Management of Complex Data Flows in the ASDEX Upgrade Plasma Control System

Wolfgang Treutterer[a], Gregor Neu[a], Gerhard Raupp[a], Dieter Zasche[a], Thomas Zehetbauer[a], Richard Cole[b], Klaus Lüddecke[b] and the ASDEX Upgrade Team

[a] *Max-Planck Institut für Plasmaphysik, EURATOM Association, Garching, Germany*
[b] *Unlimited Computer Systems, Iffeldorf, Germany*

Establishing adequate technical and physical boundary conditions for a sustained nuclear fusion reaction is a challenging task. Phased feedback control and monitoring for heating, fuelling and magnetic shaping is mandatory, especially for fusion devices aiming at high performance plasmas. Technical and physical interrelations require close collaboration of many components in sequential as well as in parallel processing flows. Moreover, handling of asynchronous, off-normal events has become a key element of modern plasma performance optimisation and machine protection recipes.

The manifoldness of plasma states and events, the variety of plant system operation states and the diversity in diagnostic data sampling rates can hardly be mastered with a rigid control scheme. Rather, an adaptive system topology in combination with sophisticated synchronisation and process scheduling mechanisms is suited for such an environment. Moreover, the system is subject to real-time control constraints: response times must be deterministic and adequately short.

Therefore, the experimental tokamak device ASDEX Upgrade employs a discharge control system DCS, whose core has been designed to meet these requirements. In the paper we will compare the scheduling schemes for the parallelised realisation of a control workflow and show the advantage of a data-driven workflow over a managed workflow. The data-driven workflow as used in DCS is based on signals connecting process outputs and inputs. These are implemented as real-time streams of data samples. Consequently, real-time signal management forms the foundation of DCS. The paper explains the principal features such as tagged samples, signal groups, algorithmic blocks and processes as well as scheduling schemes which allow DCS control applications to be defined as self-contained modular building blocks glued together by a software framework.

By virtue of this sound foundation, DCS is a mature but still evolving system for reliable, distributed control of an entire tokamak device coordinating and monitoring 20 diagnostic systems, 14 magnetic power supplies, 5 heating systems with a total power of more than 25 MW, 8 gas fuelling channels, a pellet injector and a killer gas gun.

Keywords: real-time control, software framework, process communication, synchronization, sample tag.

1. Introduction

In the attempt to permit advanced but safe plasma operation, state information must be acquired, reconstructed and computed in growing detail. Modern plasma control systems and the connected data acquisition and actuator systems thus have to process an ever-increasing amount of data. Raw information is supplied by a variety of specialised diagnostic systems either as analogue voltages or as digitised and pre-processed streams of data samples. Using these, plasma control has to address many different goals: reference tracking and plant protection, state reconstruction and output interlock, magnetic and kinetic control, manual and automatic scenario optimisation. Although all tasks are strongly linked by plasma physical interrelations, control system architectures try to manage this complexity by decoupling them as much as possible. Breaking down the control task into smaller, separable pieces results in a parallelised system topology with a multitude of data streams connecting the various algorithms.

While such structures can be found in all major fusion devices [1, 2, 3], the workflow, i.e. the synchronisation of algorithm execution and data streams is implemented with different philosophies. In the simplest case data streams are treated like continuous analogue signals and algorithms just use the instantaneous value without any synchronisation. Computational and network latencies inherent to digital signal processing as well as non-uniform data stream activities e.g. due to diagnostic system operation windows are neglected. Managed workflows solve some of these shortcomings by a pre-determined sequencing like in the MARTe framework developed at JET [4] or by a time-slot based pipelining scheme as used by industrial applications with EtherCAT, Ethernet Powerlink or RTnet real-time network protocols [5, 6, 7]. On the other hand, in complex systems such as plasma control, managed workflows turn out to be inflexible since any algorithmic modification changing the composition of data stream inputs requires a redesign of the execution

