A NUCLEAR TRAINING SIMULATOR IMPLEMENTING A CAPABILITY FOR MULTIPLE, CONCURRENT-TRAINING SESSIONS

B.J. Groeneveld, D.G. Bannister, K.R. Estes, M.R. Johnsen
Idaho National Engineering Laboratory
P.O. Box 1625, Idaho Falls, ID 83415
bjg@inel.gov

KEY WORDS

Concurrent training, interacting simulations, distributed processing, client-server, peer-to-peer.

ABSTRACT

The Advanced Test Reactor (ATR) Simulator at the Test Reactor Area of the Idaho National Engineering Laboratory (INEL) has recently been upgraded to reflect plant installation of a distributed control system (DCS). The ATR Simulator re-design implements traditional needs for software extensibility and plant installation prototyping, but the driving force behind its new design was an instruction requirement for multiple, concurrent-training sessions. Support is provided for up to three concurrent, independent or interacting, training sessions of reactor, balance of plant, and experiment loop operators. This capability has been achieved by modifying the existing design to consistently apply client-server, parent-child, and peer-to-peer processing technologies, and then to encapsulate concurrency software into all interfaces. When the resulting component-oriented design is linked with build and runtime flexibility in a distributed computing environment, traditional needs for extensibility and parallel software and scenario development are satisfied with minimal additional effort. Sensible configuration management practices coupled with the ability to perform piecewise system builds also greatly facilitate prototyping of plant changes prior to installation.

INTRODUCTION

The first simulator for the Advanced Test Reactor was built concurrently with the facility itself in the 1960s. That simulator was fundamentally an analog computer with hardware sized to the complexity of the ATR model. Though it was sufficient for training in the early days, it offered no flexibility, limited accuracy and poor reliability. The move to a digital computer-based Simulator took place in the early 1980s. Later, the Simulator was completely redesigned and placed into service in 1994. That latest upgrade is the subject of this paper.

The ATR is a 250 MW reactor designed to provide a nuclear environment for the development and testing of nuclear fuels and materials. It consists of three basic systems: the reactor itself, the experiment loops, and what we shall refer to as the balance of plant (BOP). The latter includes primary and secondary coolant piping, pumps, heat exchangers, cooling towers and fans, etc. Given the mission of ATR, there are no turbines, generators, or output connections to a power grid. ATR control philosophy is also partitioned into three distinct areas: the control of reactor power, control of up to nine (substantially independent) experiment loops, and the control of the BOP.

The requirement to upgrade the ATR Simulator came from a concurrent sister project involving the replacement of most of the instrumentation and control equipment in the BOP, Experiment Loops, and some aspects of Reactor Control, with the latest generation of DCS equipment. Thus, the upgraded Simulator had to include exact emulations of the DCS equipment.

SYSTEM OVERVIEW

From a software perspective three subsystems comprise the ATR Simulator:

- Operator Consoles
- Simulation Models
- Instructor Station

These subsystems and their interactions are described in the following sections.

Operator Consoles

The Operator Consoles subsystem includes not only the duplication of all plant control interfaces, but also includes

software to hardware interfaces and remote processing units (RPUs). From the software, run-time perspective, these hardware interfaces are an extension of the user-interface. There are three console display systems:

- reactor console display system
- experiment loops console display system
- balance of plant console display system

Hardware interfaces include those to the reactor control rods, sequential events recorder, annunciators, and PID controllers in the RPUs.

