Managing an Evolving Control Environment

G. Neu, R. Cole [*], K. Lüddecke [*], G. Raupp, W. Treutterer, D. Zasche, T. Zehetbauer
Max Planck Institut für Plasmaphysik, EURATOM Association, Garching, Germany
[*]*Unlimited Computer Systems GmbH, München, Germany*

ABSTRACT

The tokamak ASDEX Upgrade is equipped with a fully digital machine and discharge control system. Long term operation and continuous upgrading of such a system must allow for changes of physical discharge definition, real-time control procedures and I/O system peripherals. A system administration platform is under construction to keep track of the real-time system's relevant process and signal inventory along with its evolution and which allowsone to check its consistency with the discharge definition selected for the execution of actual experiments.

1 INTRODUCTION

During their lifetime the control systems of large experimental devices undergo frequent modifications. Improvements and optimization measures are carried out continuously to achieve higher performance, functionality, operability and reliability. Constant evolution is not an indicator of bad design but a measure of the important role the control system plays in the experimental set-up.

System evolution of an experimental control environment may be characterised by three separate but interdependent modification cycles, Figure 1. Device operation is based on the variation of discharge parameters collected in the discharge definition and observation of the resulting changes in physical key parameters from cycle to cycle. The set of discharge parameters is modified by the physicist in charge, typically on a hourly timescale. Knowledge gained during operation periods leads to the improvement of existing, and development of new, control algorithms and methods. Hence the real-time software will have to undergo modification by the software engineer. Change of the control software or availability of new or improved sensor signals or actuator can require modification of the peripheral I/O hardware. Component failures require instant replacement of modules during the operation of an experiment, and as a result, peripheral I/O hardware can and will be reorganized and altered by the technical staff.

To guarantee correct operation of the control system all these modifications have to be made consistently. Due to the system size, with a large number of hardware and software components and its complexity and interdependencies, it is extremely difficult if not prohibitive to keep track manually of the consequence of any intervention. This is especially difficult as many scientists, engineers and technicians collaborate in the execution of these tasks, all with their own scope and knowledge.

What is therefore needed is a management concept which, based on information about the real-time system's relevant process and signal inventory and their evolution, ensures consistency with the discharge definition selected for execution. In this paper we will decribe the ASDEX Upgrade discharge control environment and its parametrization, and derive requirements for a system administration platform, Chapters 2 and 3. In Chapters 4 and 5 adequate information representation mechanisms are introduced which help to describe the control processes and signals and retrieve appropriate information. These are used to validate the physical discharge definition and to create an executable discharge programme.

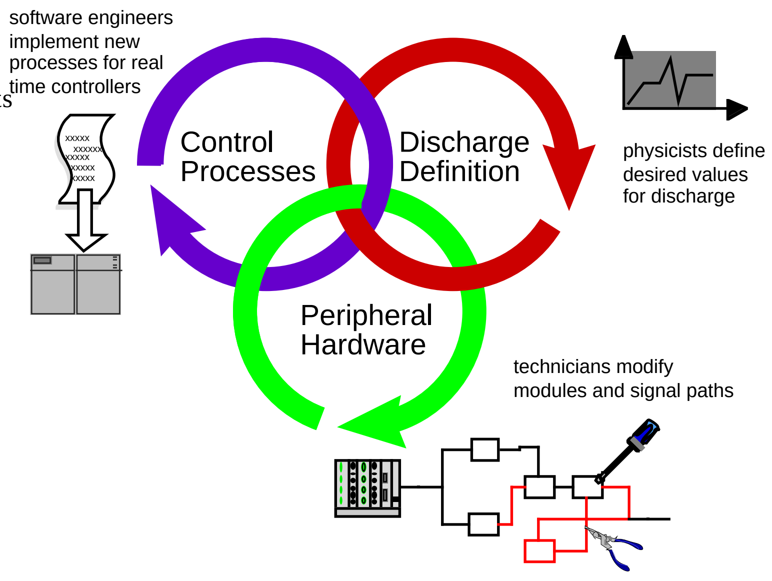
software engineers implement new processes for real time controllers
Control Processes
Discharge Definition
physicists define desired values for discharge
Peripheral Hardware
technicians modify modules and signal paths

Fig. 1 : Modification Cycles in System Evolution

2 ASDEX UPGRADE DISCHARGE CONTROL SYSTEM

ASDEX-Upgrade is a mid-sized divertor tokamak designed for investigation of plasma boundary physics and wall interaction in various geometries and under conditions similar to those in a fusion reactor. Physical control tasks range from the active stabilization of the plasma's position in the vessel and the shaping of its outer countour, to control of refueling and heating systems by feedforward access or feedback of plasma quantities and monitoring of technical subsystems and overall technical and physical discharge status. To perform these real-time control tasks and to operate the machine in the most flexible way ASDEX-Upgrade has been equipped with a fully digital control environment as indicated in Figure 2.

