Implementing DevOps practices at the control and data acquisition system of an experimental fusion device

Marc Lewerentz, Torsten Bluhm, Robil Daher, Simon Dumke, Michael Grahl, Martin Grün, Andreas Holtz, Jon Krom, Georg Kühner, Heike Laqua, Heike Riemann, Anett Spring, Andreas Werner and the W7-X team

Max-Planck-Institut für Plasmaphysik, Teilinstitut Greifswald, Wendelsteinstraße 1, 17491 Greifswald, Germany

The stellarator Wendelstein 7-X (W7-X) is a fusion device designed for steady state operation. It is a complex technical system. To cope with the complexity a modular, component-based control and data acquisition system has been developed.
During operation phases of W7-X components steadily evolve. For instance, measurement devices for diagnostics are improved, technical processes are optimized, experienced limits of the machine have to be taken into account or simple "bug fixing" is done. This requires continuous further development of the components while operating them at W7-X – a typical use case for DevOps practices.
DevOps is a software engineering practice. The term is a compound of development and operations. It aims at shorter development cycles, while the quality of the changed system must stay at a high level. This is achieved by using a highly automated tool chain.
DevOps at W7-X does not (only) mean deploying new software packages. It also comprises setting up new configurations, changing rules and constraints, improving experiment planning views etc. of components, based on new scientific and technical experiences. One crucial step of the DevOps tool chain at W7-X is the release and reconcile process. It is a well-defined and automated process that commits a change in the components configuration (Release) to the control and data acquisition system with minimal impact. During the process all existing experiment programs are adapted to this change (Reconcile). For newly added configuration parameters default values are added. Thus it is guaranteed that experiment programs are still executable at W7-X. The quality is ensured by preceding unit and integration test, configuration consistency and version checks.

1. Introduction

Fusion devices like W7-X face the challenge that they are very complex systems, which require a high reliability, availability and quality[1] of all participating components[2]. On the other hand, one often finds the requirement for high-end performance and quick extensibility. Obviously, this is also required for the control and data acquisition system. We must be able to develop new features, fix bugs or improve performance issues in a very short time. Furthermore, we have to deploy new software packages, set up new configurations, change rules and constraints, improve experiment planning views etc. of components to benefit from new scientific and technical experiences.

The CoDaC (Control, Data Acquisition and Communication) software group at W7-X is not only responsible for the development of frameworks and tools for the experiment planning, execution and data acquisition but also for the deployment, setup and maintenance of all its services. A continuous development is required, even during operation phases of the fusion device – a typical use case for DevOps practices.

2. DevOps

DevOps is a software engineering practice [5]. The term is a compound of development and operations. It aims at shorter development and deployment cycles, while keeping the quality of the changed system at a high level. This is achieved by using a highly automated tool chain and well-defined processes.

There is a wide range of application for DevOps practices at fusion experiments. This article is focused on configuration changes of components.

Often, experimental fusion devices have very individual, specific solutions for their components control and data acquisition software. There is no or only a very slim generic part implemented. In many cases, component configurations and their parametrization are hard coded, not configured. Thus, only an expert for this component is able to change something on it, a typical bottleneck situation. In this constellation, an automation of processes to change the behavior of a component is almost impossible.

In contrast, at W7-X we put great efforts to provide generic solutions with interfaces for tight, but generic integration of the components of an experiment. To be able to react quickly to requested changes we rely on automated tool chains and the high quality of our services.

3. Processes

One crucial step of the DevOps tool chain at W7-X is the "Release and Reconcile process". It commits a change in the components configuration to the control and data acquisition system with minimal impact in a well-defined

[1] Defined here as suitable to satisfy the intended purpose.

[2] At ITER components are named Plant Systems.

marc.lewerentz@ipp.mpg.de

and automated way. We call the deployment of such a configuration change “Release”. After the Release, all existing experiment programs are adapted to this change (“Reconcile”). Thus, it is guaranteed that experiment programs are still executable at W7-X. The quality is ensured by preceding unit and integration test, configuration consistency and version checks.

Except for component models and transformation functions, there is no component or project specific software. All of the developed software packages are configured according to the needs of a component. Besides using these processes at W7-X, they are also used at the WEST experiment at CEA in Cadarache, France [7].

3.1 Technical Background

The heart of the plasma control system is the W7-X Segment Control [1, 2]. The behavior of W7-X is characterized by thousands of technical parameters of the participating components. These technical parameters are also called “low-level Parameters” and get executed by a component during an experiment run. A configuration is a hierarchical structure to define all options to control a component. The parameters are used to select some of the options (implemented as sub-trees) and set corresponding values. The intended sequential change of those parameters during an experiment run is kept in so-called segments. A segment contains one “task” for each component. If a segment switch is communicated to the components, a new set of tasks is executed. Each of those tasks contain a set of component specific parameters. A special feature at W7-X is that the component continues with the execution of its task if the task to switch to is identical to the currently running. A so-called “Approve” process [3] guarantees that tasks with the same set of parameters exist only once. A corresponding life cycle state also called “approved” therefore marks a task as ready for execution.

