# Progress on standardization and automation in software development on W7X

Georg Kühner[1], Torsten Bluhm[1], Peter Heimann[2], Christine Hennig[1], Hugo Kroiss[2], Jon Krom[1],
Heike Laqua[1], Marc Lewerentz[1], Josef Maier[2], Jörg Schacht[1], Anett Spring[1],
Andreas Werner[1], Manfred Zilker[2]

[1] *Max-Planck-Institut für Plasmaphysik, Wendelsteinstraße 1, D-17491 Greifswald*
*e-mail:* kuehner@ipp.mpg.de, *phone: +49 3834 88 2511, fax: +49 3834 88 2509*
[2] *Max-Planck-Institut für Plasmaphysik, Boltzmannstraße 2, D-85748 Garching*

For a complex experiment like W7X being subject to changes all along its projected life-time the advantages of a formalized software development method have already been stated (1). Quality standards like ISO/IEC-12207 provide a guideline for structuring of development work and improving process and product quality. A considerable number of tools has emerged supporting and automating parts of development work.

On W7X progress has been made during the last years in exploiting the benefit of automation and management during software development:

- Continuous build, integration and automated test of software artefacts
    - Syntax checks and code quality metrics
    - Documentation generation
    - Feedback for developers by temporal statistics
- Versioned repository for build products (libraries, executables)
- Separate snapshot and release repositories and automatic deployment
- Semi-Automatic provisioning of applications
- Feedback from testers and feature requests by ticket system

This toolset is working efficiently and allows the team to concentrate on development. The activity there is presently focused on increasing the quality of the existing software to become a dependable product. Testing of single functions and qualities must be simplified. So a restructuring is underway which relies more on small, individually testable components with standardized interfaces providing the capability to construct arbitrary function aggregates for dedicated tests of quality attributes as availability, reliability, performance.

A further activity is on improving the development cycle. The use of release cycles has already provided favourable concentration of work and predictability of delivery times. However, the demand has risen, to react quickly on priority changes from W7X-project management. So a more agile development cycle is being prepared relying on smaller working packages, shorter release cycles and an associated release plan giving the software development responsible the possibility to react on a shorter time scale.

Keywords: W7-X; Software Development; Quality Management; Standardization; ISO/IEC 15504;

## 1. Introduction

The stellarator W7-X (2) is a large, complex and long living (~20 years) nuclear fusion experiment. Several 'generations' of scientists will do research on W7X and need support of their work by correspondingly long living software. The same applies to engineers and technicians being responsible for the operation and maintenance of the machine.

Industrial development projects of comparable size usually rely on quality standards to achieve high product quality by applying high development process quality. The expected benefit is a solution tailored to the customer's requirements as well as the minimization of effort and risk for the provider (3).

In fact quality management is practised in magnetic fusion research but astonishingly there has been hardly any activity up to now to standardize processes, domain models, terminology and consequently the architecture of CoDa-software (*Control and Data acquisition*) from a general viewpoint. There are many individual developments for the same problem comprising, too, applications for experiment preparation and data analysis and visualization.

### 1.1. *W7X-context*:

The general quality standard for W7X machine construction is based on ISO9001 (4). For CoDa-software development the standard ISO/IEC-15504 (5) has been adopted as a basis. Part 5 of this standard provides an exemplar development process model requiring a minimum set of activities to be 'performed' in order to achieve quality level 1 on a 0-5 scale.

| | |
|---|---|
| ENG | Engineering processes |
| SPL | Supply processes |
| OPE | Operation processes |
| MAN | Management processes |
| PIM | Process improvement |
| REU | Reuse processes |
| SUP | Support processes |

**Table 1: P**rocess groups of ISO/IEC 15504-5 (selected)

Groups of processes most relevant for the current development are listed in Table 1. The further discussion in this paper is based on the terminology of this standard and process improvement (PIM) is the general subject. Chapter 2 addresses introduction of processes for requirements analysis and architectural design in the engineering group. Chapter 3 motivates improvement and automation within several process groups. Chapter 4 introduces the adopted management process. Chapter 5 describes the current development cycle highlighting the relevant processes according to table 1. A sketch of the main experiences and some conclusions for fusion research are given in the closing chapters.