The discharge control system (DCS) is a hierarchical cluster of transputer-based computers running a multitude of real-time monitoring, feed-forward, and feed-back control processes. Processes require information on hundreds of actual physical and technical quantities and access to tens of technical actuators to act back onto the plasma. I/O is performed via a widespread system of peripheral hardware modules. Each controller transfers its output data to and receives input from specified interface points which relate to the diagnostic sensors and technical actuators previously configured by the machine control system (MCS).

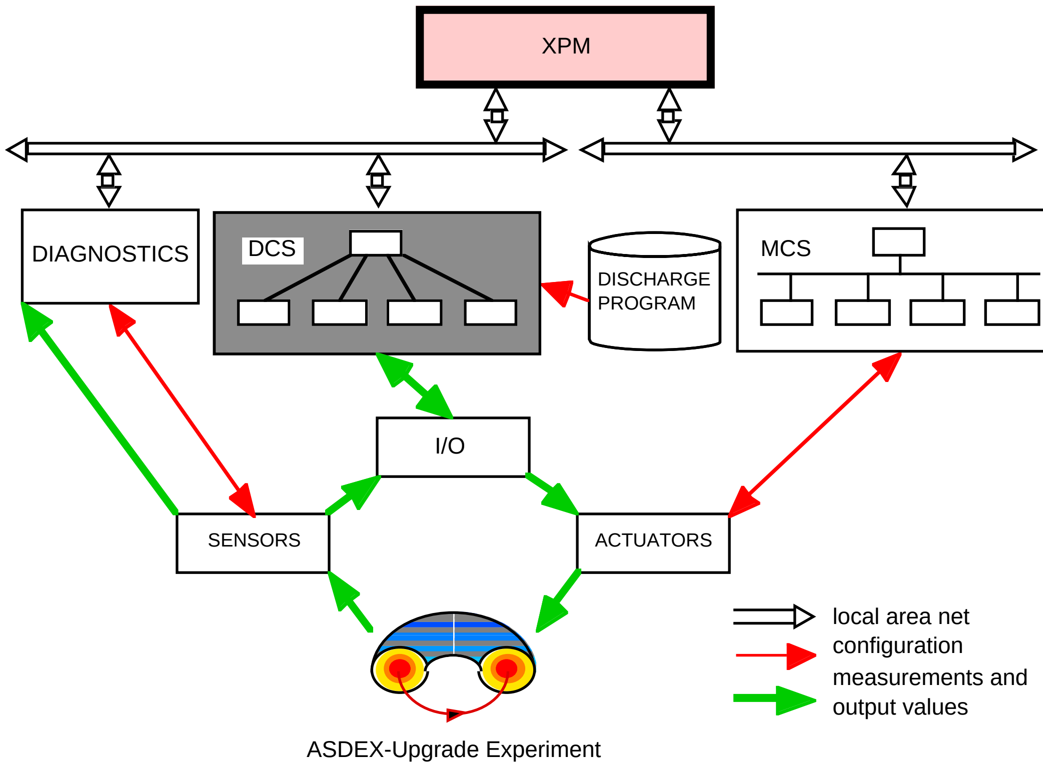

Fig. 2 : ASDEX Upgrade Control System Environment

An experiment management (XPM) software platform integrates MCS and DCS for automated experiment operation. It serves as a user interface for the technical operator and processes his commands, hiding the associated complex time ordered sequences of actions and information exchange between the distributed systems [1]. The core of the DCS is the real time control processes. These are specified by experimentalists and control system designers and implemented as code modules by the software engineer. Each process performs operations on time-varying signals. For each process an entry in a header file exists which provides identification of all its signals and defines their usage.

Signals can be subdivided into I/O ones characterized by their transfer characteristics, addresses, and format and reference parameter signals whose values must be provided in the discharge program. A manually maintainded global quantity descriptor database (QTD) contains the defining characteristics of all signals in the DCS.