Simulation Models

47 models are combined to provide training capability for all essential plant functions. Real time is simulated by cycling all models in timeslots of 10 or 100 milliseconds. The computer code of the models is divided into three *bins* for ease of maintainability, one each corresponding to the primary control functions of the plant, i.e.,

- reactor operation
- balance of plant operation
- experiment loops operation

Model bins can be combined at run time to provide several *modes* of training capability, including the following *independent* modes:

- reactor-only training (R)
- balance of plant-only training (B)
- experiment loops-only training (L)

Model bins are also combined to create several *interacting* modes of training:

- reactor interacting with balance of plant (RB)
- reactor interacting with experiment loops (RL)
- reactor, balance of plant, and experiment loops interacting (RBL)

In all but the RBL simulation, simplified models representing otherwise missing elements of the plant are added to maintain realistic plant-wide behavior. The ATR Simulator is specifically designed to permit operation of *concurrent* modes. These parallel training sessions are limited in availability only by the need for hardware consoles. The ATR Simulator accurately reflects the plant using a single duplicate of all consoles restricting the concurrent modes to:

- parallel R, B, and L sessions
- parallel RB and L sessions
- parallel RL and B sessions.

Instructor Station

The Instructor Station subsystem extends deeply into the software system, including all aspects of event scheduling and management. In essence, these are all the components of the simulator injecting changes into steady state conditions of the model subsystem. The major components include:

- event management
- scenario management
- history management
- state management

States and transitions are given in Figure 1 below. The initial, Dormant state is of significance only to the ATR Simulator as simulation mode selection occurs during transition from this state to the Freeze state. The Models- and IO-Only states separate running of Simulation Models from Operator Console processes for development flexibility, and are in fact special cases of the Run mode.

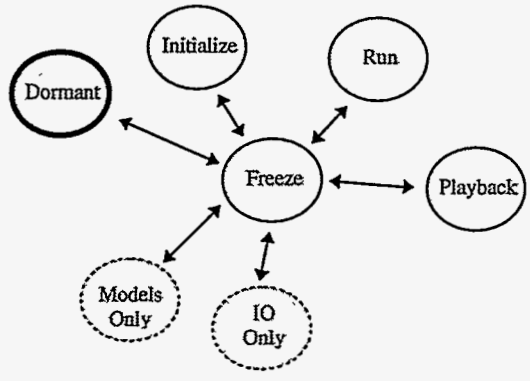

Figure 1. State transitions for the ATR Simulator.

Subsystem Interaction

Model state variables are recorded and maintained in a networked, shared-memory segment. The state variables together with all variables written to and read from hardware interfaces form the Common Control Variables (CCVs). The CCVs tie the three simulator subsystems together and provide extensive operational as well as development and test capabilities. For example, the CCVs as initial condition vectors are used to start the simulator in any previously recorded state. Instructors control the

simulator by fixing or ramping CCVs, and developers design and test by observing CCVs. Continuous, real-time recording of CCVs provides for rewind, playback, and fast-forwarding capabilities. Figure 2 illustrates how the three ATR Simulator subsystems interact through 9,652 CCVs in the ATR Simulator.

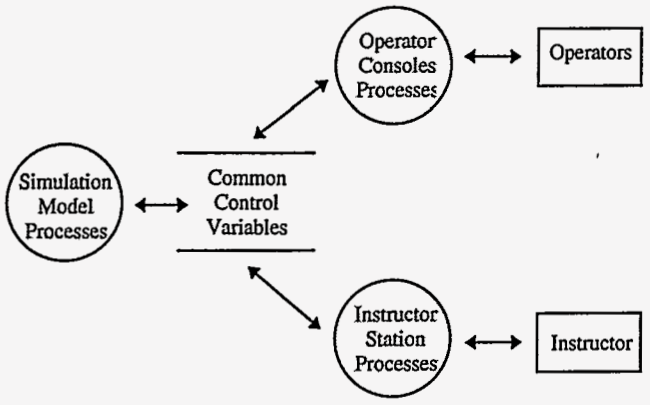

Figure 2. ATR Simulator subsystem interaction overview.

Figure 3 shows the block diagram for the computer hardware and network configuration. Simulation Model processes run exclusively on the high-performance Model computer, Instructor Station processes are distributed over the Model, Management, and Instructor computers, and Operator Console processes are distributed over the IO, Model, and Console Display System (CDS) computers. Note an additional dedicated network connecting the IO and Model computers. Each CDS consists of three to five workstations.