author's email: Wolfgang.Treutterer@ipp.mpg.de

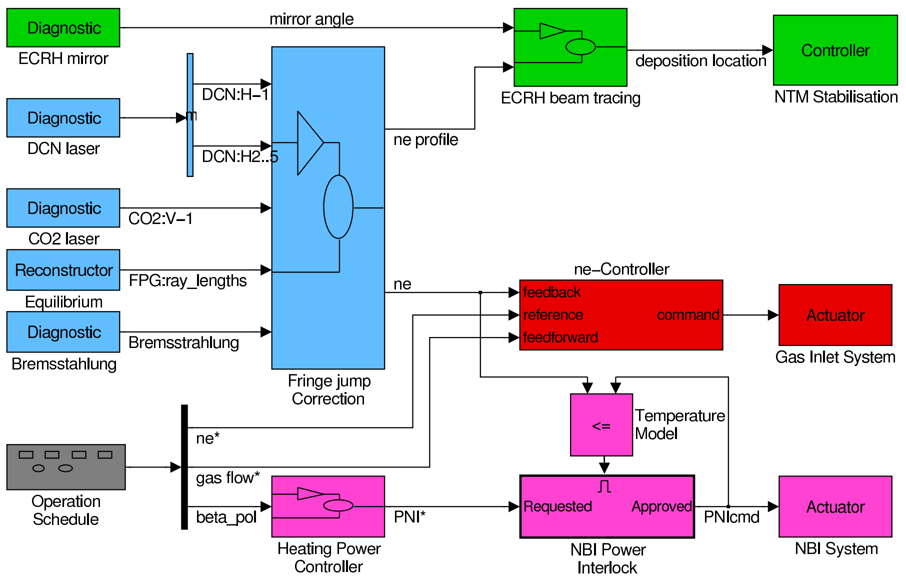

Fig. 1: Block diagram for electron density usage

sequence. Alternatively, algorithm execution can be synchronised by data streams. This data-driven, and hence self-organising workflow forms the foundation of the ASDEX Upgrade discharge control system DCS.

In the following sections we will show the effectiveness of this approach and the supplementary concepts necessary to manage the complex data flows of a comprehensive plasma control system like DCS in an easy to handle and flexible way. Some aspects of electron density usage in plasma control will serve as an illustrative example at various places. Section 2 analyses analogies between block diagrams and data flows in a control system. Section 3 will compare computation latency and algorithmic extensibility of the managed workflow and the data-driven approach. Section 4 shows how temporal stream inactivity and faulty data values are handled in a generic way by means of sample tags. Finally, section 5 presents a synopsis of the ASDEX Upgrade sample and signal framework including built-in data routing, transport and process synchronisation methods.

2. Block diagram analogies

Block diagrams are conveniently used in control system engineering to illustrate the dependencies between the functional elements of a controlled system. They utilize only two basic elements: directed lines and blocks. In our case, the lines represent data streams of signal samples and blocks stand for algorithms. Nevertheless, it is possible to model very complex systems with this simple abstraction. There are even industrial products for control system simulation and deployment whose user interface is based on block diagrams like the well-known Simulink [8] and LabView [9] signal processing tools.

This raises the question, whether some paradigms of block diagrams could also be adopted for control system design. Structuring in blocks and directed lines, allows the user to concentrate on the algorithmic part, while administrative functions are separated or hidden from the user. In addition, blocks have a generic interface with other blocks comprising just input and output ports connected to signals. Thus, signal exchange is the only means to link algorithms. In addition, the connections between blocks are not part of the blocks themselves allowing blocks to be encapsulated and re-used. The direction of signal lines defines a signal flow and imposes a constraint on the block execution sequence. Modern control system frameworks like MARTe and ASDEX Upgrade DCS employ the same simple principle. Generic Application Modules (MARTe GAMs) and Application Processes (DCS, see section 5) correspond to blocks. In both cases they exchange information exclusively via signals.

Figure 1 shows an example block diagram for electron density reconstruction and usage in plasma control drawn with the Simulink tool. Here, most of the blocks are subsystems containing again other blocks and signals, which implement the detailed function. Electron density is a characteristic property of plasma scenarios and therefore regulated to defined values by a feedback controller. It is measured by DCN laser interferometers, but the reconstruction can be subject to fringe jumps caused by MHD mode activity or pellet injection. Using other information sources like alternative laser interferometers, Bremsstrahlung and magnetic equilibrium information, such failures can be detected and often corrected with some quality degradation. The reconstructed density also serves as an input for wall-protection from overheating by NBI beams, and density profile reconstruction is needed for proper ECRH beam guidance in NTM control schemes as the beam gets deflected by spatial density variations.