It is via the peripheral hardware that processes access actuators and sensors to interact with the experiment. When a signal coming from a diagnostic sensor's interface point reaches the computer port, it will have passed through a series of modules such as decoupling and pre- amplifiers, ADC or digital input, multiplexors, parallel to serial converters and optical and electrical transmission lines. This also holds for an output signal on its way to an actuator's interface point. Signal paths are specified by the control system designer and set up by the technical staff to meet the format and transmission characteristics required by the control computer to correctly access, identify and interpret a signal. Beyond the interface points, sensor and actuator characteristics have to be known to correctly interpret an I/O signal. Whereas the latter are set in the MCS, the former are input both into the QTD entry of the corresponding input signal and into a diagnostics database.

To prepare specific experiments the physicist designs a discharge definition consisting of the full set of desired signals required to parametrize a given control software version. A signal oriented editor is used to input values for configuration switches for process activation, desired values for monitoring trajectories, parameter switches and associated timebases. The executable discharge programme is created by linking the discharge definition, the MCS operation parameters defining actuator characteristics, and the peripheral I/O system's transfer characteristics and sensor characteristics as given in the QTD. The dependencies of these various inputs are shown in Figure 3.

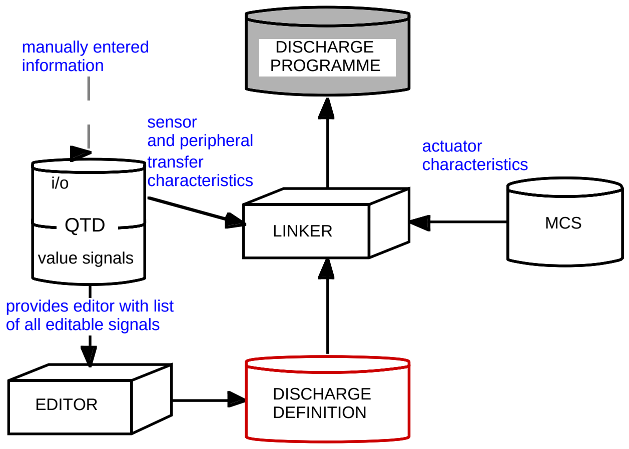
manually entered information
DISCHARGE PROGRAMME
sensor and peripheral transfer characteristics
actuator characteristics
i/o
QTD
value signals
LINKER
MCS
provides editor with list of all editable signals
EDITOR
DISCHARGE DEFINITION

Fig. 3 : Assembling a Discharge Programme

3 REQUIREMENTS FOR CONTROL SYSTEM ADMINISTRATION

There are a number of ways in which inconsistencies can be introduced into this system. Major changes in functionality begin with the specification of a new process description which leads to the implementation of a new code version. If new signals are required these are specified in the QTD. Necessary modifications of the I/O configuration (e. g. additional I/O signals, changes in I/O format, rearrangement of already existing signals) have to be implemented in the peripheral hardware, including diagnostics sensors, and the corresponding updates made in the QTD. One should note that inserting or replacing a single mux module may mean having to recompute the transfer characteristics of several signals. Finally, the I/O routines of the control computers have to be adapted. Discharge definitions which were not designed for the new software must be enhanced and/or modified to account for new signals or signals whose usage has changed.

Clearly consistency is not enforced by the system described above and information required to validate a scheme is not provided automatically. The description of the process structure and lists of signals of a given software version - needed to define a consistent discharge description - must be manually assembled by inspecting the configuration files in the source code. There is also no true description of the I/O hardware structure at the module level - needed by the technician to assess the impact of changes on the I/O signal entries of the QTD. Modifications in the sensor characteristics are usually protocolled in the diagnostics database from which they have to be manually transferred into the QTD.

From the above considerations it is clear that maintaining consistency within the control system is far from trivial. It relies on the correct interplay between control processes and peripheral hardware and between control processes and discharge definition. All of these are independently configured and modified by different people and on different timescales.

An ideal solution would be a complete model descriprion of the entire system software and hardware, which could be used to allocate and parametrize real-time processes and activate and configure real-time I/Oo hardware. With the given system complexity and implementation resources, however, such an approach is currently far from being realised.

To provide the flexibility required by experiment goals or operation procedures, the ability to change control system characteristics is fundamental. Hence, what has to be provided are supporting measures for the prevention and the detection of inconsistencies.

A first step towards prevention is an organization of system information avoiding multiple definition, and an adequate and correct representation of this information. Whenever changes are made it should be clear where the corresponding information is maintained, and mechanisms should exist which update related information automatically. Any intervention should be based on reliable and precise knowledge of the set-up of the components to be handled. For the physicist designing a discharge definition, the information of interest would be an inventory list organized by process of all signals required in a downloaded software version; a software engineer can draw valuable information from a visual representation of a controller's process structure; and a technician will appreciate an I/O hardware representation which will allow him/her to modify module descriptions instead of having to go through endless lists of I/O signals

Detection of inconsistencies relies on the specification of rules which define the relations between information from the various components of the system. Instances must be created which implement these rules and extract the relevant information.

