Fusion Engineering and Design 89 (2014) 146–154

Contents lists available at ScienceDirect

Fusion Engineering and Design

journal homepage: www.elsevier.com/locate/fusengdes

ASDEX Upgrade Discharge Control System—A real-time plasma control framework

W. Treutterer[a,*], R. Cole[b], K. Lüddecke[b], G. Neu[a], C. Rapson[a], G. Raupp[a], D. Zasche[a], T. Zehetbauer[a], ASDEX Upgrade Team

[a] Max-Planck-Institut für Plasmaphysik, EURATOM Association, Boltzmannstraße 2, 85748 Garching, Germany
[b] Unlimited Computer Systems GmbH, Iffeldorf, Germany

HIGHLIGHTS

- The ASDEX Upgrade Discharge Control System (DCS) is a comprehensive control system to conduct fusion experiments.
- DCS supports real-time diagnostic integration, adaptable feedback schemes, actuator management and exception handling.
- DCS offers workflow management, logging and archiving, self-monitoring and inter-process communication.
- DCS is based on a distributed, modular software framework architecture designed for real-time operation.
- DCS is composed of re-usable generic but highly customisable components.

ARTICLE INFO

Article history:
Received 2 May 2013
Received in revised form
22 November 2013
Accepted 8 January 2014
Available online 16 February 2014

Keywords:
DCS
Plasma control
Exception handling
Data-driven workflow
Real-time network
Configuration

ABSTRACT

ASDEX Upgrade is a fusion experiment with a size and complexity to allow extrapolation of technical and physical conditions and requirements to devices like ITER and even beyond. In addressing advanced physics topics it makes extensive use of sophisticated real-time control methods. It comprises real-time diagnostic integration, dynamically adaptable multivariable feedback schemes, actuator management including load distribution schemes and a powerful monitoring and pulse supervision concept based on segment scheduling and exception handling. The Discharge Control System (DCS) supplies all this functionality on base of a modular software framework architecture designed for real-time operation. It provides system-wide services like workflow management, logging and archiving, self-monitoring and inter-process communication on Linux, VxWorks and Solaris operating systems. By default DCS supports distributed computing, and a communication layer allows multi-directional signal transfer and data-driven process synchronisation over shared memory as well as over a number of real-time networks. The entire system is built following the same common design concept combining a rich set of re-usable generic but highly customisable components with a configuration-driven component deployment method.

We will give an overview on the architectural concepts as well as on the outstanding capabilities of DCS in the domains of inter-process communication, generic feedback control and pulse supervision. In each of these domains, DCS has contributed important ideas and methods to the on-going design of the ITER plasma control system. We will identify and describe these essential features and illustrate them with examples from ASDEX Upgrade operation.

1. Introduction

In view of shrinking fuel resources for conventional power generation, opening up alternative methods of energy production has become a pressing task. Besides the renewable sources like solar and wind energy, thermonuclear fusion can produce energy in a sustainable, almost inexhaustible way. The fundamental principle seems simple – two hydrogen atoms are combined to a helium atom, releasing an amount of binding energy. Realisation, however, is one of the most challenging projects of scientific and technological research.

In a fusion reactor built on base of the Tokamak principle, a magnetic shaping field confines hot Deuterium and Tritium plasma to avoid direct contact with the reactor vessel. Magnetic induction, neutral beam injection, ion cyclotron and electron cyclotron frequency heating systems provide auxiliary power to start the exothermic reaction. Gas injection and frozen hydrogen pellets bring fuel to the plasma. Fluid mechanic and electromagnetic

* Corresponding author. Tel.: +49 89 3299 1496; fax: +49 89 3299 2580.
E-mail address: Wolfgang.Treutterer@ipp.mpg.de (W. Treutterer).

0920-3796/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.fusengdes.2014.01.001

effects determine the inner plasma structure. They lead to a variety of temporal and localised effects with strong influence on the efficiency of the fusion reaction. Internal particle transport processes paired with dosed feeding of radiating gas species and localised heating allows indirect control of the plasma structure, control of power flows as well as removal of impurity and exhaust particles.

Establishing a sustained fusion reaction thus is a complex task and a sophisticated control system is needed to coordinate the multitude of involved systems, to ensure stable operation and to assist in protecting the device and its environment. On the way towards an economically viable fusion power plant many experimental devices over the world are exploring the principles of plasma physics as well as methods for plasma control [1–4]. One of them is ASDEX Upgrade with its Discharge Control System DCS [5].

DCS provides all functionality needed to conduct plasma experiments, coordinating a multitude of measurement and actuation devices to control and optimise the behaviour of plasma and technical systems. As described in Section 2 these functions do not only comprise classical feedback control but also exception handling with intelligent methods to deal with special situations in a fault tolerant way. Section 3 summarises the main requirements and the derived main concepts resulting in the distributed, modular and configuration-driven framework system architecture of DCS. Sections 4–7 highlight selected aspects contributing to the effectiveness and versatility of DCS: control workflow management, testing and inspection facilities, extension interfaces and deployment.

2. Functions for plasma control

DCS is an entirely digital control system and contains all basic elements of a feedback controller from sensor data sampling and measurement pre-processing (blue boxes in Fig. 1) over single and multivariate control algorithms (magenta boxes in Fig. 1) to command output (red boxes in Fig. 1). A reference generator and a segment scheduler (white boxes in Fig. 1) do not only transform waveform definitions to actual reference signal values. The reference generator can also intelligently adapt waveforms with respect to the measured plant state, while the segment scheduler allows to select alternative sequences from the pulse schedule in response to unplanned events.

2.1. Measurements and plasma reconstruction

More than twenty external diagnostic systems with hundreds of channels supply information about plasma and plant state. For control and monitoring in DCS raw sensor signals are processed to reconstruct main physics quantities such as plasma separatrix geometry [6], magnetic flux coordinates [7], electron and neutral density and density profiles [8], radiated power, power fluxes, effective heating power components, stored energy, plasma confinement parameters [9,10], MHD and locked mode amplitudes and locations [11,12] and disruption precursors [13,14]. To distribute and parallelise the working load, part of these quantities are not computed by DCS but directly by real-time enabled digital diagnostic systems [15].

The condensed information is subsequently fed to feedback controllers and monitoring algorithms. But also exception handling makes use of it: the segment scheduler selects pulse schedule segments based on identified events and their context. These, in turn, are derived from processed plasma and plant. Moreover, the reference generator can intelligently generate state-adapted waveforms during runtime. This feature is exploited by the ASDEX Upgrade soft-landing function (see also Section 2.3).

2.2. Feedback control

DCS feeds the condensed measurement information to a collection of mostly multivariable control loops. Control of elementary plasma parameters like current, position shape [16–18], pressure, density and radiated power [19] provide the fundament to ensure stable and reproducible experiment conditions even in the presence of external perturbations. Nested control schemes address advanced use cases like plasma current profile shaping or MHD mode stabilisation with mirror guided electron cyclotron beams [20–22]. As a tool for basic exploration DCS also supports the full or partial application of feed-forward commands.