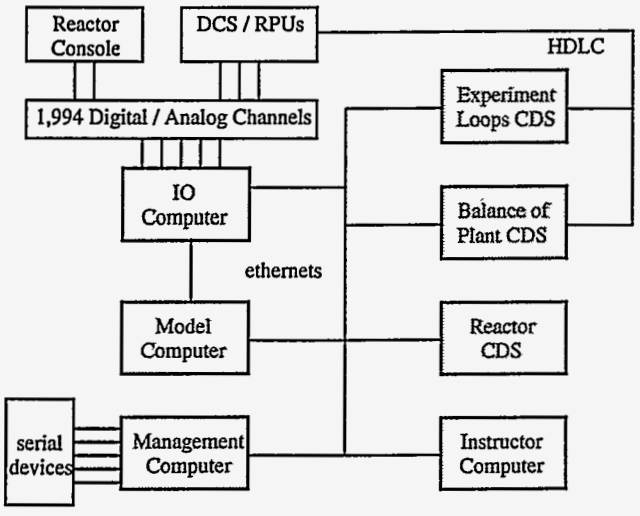

Figure 3. Networked computer hardware block diagram.

MODELS

The criterion for model accuracy was whether or not an operator could see a particular plant response in the course of operating the plant. If the response could be seen in the plant, then it must be seen in the Simulator. This applied to both static and dynamic responses. With this as a guideline, certain assumptions were made. The accurate simulation of catastrophic events such as guillotine primary coolant pipe breaks was not required; under these extreme conditions the operator's action was simply to exit the facility, leaving the safety systems to react as they were designed. Thus, for example, the modeling of two-phase coolant conditions was unnecessary except in certain portions of the Experiment Loops. Turbulent flow could be assumed in most piping.

The models do, however, allow the simulation of a great variety of anomalies that are relevant to operator training, such as smaller pipe breaks, heat exchanger leaking and blockage, electrical power failures, pump trips, and so forth. The effects of cooling fan speeds and weather conditions on cooling tower efficiency are also included so that, for example, the effect of a weather front or seasonal changes can be simulated.

The creation of computer-based models for the various ATR subsystems proceeded as follows. First, mathematical models were created, using existing plant documentation, actual static and dynamic measurements, and of course appropriate physics. Model complexity, precision, and accuracy were established on the basis of the mission, i.e., operator training, as opposed to precise engineering analysis. Second, corresponding computer models were created using a design package called XANALOG, a graphical simulation tool similar to MATRIX-X and CTRL-C. Model response, accuracy, and stability were carefully verified in this domain before proceeding to the next step of creating corresponding code that could be ported to the simulator computers. XANALOG is capable of automatic code generation in either C or Fortran, but as a practical matter this step was not quite as trivial as expected.

Computer throughput was sized to accommodate the required model response. Many thermal equations run on a 100 ms cycle time. Most of the flow and pressure equations run at 20 ms, but some hydraulic model portions must run at 2 ms to maintain correct dynamic response.

DESIGNING FOR MANY OBJECTIVES

The goal of upgrading the ATR Simulator to reflect plant changes had several objectives:

- add balance of plant and experiment loops simulation
- add independent, interacting, and concurrent simulation modes
- retain existing reactor simulator functionality

Buried in these objectives were requirements for plant prototyping, scenario and software development, and system extensibility.

Of most significance was the need to support additional and concurrent simulations. Without regard to implementation, this would increase computing demands by two orders of magnitude. Not only would the lines of commented source increase from 80,000 to more than 220,000, but up to three simulations would require simultaneous processing. Simulation codes themselves account for 45,000 lines of C implemented using Euler (40%) and Runga Kutta 4 (RK4, 60%) numerical integration methods. Modeling codes using RK4 linearly increase time complexity 20 times, and for pressures and flows (34% of modeling code) 200 times. Compute power needs were in part met by upgrading existing equipment, i.e., replacing a 4 specfp92 HP9000 series 350 with a 168 specfp92 HP9000 series 730. With the exception of turn-key components, the ATR Simulator runs on HP-UX versions 7.05 and 9.07.

