A Simulation Environment for ITER PCS Development

M.L. Walker[a], G. Ambrosino[b], G. de Tommasi[b], D.A. Humphreys[a], M. Mattei[c], G. Neu[d], G. Raupp[d], W. Treutterer[d], and A. Winter[a]

[a]*General Atomics, P.O. Box 85608, San Diego, California 92186-5608, USA*
[b] *CREATE/Università di Napoli Federcico II, Napoli, Italy*
[c]*Seconda Università di Napoli, Napoli, Italy*
[d]*Max-Planck-Institut fuer Plasmaphysik, EURATOM Association, 85748 Garching, Germany*
[d] *ITER Organization, Route de Vinon-sur-Verdon, 13115, St. Paul-lez-Durance, France.*

A simulation environment known as the Plasma Control System Simulation Platform (PCSSP), specifically designed to support development of the ITER Plasma Control System (PCS), is currently under construction by an international team encompassing a cross-section of expertise in simulation and exception handling for plasma control. The proposed design addresses the challenging requirements of supporting the PCS design. This paper provides an overview of the PCSSP project and a discussion of some of the major features of its design. Plasma control for the ITER tokamak will be significantly more challenging than for existing fusion devices. An order of magnitude greater performance is needed for some types of control, which together with limited actuator authority, implies that optimized individual controllers and nonlinear saturation logic are required. At the same time, consequences of control failure are significantly more severe, which implies a conflicting requirement for robust control. It also implies a requirement for comprehensive and robust exception handling. Coordinated control of multiple competing objectives with significant interactions, together with many shared uses of actuators to control multiple variables, implies that highly integrated control logic and shared actuator management will be required. It remains a challenge for the integrated technologies to simultaneously address these multiple and often competing requirements to be demonstrated on existing fusion devices and adapted for ITER in time to support its operational schedule. We describe ways in which the PCSSP will help address these challenges to support design of both the ITER PCS itself and the algorithms that will be implemented therein. and at the same time greatly reduce the cost of that development. We summarize the current status of the PCSSP design task, including system requirements and preliminary design documents already delivered as well as features of the ongoing detailed architectural design. The methods being incorporated in the detailed design are based on prior experience with control simulation environments in fusion and on standard practices prevalent in development of control-intensive industrial product designs.

Keywords: plasma control system, simulation, architecture

1. Introduction

The high cost and limited number of discharges planned for ITER, as well as the constraints imposed by its nuclear mission, imply both minimal time for scenario and control tuning and a greater level of confidence needed in discharge performance prior to execution. The use of simulation for control development and verification has been well-established in research and commercial applications to support both of these requirements [1]. Several operating tokamaks have made significant use of simulation tools in the development of control algorithms or key components of plasma control systems themselves [2-9]. The broad success of this approach, both commercially and in the fusion community, led to an IO-funded task to develop such a simulation tool, known as the Plasma Control System Simulation Platform (PCSSP), to aid in development and testing of the ITER Plasma Control System. The scope of the current project is limited to deployment of a demonstration prototype environment with selected components. It is envisioned that this prototype may be extended in subsequent efforts to provide a fully capable control simulation tool to also support ITER machine/system design and configuration evolution and discharge scenario development, and to support plant troubleshooting during operations.

A draft of requirements and use cases [10] and a preliminary architecture definition [11] have been delivered and a detailed design addressing a subset of these requirements [12] is now being developed. At the end of 2013, the design and a prototype implementation will be delivered.

2. The vision for PCSSP

The PCSSP is envisioned as a system of components including a computational framework (referred to as a Control Simulation Environment, CSE) in which multiple simulation modules can be implemented, to which external processes can be connected, and which can support input and output interfaces for programming the simulation and interpreting results. The principal goal of ITER PCS development and testing requires a simplified PCS connected to a simplified simulated Plant (system to be controlled), both implemented within

author's email: walker@fusion.gat.com

PCSSP. Matlab/Simulink has been chosen as the host environment.