The magneto-hydrodynamic nature of plasma gives rise to strong non-linear interactions between plasma parameters. Spatial distribution of plasma current, density, temperature, radiative losses, transport effects and electromagnetic fields induce global plasma regimes with distinct characteristics like L- and H-mode, as well as local instability phenomena like tearing modes. They are the reason for operation limits, such as density limit or beta limit. Variations in the current density or pressure gradient lead to localised dynamic limit cycle effects like ELMs or sawteeth with consequences on overall plasma behaviour, stability and load of structural components like the divertor. DCS controllers need the capability to adapt to the changing regimes. A large choice of algorithms from simple proportional to non-linear relay control is at disposal for dynamical selection with a control mode switch. Likewise, controller gains and even the selection of controlled parameters can be changed dependent on pulse schedule references and plasma state [23].

Commonly, plasma control systems have to implement a number of feedback control functions with different goals but requiring the same set of actuators. Examples are plasma shape control functions by gap control or by isoflux control. Both use the same poloidal field coils as actuators. DCS avoids possible conflicts by design. Actuators are bound to dedicated feedback controllers, whose controlled parameters are determined by the control mode. As each feedback controller has exactly one active control mode at a time, no conflicts can occur. Instead of sharing actuators between different control goals, the user has to define a combined control goal and a corresponding control mode.

In order to avoid steps in the command output when changing gains or controlled variables, DCS controllers offer a smooth transition option. The implementation of smooth transitions depends on the control algorithm. In general, smooth transitions are be achieved by adding a transient decay term to the control algorithm output. Whenever the control mode changes the initial term value is set to the difference between the last and the newly calculated controller output. For I, PI or PID controllers smooth transitions are implemented by manipulating the initial state of the integral component.

Since integrating control algorithms are also prone to windup in case of actuator saturation, DCS comes with an anti-reset windup management resetting the integrator while the command output is limited and the increment of the integrator component would aggravate the windup (see also [24]).

Command outputs can be confined by a configurable number and type of limits. Dependent on the actuator, limits originating from different reasons can be accumulated and the most stringent one will become active. The currents in the ASDEX Upgrade divertor coils OH2u and OH2o, for example, must not only stay below the static power supply limits. Their maximum values are also restricted by coil suspension forces. Due to magnetic attraction the current command allowable for these coils is dependent on the varying current in the ohmic solenoid. Limits can also be related to change rate, energy, or other dynamic quantities.

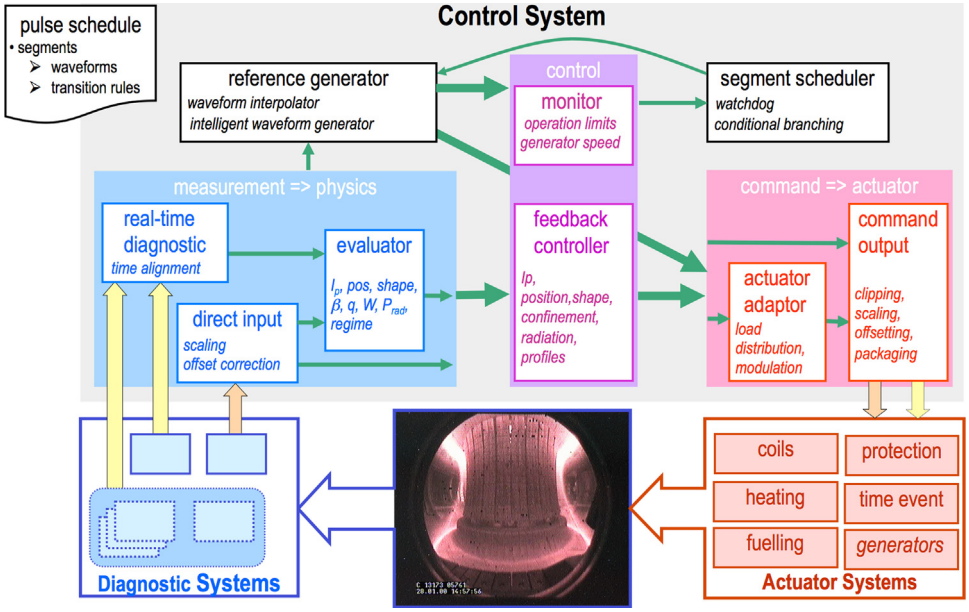

Fig. 1. DCS control system function overview. (For interpretation of the references to color in the text, the reader is referred to the web version of the article.)

2.3. Actuator management

ASDEX Upgrade is equipped with a number of actuators: ten independently controllable magnetic coil circuits for ohmic heating and poloidal field, eight neutral beam boxes with different injection angles, eight ECRH beamlines, four of which are equipped with steerable mirrors, two ICRH antennae, five gas channels, a pellet injector and a massive gas injection valve for disruption mitigation. While the magnetic field coils generally are unique actuators, heating systems often are interchangeable in their effects. As long as only part of the installed power is required for control, this redundancy can be used to distribute load and reduce fatigue, or to overcome temporal trips of individual actuators. Functions like this or transforming continuous control commands to binary pulse-width modulated actuator commands are assembled in DCS actuator management. This component keeps custom properties of actuator systems separate from re-usable generic feedback algorithms and allows versatile combinations.

2.4. Exception handling and reference generation

The complexity and non-linearity of plasma behaviour, as well as the large number of plant systems raises the potential for the occurrence of unplanned events – be it the degradation or failure of technical components, plasma instability effects, or an accidental or correlated combination of both. Most situations are not safety critical and by applying smart handling policies DCS can cope with them without annoyingly shutting down operation. Moreover, ASDEX Upgrade has a size and design, that requires a coordinated system-wide reaction by DCS to protect the investment, and this requirement is even more stringent for larger devices like JET or ITER. Therefore, exception handling is an integral part of DCS [25].

As far as possible, events are detected and handled on DCS component level, thus preventing propagation over the entire system. Measurement outliers or failed sensors, for example, are intelligently detected in the diagnostic systems or reconstruction modules. Often, they can be repaired using redundant information from neighbouring sensors. Equilibrium reconstruction, for instance, uses the function parameterisation approach [6], which is based on principal component analysis. This method allows specifying confidence limits for the measurements of magnetic pickup coils. DCS checks the sensor measurements against these limits and replaces outliers by calculating a weighted sum from values of the remaining sensors. Similarly, actuator management can replace a tripped neutral beam by another one. Thus, e.g. in case of a beam interlock due to overheated wall structures, the physics experiment can be continued without interruption. Feedback controllers encountering defective feedback input signals have a built-in recursive fallback strategy, which switches the controller to a more robust mode with other inputs and ultimately to feedforward operation.

Plasma events, however, need adaptation of control strategies spread over a variety of feedback control loops. Locked modes, for instance, often leading to disruption, can potentially be unlocked again with a combination of localised ECRH heating and increased density [22]. Applying additional central heating at reduced density can eliminate MARFEs and prevent disruptions [26]. Such and similar events are treated by the central exception handler in close interaction with the DCS reference computation [27]. Fig. 2 shows, how a repair segment with the corresponding actions interrupts the nominal pulse schedule of shot 24137 to counteract a locked mode and save the discharge.