4 REPRESENTATION OF THE PERIPHERAL HARDWARE

In the previous chapters we have seen that a useful description of the peripheral hardware must be module oriented. It must mirror the installed I/O system's structure to facilitate maintainability, and it must allow the retrieval of the actual signal I/O characteristics to be processed by the controller at a given port.

The general strategy in modelling the peripheral hardware is straightforward - the real-world hardware modules are represented as module descriptions, including board and transfer characteristics. The real-world hardware connections between module I/O ports are represented as connection descriptions. The end points of this peripheral I/O system to the other system components are a sensor/actuator description on the peripheral side to sum up the characteristics of the signal preprocessing hardware outside the control system, and a controller port description as the end point of a fiber optic transmission line to define the interface to the controller internal process description.

Any relational database can serve as an implementation platform. For each type of module templates must be provided which define its static and variable properties and characteristics. Examples of static properties are the type description, front panel set-up, number and kind of I/O connections, and connector type. Even before being actually used a specific module will allow personalization of its representation. This is done by assigning values to some of the module's variables: a serial number for future identification is defined, values for specific parameters are set (e.g. the addresses of multiplex channels or amplification factors), status (never used, repaired, modified ...). Finally, when the module is installed, a location description and interconnections with other modules can be put into the description. Special 'modules' describe the two ends of the peripheral hardware: on the one side the controller I/O boards with ports, and on the other side virtual modules which describe the interface points with the sensors and actuators of the experiment. A sensor module will typically contain the name and characteristics of a sensor and one entry describing its connection to some module of the peripheral hardware.

Such a model representation describes the entire I/O hardware system topology and transfer characteristices, using all required information and avoiding redundancy. The transfer scalings of a peripheral signal on its way to a controller can simply be inferred by stepping through the modules and extracting the local transfer characteristics along the signal path. To check whether a signal path is consistent with given controller software all that is needed is the relation connecting the signal name to an actuator or sensor and the information concerning at which port that signal is expected by the controller, Figure 4.

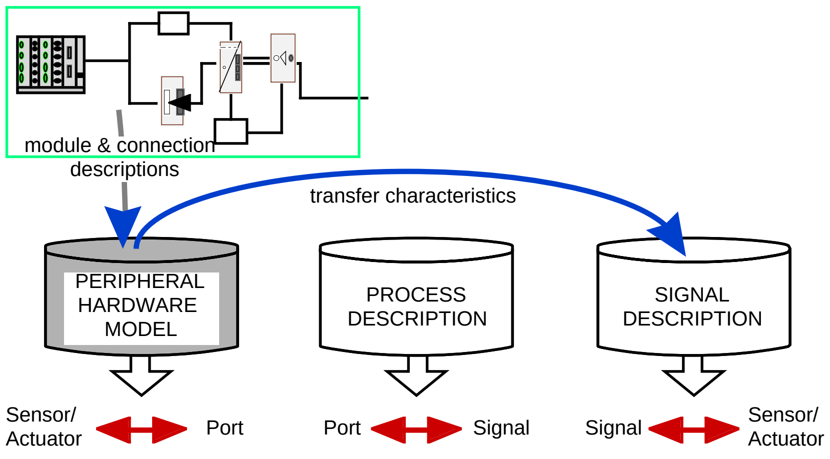

Fig. 4 : Application of the Peripheral Hardware Representation

If the loop indicated in the figure can be closed the signal path exists and the signal can be accessed with the actually derived I/O characteristics. The latter are then automatically updated in the signal description of the QTD. Otherwise, the signal path hardware is not installed correctly or the physicist's specification to use a particular sensor/actuator as a control signal is wrong or the control process expects the signal at another port or not at all. In such a case, the model description can assist in debugging by giving information on where the loop is broken, where the signal path ends or which set of signals is actually sent to a given port.

5 ADMINISTRATION OF THE PROCESS STRUCTURE

Enhancing the software of an evolving real-time system to provide the functionalities required for administration is both difficult and inefficient. This is especially true for a distributed system running on a variety of platforms and programmed in different languages. Checking the compatibility of a given peripheral hardware set-up or the completeness and correctness of a discharge description would mean having to activate all controllers and related subsystems.