Clients, Servers, Peers, and Managers

Supporting the requirements without compromising real time was further achieved through application of detailed analysis and synthesis software engineering procedures.

First, the existing reactor simulator was analyzed and modularized into a set of interacting components. Berkeley sockets and link-level communications were used to scale processing from two to three hosts. Scenario management, event management, and history management were converted or extended to clients and servers. Simulation models and hardware input and output were separated and distributed into peer-to-peer processes using link-level communications. The ethernet between the IO and Model computers (Figure 3) is dedicated to connecting the peer processes, which transmit 40 packets per second. While this provides for high-speed network communications, it dictates the need for custom, exception-based error handling. This was accomplished through careful definition of interactions using a simple finite state-machine shown in Figure 4 (*e* indicates an *error* state; event sequences omitted for clarity).

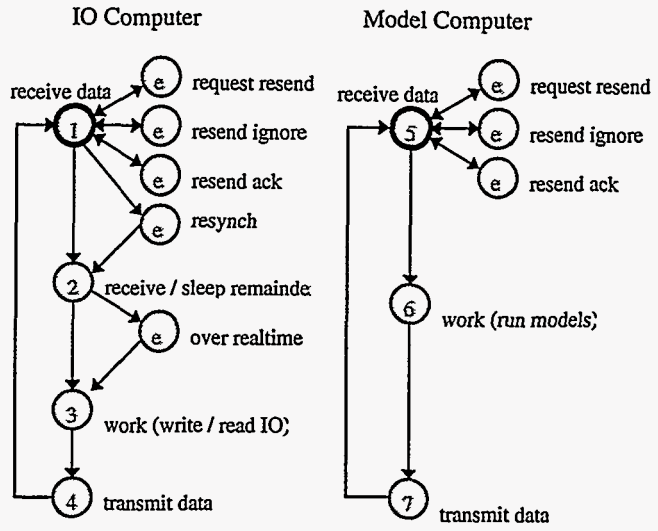

Figure 4. Finite state machine for IO-Model handshaking.

Second, a new service called the Mode Manager was introduced for inclusion at all newly defined interfaces. This layer of concurrency control software involves correctly matching a service request to the service provider. For example, with three concurrent simulations running a request to ramp a variable must be properly matched with the desired simulation. The Mode Manager accomplishes this by maintaining a *mode mask* in the CCV containing slots for each simulation and corresponding simulation state. Three slots permit three concurrent simulations each with a private set of CCVs. The mode mask itself resides in a global CCV section. Only the Mode Manager may manipulate the mode mask.

Parents And Children

The last step was the redesign of the Model Processes to understand modes. A parent-child model was used to implement the need for rapid state transitions, to manage multiple simulation model processes, and to provide development and test capability concurrent with operational use. In fact, operational use of several fully tested ATR Simulator modes commenced prior to full completion of the upgrade.

A parent *control* process monitors the mode mask for changes from and to the Dormant state, spawning and killing up to three concurrent *model* processes, respectively. Model processes themselves check the mode mask to identify themselves, i.e., what simulation mode they represent. The model processes also use the mode mask to determine what state they are in. For example, entry into

the *Initialize* state requires special first-time processing to properly set hardware channels and state variables.

The parent, control process is responsible for running the children, digital-to-analog conversions, engineering unit conversions, and network communications software within a 100 ms timeslot. Semaphores are used to synchronize processing within this timeslot. Network communications involve sending several link-level packets from the Model computer to the IO computer, which is dedicated to hardware writes and reads. Figure 5 shows the resulting use of the CCV shared memory segment. In this case three slots are in use by independent simulations. For each simulation only those variables associated with the operator console are IO bound.