The product quality of W7X CoDa-software is specified on the basis of the ISO/IEC 25000 (6) family of standards (See Table 2 for an overview).

| |
|---|
| Functional suitability |
| Performance efficiency |
| Compatibility |
| Usability |
| Reliability |
| Security |
| Maintainability |
| Portability |

**Table 2:** Product quality model of ISO/IEC 25010

A significant part of software for W7X has been developed in the Java programming language (data acquisition, applications for

configuration, experiment preparation and data browsing). Therefore the quality standards have been introduced in this field as a start. First steps were taken mainly for the engineering processes (software construction, integration, testing) to standardize and automate the software build process (1).
Increasing demand of responsivity against users, quality of system tests, testability in general and traceability of architectural and detailed design has governed the recent development and gave rise to serious enhancements of standardization and automation in the engineering as well as the other processes.

## 2. Top-down development

Quite some effort has been made to pursuit a top-down development process appropriate for large projects to derive dependable requirements from a magnetic fusion specific domain model (7) and to develop an appropriate software architecture which fulfils quality requirements sufficiently and minimizes the overall development and maintenance effort. This work is still developing and is put into practice for cases with high priority.

Requirements are collected in a management system. For sub-projects simple requirements lists are generated. Two prominent recent examples concern infrastructure projects which usually have single (simple) functional requirements:

- Communicate sensor data via an appropriate network to storage.
- Store sensor data.

The corresponding quality attributes (performance efficiency, reliability, maintainability etc.) were specified making systematic use of the ISO:25000 standard.

From the requirements an appropriate software subsystem architecture is developed. For architecture documentation on W7X the *arc42* (8) documentation template has been adopted which covers the necessary textual and model specifications. An *arc42* document can also be generated from UML (9) models, where the layout can be adapted to the W7X documentation standard. This procedure has been used successfully for (still internal) documentation of:

- A unified interface of data analysis services (during experiment and off-line; ->REU).
- An abstract communication model for experiment control driven by resource availability.
- A reviewed version of the CoDaStation (Control and Data acquisition station) to improve testing capabilities of system elements and arbitrary aggregates and to provide independent tests of quality attributes like stability, latencies, etc.

## 3. Bottom-up development

Much of the software development so far has been done in the bottom-up manner typical for a research environment, an activity which was driven mainly by the evolving prototype- and lab-systems. However, these initial designs and implementations represent solutions appropriate for small experiments and shot based operation.
In order to fulfil the quality attributes required for a large fusion experiment on the long term the architecture must be improved in order to reduce component sizes, dependencies and complexity, simplify interfaces, increase scalability, etc.
The existing CoDa-software is appropriate for use in the commissioning phase of W7X starting in 2014 which has increased the demand on its reliability enormously so that presently the focus in this area is on consolidating of the existing.
Basic engineering and supporting workflows had been introduced as a start, e.g. using SVN (10) for source code versioning, using *Ant* (11) for build standardization and employing *Hudson* (12) for continuous integration. Work had begun to provide separate environments for development (“snapshot”) and release builds, to provide stable operation conditions for

users in the labs. This has worked successfully for some time.
However, demands increased on

- Reliability of software for users.
- Maintainability of software for the developers.
- Reduction of response times after bug reports.
- Reduction of response times due to changing priorities in the W7X project plan.
- Management capabilities of the development process itself, especially planning and measurement of working units.

As a consequence the engineering, support, management, operation and supply processes had to be optimized, and even more tool support and a higher degree of automation was required to increase productivity and product quality.

## 4. Project management (MAN)

CoDa-software project management so far had long release cycles and lacked sufficient control and communication with the W7X project management. To improve this situation an 'Agile' (13) project management process has been agreed which is largely inspired by the 'Scrum' (14) method and provides the following benefits:
- risk reduction by short feedback cycles;
- quick reaction on changing priorities;
- product owner represents user's needs;
- product owner manages product configuration (i.e. prioritizes and filters requests for iterations and product release);
- iteration progress visible for W7X management;
- team: daily progress discernible, daily feedback, concentrated work.