Therefore, to deal with this problem at ASDEX Upgrade a generic model was devised, which can be configured for various software versions and parametrized by arbitrary discharge programs. To keep it as simple as possible, the model only contains those data structures and rules or functions strictly necessary to perform the desired administrative tasks.

The hierarchical architecture of the model reflects both the existing software structure and the tasks to be performed. Rules can be classified according to their complexity and generality. Whereas some checks can be performed locally and by every process (e.g. determining whether all signals it requires are included in a given discharge program), others will decompose into sequences of specific actions. As an example for the latter, consider the real time task of controlling the antenna coupling of the ICRH. Its implementation in ASDEX Upgrade's control system software requires the cooperation of two control processes running on different controllers [2]. When activated, the first one feedback-controls the antenna coupling by computing a desired gap between antenna and plasma boundary. Provided the second process is configured to control the outer radius of the plasma and to receive this desired value as input, it will compute currents for fast control. The correct execution hence relies on particular settings of various software control switches and on the definition of desired values and gain factors for both processes.

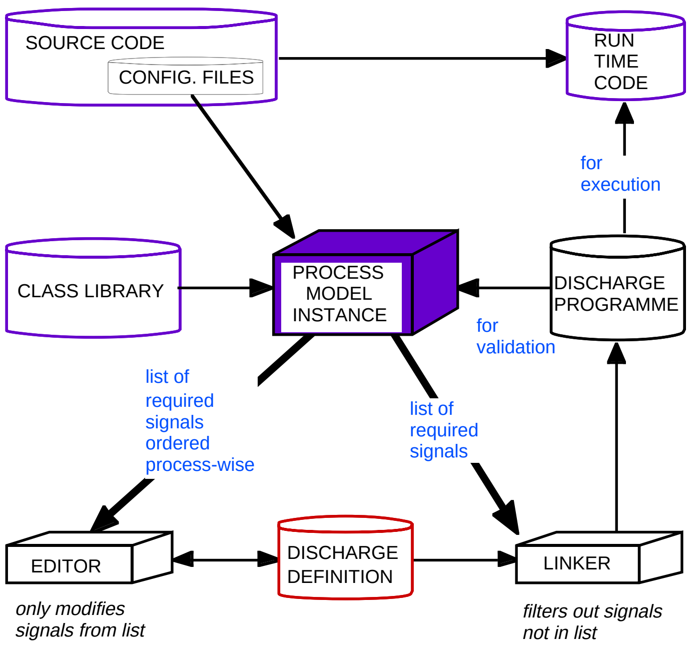

Fig.5 : Process Model: Creation and Application

The model has been specified using an Object Oriented (OO) paradigm, and a prototype is currently being implemented in C++. The process class library contains classes ranging from basic template processes, elementary processes (e.g. single-variable PID controller) to derived classes defining the complete process structure of a real-time controller. Inherent properties of OO methodologies such as encapsulation of methods and structures into a class, inheritance in derived classes and polymorphism of methods were helpful in designing a library of reusable and easy-to-enhance modules.

The generation of a particular model instance is performed using customized methods which address configuration tables declared in header files of the original source code of the control software. This guarantees that the model structure corresponds to that of the target system, see Figure 5.

Once a model has been activated, it can provide the discharge definition editor with the lists of processes and those signals which have to be defined for a complete discharge definition. A similar list is passed on to the linking editor, which will add to the discharge programme only those I/O-signal descriptions which are actually needed. Finally, the discharge programme can be validated according to the methods defined in the classes of the process class library.

6 CONCLUSION

In an evolving control system for a large experiment such as ASDEX Upgrade, control processes, discharge definitions and peripheral hardware are modified continuously. In the past, a system description was obtained by entering critical information manually into a set of unrelated databases. This carries the risk of introducing inconsistencies. With the new approach presented here a system description is extracted from the installed hardware and software modules, and rules are added. Within this system model, consistency checks can be performed and discharge programmes can be generated and validated. We expect this approach to considerably improve the management of change within ASDEX Upgrade's control environment

REFERENCES

[1] Richter, et al.:"Overview of the ASDEX Upgrade Experiment Management Software" ; Proc. 17th Symposium on Fusion Technology, Roma (I), 1992, p. 1077

[2] T. Zehetbauer, et al. :"Management of RT Processes for Plasma Parameter Optimization at ASDEX Upgrade" ; Proc. 9th Conference on Real-Time Computer Applications in Nuclear, Particle and Plasma Physics, Chicago, (USA), 1995