Segments are the elementary temporal parts to define an experiment program from. A sequence of segments gives a “Scenario” and a sequence of Scenarios gives an experiment program. Planning such an experiment program is a crucial and complex job. To reduce the complexity an abstract, more physics-oriented high-level layer has been introduced earlier [3]. The so-called high-level (physics) parameters are used to encapsulate technical details. In high-level segments, the high-level behavior (also called high-level task) of the components is described using high-level parameters for a certain period of time. A high-level configuration describes all options to control a component in a more abstract, physics view. Via transformation functions, the high-level segments can be transformed to low-level segments containing a task for each component. These can be executed by the participating components. The transformation functions are implemented as Java code and define a mapping from high-level to low level parameters.
While high-level parameters describe how a component is parametrized, component models [4] give a comprehensive view of the component. They are implemented as Java code too and are used to generate several views, e.g. constraints checks, preview curves or meta data for tags. Figure **1** gives an overview about the artifacts for experiment planning and execution.

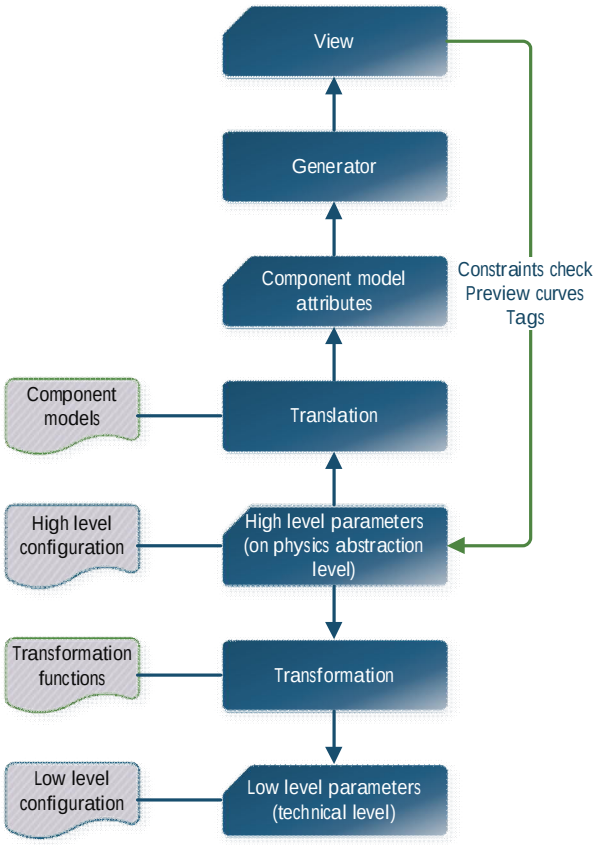

Figure 1 Experiment planning and execution artifacts

3.2. Release Process

At a fusion experiment, you usually have very complex components comprising hundreds of parameters to tweak your measurement or control algorithms. Changes to the configuration of a component can be complex. They often require intense testing before being used productively at the experiment. Therefore, we implemented for W7-X a preparation and a productive environment, each with a dedicated database. The productive database holds the actively used configuration of all components and the corresponding experiment programs. The preparation database contains configurations presently under development, as well as experiments programs for dry runs. Figure **2** shows the release process from the preparation to the productive environment.