Generation of reference and feedforward signals for controllers and actuators is organised in segments, where each segment contains time-value data points defining the waveform for each signal relative to the start time of the segment. DCS implements reference computation with two processes: a segment scheduler and a reference generator. While the segment scheduler determines the “active” segment, the reference generator interpolates the waveform data points in the active segment to obtain the reference signal values.

Segments structure the pulse schedule and enable re-using waveform definitions in other pulse schedules. In nominal operation, segments are chained according to elapsed time (Fig. 3a). The combination with rule-driven sequencing driven by the detection of planned as well as unexpected events makes segment scheduling one of the principal tools for central exception handling. In an ASDEX Upgrade pulse, for example, the magnetic coils may be powered on only after the flywheel generator spin-up is completed. This procedure is implemented with a segment, whose exit condition is the achievement of the generator working speed. This planned event triggers the scheduling of the continuation segment

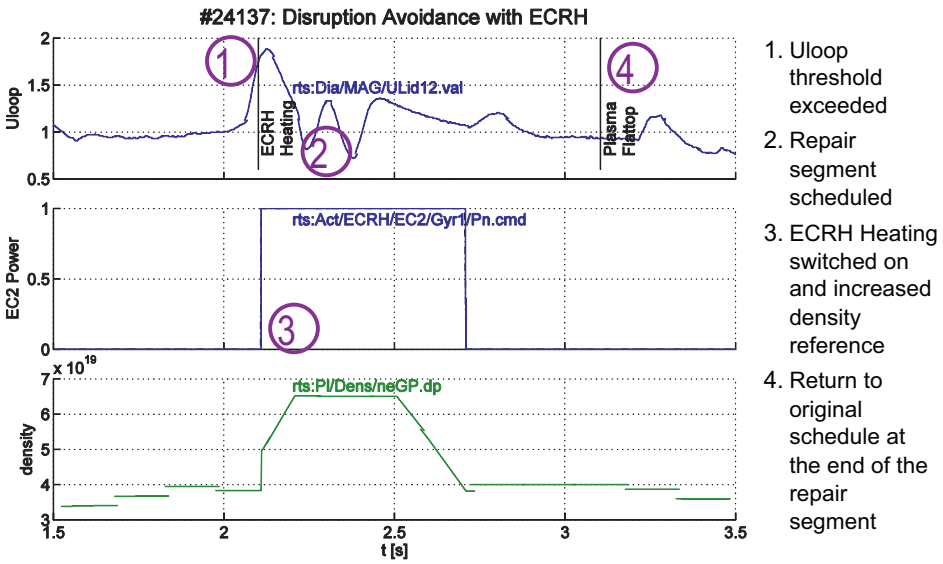

Fig. 2. Locked mode rejection to avoid a disruption inserting a repair segment that interrupts the normal pulse schedule.

where the coil currents are ramped up. If, instead, a fault should be detected, the scheduler selects a segment leading to pulse abortion (Fig. 3b). Scheduling rules and precedence of concurrent events are configurable and laid down in the pulse schedule [28]. As the above example on locked mode rejection illustrates, the segment scheduler can even interrupt a segment with a repair segment and then return control back to continue the original schedule (Fig. 3c).

Also the reference generator comes with a number of smart features. Step interpolation is applied to signals with discrete value domain like control mode switches, while continuous signals are linearly interpolated. For these, smooth transitions between segments are automatically generated if the waveform data point of a signal is placed with an offset to the segment start time a (Fig. 3a start of the "Pulse" segment). The last data point of a waveform is carried on, if the segment duration is longer than the waveform definition (Fig. 3a, "Init" and "Stop" segments). In addition, the reference generator has the capability to re-compute waveform data points, derived from the current plasma state, in real-time (Fig. 3b, "Abort" segment). This feature is most valuable in exception handling for adjusting the references to match the current plasma and plant conditions. ASDEX Upgrade uses it on a regular basis to implement the soft-landing procedure. If the soft-landing segment is scheduled due to some event, new reference waveforms and control mode settings will be computed for plasma current and position, as well as for shaping coil currents, external heating power and fuelling systems, starting from the current situation. In the first phase, plasma shape is smoothly guided to a more stable circular shape combined with a moderate plasma current reduction and a staged shutdown of the additional heating. Subsequently, the plasma current is ramped to zero at a high rate, maintaining plasma shape and position. After the plasma phase, the remaining coil currents such as the currents in the ohmic solenoid and in the position control coils are ramped down (Fig. 4).

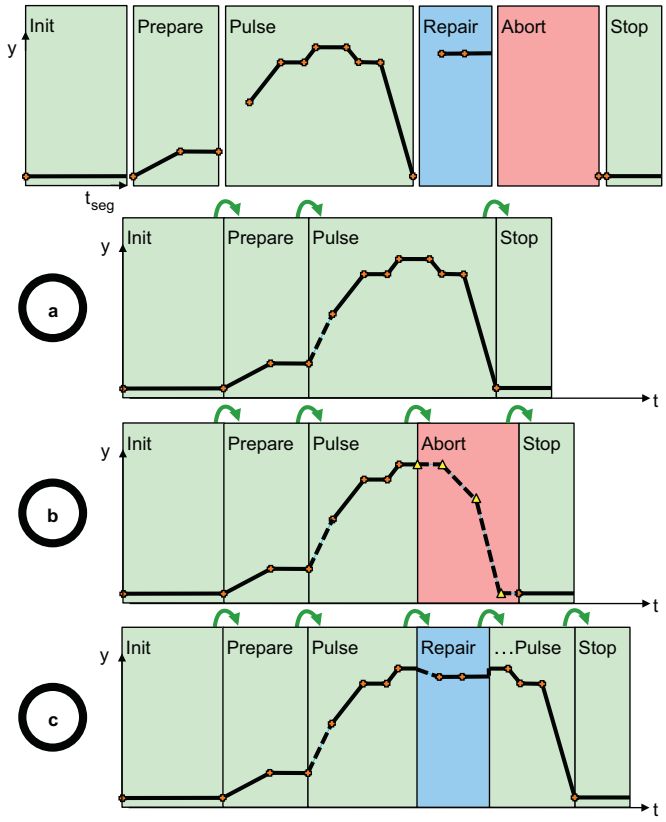

Fig. 3. Illustration of segment chaining.

In the case of soft-landing, as well as in other cases of exception handling a coordinated response of the actuators for plasma shaping, heating and fuelling is required to ensure stable and consistent behaviour. Because of its coordinating role, reference generation.

3. DCS architecture

The abundance of required control and exception handling functionality has embossed the architecture of DCS. DCS has been designed to control a complex matter under the aggravation that it is used for a research experiment where frequent modifications accounting for the latest findings must be anticipated. Selection between large variety of options, amendment of algorithms, adaptation of control structures and the addition of novel features must be facilitated.