The *Redmine* (15) multi project management system which is now in use for software development on W7X fortunately provides support for agile development:

– For each product a backlog (requirements for a milestone in the project plan) is available.
– Iteration backlogs (requirements for a single iteration).
– work breakdown of requirements into single tasks and a time estimation for each task.

About 2000 Tickets (bugs, features, tasks) have been processed so far.

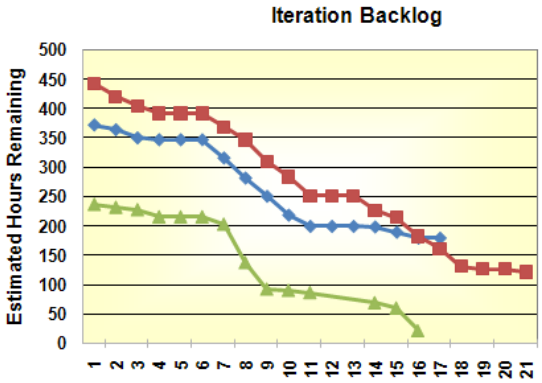


**Fig. 1**: "burn down chart" showing the remaining working hours for three iterations each of three weeks length.

The iteration cycle has been set to three weeks. Fig. 1 displays the 'burn down chart' for the first three iterations showing up the stepwise resolution of the planned tasks.
Each iteration is to produce a releasable product (Fig. 2) which has a well defined configuration represented by the backlog, and which is subject to a quality assurance process. At iteration start each requirement has to specify an acceptance criterion for successful completion. At iteration end all tasks are checked for completion and the requirements' acceptance criterion are tested and closed on success.
Actual releases to the end-users are agreed per iteration and are communicated as milestones to the W7X project plan.

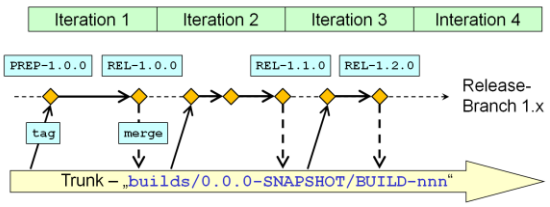


**Fig. 2:** Release plan via sequence of iteration backlogs.

## 5. Development cycle

Starting from the ISO/IEC 15504-5 example process an efficient development cycle

was introduced to achieve rapid response times in case of bug reports or feature requests. Such a process is required independently of the development method (top-down or else) and provides a narrow coupling between users and developers. The following topics are marked with the corresponding process identifiers of ISO/IEC 15504-5 (where possible).

*5.1. OPE.2 support*

User feedback is given via the ticket system of the W7X computer maintenance group. On a daily basis the developer team looks up tickets categorized as "CoDa" and transfers them to the *Redmine* development ticket system. At present more than 50 tickets have been collected this way.

*5.2. SUP.9 problem resolution*

Most bug reports, however, come from testing by the developers themselves. The bug relevance, the assignment to the proper development (sub-)project and the solution strategy are discussed by the team members.

*5.3. SUP.10 change requests*

The product owner alone has the privilege to transform a bug or feature request into a change request by including it in the product backlog. Bug fixes are agreed during the daily meetings, feature requests are agreed during the preparation of an iteration (see Fig. 3). Both become tasks to be resolved during the following development (construction) step. The sequence of iterations containing sets of requirements represents the release plan for a product (Fig. 2).

*5.4. ENG.6 software construction*

Java development is presently done purely with the eclipse IDE. Detailed software design is mostly a code centred process. The design documentation in UML is done by reverse engineering of existing code which is a standard capability of present day UML modelling tools.