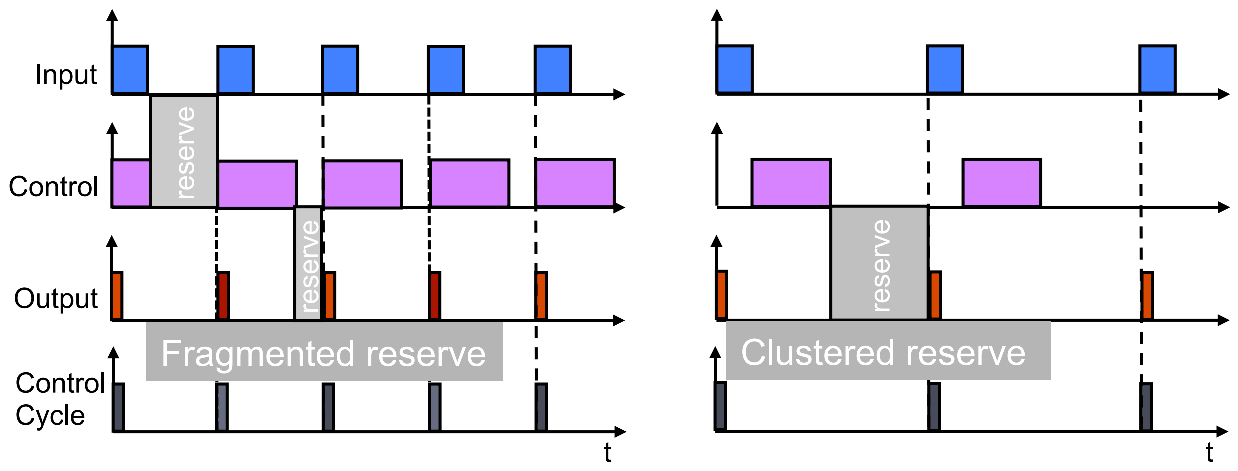

Fig. 2: Activity diagram for a managed (left) and data-driven (right) workflow

In figure 1 three basic topological data stream patterns can be identified. The first one is a chain formed by measurement, reconstruction, correction, feedback control and output to the gas inlet system, where the blocks should be executed in sequence. The data-driven workflow would be a natural implementation of sequence chains.

Stream branches, where the same signal is fed into several blocks, are another pattern. Blocks on such branches, in our example the feedback controller and the temperature model for NBI power interlock, both using the reconstructed density, can execute in parallel.

Finally, algorithms frequently need a combination of several input data streams for processing. These streams might originate from different sources with their individual sampling rates and operation windows such as the inputs to the fringe jump correction block. Synchronising block execution with these inputs is a challenging task, which, however, should be transparent to the algorithm.

The concepts found in block diagrams can serve as a sound basis to design a framework for plasma control. There are, however, also facets in control system design outside the scope of block diagrams. Real-time aspects as well as distributed deployment and the associated real-time networking are usually not in their main focus. Generally, continuous stream activity is assumed, as well as good signal value quality. Efficient solutions of the above issues are mandatory to obtain an applicable and powerful system.

3. Managed versus data-driven workflows

In managed workflows a dedicated managing instance, deterministically decides when which algorithm executes. For procedural codes this instance is the sequence of source code statements. Higher-level programs might employ a dispatcher process; a method that works also in distributed service oriented architectures [10]. Another example are time-slot based processing pipelines which often occur in combination with isochronous fieldbus network protocols like EtherCAT, Ethernet Powerlink or RTnet. All algorithms are processing in parallel. Each one is assigned a dedicated time slot, where the outputs are propagated to follow-up algorithms. This technique is also known as Time Division Multiple Access (TDMA).

Certainly, the determinism of the managed approach makes it interesting for real-time applications. On the downside, each change in the algorithms also requires the assessment and adaption of the managing instance. In complex systems with many parallel signal branches this can become a cumbersome exercise, which, unfortunately occurs frequently in existing fusion experiments.

Data-driven workflows are more robust in this respect. In their context, blocks are associated with computational processes or threads. Each block is ready for execution as soon as its input data have become available. It is up to the operation system scheduler to assign CPU resources to these processes using standard techniques like semaphores, priority and pre-emption. No additional program instance is required to trigger the start of execution and no user intervention is necessary to re-order the sequence. The self-organising property of this approach also allows for a seamless handover from one to the next process in a chain. The reserve time between the end of computation and the start of the next control cycle is accumulated at the end of the process chain, better exploits the computational resources and allows even to add new algorithms to the chain, as long as they require not more as the total reserve time.