PCSSP will provide a number of capabilities that can substantially reduce the time and cost of PCS development, which can be best understood by outlining the steps in this development.

1. Develop and test ITER PCS architecture,
2. Develop and test ITER control algorithms,
3. Implement in realtime code and test,
4. Deploy in ITER operation.

It is expected that Step 1 will be performed only once, but it may not be fully complete until sometime after ITER begins operation. Steps 2–4 will be repeated many times during the ITER lifetime as new control capabilities are developed and brought on-line. PCSSP is envisioned to play a key role in each step but the last.

The ITER PCS will incorporate several architectural features common in operating devices as well as two functions, exception handling (EH) and actuator sharing, that require a level of sophistication not yet demonstrated. PCSSP provides a method for prototyping and evaluating candidate architecture and controller solutions with minimal coding effort. In particular, the Plant simulation will contain an event generator (EG) module that can trigger simulated performance-challenging plasma and hardware events (known as *exceptions*) during control simulation, to evaluate the ability of the EH to handle them [13]. It will also contain a model of the Central Interlock System (CIS), which can be used to test the interaction of the PCS and CIS in responding to events occurring in the Plant [14].

Controller development is facilitated through use of control design *toolboxes* available for Matlab and use of Simulink for controller implementation, debugging, gain tuning, and evaluation of control performance and robustness via simulation.

When satisfied with controller performance following the initial design and simulation process, the Simulink *Embedded Coder* can generate real-time capable C code versions of the algorithms, to incorporate into the PCS. The actual ITER PCS can then be connected to the plant simulation to verify both performance and correct implementation via closed loop simulation.

3. Key use cases and requirements

The Requirements document [10] details the use cases and requirements for the PCSSP, which currently focuses on simulation. Almost all expected simulation use cases can be summarized by the following sequence of actions:

1. Select a version of PCS to be used (e.g., simulated or actual).
2. Select a version of Plant to use (Fig. 1).
3. Define PCS configuration data, including the ITER pulse schedule.
4. Define configuration parameters that customize plant simulation modules.
5. Define PCS initial state to begin simulation.
6. Define Plant initial state to begin simulation.
7. Connect the PCS and Plant objects (both can be in PCSSP or one can be external).
8. Initiate execution of the simulation.
9. Monitor the simulation.
10. Terminate the simulation.
11. Archive the simulated data.
12. Analyze the simulated data.

Variability in choices of components and detailed actions depends on the stage of PCS development (Fig. 1). The PCS function may be implemented by a simulated PCS in the CSE, actual PCS code running on simulation (non-realtime) computers, or PCS code running on PCS realtime hardware. Algorithm development can use the simplified PCSSP Plant, while more detailed testing and eventual validation of ITER pulse schedules require more accuracy.

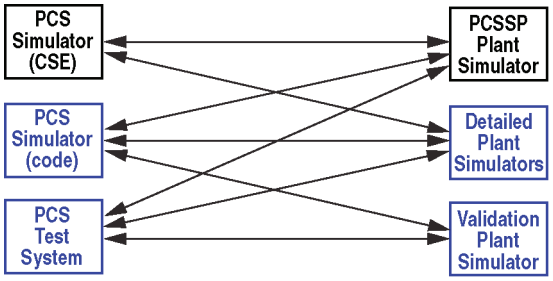

Fig.1 Expected connections of PCS and plant simulators for closed-loop simulation. Blue blocks indicate objects external to PCSSP.

Since simulations will be used to evaluate effectiveness of controllers for ITER, plant models used in simulation must be predictive, i.e., responses seen in simulation must be reflected in similar responses seen in physical plasma operation. To gain this confidence, PCSSP will support model validation efforts by providing the capability to execute simulations of currently operating devices as well as ITER. Equally important, simulation execution must be sufficiently fast to allow iterative evaluation and tuning of candidate PCS algorithms. The need to flexibly replace plant modules (e.g., to simulate different devices) and PCS control algorithms leads to a requirement for a modular architecture that can support such use.

