A first W7-X experiment program editor

Anett Spring[a,*], Marc Lewerentz[a],
Torsten Bluhm[a], Peter Heimann[b], Christine Hennig[a], Georg Kühner[a], Hugo Kroiss[b],
Heike Laqua[a], Josef Maier[b], Heike Riemann[a], Jörg Schacht[a], Andreas Werner[a],
Manfred Zilker[b]

[a] *Max-Planck-Institut für Plasmaphysik, EURATOM Association, Teilinstitut Greifswald, Wendelsteinstraße 1, D-17491 Greifswald, Germany*

[b] *Max-Planck-Institut für Plasmaphysik, EURATOM Association, Boltzmannstraße 2, D-85748 Garching, Germany*

[*] *Corresponding author; Tel.: +49-3834-88 2757; Fax: +49-3834-88-2509; E-mail address: anett.spring@ipp.mpg.de*

The long Wendelstein 7-X experiment programs will be segmented into arbitrary time slices. For each of these segments the planned behaviour is stored in a configuration database - and that for every component within the W7-X hierarchical layout. Up to now we have an editor to set up these segments for each single component. But generating the compound program description out of the components' segments is time-consuming experts' hand work. It is planned to implement a top-down program editor which is able to break down a high level physics program proposal to component segments while observing all the physical and technical constraints – a quite ambitious intention.

This paper describes the first step to an experiment program editor: The "eXpress program editor" is able to modify parameters of a single, a subset, or all segments within a given program, thus adapting its behaviour in an easy way. A graphical program structure overview has been implemented giving fast access to program parameters for comparison and editing. Appropriate parameter editors regard type and constraints. In a single transaction all changes are saved, resulting in a new experiment program ready-to-use for the next experiment run.

Even though changing the program's structure is not supported in this version, the eXpress program editor serves as a tool for quick program adaptations in the daily experiment routine. Furthermore this implementation will be used as a design and usage study for the later high-end W7-X program editor.

Keywords:

Wendelstein 7-X, Graphical user interface, Editor, Segment control

1. W7X control system design

Wendelstein 7-X is currently under construction as a highly complex fusion experiment with a huge quantity of subsystems, both technical components and diagnostics. The W7-X control system has to manage the co-ordinated operation while observing a multitude of restrictions. Because of the different nature of these constraints, the fast control system for the physics programs is backed up by an operation management which observes process limits and all slow procedures, and a safety system. The latter communicate with the fast control system but operate independently **Fehler! Verweisquelle konnte nicht gefunden werden.**.

To ensure flexibility, upgradeability, and programmability the control and data acquisition system design follows three basic principles: strict hierarchy, high modularity, and time segmentation. On top of the hierarchy resides a so-called 'project', underneath followed by 'components' (technical or diagnostic components, also referred to as 'groups'), 'control and data acquisition stations' (normally PCs running a real-time operation system or Java, depending on time requirements), and (software) 'modules'. As a general rule each subsystem is equipped with as much intelligence as possible, and the superordinate system does not need detailed knowledge about the properties of subordinate components.

Fast control at W7-X is PC based. Control and data acquisition features are coded in software modules which access suitable hardware. The behaviour of a component is defined by individual configuration of these modules and sub-modules and can be changed dynamically by replacing certain parameter objects at runtime. The time line of an experiment program is divided into arbitrary time slices, the so-called 'segments', fully describing the behaviour of the whole system for this fraction of time. The above-mentioned parameter change stands for a switch from one segment to another.

Both the configuration and the segment descriptions are kept in an object-oriented database **Fehler! Verweisquelle konnte nicht gefunden werden. Fehler! Verweisquelle konnte nicht gefunden werden.**. During the boot procedure a control station loads its configuration and a prepared pool of segment descriptions. While the configuration is fixed, segments can be reloaded at any time. A central segment sequence controller on top of the hierarchy gets the planned segment sequence from the user interface Xcontrol **Fehler! Verweisquelle konnte nicht gefunden werden.** which as well announces the contained segments to all components. At runtime the sequence controller checks the program feasibility and conducts the segment switches.

The editor for the experiment program preparation has to meet different features. Setting up a new program from scratch will be the most sophisticated task and requires detailed knowledge about the machine which might not be available at the beginning of the W7-X lifetime. Instead of that the focus will lie on more or less small changes in existing programs to explore the machine's behaviour step by step. Beyond this, at W7-X it will be necessary to edit an already running program in cases of foreseeable problems without the need to stop and restart a discharge. Therefore a program editor has to provide a quick method to change or even replace parts of a program.