Figure 2 shows the difference to a managed workflow based on pipelines assuming a simplified process chain comprising input/output and control for electron density feedback. Both cases have been chosen such that the total latency from input to output is the same. In the managed case, the control cycle frequency must be two times higher than in the data-driven scenario to account for pipelining. Moreover, the reserve times are fragmented. Insertion of an additional process without sacrificing latency would require a further reduction of the cycle time. The reduction must

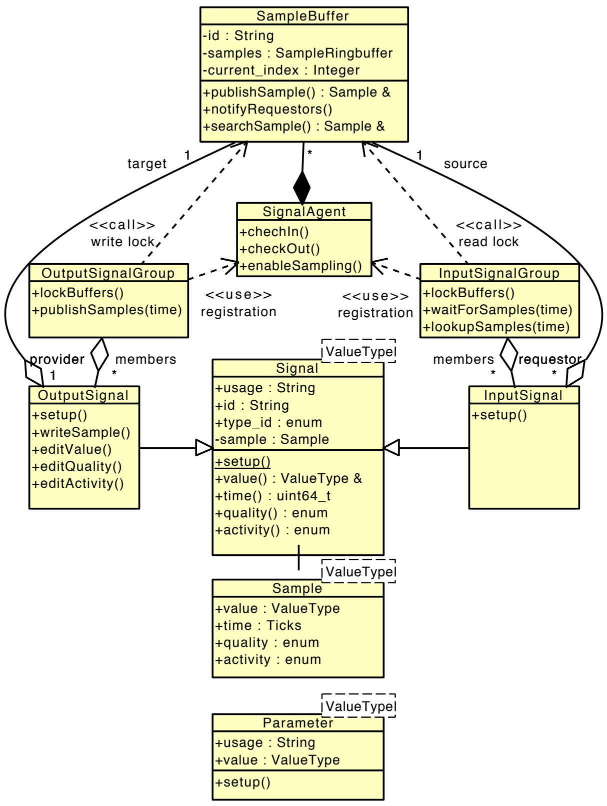

Fig. 3: Parameter, Sample and Signal class hierarchy in DCS

not exceed the shortest process reserve time. It is obvious, that this situation deteriorates with the pipeline depth. Hence, the data-driven approach appears to be more robust and flexible. In combination with the concept of encapsulated algorithms data-driven workflows also have the advantage that processes not bound to peripheral devices can freely be deployed on any computational node of a distributed system for load balancing purposes.

The implementation of data-driven workflows is based on synchronisation methods for data samples. In a distributed architecture data exchange is best organised via a shared memory with adaptors to real-time networks. Especially well suited networks for this purpose is reflective shared memory technology but also Ethernet with UDP or multi-cast protocol and also time-slot based products in combination with a shared memory layer like RTnet with NETSHM can be employed [11], [12].

The DCS infrastructure offers a choice of methods from simple lookup over blocking wait to sample subscription. As the majority of processes has more than one input and output signal, synchronisation efficiency can be boosted by the formation of signal groups. In input signal groups the process execution is triggered only when the samples of all group members have become available. Sophisticated policies are provided to handle mixed member data rates and stream activity phases. Output signal groups allow simultaneous publishing of related outputs. This feature reduces the operation system's scheduling overhead and can also be used to define data packages sent over a network.

4. Handling exceptions with sample tags

So far, data streams were considered to be continuous and the transported values were all assumed to be valid. In the reality of plasma control systems these conditions do not always apply. In the introduced example, where electron density is used for NBI interlock, the process chain passes through many stages. At any stage problems such as fringe jumps, absence of plasma, division by zero or network interruption may arise. The workflow must be perpetuated in all these cases and the protective function should be guaranteed. This can be accomplished by a local event handling strategy: if it is not possible to repair the fault, the process adds a tag to the output sample data so that subsequent consumer processes can adapt their processing algorithm appropriately. Tag values must be globally defined so that they can be interpreted by any kind of algorithm. The ASDEX Upgrade DCS utilises two types of tags, a quality tag, whose value ranges from GOOD over CORRECTED to INVALID, and a stream activity tag informing recipients, whether further samples may be expected. If the stream activity is set to STOPPED, the synchronisation methods of consuming processes will no longer wait for further samples. Local event handling is an extremely powerful mechanism to deal with exceptional situations. However, due to the separation principle, decisions in the processes are taken based on the limited scope of the individual functionality. For optimal overall response the control system must be supplemented with central global event handler processes.

