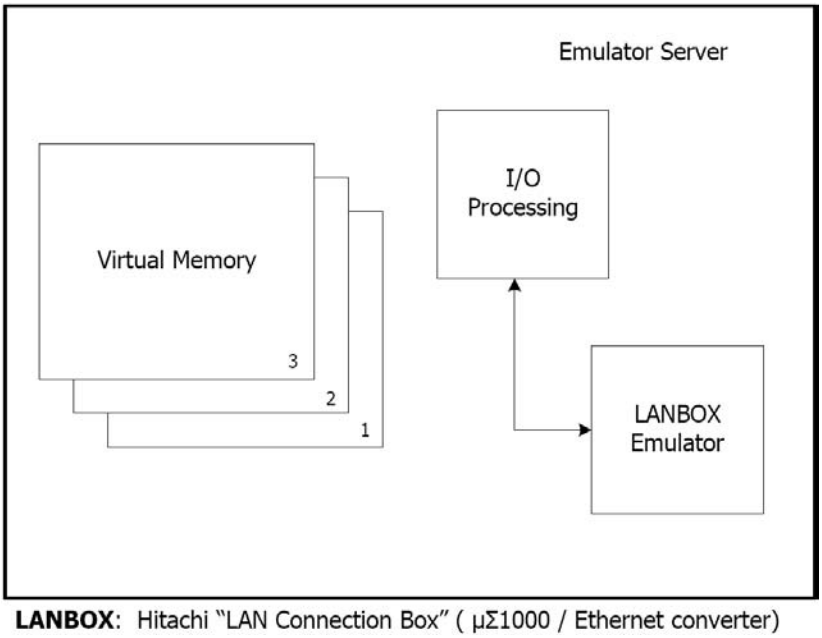

Figure 3 DCS emulator components

Maintaining the timing integrity of logic execution without reproduction of the entire task scheduling system can be a daunting and complex task. To achieve this, Hitachi's DCS emulator reproduces the timing enforced by the task scheduler on the DCS hardware by taking one cycle and looping it. When a request is made to the simulator to speed up or slow down, the interval that this cycle is executed is adjusted accordingly. To minimize the effects of reproducing controller timing in the emulator, it is necessary to optimize the time constant values used in the controller's logic, especially in PID algorithms. For example, if a controller's emulated cycle time doubles, then constant values embedded in its logic need to be doubled also to avoid mismatch.

The DCS emulator may need to emulate systems that use logical partitions to split up processes by function or by system. These partitions can be composed of multiple controllers, some redundant, some standalone. Consequently, the DCS emulator must be designed to accommodate the emulation of multiple controllers and the communications between them. Hitachi has chosen to address these design requirements by providing a DCS emulator that is capable of emulating a large number of (multiple) controllers in a shared memory environment as illustrated in Figure 3, this allows controller information, internal memory, and network memory to be assigned regions in one physical memory space so that multiple controllers can be emulated on a single PC.

The intent of the DCS emulator is to be able to take any application logic and configuration that has been created in a development environment, and drop it in unchanged into the emulator environment. This emulator environment supports all of the configurations that would be possible with multiple physical controllers and networks. In addition, the behaviour of the emulated network, configuration handling, memory management and I/O will mirror that of a physical DCS configuration. This enables the DCS emulator to have a high fidelity to the actual DCS and provides an effective platform for design validation and the development of plant software. In addition, this high fidelity lends itself to the development of an effective simulator for training the operators to troubleshoot or maintain the DCS in an operating environment.

A major component of DCS support for the NPP mock-up and the various simulators is the ability to interface with external sources. This capability provides developers with the ability to test and validate design concepts in a realistic environment. In addition, it enables instructors to have the flexibility required to create an authentic environment for training operators. One key requirement of any simulator is the ability to be driven by a model supplying real-world data. This requirement often poses significant challenges as models may be created by other entities or use diverse and proprietary technologies. Thus, a universal method of access is required. Hitachi provides an API that will serve as an interface between the emulated interpreter and the plant model. This will enable the simulator to be driven by any plant model.

Figure 4 shows a conceptual connection diagram depicting the role of the DCS emulator.