2. Experiment program structure

Modular, hierarchic, sequential

The control system design leads to a likewise modular, hierarchical and time consecutive experiment program structure. For each component the behaviour has to be defined – and this for each point of time. The smallest entity of a program is a segment description for a component containing a structured composition of modules which fully describe the

component's activities. The aggregation of all components' segments constitutes a project segment which specifies the project's over-all activities at this point of time. By the position of segments within a program the complete time characteristics during an experiment are given. If only one component has to change its behaviour, a new project segment has to be defined. On the other hand, consecutive project segments may not change the behaviour of a certain component at all. In order to avoid unnecessary switch operations on the components' level, the reuse of segment descriptions is a strict rule, i.e. consecutive project segments may contain the identical group segment for a component.

Several segments can be combined to scenarios in order to divide a program on a higher logical level (e.g. start-up, physics relevant program(s), shut-down scenarios) and to give the possibility to reuse these scenarios in other programs. Such as the segment descriptions, the scenario and program descriptions are stored in the database.

High level parameter concept

An experiment program has to define each single parameter on a technical level as given by the configuration. To handle details on this level with all their restrictions requires comprehensive knowledge, which generally is neither possible nor sensible. To facilitate the program set-up a high level parameter concept **Fehler! Verweisquelle konnte nicht gefunden werden. Fehler! Verweisquelle konnte nicht gefunden werden.** has been developed. This approach allows to create a more physics oriented view on the program set-up while hiding technical parameters and automatically transforming physical settings to the corresponding technical low level parameters. Choices permit to choose different views, e.g. a more technical or physics view, or a project driven or local component operation view. Suitable transformation functions compute the related low level values thus providing an executable segment program for the segment control on the technical level. The parameter structure and transformation functions have to be defined and stored as part of the configuration in the database. In contrast, the real parameter values are part of the segment descriptions.

On group segments' level this concept has been already established. On the level of project segments, scenarios, and whole programs it is under discussion.

The high level parameters do know their own limits, default values and/or constraints. This information is defined using a property attribute within the parameter structure description. It can be used for instant feedback within an editor user interface to assist the user. Furthermore, high level parameters can be marked with more meta-properties. For example, parts of the segment parameters are context sensitive, e.g. they might only be relevant when running this segment in a project program. Or part of the parameters will be changed only seldom – so they can be hidden in editors depending on a defined display level. A schema to use part of the parameters for automatic name generation and graphic previews has been installed.

3. An editor to modify an existing program

At the moment we have an editor to set up the group segments for each single component **Fehler! Verweisquelle konnte nicht gefunden werden.**. Generating the compound program description out of the components' segments requires to aggregate the project segments and to line them up within an experiment program. This is time-consuming (and error-prone) experts' hand work. Since we realized a W7-X control and data acquisition system prototype at the small stellarator WEGA **Fehler! Verweisquelle konnte nicht gefunden werden. Fehler! Verweisquelle konnte nicht gefunden werden.** the need for a program editor has become obvious.

As a first step to a program editor a slim “eXpress program editor” has been realized: from the users’ point of view it resolves the structure of a program down to the group segments and displays the (context relevant) high level parameters. The user can choose between a compact and a full view. He/she can edit the parameters of one segment or a subset of segments at a time. The editor view shows the predefined constraints for each parameter, marks off-limit settings and checks for inconsistencies. The modified program can be overwritten or saved as a new program.