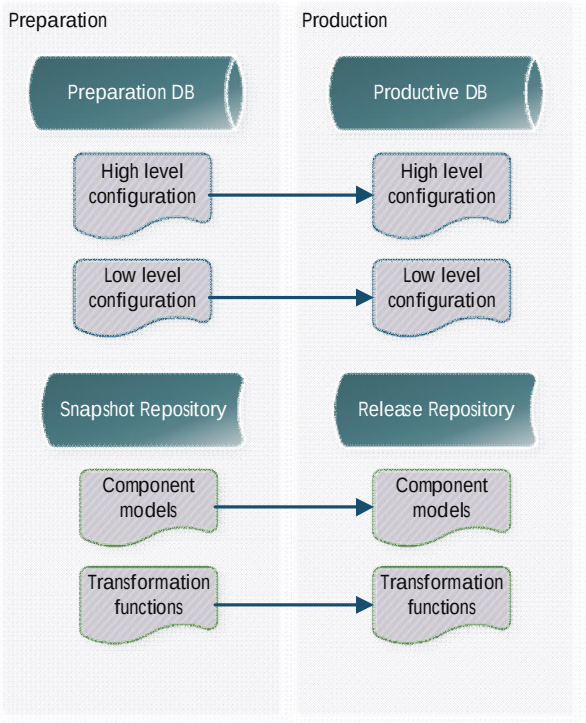

Figure 2 Release process from preparation and productive environment

We differentiate between two types of releases: releases of software artifacts and releases of configuration objects. Software artifacts are our component models and transformation functions. During the release process, they are transferred to a release repository and get a release version (e.g. 4.3.1). Our versioning database keeps information about valid versions of our software artifacts and configurations objects. A released version has to pass several tests [8], among them unit tests [9] covering all transformation functions and component models. It is possible to test against Mockups or real configurations from the productive and the preparation environment. All tests are executed automatically by a continuous integration server [10].
When all tests are passed successfully, we add the new release to the list of valid component models and transformation functions and mark outdated ones as invalid. Only releases of the software artifacts with a valid version number are granted access to the productive environment.

Releasing configuration objects is the second type of a release. A high-level and/or low-level configuration of a component is selected in the preparation database and then migrated to the productive system. Before configuration objects are released, they are tested using the corresponding software artifact releases in the preparation environment. In the tests, we check for instance whether low-level tasks are generated as expected. During integration tests, the correct operation of components is checked against the configuration objects to be released. Only if those tests are successful the release of the configuration objects is done.

3.3 Reconcile Process

After a release of configuration objects or software artifacts, the parameters in the experiment programs have to be adapted to match the changed configuration and transformation functions. We call this process “Reconcile”. It is a complex, hierarchical process that starts on task level. Changing features of a component usually means that the low-level as well as high-level configuration has changed and the transformation functions have to be adapted. Typically, the component model has to be adjusted too, because for instance, new tags must be generated or new constraints have to be checked. Figure **3** shows the steps, executed in the reconcile process.

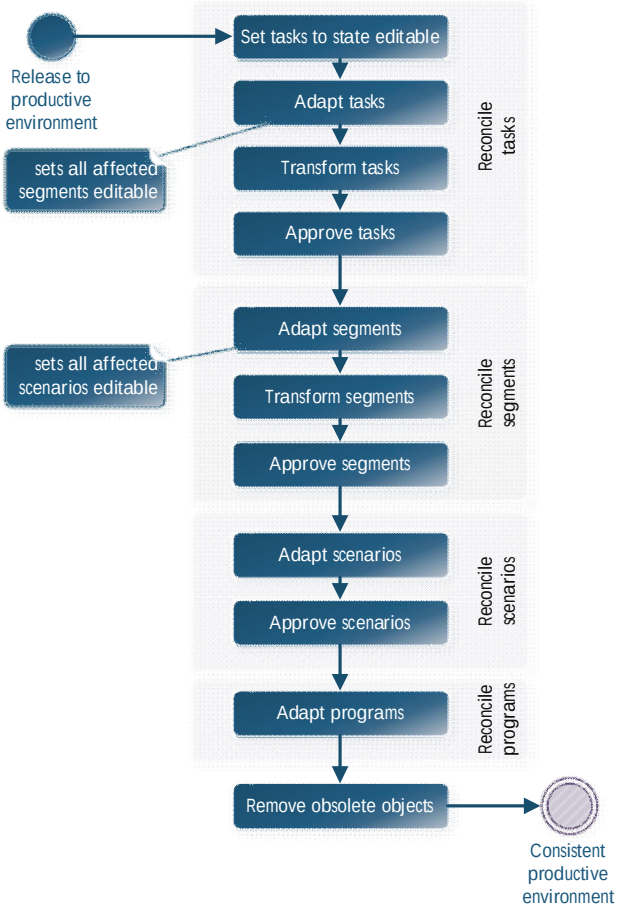

Figure 3 Reconcile Process

The reconcile process starts with the high-level tasks of the changed component. All tasks of this component are set back to state editable. The life cycle state “editable” means that those objects are changeable. The other state “approved” means that the objects are immutable and no duplicate set of parameters exists. In the so-called “Adapt” step, it is analyzed whether the set of parameters in a task fits the configuration of the component. If there is no parameter for an option (implemented as sub-tree) of the configuration, a default parameter, which is defined in the configuration, is added. If a parameter is found, which has no corresponding definition in the configuration, the parameter is removed from the task. In this way, all tasks of the component are adapted. For each adapted task, all referencing segments are determined and also set to life cycle state editing.