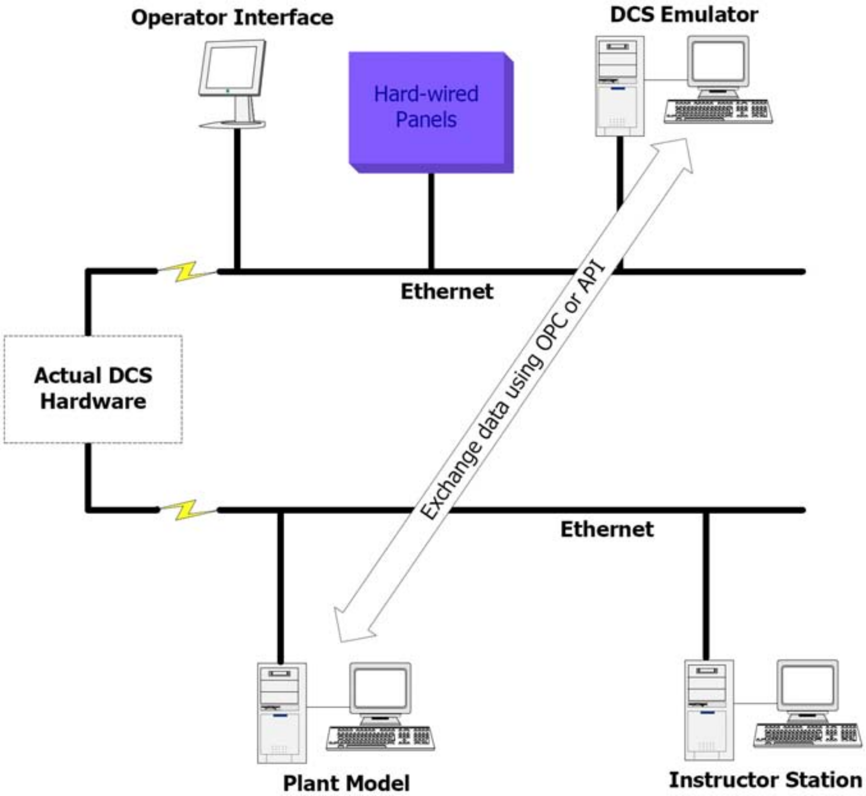

Figure 4 Training simulator architecture

Communications with the plant model requires "hooks" to be established in the DCS emulator so that the emulator can be synchronized with the plant model. Governed by the simulator, the simulation speed can be increased or decreased depending on the requirements of the trainer using the simulator. The model, in turn, will respond to these speed adjustments. In order to execute correctly, the DCS emulator responds to these speed changes by adjusting the emulation of the controllers accordingly. In Hitachi's product, this functionality is provided both in the supplied API and an OPC interface. OPC is a communications standard that uses a tag-based mechanism designed to facilitate the communication of real-time data between emulated controllers and/or physical controllers in a network environment. Hitachi includes an OPC interface to the emulated controller memory and network memory. The option to use OPC or the supplied API enables the simulator design to be adaptable and accommodate a wide variety of configurations. In addition to varying the speed, the DCS emulator must be able to recall pre-set and initial conditions, freeze and restart execution, and store and restore system states. To aid the instructor in operator training, the DCS emulator should also be able to take snapshots and replay recorded scenarios or earlier simulations.

A maximum flexibility FSS will be capable of interfacing with actual DCS hardware and an Operator Interface/Plant Data network. This function will enable instructor stations to initiate preset or alarm conditions for scenario-based training. Consequently, this provides flexibility for building a highly integrated instructor station package with plant model that can be effectively employed by trainers.

4. Hitachi's approach

Hitachi has been reviewing the details of the simulator design described above. Some of this work is described by Nakashima et al. [1]. The most recent topics of investigation are the interface with an external source (HMI) and synchronization between software modules. This paper describes these 2 topics in the sections below.

4.1 Interface between HMI and the DCS emulator

An actual HMI (Human-Machine Interface) connection architecture is shown in Figure 5. The μΣ-Network1000 (μΣ-1000) is Hitachi's proprietary control system communication network. The physical medium is a Hitachi in-house optical fibre and its communication method is based on a shared cyclic transfer memory. Each μΣ-1000 has several controllers that read from / write to the shared memory. LANBOX is a Hitachi hardware module which transfers data from μΣ-1000 to standard Ethernet and vice versa. The LANBOX Driver accesses LANBOXes and requests the LANBOXes to read from / write to a shared memory within the μΣ-1000 by using UDP packets.