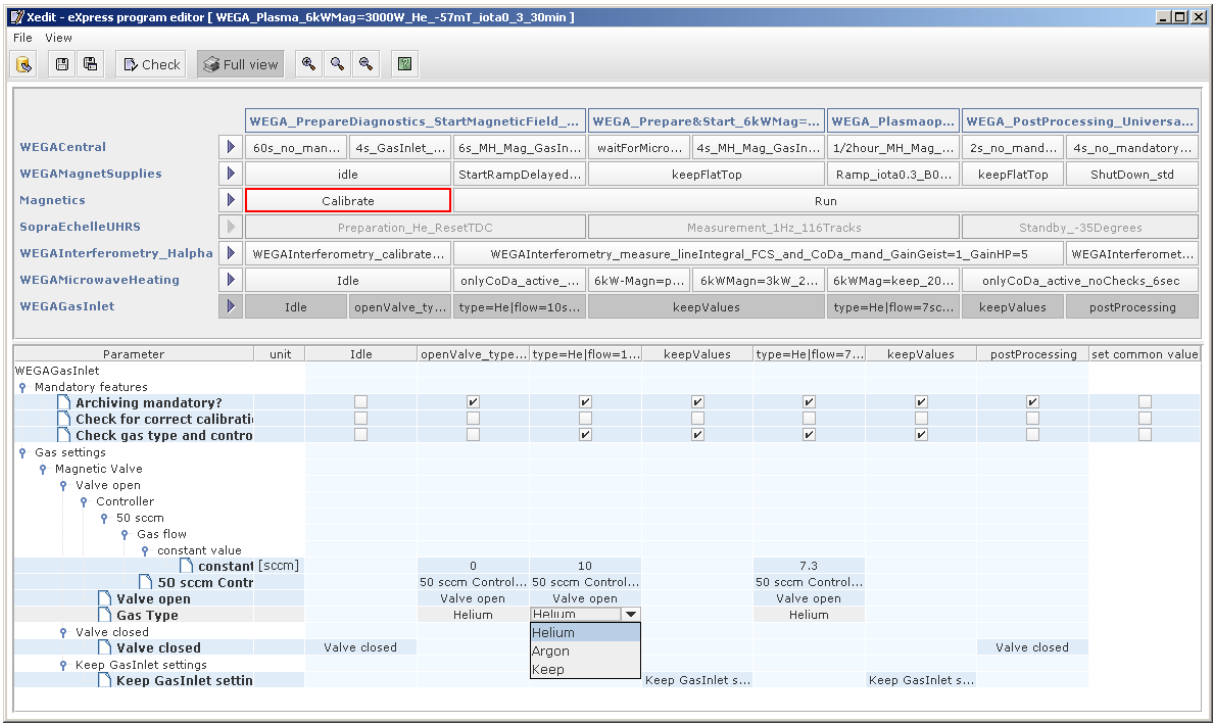

Fig. 1: A project program while editing the parameters for the component “GasInlet” (dark grey selected segment row). The user is informed about illegal settings in one of the “Magnetics” segments (marked by a bold red border).

On implementation side, displaying and editing the segment parameters is mainly a question of GUI (graphical user interface) design and user guidance. One has to keep in mind that the editor user is usually not identical with the configuration creator, so as a rule the layout has not to follow the technical implementation but the users’ view on the problem. The display implementation has to merge both the configuration and the segment descriptions for all involved components. The program structure is translated into a table with a row for each involved component and a column for each segment (the upper part in Fig. 1). Identical group segments in subsequent project segments will be shown as one segment spanning several columns. Selecting one of the table cells initiates a detailed view on the parameter structure and contents of this segment (the lower part in Fig. 1). Several segments belonging to the same component can be selected, thus offering the possibility to modify their conjointly used parameters at one go. Generally, the eXpress editor works for component programs, too, such as used for local component operation. Here the program table consists of a single row only.

For a general solution, the intelligent application of meta-properties for the high level parameters during the configuration set-up is indispensable. In this manner different views can be created and tooltips can be customized (using the display level property). Off-limit values can be marked using the limits/constraints properties. Drop down lists for discrete values and specialized editors depending on parameter type (e.g. for Boolean or Integer parameters) prevent illegal settings. Parameter arrays can be changed in array size using a

pop-up menu. A graphic preview showing shapes of certain parameter characteristics is under development.

By locally buffering the original values, the user can undo editor inputs. To handle long programs, basic zooming is implemented, but has to be enhanced to non-linear expansion of certain program sections.

Saving a modified program has to follow a more or less complicated but straight-forward algorithm: check for already existing equal segments, transformation (if necessary), re-generate the program structure and save new objects. This does not need any user input and runs fully automatically.

Because of the reuse-rule we check for each locally modified segment whether an equal segment (with the same high level settings) already exists in the database. If this is the case, we just replace it (and the possibly neighbouring identical segments) locally with the database segment. If not, each changed segment – exacting, the values of its high level parameters – has to be transformed into the executable (low level) segment. During an approve procedure a deep check for equal objects (including all sub-objects) will be performed. This procedure uses already established database methods **Fehler! Verweisquelle konnte nicht gefunden werden.**. The result is a new segment which in turn will replace the local segment and its identical neighbours. Finally, the hierarchic program structure has to be reconstructed. The column-wise aggregation of the group segments builds a new project segment. Again we check whether an equal object already exists in the database and reuse it or get a new one during the approve procedure. The same has to be done for the scenario (aggregating segments row-wise) and the whole program (aggregating all scenarios). During this process the lifecycle status of all parts of the program is set to "approved" and the ready-to-use program is stored in the database, available for the session leader to be executed.

The eXpress program editor has been realized as a Java/Swing application. To ensure fast user interaction response times it is coded highly multithreaded. During the transformation process which may take up to several minutes (depending on the amount of program modifications), the user is informed about the progress. At this stage, Swing provides all necessary methods and sufficient flexibility to ensure high GUI responsibility.

Prior to the first release the eXpress program editor has to undergo thorough tests. Because it is impossible to fully cover a GUI driven application by Unit Tests, systematic practical tests using a test-bed are the only way to ensure the application's accuracy. For these tests we use a (non-productive) test database and examine the results with low level database tools [2]. The application design with immediate user input checks (against the given parameter properties) should widely avoid faulty operation. The target user group is involved in these tests from the beginning to detect other potential malfunctions in the user interaction at an early stage.