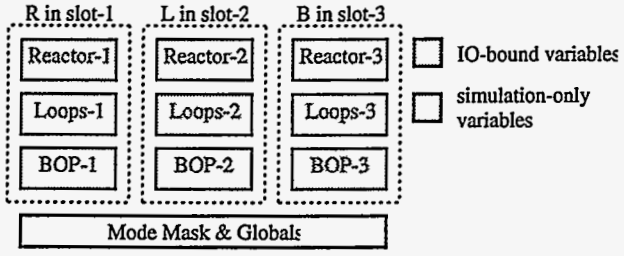

Figure 5. CCV occupied by three independent simulations.

The interacting RL case is illustrated by Figure 6, where variables for both Reactor and Experiment Loops are IO bound.

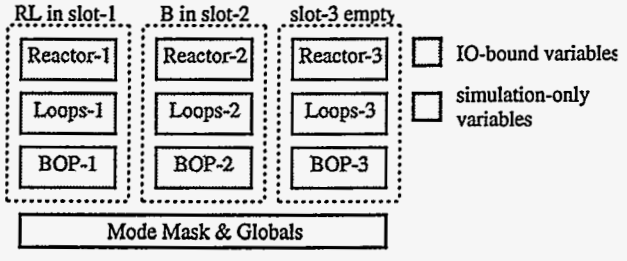

Figure 6. CCV occupied by two independent simulations of which one uses an interacting mode.

Figure 7 shows a control panel available to instructors for mode selection. This control panel is a *tcl/tk* wrapper around the *Scenario Manager*. The Scenario Manager is an interpreter that acts in a client-server relationship with the *Event Manager* to inject changes into the Simulation Model subsystem.

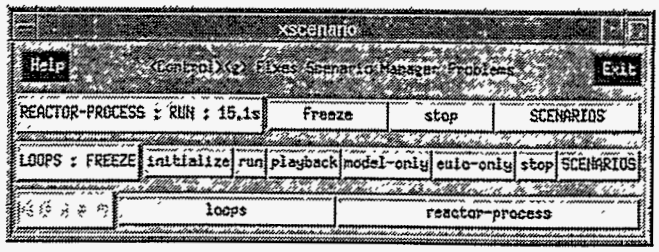

Figure 7. Simulation mode management control panel.

In this figure we see a *REACTOR-PROCESS* (RB) simulation 15.1s into a run, a frozen *LOOPS* simulation, and a third, dormant slot.

Besides controlling modes through points and clicks, simulation selection from this control panel generates a *gnu-client* request to an *emacs-server* responding with a Scenario Manager interpreter for precise event scheduling. Instructors can develop sophisticated scenario programs using tcl/tk interactions with the Scenario Manager.

IMPLEMENTING FOR THE LONG TERM

Mature software development practices are a prerequisite to the production of quality software. Software experiences specific to nuclear training simulators have been documented (Davis and Webb 1988). At the ATR Simulator, several steps were taken to address the need to develop within short time frames and produce intermediate deliverables. These included:

- sensible configuration management (CM) practices tightly woven into the development process
- component-oriented development
- strict adherence to build- and run-time flexibility rules
- extensive use of Unix and other productivity tools

In order for CM to aid rather than restrict the development process the software engineer must see its benefits. Every developer must be accountable to perform code checkouts and checkins, perform system builds, establish baselines, and work directly with customers to sign off change requests.

In a concept overloaded software world, many basic computer science precepts retain high value. Data-driven processing and loosely-coupled components are inherent to producing flexible and maintainable code. This includes decoupling source from its build- and run-time environments to produce build and run anywhere software. And, it also includes the requirement to permit piecewise system builds.

Applying CM sensibly and linking the component-oriented design with build- and run-time flexibility in a distributed computing environment has produced many benefits. Several *software simulators*, each offering concurrent simulations, can run simultaneously without the need for additional hardware, resulting in minimal hindrance between the following activities:

- software development
- training sessions
- scenario development
- plant prototyping

This is in contrast to offering similar capabilities using duplicate, independent hardware (Gregory et al. 1991).