The low-level tasks are newly generated from the high-level tasks using the transformation functions. Afterwards the tasks are "approved". The approve step guarantees that we have no duplicate tasks in the database and sets the life cycle state.
Now the segments get reconciled. If a component is added to or removed from an experiment, all segments have to be adapted. If a new component is added, a new default task generated from the default values of the component's configuration is added to each segment. If a component is removed, the task of that component is removed from all segments. If a component is only changed, which is the most frequent use case, the segments are adapted to the changed tasks. The list of editable segments is checked against a list of changed tasks. During the approve process of the tasks, duplicates and no longer used ones might have been sorted out. The remaining task has to be used to replace the sorted out one in a segment. The adapted segments now have to be transformed to low-level segments. Subsequently, the approve process again sorts out duplicates.
For scenarios, a sequence of segments, the process is almost identical. Sorted out duplicate segments have to be replaced by the remaining ones. The following approve process guarantees that there are no duplicates on scenario level. Experiment programs are reconciled by replacing sorted out scenarios by the remaining ones. After this is finished, a clean-up process consolidates the database by removing sorted out object.

3.4 Overall deployment process

The release and reconcile processes are embedded in an overall deployment process that guarantees an ordered operation of an experiment. The schematic presentation of this overall process is illustrated on Figure **4**.
Usually the release of transformation functions and component models is done some time before the configuration release and the reconcile processes are executed.

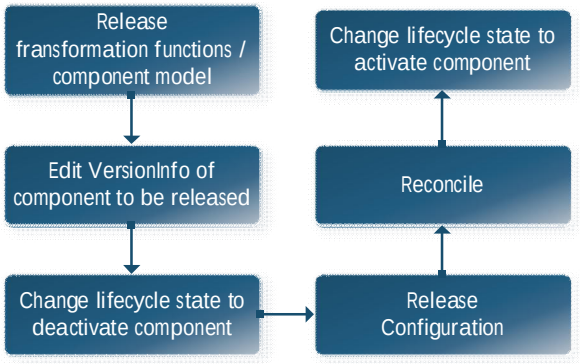

Figure 4 Overall deployment process

We have two mechanisms to assure an ordered operation of an experiment: a versioning framework and life cycle states of the components.

Immediately before the release process for a configuration starts, it must be verified, that it is not possible to run experiments at the same time. The life cycle state of the concerned component and the experiment (e.g. W7-X) is changed to prevent further operation. All other components are allowed to continue to operate autonomously and are not affected. Furthermore, our applications for experiment planning and control must be aware of the component configuration change. Via the versioning framework, the loaded component configurations used in the applications are marked as invalid.
After the reconcile process life cycle states are set back to allow operation again. Additionally the allowed versions for productive use of the component models and transformation functions are updated. The applications are able to use the changed configurations and reconciled experiment programs.
The productive database always keeps only the actual configuration of W7-X and all its components. We do not keep a history of old parameters and configurations in the productive environment. This information is found in our central archive [6].

Additionally, we execute nightly consistency checks on our preparation and productive database. Thus, we monitor the flawless work of our automated tool chain.

4. Conclusion

By implementing DevOps techniques in our way of working, we are able to roll out changes to the control and data acquisition system rapidly.

A separation between productive and preparation environment allows the development of component changes while operating the fusion device. The preparation environment allows thorough testing of component specific software and configurations before the roll out to the productive environment. Thus, we can guarantee a high quality for changed configurations of components.

We minimize the time when executing experiments using W7-X are not possible, because we use a highly automated, well-defined process.

Acknowledgments

The authors like to thank the CEA CoDaC team for many fruitful discussions.

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] H. Laqua et al., Control system of Wendelstein 7-X experiment, Fusion Engineering and Design, 66–68 (2003) 669-673.

[2] H. Laqua et al., Resource checking and event handling within the W7-X segment control framework, Fusion Engineering and Design 87 (2012) 1958-1960.

[3] H. Riemann et al., From a physics discharge program to device control – linking the scientific and technical world at Wendelstein 7-X, Proceedings of the 25th Symposium on

Fusion Technology, June 2009, Fusion Engineering and Design, 84 (1598-1601)

[4] M. Lewerentz et al., Experiment planning using high-level component models at W7-X, Fusion Engineering and Design 87 (2012) 1949-1953.

[5] Bass, Len; Weber, Ingo; Zhu, Liming. DevOps: A Software Architect's Perspective. ISBN 978-0134049847

[6] Ch. Hennig, et al., ArchiveDB—Scientific and technical data archive for Wendelstein 7-X, Fusion Engineering and Design 112 (2016) 984-990.

[7] N. Ravenel et al., Status of the new WEST plasma control system, Fusion Engineering and Design 112 (2016) 667-672

[8] G. Kühner et al., Progress on standardization and automation in software development on W7X, Fusion Engineering and Design 87 (2012) 2232-2237

[9] JUnit, Unit testing framework, https://junit.org

[10] Jenkins, Jenkins continuous integration server, https://jenkins.io/