3.1. Core concepts

A very important demand to a control system like DCS is to bridge between the world of physics, which is the object of research,

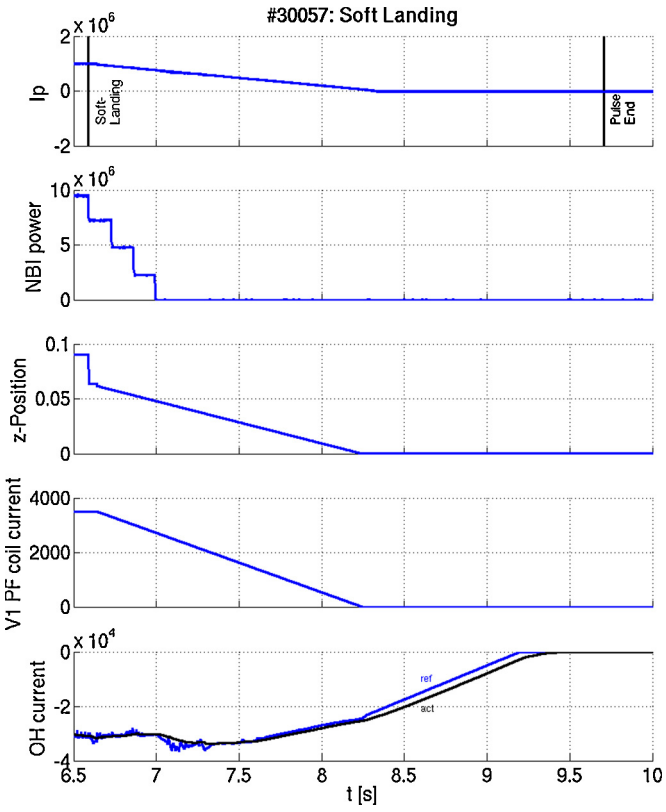

Fig. 4. Timetraces of an ASDEX Upgrade soft-landing.

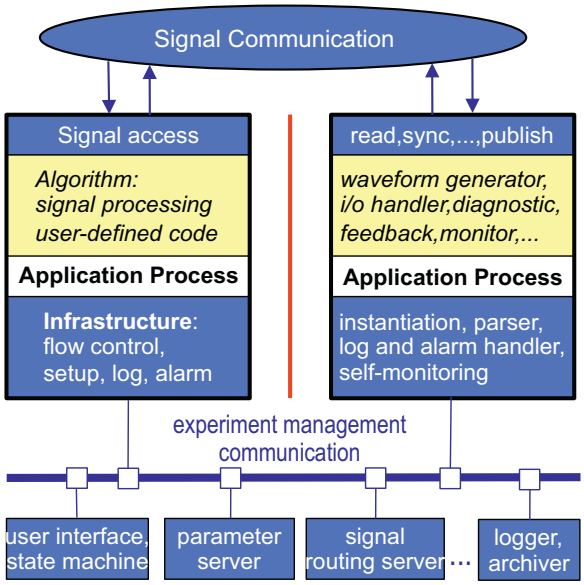

Fig. 5. Application process: separation between user algorithm (yellow) and DCS infrastructure (blue). Signal communication is the only way to exchange real-time information between the two application processes. The left box shows process functions, examples are listed in the right box. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

and the world of computers where the ideas get implemented. Physics researchers are experts in modelling the nature of fusion plasmas and in designing methods for control. However, they are not necessarily computer and software experts knowing, how to effectively implement such methods. The same dilemma applies in the reverse direction. DCS therefore has been designed to assist researchers in developing and improving control methods, while it hinders them to make mistakes and relieves them from tedious repetitive administrative tasks. Fig. 5 shows how DCS offers a user environment in the form of application processes (AP) holding the algorithmic part of control (in yellow) embedded in a framework infrastructure (in blue). Built-in consistency checks, e.g. for data types, implicit memory allocation and the preferred object lifetime management via class constructors and destructors prevent user errors. Application processes are strictly separated from each other and communicate only indirectly via in- and output signals. Signal routing between application processes is automatically established with a publish-subscribe mechanism on base of configuration data. That way the composition application processes can be easily restructured because the users of a signal need not be adapted when the producer is modified and vice versa.

Complementary, DCS offers infrastructure services comprising function libraries, real-time computing, data communication, process configuration and experiment management [29]. The latter is depicted at the bottom of Fig. 5 and contains graphical user interface, coordinating state machine, parameter server, signal routing server and back-ends for the logging and data archiving facilities. All infrastructure services are developed and maintained by software and control engineers. User environment and infrastructure services are glued together by modular framework architecture, forming the backbone of DCS.

As a modern product, DCS is committed to address the RAMI requirements: reliability, availability, maintainability and inspectability. Inspection by logging and automatic data archiving helps in analysing normal and off-normal system behaviour. The modular architecture and the clear separation of algorithms from infrastructure services make maintenance and modification straightforward. Reliability and availability are addressed by sophisticated methods for exception handling (Section 2.4) and testing, as well as by coding standards (Section 5). Thus, DCS ensures high quality in operation, as well as scalability towards future extension.

Finally, sufficient computation performance is a key requirement for real-time systems. Luckily, the on-going progress in computer technology is already a big help. Nevertheless, DCS exploits multi-core processor technology, distributed parallel processing and real-time operation systems like real-time enabled Linux or VxWorks for maximum speed and deterministic timing. The code base is written in C++, a very powerful, feature-rich, high-level but yet very fast object oriented programming language.

3.2. Core components

As a workflow scheme DCS adopts the basic idea of a block-diagram, where blocks represent control processes and linking signal lines stand for the information data flow [30]. By virtue of the polymorphic features of C++ it is possible to implement all infrastructure functions in base classes for the blocks, signals and other core components. For customisation of user algorithms it is sufficient to define derived classes, inheriting all the common functions. At the same time the language ensures, that all components adhere to uniform interfaces and follow the same standards.

The main components in DCS can be divided in function elements in the form of processes and DcsObjects, as well as data elements represented by signals, signal groups and parameters. Fig. 6 shows these elements.

Processes are mapped on one or more threads and execute commands of the central DCS state machine, the most prominent of which is the execution of the real-time algorithm. In addition, the process base class supplies a unique interface to format and stream log messages to the operator's console and web-based graphical frontends, documenting internal events and decisions for subsequent analysis. Specialised classes – in this article referred to as