As in any well-designed, distributed system, the ATR Simulator's clients and servers can run within the domain of a single machine. This is illustrated in Figure 8 by the use of the xCCV development tool under *model-only* mode. This tool plots any CCV variable for any simulation mode (concurrent and / or integrated) against time. The figure below shows a selection of variables for RB (REACTOR-PROCESS, interacting) and L (LOOPS, independent) simulations.

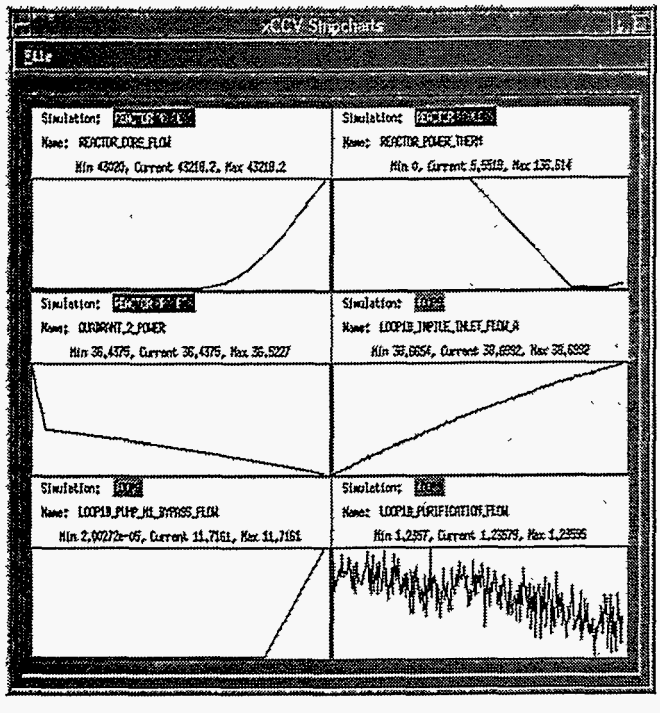

Figure 8. xCCV's unlimited variable display.

FUTURE DIRECTIONS

The current generation technology ATR Simulator was originally completed in 1989 (Burtt et al. 1989). This upgrade was completed in 1994. The software life cycle is currently at a maintenance stage where bug fixes and minor enhancements are ongoing. Additional upgrades are currently not planned.

Were additional simulation capabilities required, the team involved with the ATR Simulator would draw heavily from the ATR Simulator's concepts and experiences. In particular, the component-oriented approach would likely be extended to include additional client-server and peer-to-peer processing for decreased coupling of components. This would result in yet more flexibility, maintainability, and reliability. This would be of most interest when the use of independent, interoperating, and concurrent simulation modes are generalized without restrictions on instances (currently three) or distribution resulting in a fully scaleable simulator system. This would correspond to several trends in simulation capability, including both larger and smaller scale simulators (White 1992).

ACKNOWLEDGMENTS

The authors wish to thank those involved in making the work described in this paper possible, including Lee Bowen, Dave Brooks, Ann Egger, Marti Ehlinger, Glynn Ellis, Louise Richardson, Jim Rose, Chris Russell, Stan Scown, and Rita Wells.

REFERENCES

Burtt, J.D., Laats, E.T., and Venhuizen, J.R. 1989. "The Development Of A Reactor Training Simulator At The ATR." *Transaction of the American Nuclear Society*, Vol. 59: 191~2.

Davis, R.M. and Webb, N.J. 1988. "Updating a Nuclear Training Simulator." *Simulators V.* Proceedings of the Society for Computer Simulation Simulators Conference: 419~421.

Gregory, M.V., Mann, J.L., and Sundal, H.W. 1991. "Use Of The RTMC, A Full-Scope Training Simulator Clone (U)." *Simulation Multiconference*, Society for Computer Simulation, New Orleans, Louisiana (April 1-5).

White, J. 1992. "Simulation Enters a New Era." *Nuclear Engineering International*, Vol. 37, No. 458 (September): 1992, 46~47.

DISCLAIMER