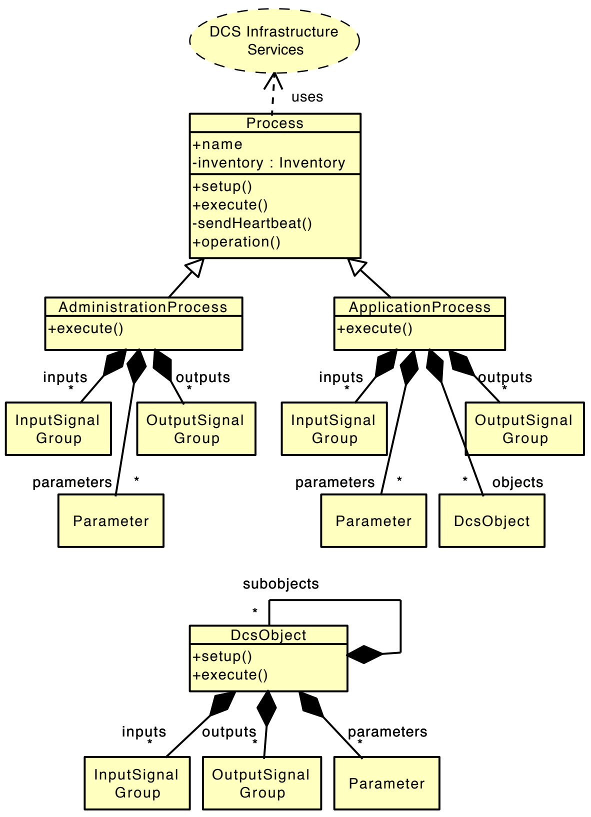

Fig. 4: Realisation of blocks as DcsObject and Process classes

5. DCS signal framework

The object oriented DCS framework defines a hierarchy of classes for data stream processing. Figures 3 and 4 show a simplified logical view of this hierarchy. The class diagram in Figure 3 focuses on signal handling and parameters. While parameters hold constant values, the Sample class forms the payload of all data streams. Samples represent snapshots of a variable quantity. This quantity can be a scalar, vector, or matrix value of an elementary data type. As already explained in the previous section the value is accompanied by a quality and an activity tag for event propagation and handling. A timestamp is attached as sample identifier. It is essential for signal synchronisation, especially in groups with mixed sampling periods.

All sample streams are conveyed via the Sample Buffer. This is a virtual global shared memory of Samples for all signals. Local instances on each computation node managed by a SignalAgent store the samples in ring-buffers and transparently transfer them to other local processes as well as to processes on remote nodes connected via real-time networks. Thus, each process has access to any signal. The DCS framework sets up the network transfer automatically based on configuration data.

The task of the Signal class is the administration of sample streams. It holds the name and type attributes of a signal as well as the connection data to the Sample Buffer. The signal class comes in two variants as InputSignal with sample query and synchronisation methods and as OutputSignal equipped with sample publication functions. As outlined in section 3, signal collections are important for efficiency. For this purpose DCS furnishes Input- and OutputSignalGroups which have to register at the SignalAgent to obtain access to the SampleBuffer.

Blocks can be realised in various ways as illustrated by figure 4. The DCS Object class forms the frame for low-level algorithms. It comprises input and output signal groups and a generic interface for configuration and execution. DCS Objects are customised building derived classes, which add algorithms and required parameters. Like blocks built from subsystems, complex DCS Object descendants can be comprise various subobjects. The DCS framework contains block-library like collections of DCS Object descendants for filters, feedback controllers and signal monitors.

The Process classes are executable threads to run algorithms. They are built from signals, signal groups, parameters and, DCS objects. Processes implementing control algorithms belong to the ApplicationProcess category. Similarly, AdministrationProcess category processes offer infrastructure services, like sample buffer administration, control cycle generation, log message forwarding, system self-monitoring and network transport. While each of theses processes has its individual composition of signals, parameters and DCS Objects, the Process base class offers general administration services like configuration and initialisation of the composite parts.