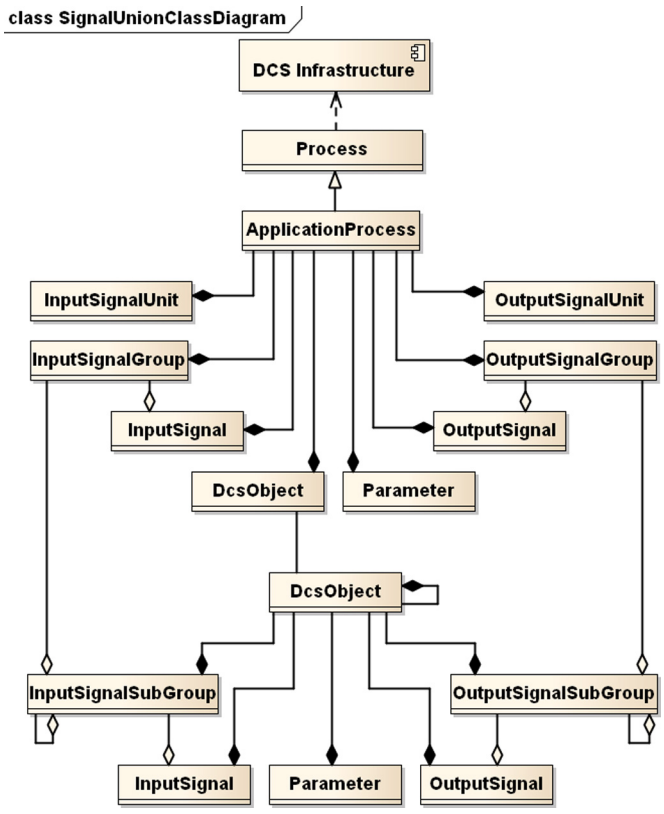

Fig. 6. DCS core components as a SysML diagram.

application processes – are dedicated to hold the users' algorithms. But DCS employs the same mechanism in form of AdministrationProcesses to implement services like control cycle pace making, self-monitoring, alarm propagation, or signal data collection and archiving. Each process specialisation defines its individual composition of parameters, signals and/or signal groups.

Signals are communication elements for information exchange between processes. They represent streams of data samples, consisting not only of the pure value. In addition, they comprise metadata like a time stamp, a confidence state classifying the data quality, and a production state indicating process activity (Fig. 7). How these meta-data are used will be explained in the following sections.

Input and output signals provide different methods depending on the flow direction. The InputSignal and OutputSignal base classes comprise properties like identifier and sampling period, as

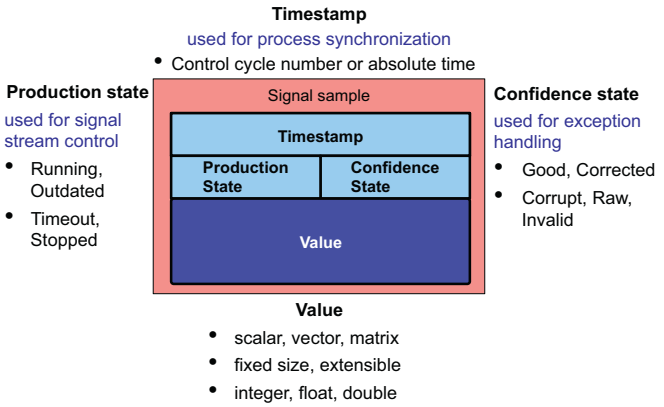

Fig. 7. Signal sample components.

well as access methods to the current sample. The derived InputSignalUnit and OutputSignalUnit classes supply particular methods to establish synchronised data-driven workflows, one of the outstanding features of DCS. Details are explained in Section 4. The actual data transport of signal samples is part of the infrastructure. Hiding the details from the application code allows to transparently convey data via memory or networks and gives DCS maintainers the freedom to adjust process deployment on computation nodes without changes to the control processes.

A common pattern in control processing are algorithms requiring a number of input signals sampled at the same time and in return producing a number of output signals. Complementing individual InputSignalUnits and OutputSignalUnits, DCS accounts for this pattern with the definition of InputSignalGroups and OutputSignalGroups, thus facilitating efficient process scheduling and network package transfer.

Processes can also be built combining sub-entities consisting of executing function elements, parameters and private process signals and signal subgroups. These entities are called DcsObjects and bundle non-trivial sub-functions, like signal filters or control algorithms to generic re-usable blocks. The ability to recursively compose DcsObjects from sub-objects and to derive super classes which can even be used polymorphically makes the concept extremely powerful. DCS furnishes a constantly growing library of DcsObjects, which are all highly configurable by configuration data.

4. Control workflow

In general, DCS control operation is structured in periodic control cycles. The Cycle Master, a pace-making process, starts control cycles in defined time periods sending new samples of a dedicated signal – the cycle number signal, whose payload is the number of the current control cycle. These samples trigger outputting DCS commands from the previous cycle, taking sensor data snapshots for the new cycle and generating reference signals for control. The duration of DCS control cycles is defined by a reference signal in the pulse schedule and can be varied during a pulse. Usually, during plasma phases a length of 1 ms is used, whereas in technical phases like fly-wheel generator spin-up, control cycles can be as long as 0.5 s.

Inside a control cycle the main workflow – the sequence in which the control blocks are scheduled – is data-driven. As soon as all requested input samples are available an application process becomes executable. In general, the process will subsequently produce output signal samples, such that follow-up processes become executable and so on, until all involved processes are done. On a system with distributed computers, multi-core CPUs and real-time operating system the method inherently opens parallel execution branches whenever more than one process is executable. Thus, DCS can exploit the computational power and minimise processing latency in response to varying processing needs during a plasma pulse without a prior, inflexible sequence specification. With such a self-organising workflow it is also straightforward to insert new processes or to decommission obsolete ones [30]. DCS' self-monitoring infrastructure function monitors that each control cycle is completed timely and executes a controlled pulse stop, if any process should not be responsive.

4.1. Synchronisation methods

Depending on the nature of a process task, different synchronisation modes can be appropriate. All methods make use of the timestamp, which is an intrinsic constituent of each sample. In most cases, processes such as plasma re-constructors and feedback controllers use the blocking-wait method to obtain input data

associated with a timestamp. The process is suspended until the data is delivered and then carries out its control function. To accomplish this, DCS uses counting semaphores for each input signal unit or input signal group. The process scheduling itself is implemented by the native operating system's methods.

Subscription mode, however, is preferred for generic output processes. These apply scaling and other transformations to a large number of signals, which may be produced by different processes and thus arrive in arbitrary order. In subscription mode, a process executes a callback function for each arriving sample of a signal or a signal group.

All synchronisation methods require that all signals in a group be produced with a same sampling rate being equal or a multiple of the sampling rate of the synchronised consumer process. This condition is true for most signals produced by the control system, since they are based on a common control cycle, and DCS verifies this condition at configuration time before each pulse.

Signals from external systems, however, are not necessarily synchronous with the control cycle. At ASDEX Upgrade, each real-time diagnostic can have its individual sampling rate. MHD control, for instance, requires NTM mode locations from ECE, which are published every 6 ms, as well as ECRH beam deposition locations, which the TORBEAM ray-tracing diagnostic calculates every 35 ms [20]. Rigid synchronisation would not be applicable in such cases. As an alternative, DCS offers also non-blocking sample access based on sample timestamps. As further detailed in Section 4.2, DCS infrastructure stores all distributed signal samples in internal sample buffers. DCS can search these buffers for input signal samples that most closely match a given cycle time and thus assemble all data to a consistent snapshot.

Some of the signals are not produced in periodic intervals. These indicate asynchronous changes such as the occurrence of an event or the change of an operation mode. For convenience, DCS supports the grouping of such sporadic signals with periodic ones, in which case the sporadic signals are exempted from the original synchronisation method and non-blocking lookup is applied for them, instead. On the contrary, a process can also be instructed to wait for a sporadic signal to handle an associated event.