4. Architecture overviews

The goals summarized in Sec. 2 are fundamentally concerned with simulations of interactions between the ITER PCS and the ITER plant, so the PCSSP must include those two main functional blocks (Fig. 2), input processes to manage inputs to the simulation (Simulation Input Managers, SIM), an output process to archive and display results (Simulation Results Managers, SRM), as

well as the interfaces between them. Pulse Schedule input will be a distinct component within PCS SIM to enable replacement by the actual ITER pulse schedule when it becomes available. A functional block representation in Fig. 2 does not necessarily imply that all functions in that block are implemented as a single module.

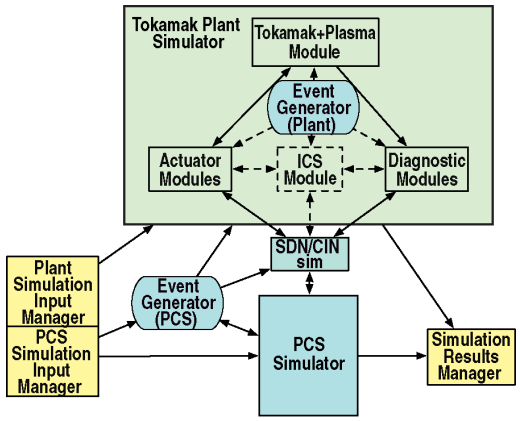

Fig.2 PCSSP functional block diagram. Major components are plant simulator, PCS simulator, SIM, and SRM. PCS and plant simulators are in CSE. Either of the PCS or plant blocks can be replaced by an external simulation or source of data to support use cases shown in Fig. 1.

In the plant, actuator modules simulate actuator responses to commands (either for ITER or a currently operating tokamak), diagnostic modules simulate processes involved in transforming physics quantities to real-time measurements, and the Tokamak+Plasma module simulates the combined plasma and device responses to actuator outputs. The SDN/CIN module simulates delays introduced in moving measurement data from plant to PCS and commands from PCS to plant. SDN = Synchronous Databus Network (commands & diagnostic data) and CIN = Central Interlock Network (machine protection control data). The EG module(s) serve to trigger simulation of user-specified off-normal events in plant.

The PCS simulator will contain multiple components, whose description is not completely specified and in fact will be defined by development of the PCS using PCSSP. However, it must include a set of Control Units, which receive input signals and produce output signals to perform specific functions such as feedback control, an Exception Handling function that detects and produces responses to those off-normal events that require triggering a change in control action, and a PCS Supervisor function to interpret the Pulse Schedule.

The Event Generator and Exception Handler functions are sufficiently large and novel developments that they are described in a separate paper [13].

The Interlock Control System (ICS) module incorporates the CIS and PIS (CIS = Central Interlock System, an ITER overall protection circuit that responds to dangerous events, whose simulation module is provided by ITER [14], PIS = Plant Interlock Systems, individual plant system, e.g. actuator/diagnostic, protection circuits).

The Simulation Input Managers define all data needed to execute a simulation, which includes configuration data to customize individual modules or an entire simulation. For example, PCS modules will be configured from data to support iterative algorithm development, plant modules will be configured from data (whenever feasible) to model multiple tokamaks, and the SDN/CIN module will be configured from data to support an evolving definition of signals that travel on the ITER SDN [15].

The Simulation Results Manager provides methods to support user understanding of simulation results, and includes data visualization, during and after simulation, archiving and restoration of simulated data, and data analysis. All data for a simulation are archived as a group, identified by label. This includes data needed to restart a simulation at specified times. The SRM must be able to read from data archiving systems of operating devices as well as of ITER.