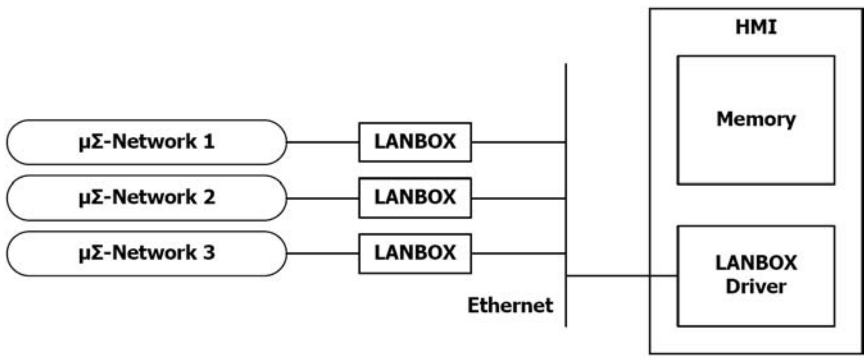

Figure 5 Interface between HMI and actual LANBOX

The HMI already uses an Ethernet connection to a LANBOX in the actual environment. A more flexible solution would be to provide a LANBOX emulator within the DCS emulator as shown in Figure 6. The actual LANBOX Driver connects with several LANBOXes each with its own IP address. The LANBOX driver can recognize each LANBOX by using its IP address; however, in the emulator case, the DCS emulator has only one IP address. In order to allow the LANBOX driver to recognize each virtual μΣ-1000 network, the LANBOX emulator uses virtual IP addresses. The virtual IP addresses will identical to the actual LANBOX IP addresses so that the LANBOX driver can connect to the virtual μΣ-1000 network as if connecting to the actual network. Recognizing a target virtual μΣ-1000 network, the LANBOX emulator can access each of these. The virtual μΣ-1000 networks have the same memory allocation as the actual networks they are representing. With this approach, any virtual μΣ-1000 network (within the DCS emulator) that requires a physical connection to HMI can utilize this LANBOX emulator.

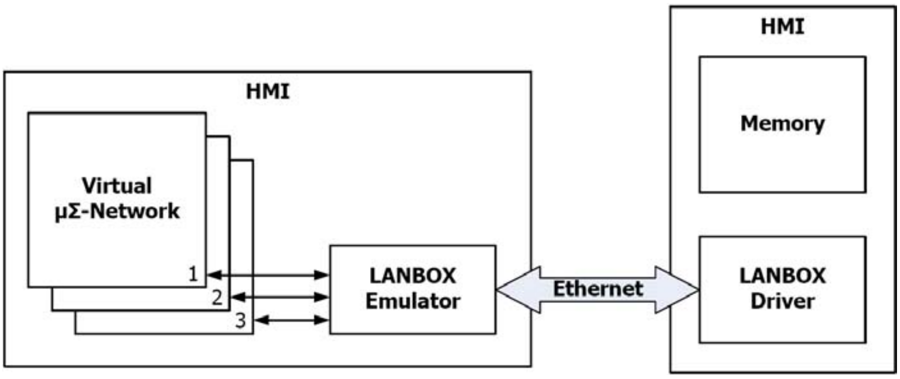

Figure 6 Interface between HMI and LANBOX emulator

4.2 Synchronization

In order to ensure high performance, Hitachi recommends that the Simulator Software architecture have a stand-alone software structure, which is based on Hitachi's NS-3 simulator; however, in order to improve its flexibility and usability (e.g., with external plant models), the DCS Emulator provides a synchronization communication interface.

This communication interface provides three major functions that allow the DCS Emulator to stay synchronized with the simulator software: (1) execute, (2) input/output and (3) store/restore. The execute function (1) governs the speed at which the emulator runs, allowing the DCS emulator to run in step with the simulator. The input/output function (2) allows the simulator to read or write from a common memory area that is shared between the simulator and DCS emulator, providing an effective way for the simulator to initiate external DCS events. The store/restore function (3) enables the DCS emulator to capture its internal state and restore it so that the simulator can replay and record different scenarios. Together, these three functions provide a great deal of flexibility to the simulator software to control the behaviour of the DCS emulator. The communication interface will be released as API and/or OPC commands. A trigger command released from the simulator software executes the DCS Emulator functions.

5. Hitachi's NPP simulator functions

Hitachi's NPP simulator includes functions to allow instructors and trainees to experience efficient training sessions. This section describes these functions.