Not all components in large systems might be operational at all times. Real-time diagnostics for plasma investigation for example often are in a preparation stage before plasma breakdown and do not deliver data. With the aid of the sample production state each process announces, whether it is active and clients can expect further samples. DCS uses this information for workflow management. Like sporadic signals, signals from inactive processes are exempted from synchronisation.

4.2. Signal communication

Behind the scenes, DCS infrastructure uses sophisticated methods to store and buffer samples, transport them across the boundary of computing nodes and notify sample consumers that a requested sample is available. Availability of the sample, in turn, triggers algorithm execution, and thus establishes the data-driven workflow.

When a process publishes a sample, it is stored in a ring-buffer on the local computation node, the sample buffer (Fig. 8). Buffering allows delayed sample access by consumers without the sample being overwritten in the meantime. This feature is important for background processes like signal archivers, downstream processes operating at lower rates and for stateful sample consumers like filters.

When a consumer process requests a sample, a registration is stored in the sample buffer of the corresponding signal. If the matching sample is stored in the buffer, the buffer issues a notification leading either to the release of a semaphore or to the execution of a callback function, depending on the synchronisation method employed by the consumer. Finally, the requested samples are copied to the consumer's input signals, just before it starts processing.

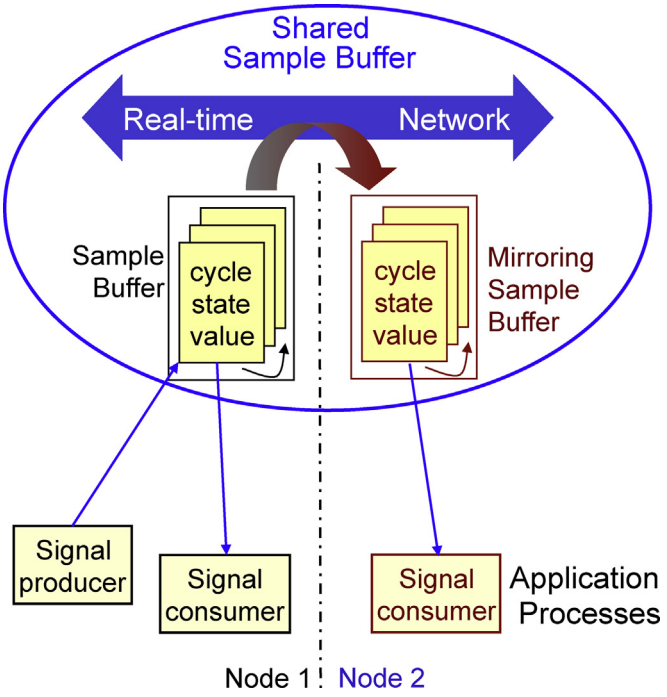

Fig. 8. Signal communication with the shared sample buffer.

When consumers and producers reside on different computation nodes, the samples are transported via real-time networks. Dedicated gateway processes adapted to network technology and transport protocol manage this operation such that an arbitrary number and arbitrary types of networks can be attached to the sample buffers. On the remote consumer side DCS automatically installs a mirroring sample buffer, where the receiving network gateway serves as a deputy producer for incoming signals. Gateway processes currently support shared memory, reflective memory and Ethernet transport with UDP protocol [31]. In addition to sample transport they also monitor the sample stream detecting and handling timeouts, which might result from network problems as well as malfunctions of the signal producers.

While nowadays the control system processes of DCS all fit on one modern multi-core CPU, real-time networks are still indispensable to provide a scalable architecture and to connect to real-time diagnostic systems.

5. Testing and simulation

Before a new or modified control algorithm is put into service it should be tested for faults or unexpected behaviour in order to minimise risks to the plant. As a first line of defence DCS is designed to exploit the strict compiler checks for consistency of data types, function arguments and class member visibility. Most DCS classes, for example, are based on the C++ template concept allowing the compiler to sort out accidental type conversions.

A lot of issues, however, arise only when an algorithm is used in a realistic context. By virtue of the strict separation of application processes in DCS a dry-run is easy to accomplish by just replacing the original sensor data sampling processes with test data injectors streaming archived data from an earlier experiment. With a plasma simulator process as replacement, even hardware-in-the-loop simulation is feasible (Fig. 9).

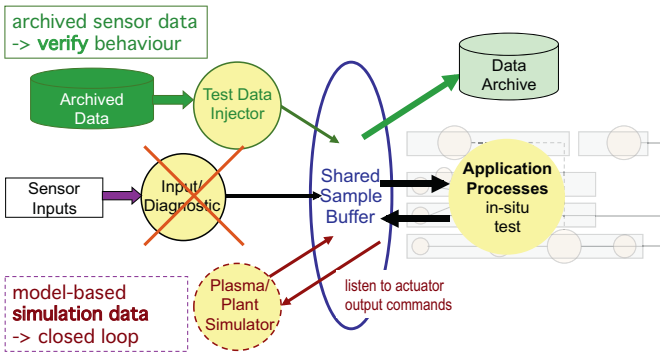

Fig. 9. Test mockup for replaying archived data or embedding a plasma simulator.

Testing with archived data is a standard operation in DCS called Replay. By choice of graded replay levels it is also possible to customise the coverage of replaced diagnostic processes. In level-0 replay all data from external components, also the outputs of real-time diagnostics, are read from the archive. In level-1 replay the real-time diagnostics are part of the test but are fed with raw measurements from the archive.

6. Integration of external systems

Although DCS is already a quite comprehensive control system, occasionally challenges occur, which cannot be satisfied or which would require enormous effort, while specialised third-party systems can provide an attractive solution.

Real-time diagnostic systems with dedicated frontend electronic and dedicated computer hardware equipment belong to this category. Real-time diagnostics often perform sophisticated reconstruction of physics quantities. TORBEAM-rt, for instance, estimates the deposition location of electron cyclotron beams, essential information for NTM control.

Another case is the trend to develop and test algorithms in a simulation environment like Simulink and generate programme source code or even executable binaries directly from there. A recent application is the RAPTOR code, combining magnetic equilibrium with a kinetic transport model to reconstruct plasma temperature and pressure profiles in real-time [10]. Such data open the door for designing advanced control methods needed not only at ASDEX Upgrade but also in future fusion devices like ITER and DEMO.

Moreover, in the collaborative fusion research community, physics reconstruction and control algorithms developed on one experimental device are often validated on a different device, which might be equipped with its own control system. MARTe, for instance, is another well-known control system framework used at several fusion devices [32]. DCS can cooperate with a MARTe installation such that MARTe algorithms can be integrated in the DCS control workflow.

In all three examples, connectivity to external systems greatly extends the capabilities of the control system. The integration interface covers data exchange and a general experiment workflow management consisting of configuration, initialisation and execution.