PCSSP does not require that a user make use of the SIM and SRM functions, but doing so automates the bookkeeping involved with manipulating input and output data associated with simulations. For example, if SIM functions are used to specify all input, then SIM creates a "package" of all inputs that can be automatically archived with the simulated data.

5. Status of PCSSP development

Figure 3 shows a high level view of the Simulink implementation of the architecture under development, which is not yet final. It shows simulators for the plant (top section), PCS (bottom section), and SDN/CIN (small block in-between). The high-level plant design shown is nearly complete. The PCS represents only a candidate design, a version of which is to be delivered as part of the current PCSSP task. The PCS development in this work will serve as an input to a separate PCS design task. In particular, the EH architecture will likely be the starting point for the final EH design in the PCS. However, the separate PCS design task will determine the final PCS architecture, which can then be evaluated and eventually converted to realtime implementation code using PCSSP.

The PCSSP design philosophy is to develop custom infrastructure only if it makes simulations easier to perform. Otherwise, the project relies on capabilities provided by Simulink. In the design, tradeoffs must be made, e.g., standardizing a data interface methods allows more automation of configuration tasks and thereby improves ease of use, but reduces flexibility. Some methods to improve ease of use can also have the side effect of increasing simulation execution time.

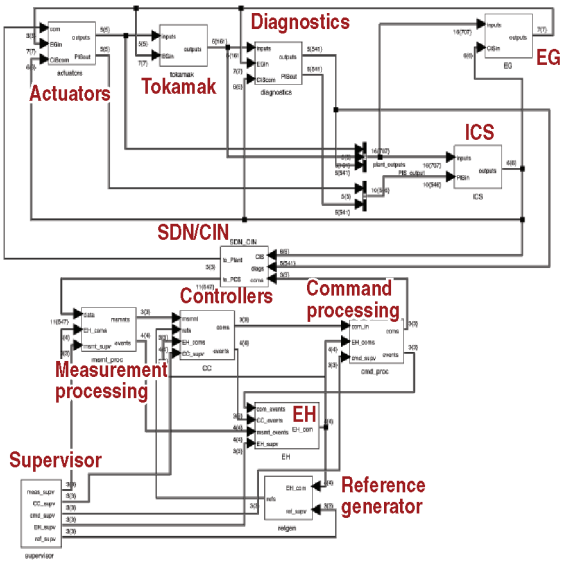

Fig.3 Simulink top-level architecture implementation.

To maintain flexibility, PCSSP imposes no constraints on internal structure of individual modules. However, some standardization is imposed on inter-module connections to allow modules provided from multiple sources to be easily connected and to take advantage of capabilities provided by PCSSP. For example, it is expected that signal lines connecting modules will be standardized to be Simulink bus signals (thick lines in Fig. 3). This choice allows individual module developers to add outputs to their module without becoming inconsistent with users' previously constructed simulations. It also provides a form of self-documentation of signal line content.

Data used to configure and initialize simulation modules is also standardized to some extent. Although data content for each module will be determined by the module providers, the data is required to be represented by a Matlab data structure with some required fields representing data for configuration, initialization, time series inputs, and module documentation. The documentation field itself has some prescribed content, such as descriptions of module input and output signals. Use of a single data structure provides a method of grouping all data needed to use a module for simulation. This helps the user to identify the data needed to prepare the module for simulation and makes it easier to develop and maintain the methods used by the SRM to archive and restore this data.

Appropriate standardization can also support validation. Identifying system characteristics that can be modeled by the same calculations for all devices allows the same modules to be used in simulations for existing devices and for ITER. The only change typically required is in the data used to configure the modules. Most plasma models can be represented this way, but many plant systems cannot. Methods are being developed to manage the modeling and simulation of multiple devices so as to exploit commonalities whenever available. Figure 4 illustrates the file structure that will be used to manage Simulink model files and input data for multiple device simulations.