5.1 Full scope simulator architecture

Figure 7 Overview of the latest ABWR simulator

Figure 7 shows a view of Hitachi's latest ABWR Full Scope Simulator basically consisting of the same hardware (mimic board, control panel, HMIs, switches, buttons and so on) as the actual plant operator control room. The simulator also uses the same computers:

<u>Servers</u>

- Simulator Server : 1
- Process Computer Server : 1
- Human Machine Interface (HMI) Server : 1
  - ✓ CPU : Intel$^{®}$ Core^{TM}2 Duo 2.4GHz
  - ✓ Memory : 4 GByte
  - ✓ OS : Hitachi's Linux based real time Operating System

<u>HMIs</u>

- Plant Level Interfaces : 6
- System Level Interfaces : 26
- Integrated Interface : 1
  - ✓ CPU : Intel$^{®}$ Core^{TM}2 Duo 2.4GHz

- ✓ Memory : 2 GByte
- ✓ OS : Windows XP SP3

Consoles

- Operator Consoles : 7
- Instructor Consoles : 6
  - ✓ CPU : Intel® Core™2 Duo 2.4GHz
  - ✓ Memory : 2 GByte
  - ✓ OS : Windows XP SP3

5.2 Classroom and desktop simulator capability

For more efficient and flexible training, Hitachi's simulator can consist of PCs only. The simulator hardware (panels, switches and buttons) are represented by software applications and GUIs; trainees can operate them using a mouse. Figure 8 shows two examples of such virtual interfaces.

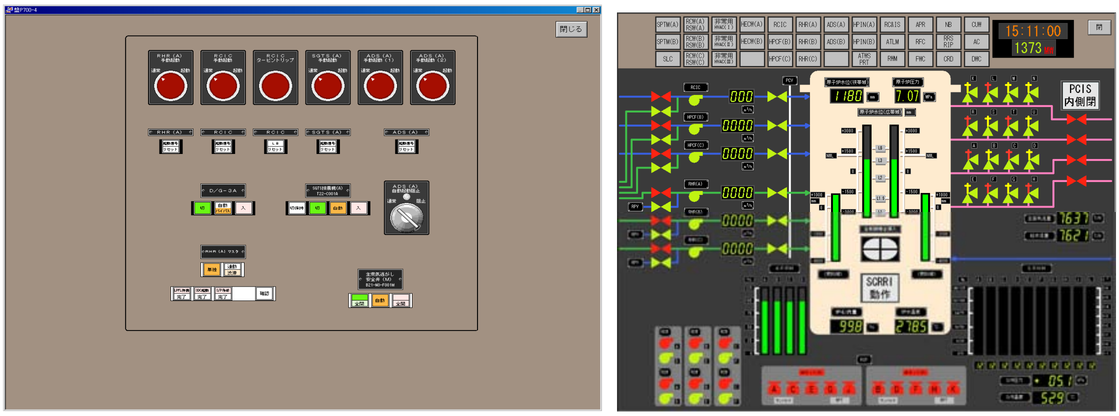

Figure 8 Virtual simulator displays

By using these virtual displays, a classroom simulator can consist of a minimum of hardware. The simulator servers described in section 4.1 can be integrated into a virtual PC environment. Specifications of this virtual PC environment are shown below:

- Host Computer CPU : Intel® Core™ i7 2.8GHz
- Host Computer Memory : 8 GByte
- Host Computer OS : Windows 7 Professional 64-bit
- Virtual Environment : VMware® Workstation ver. 7.1

In addition to virtual servers, the classroom simulator requires several PCs. Figure 9 shows a minimum classroom simulator. From left to right, the classroom simulator consists of a virtual mimic board, control panel, HMI and instructor console. Each of these PCs is able to change its functionality depending on the requirements of the specific situation.

Figure 9 Classroom simulator

5.3 Instructor functions

Hitachi's simulator gives instructors access to functions that make training efficient and comprehensive. The full scope simulator has 6 Instructor Consoles with hardware specifications as described in section 5.1. Instructor functions provided as described in the following sub-sections.

A. Simulator control function
This function provides the capabilities of "Freeze / Run", "Store / Restore", "Speed Up / Speed Down", "Backtrack / Restore Backtrack", "Cause Plant Malfunction, and "Forcing and Tracking of Operator Actions". This function enables the Instructor to manage training sessions efficiently and effectively.