DCS supplies two basic approaches of interfacing. The wrapper process approach is used to integrate algorithms designed with Simulink. Code for a DCS application process is generated from the algorithm according to a given pattern. This process wraps the algorithm's entry functions behind a DCS compatible facade. The second approach is used when the external systems are independent computational entities like real-time diagnostics or MARTe systems. Here a DCS gateway application process communicates with the external system, via real-time networks or dedicated shared memory objects like message queues.

With its plug-and-play like extension interface DCS opens up a new field of capabilities and is ready to utilise state-of-the-art power tools also in future.

7. Deployment

Fig. 10 shows context and deployment of a standard DCS installation. It comprises a single-core 1 GHz PC with VxWorks operating system for compatibility with legacy hardware and a modern multi-core 3.33 GHz PC running Concurrent Linux [33], a real-time optimised Linux kernel. Six of its twelve cores are reserved for real-time processing. For robustness reasons, in particular for memory protection, DCS applications on the multi-core computer are distributed over seven logical units (LCx0–LCx6), which, however, need not necessarily be bound to individual cores. The allocation of input, reconstruction, monitoring and feedback processes on dedicated units is defined in a configuration file and can be tuned to optimise the overall performance. Inside a computation node, signal data exchange is accomplished via shared memory. External nodes are connected by a variety of real-time networks, such

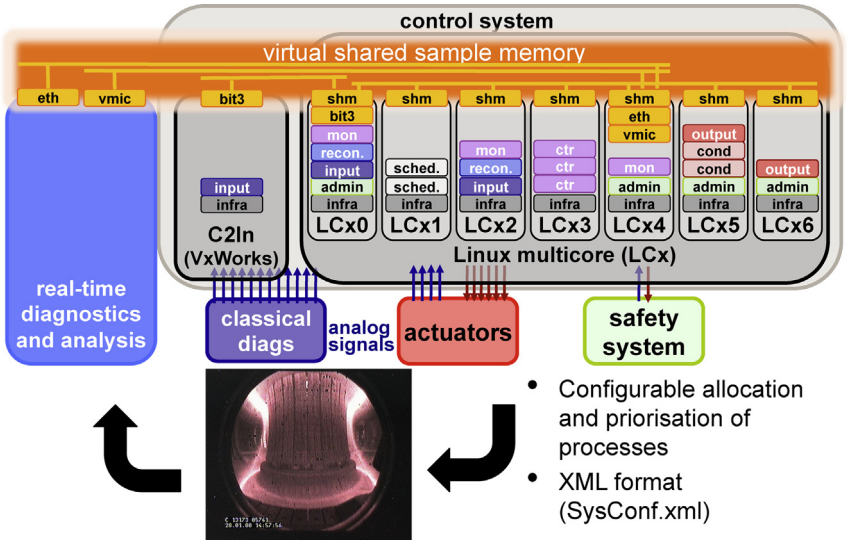

Fig. 10. Typical DCS deployment and context.

as Ethernet and two types of reflective memory, Bit3 and VMIC ([31], see also Section 4.2). Gateway processes, which are part of DCS infrastructure, encapsulate data transport on and between these networks, creating a virtual shared sample memory for signal exchange between application processes. Real-time networks also connect to real-time diagnostic systems, while classical diagnostic and actuator systems are attached directly to DCS computation nodes with custom optical fibre links. In this configuration, the entire control loop is executed in a base cycle of up to 1 ms duration. This length is given by the response time of ASDEX Upgrade fastest actuator, the power supply for the vertical stabilisation coils. DCS would even be capable of faster cycle times.

8. Conclusion

Due to the complex matter of plasma physics, control systems for experimental thermonuclear fusion devices belong to the most ambitious exponents of their species. DCS, the control system of the ASDEX Upgrade fusion experiment, tackles this challenge, employing a modular framework, which encapsulates control algorithms in application processes with uniform interfaces to common infrastructure services like data exchange. The design has been optimised to relieve algorithm developers from recurring but error-prone general tasks and offer them a simple yet powerful toolset for implementation. In the background sophisticated synchronisation methods, versatile signal transport via real-time networks, customisable process communication topology based on the block-diagram concept ensure performance, scalability, easy maintainability. The powerful integration interface to external systems opens up a promising evolution path. The concepts are even generic enough, that the system could be used for any demanding control task – not only in the fusion domain.

References

[1] R. Felton, E. Joffrin, A. Murari, L. Zabeo, F. Sartori, F. Piccolo, et al., Real-time measurement and control at JET experiment control, Fusion Eng. Des. 74 (2005) 561–566.

[2] D.A. Humphreys, R.D. Deranian, J.R. Ferron, R.J. Jayakumar, R.D. Johnson, R.R. Khayrutdinov, et al., High performance integrated plasma control in DIII-D, Fusion Eng. Des. 74 (2005) 665–669.

[3] J.I. Paley, S. Coda, B. Duval, F. Felici, J.-M. Moret, Architecture and commissioning of the TCV distributed feedback control system, in: Real Time Conference (RT), 2010 17th IEEE-NPSS, vol. 1, May, 2010, pp. 24–28, http://dx.doi.org/10.1109/RTC.2010.5750487.

[4] K. Kurihara, J.B. Lister, D.A. Humphreys, J.R. Ferron, W. Treutterer, F. Sartori, et al., Plasma control systems relevant to ITER and fusion power plants, Fusion Eng. Des. 83 (7–9) (2008) 959–970, http://dx.doi.org/10.1016/j.fusengdes.2008.06.027.

[5] G. Raupp, K. Behler, H. Blank, A. Buhler, R. Drube, H. Eixenberger, et al., ASDEX Upgrade CODAC overview, Fusion Eng. Des. 84 (7–11) (2009) 1575–1579, http://dx.doi.org/10.1016/j.fusengdes.2009.01.031.

[6] P.J. McCarthy, An Integrated Data Interpretation System for Tokamak Discharges (Ph.D. thesis), University College Cork, 1992.

[7] L. Giannone, M. Reich, W. Treutterer, R. Fischer, J.C. Fuchs, K. Lackner, et al., Real-time magnetic equilibria for NTM stabilisation experiments, in: A. Becoulet, T. Hoang, U. Stroth (Eds.), Europhysics Conference Abstracts (CD-ROM, Proc. of the 38th EPS Conference on Plasma Physics, Strasbourg France, 2011), vol. 35G, European Physical Society, Geneva, 2011, P2.105.

[8] A. Mlynek, M. Reich, L. Giannone, W. Treutterer, K. Behler, H. Blank, et al., Real-time feedback control of the plasma density profile on ASDEX Upgrade, Nucl. Fusion 51 (4) (2011) 043002, http://dx.doi.org/10.1088/0029-5515/51/4/043002.

[9] H. Höhnle, J. Stober, K. Behler, A. Herrmann, W. Kasparek, R. Neu, et al., Electron cyclotron resonance heating in the second harmonic ordinary mode at ASDEX Upgrade, in: 38th EPS Conference on Plasma Physics, vol. 35, 2011, P2.095.