4. Known problems and outlook

Aside from the successfully demonstrated practical use, this editor implementation is a prototype development. During the daily work we want to explore both the performance, reliability, and scalability of the technical procedures, and the users' behaviour.

There are several known problems with the current implementation, such as the long response times when fetching or writing deeply nested objects (requiring granular database access) and still unsolved concurrency problems during database transactions **Fehler! Verweisquelle konnte nicht gefunden werden.**. At the current development stage it is not possible to store intermediate results, therefore it is recommended to save the modified program immediately. This may be practicable with small changes and short programs, but because of the long

transformation procedure this does not scale with long programs and many modifications. A strategy has to be developed.

According to the well-known maxim when implementing human-computer interfaces: “Watch closely what your users do, but never ask them what they want.” we will use this first program editor to investigate and complete the edit workflows from the user’s point of view. The way will be to deploy the eXpress program editor at the prototype control and data acquisition system **Fehler! Verweisquelle konnte nicht gefunden werden.** and observe the user activities during the practical usage. Step by step the program and parameter presentation will be enhanced: introduce a compressed plain parameter view, a graphic preview, or what will crystallize as the users’ essential needs.

In the first version the eXpress editor is not able to change the selected choices within a segment definition or the structure of a program. But possibly the user will soon ask for a feature to change choices, or to remove, add, duplicate or replace group segments or whole scenarios. A suitable horizontal and/or vertical grouping of neighbouring segments can be a first step to reusable program parts. In this context the expansion of the high level parameter concept on a component-spanning or project level has to be forced. Approaches are under discussion and will complement the new program editor concept based on the so-called “Segment Program Boxes” **Fehler! Verweisquelle konnte nicht gefunden werden.**.

References

[1] J. Schacht, H. Laqua, M. Lewerentz, I. Müller, St. Pingel, A. Spring and A. Wölk, Overview and status of the control system of WENDELSTEIN 7-X, in: Proceedings of the 24th Symposium on Fusion Technology – SOFT-24, Fusion Engineering and Design 82 (5–14) (2007) 988–994, doi:10.1016/j.fusengdes.2007.02.009, ISSN 0920-3796.

[2] P. Heimann, T. Bluhm, Ch. Hennig, H. Kroiss, G. Kühner, J. Maier, et al., Database structures and interfaces for W7-X, in: Proceedings of the 6th IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research, Fusion Engineering and Design 83 (2–3) (2008) 393–396, doi:10.1016/j.fusengdes.2007.08.008.

[3] J. Maier, T. Bluhm, P. Heimann , Chr. Hennig, H. Kroiss, G. Kühner, et al., Concurrent Object Access for the W7-X Configuration Database, this issue.

[4] A. Spring, H. Laqua, J. Schacht, User control interface for W7-X plasma operation, in: Proceedings of the 24thSymposiumonFusion Technology – SOFT-24, FusionEngineering and Design 82 (5–14) (2007) 1002–1007, doi:10.1016/j.fusengdes.2007.05.052.

[5] H. Riemann, T. Bluhm, P. Heimann, Ch. Hennig, G. Kühner, H. Kroiss, et al., From a physics discharge program to device control—linking the scientific and technical world at Wendelstein 7-X, Proceeding of the 25th Symposium on Fusion Technology - SOFT-25, Fusion Engineering and Design 84 (7-11) (2009) 1598-1601, doi:10.1016/j.fusengdes.2008.12.012.

[6] H. Riemann, T. Bluhm, P. Heimann, Ch. Hennig, G. Kühner and H. Kroiss et al., Experiment planning using the high level parameter concept, Fusion Engineering and Design 85 (3–4) (2010), pp. 478–481.

[7] J. Schacht, T. Bluhm, U. Herbst, Ch. Hennig, St. Heinrich, G. Kühner, et al., Overview and status of the prototype project for Wendelstein 7-X control system, Proceeding of the 25th Symposium on Fusion Technology - SOFT-25, Fusion Engineering and Design 84 (7-11) (2009) 1723-1728 , doi:10.1016/j.fusengdes.2009.01.067, ISSN 0920-3796.

[8] H. Laqua, D. Aßmus, T. Bluhm, P. Heimann, St. Heinrich and Chr. Hennig et al., Test of the steady state W7-X control and data acquisition system at the WEGA Stellarator, Fusion Engineering and Design 85 (3–4) (2010), pp. 520–524.

[9] J. Schacht, H. Laqua, M. Lewerentz, A. Spring, A new concept for experiment program planning for the fusion experiment Wendelstein 7-X, Real Time Conference 2009, Beijing, IEEE Transactions on Nuclear Science, to be published.