B. Plant condition monitoring function
This function provides Trending and Data Monitoring for the plant conditions. The Trending and Data Monitoring are provided at the control panel and operator consoles.

C. Plant level interface monitoring function
This function provides Trending and Data Monitoring for the plant level interface. The Trending and Data Monitoring are provided at the mimic board and control panel.

D. System level interface monitoring function
The HMI monitoring functions described in B and C provide the instructor with the ability to monitor which information screens the operators are checking. This function also provides the instructor with the ability to change screens remotely in order to direct operators (i.e., trainees) to the proper screen.

E. Scenario development function

For advanced training, the "Scenario Development Function" provides the ability to schedule events which cause malfunctions or to force events. Figure 10 shows a scheduled scenario and its logic. These events are caused or reset by a timer, logic or instructor action. With this function, instructors can provide flexible and advanced training. To keep even standard training scenarios fresh, instructors can provide interrupt malfunctions or force events.

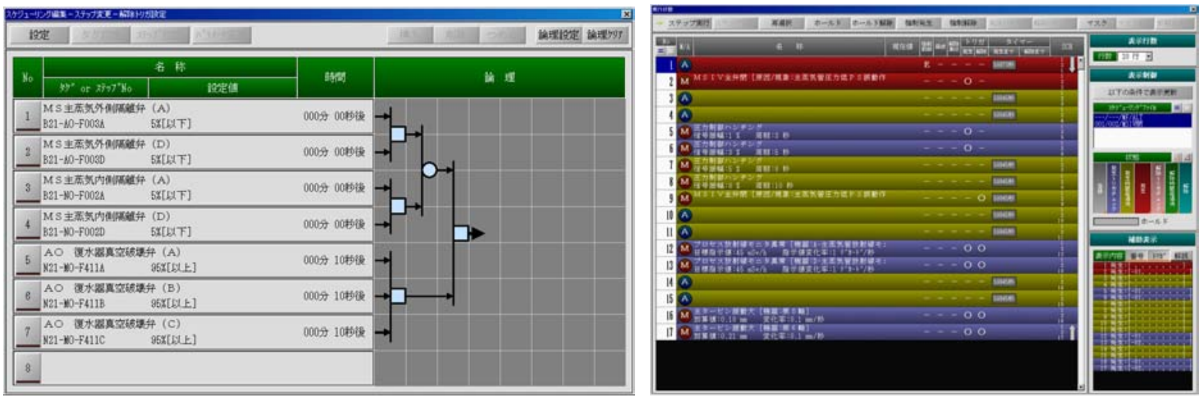

Figure 10 Scenario development function

F. Operator console function

This function is identical to that of the actual operator console: instructors are able to obtain plant information in the same way as operators.

With these functions, Hitachi's NPP simulator is able to provide high-quality training. Also, the classroom and desktop simulator capability provide a flexible training environment.

6. Conclusions

Strategies for the development of NPP DCS architectures and associated software applications could rely on an NPP mock-up as a proving ground for design concepts and control analysis. At the centre of this design is Hitachi's DCS emulator, which, without involving the actual DCS hardware, enables plant application software development and analysis as overall development and design progress. Once development is complete, the software can be moved from the NPP mock-up environment to the actual DCS with no modification. Aside from accurately reproducing DCS execution, the DCS emulator also provides external integration with a model through a flexible interface. Furthermore, Hitachi can support such an NPP mock-up initiative by providing an interface between the DCS emulator and an actual LANBOX, allowing the DCS emulator to communicate to an actual HMI and data gathering applications using Hitachi's control system communication network (μΣ-1000). This proving ground can then evolve into a full-scale simulator for training operators and conducting operation scenarios. Throughout the entire development of the NPP, the Hitachi DCS emulator would play an essential role in the efficient development, testing, and validation of the plant software.

7. References

[1] Y. Nakashima, R. Trueman and K. Ishii, "DCS Emulator Development for the ACR-1000® Simulator", 31st Annual Conference of the Canadian Nuclear Society, Montreal, Quebec, Canada, May 24, 2010.

[2] Y. Nakashima, K. Ishii and D. Chiba, "DCS Emulator Development for Nuclear Power Plants", 30th Annual Conference of the Canadian Nuclear Society, Calgary, Alberta, Canada, June 3, 2009.