[10] F. Felici, M.R. de Baar, M. Steinbuch, E. Fable, E. Fokina, L. Giannone, et al., Real-time plasma state reconstruction and fault detection using a model-based dynamic observer, in: Proceedings of the EPS 2013, 2013, P2.147.

[11] M. Reich, K. Behler, A. Buhler, H. Eixenberger, J. Hobirk, W. Treutterer, et al., Real-time current profile measurements for NTM control, in: Europhysics Conference Abstracts (CD-ROM, Proc. of the 36th EPS Conference on Plasma Physics), vol. 33E, Sofia, Bulgaria, 2009, P1.160.

[12] M. Maraschek, Control of neoclassical tearing modes, Nucl. Fusion 52 (7) (2012), http://dx.doi.org/10.1088/0029-5515/52/7/074007.

[13] L. Giannone, A.C.C. Sips, O. Kardaun, G. Pautasso, F. Spreitler, W. Suttrop, et al., Regime identification in ASDEX Upgrade, in: P. Norreys, H. Hutchinson (Eds.), Europhysics Conference Abstracts (CD-ROM, Proc. of the 31st EPS Conference on Plasma Physics, London, 2004), vol. 28G, EPS, Geneva, 2004, p. 4.131.

[14] Y. Zhang, G. Pautasso, O. Kardaun, G. Tardini, X.D. Zhang, et al., Prediction of disruptions on ASDEX Upgrade using discriminant analysis, Nucl. Fusion 51 (6) (2011) 063039, http://dx.doi.org/10.1088/0029-5515/51/6/063039.

[15] W. Treutterer, K. Behler, A. Buhler, R. Cole, L. Giannone, A. Kagarmanov, et al., Integrated operation of diagnostic and control systems, Fusion Eng. Des. 86 (6–8) (2011) 465–470, http://dx.doi.org/10.1016/j.fusengdes.2010.12.074.

[16] W. Treutterer, C. Aubanel, O. Gruber, G. Neu, G. Raupp, U. Seidel, et al., Redesign of the ASDEX upgrade plasma position and shape controller, IEEE Trans. Nucl. Sci. 43 (1 Part 1) (1996) 217–221.

[17] W. Treutterer, O. Gruber, P. McCarthy, G. Raupp, et al., Progress in shape control at ASDEX upgrade, in: Fusion Technology (Proc. of the 20th Symposium on Fusion Technology), vol. 1, Marseille, 1998, pp. 521–524.

[18] J. Santos, L. Guimarãis, M. Zilker, W. Treutterer, M. Manso, et al., Reflectometry-based plasma position feedback control demonstration at ASDEX Upgrade, Nucl. Fusion 52 (3) (2012), http://dx.doi.org/10.1088/0029-5515/52/3/032003.

[19] A. Kallenbach, M. Bernert, T. Eich, J.C. Fuchs, L. Giannone, A. Herrmann, et al., Optimized tokamak power exhaust with double radiative feedback in ASDEX Upgrade, Nucl. Fusion 52 (12) (2012), http://dx.doi.org/10.1088/0029-5515/52/12/122003.

[20] M. Reich, K. Behler, A. Bock, L. Giannone, M. Lochbrunner, M. Maraschek, et al., ECCD based NTM control at ASDEX Upgrade, in: Europhysics Conference Abstracts, vol. 36F, 2012, p. PD4.004.

[21] C. Rapson, F. Monaco, M. Reich, J. Stober, W. Treutterer, et al., Simulation of feedback control system for NTM stabilisation in ASDEX Upgrade, in: Proc. of 27th SOFT Conference, Liege, 2012.

[22] B. Esposito, G. Granucci, M. Maraschek, S. Nowak, A. Gude, V. Igochine, et al., Avoidance of disruptions at high βN in ASDEX Upgrade with off-axis ECRH, Nucl. Fusion 51 (8) (2011), http://dx.doi.org/10.1088/0029-5515/51/8/083051.

[23] W. Treutterer, T. Zehetbauer, V. Mertens, G. Neu, G. Raupp, D. Zasche, et al., Plasma feedback controller reorganisation for ASDEX Upgrade's new discharge control and data acquisition system, Fusion Eng. Des. 74 (2005) 501–505, http://dx.doi.org/10.1016/j.fusengdes.2008.06.027.

[24] K.S. Walgama, J. Sternby, Inherent observer property in a class of anti-windup compensators, Int. J. Control 52 (3) (1990) 705–724, http://dx.doi.org/10.1080/00207179008953562.

[25] G. Raupp, V. Mertens, G. Neu, W. Treutterer, D. Zasche, T. Zehetbauer, Real-time exception handling—use cases and response requirements, Fusion Eng. Des. 87 (12) (2012) 1891–1894, http://dx.doi.org/10.1016/j.fusengdes.2008.06.027.

[26] V. Mertens, G. Raupp, W. Treutterer, Plasma control in ASDEX Upgrade, Fusion Sci. Technol. 44 (3) (2003) 593–604.

[27] W. Treutterer, G. Neu, C. Rapson, G. Raupp, D. Zasche, T. Zehetbauer, Event detection and exception handling strategies in the ASDEX Upgrade discharge control system, Fus. Eng. Des. 88 (2013) 1069–1073, http://dx.doi.org/10.1016/j.fusengdes.2013.03.004 (in press).

[28] G. Neu, K. Engelhardt, G. Raupp, W. Treutterer, D. Zasche, T. Zehetbauer, et al., The ASDEX Upgrade discharge schedule, Fusion Eng. Des. 82 (2007) 1111–1116, http://dx.doi.org/10.1016/j.fusengdes.2013.03.004.

[29] G. Neu, A. Buhler, K. Engelhardt, J.C. Fuchs, O. Gruber, V. Mertens, et al., Experiment planning and execution workflow at ASDEX Upgrade, Fusion Eng. Des. 86 (6–8) (2011) 1072–1075, http://dx.doi.org/10.1016/j.fusengdes.2008.06.027.

[30] W. Treutterer, G. Neu, G. Raupp, D. Zasche, T. Zehetbauer, R. Cole, et al., Management of complex data flows in the ASDEX Upgrade plasma control system, Fusion Eng. Des. 87 (2012) 2039–2044, http://dx.doi.org/10.1016/j.fusengdes.2012.01.038.

[31] W. Treutterer, G. Neu, G. Raupp, T. Zehetbauer, D. Zasche, K. Lüddecke, et al., Real-time signal communication between diagnostic and control in ASDEX Upgrade, Fusion Eng. Des. 85 (2010) 466–469, http://dx.doi.org/10.1016/j.fusengdes.2010.04.

[32] A. Neto, F. Sartori, F. Piccolo, R. Vitelli, G. De Tommasi, F. Sartori, et al., MARTe: a multiplatform real-time network, IEEE Trans. Nucl. Sci. 57 (2) (2010) 479–486, http://dx.doi.org/10.1109/TNS.2009.2037815.

[33] Concurrent Computer Corporation, iHawk Real-Time Multiprocessors, 2014 http://real-time.ccur.com/docs/default-source/data-sheets/ihawk-real-time-multiprocessors-data-sheet.pdf?sfvrsn=8