Finally, all these class definitions and administrative services are part of DCS Infrastructure, which provides further basic definitions, as well as elements for process instantiation, and configuration.

6. Conclusion

Although, ASDEX Upgrade currently is the only fusion plasma experiment with a control system that rigorously implements a data-driven workflow, this approach has demonstrated its capability to master a full-blown complex system since years. About 20 versatile control processes, 12 real-time diagnostic and 8 actuator systems are connected exchanging 800 signals with a 1 millisecond control cycle period.

This success has been made possible by just a few fundamental principles. Sample tags are used to mark sample quality and stream activity. The powerful concept of an abstract representation of the system state enforces provident algorithm definition considering exceptional cases. Sample tags form the fundament of a local event handling strategy and are an essential component for a self-organising workflow. Signal groups provide coherence of interrelated signals and increase the efficiency of process synchronisation. Finally, a data-driven workflow helps in designing a modularised system with generic application blocks and provides high flexibility for amending and extending the system.

Future development of the ASDEX Upgrade control system will put special emphasis on optimising the access to the sample buffer as a shared resource in a multi-core processor architecture, in advanced sample query methods including time interpolation and in extending general purpose DCS Object libraries.

References

[1] 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 Engineering and Design 83 (7–9) (2008) 959–970, doi:10.1016/j.fusengdes.2008.06.027.

[2] W.Treutterer, K. Behler, A. Buhler, R. Cole, L. Giannone, A. Kagarmanov, K. Lüddecke, G. Neu, G. Raupp, M. Reich, D. Zasche, T. Zehetbauer, Integrated operation of diagnostic and control systems, Fusion Eng. Des. (2011), doi:10.1016/j.fusengdes.2010.12.074

[3] G. De Tommasi, G. Alves, D. Bellizio, T. Felton, R. Neto, A. Sartori, F. Vitelli, R. Zabeo, L. Albanese, R. Ambrosino, G. Lomas, P., Real-Time Systems in Tokamak Devices. A Case Study: The JET Tokamak, to be published in IEEE Transactions on Nuclear Science, doi:10.1109/TNS.2011.2147332

[4] A. Neto, F. Sartori, F. Piccolo, R. Vitelli, G. De Tommasi, et. al., MARTe: a multiplatform real-time network, IEEE Transactions on Nuclear Science 57 (2) (2010) 479–486, doi:10.1109/TNS.2009.2037815

[5] http://www.ethercat.org/en/technology.html

[6] http://en.wikipedia.org/wiki/Ethernet_Powerlink

[7] Kiszka, J.; Wagner, B.; , RTnet - a flexible hard real-time networking framework, Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on , vol.1, no., pp.8 pp.-456, 19-22 Sept. 2005
doi: 10.1109/ETFA.2005.1612559
(http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1612559&isnumber=33857)

[8] http://www.mathworks.com/products/simulink

[9] http://sine.ni.com/labview

[10] M. Bell, Introduction to Service-Oriented Modeling, Wiley&Sons, 2008, p.3, ISBN 978-0-470-14111-3.

[11] W. Treutterer, L. Giannone, K. Lüddecke, G. Neu, G. Raupp, D. Zasche, T. Zehetbauer, ASDEX Upgrade Team, Real-time diagnostic integration with the ASDEX upgrade control system, Fusion Engineering and Design, Volume 84, Issues 7-11, June 2009, Pages 1871-1874, ISSN 0920-3796, 10.1016/j.fusengdes.2008.12.026. (http://www.sciencedirect.com/science/article/pii/S0920379608004419)

[12] L. Boncagni, C. Centioli, F. Iannone, C. Neri, M. Panella, L. Pangione, M. Riva, M. Scappaticci, V. Vitale, L. Zaccarian, Synchronous Databus Network in ITER: Open source real-time network for the next nuclear fusion experiment, Fusion Engineering and Design, Volume 83, Issues 2-3, April 2008, Pages 504-510, ISSN 0920-3796, 10.1016/j.fusengdes.2007.10.007. (http://www.sciencedirect.com/science/article/pii/S092037960700508X)