Methods provided by PCSSP will support construction of new simulations using PCSSP-installed modules. For example, separate Simulink model files containing generic and device-specific modules will be provided. Drag-and-drop of a PCSSP-provided module will automatically create the data structure needed to set up that module. Many of the configuration data fields will have default values, but all are modifiable by the user.

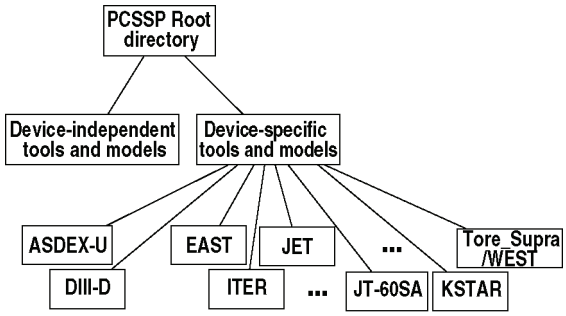

Fig.4 File management structure to support multiple device simulation. Actual devices supported will depend on IO deployment strategies.

PCSSP is required to provide methods for connecting with existing detailed simulation codes. Combining two or more simulation components into a single simulation can use either an integrative approach, i.e., all components fully integrated into the same environment, or a co-simulation approach, in which some of the components execute separately but exchange data in such a way as to synchronize the separate simulations. Connections between PCS simulations in PCSSP and external full Plant simulations such as those in Fig. 1 will be made through co-simulation. Incorporation of individual external modules in PCSSP will be done by embedding the module code in a Simulink S-function.

Acknowledgment

This work was supported by the the ITER Organization under ITER/CTS/6000000037. The views and opinions expressed herein do not necessarily reflect those of the ITER Organization.

References

[1] R. Meyers, et al, Modeling of Plug-In Series Hybrid Powertrain for USPS Carrier Route Vehicle, Proc. SAE World Congress, Detroit, Michigan, Mar. 2007; T. Denery, et al., Creating Flight Simulator Landing Gear Models Using Multidomain Modeling Tools, *AIAA Modeling and Simulation Tech. Conf.,* Keystone, Colorado, 2006; M. Castillo-Effen, et al, Modeling and Visualization of Multiple Autonomous Heterogeneous Vehicles, *IEE Systems, Man and Cybernetics Int. Conf.*, v.3, 2005, 2001

[2] W. Suttrop, et al., Predictive Simulation of Tokamak Discharge Behaviour based on Simple Scalings, *Proc 32nd EPS Conf Plasma Phys, Tarragona*, vol. 29C, 2005, P-4.076.

[3] G. Raupp, et al., Fusion Eng. & Design **82**, 1102 (2007).

[4] R. Albanese, et al., Fusion Eng. & Design **66–68**, 715 (2003).

[5] G. de Tommasi, et al., IEEE Trans. Plasma Sci. **35**, 709, (2007).

[6] J.B. Lister, et al., Proc. 38th European Phys. Soc. Conf. on Plasma Physics, Strasbourg, France, 2011, P1.105.

[7] M. Ferrara, et al., Alcasim Simulation Code for Alcator C-Mod, Proc. 45th IEEE Conf. on Decision and Control, San Diego, CA, 2006, 2238.

[8] M.L. Walker, et al., Fusion Eng. & Design **82**, 1051 (2007).

[9] D.A. Humphreys, et al., Nucl. Fusion **47**, 943 (2007).

[10] PCSSP Final Requirements Document v.1.2, 4FK397 (2012).

[11] Preliminary Architecture for PCSSP v1.0, 13, CAARKY (2012).

[12] PCSSP Functional Specification, Draft for IO review v1.0, C9766M (2012).

[13] G. Raupp, et al., Event Generation and Simulation of Exception Handling with the ITER PCSSP, O1-5, this conference

[14] Vergara Fernández, et al., Fusion Eng. & Design **86**, 1137 (2011).

[15] H.G. Kim, E.J. Lee, SDN Software Architecture Design Description v.1.4, B7AWBK (2013).