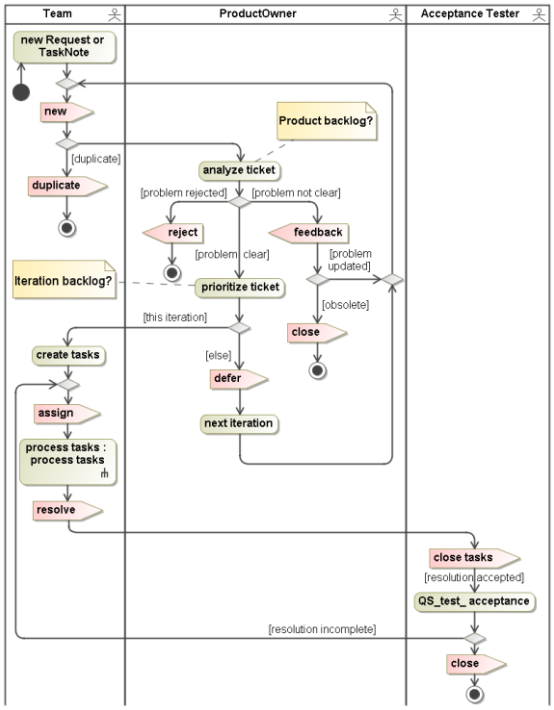


**Fig. 3:** Problem ticket resolution process

Local developer builds use the standard eclipse build (*Ant* based) as well as a *Maven* (16) build supported by the Eclipse-M2-plugin (17) (see below).

Several tools are used to ensure code quality and can be applied within eclipse:

- *Checkstyle* (18) checks *javadoc* (20) statements, coding style and appropriateness of special programming constructs.
- *PMD* (19) checks the java byte-code structure.
- *Findbugs* (21) checks explicitly for dangerous java constructs.

Since code quality checks have been introduced lately the number of warnings is generally high. Fortunately all tools are configurable so that a smooth improvement process can be defined to reduce the warnings in several steps that can be arranged appropriately into the iteration plan.

Unit tests are performed using *junit* (22), test coverage is done by *Cobertura* (23). Improvement of test coverage and traceability of tests to requirements needs still some work.

*5.5. ENG.7 software integration*

It focuses build standardization, continuous (automatic) builds and build artefact management.

The first approach had used the *Ant* build system and the continuous integration system *Hudson* for build automation and report generation. The generated *jar*-files (Java archive) were deployed into a single directory of the file system.
A radically new approach was pursued to reduce the drawbacks of this process.

*5.6. Build standardization:*

The *Ant* build system was replaced by the *Maven* system which uses a process model configured in a "`pom.xml`" file and provides many functions (like *javadoc*, *junit*, etc.) as a standard. The main task when setting up a new build project is management of dependencies to existing artefacts (jar-files). *Maven* stores build artefacts in a repository which is used, too, as intermediate storage for required open source software artefacts that are available via a worldwide central repository.
Additional functions like code quality checks, code metrics, statistics, report output can easily be introduced as plug-ins and made available as a standard '`master-pom`.xml' for all active development projects.
It required quite some effort to introduce the *Maven* system. A big refactoring effort was necessary to resolve cyclic dependencies between java sub-projects as name spaces were distributed over several jar-files. Some effort was necessary to provide a standard project layout for the 23 java projects existing on W7X. Some effort is still necessary to make the structure of name spaces and java projects and artefacts compliant to the OSGi (24) conventions. Presently the *Maven* builds are easy administrable, the plug-in documentation on the internet is sufficient for proper configuration.

*5.7. Continuous builds:*

Automatic integration builds (snapshots) are performed by the *Hudson* system. The configuration of builds is rather simple as the *Hudson* system exploits the contents of the *Maven* '`pom.xml`' file. Thus even the dependencies are reused to control the rebuild of java projects depending on a newly built sub-project. Thus the snapshot build of software products is fully automatic when a change has been checked in into the SVN versioning system.
The output of all plug-in functions (*junit*, code metrics, etc.) is condensed in a standard web-page allowing to navigate quickly to all kinds of reports, corresponding warning messages and source code regions. History graphs allow to follow the trend of error messages, number of tests, etc.
The *Hudson* system provides, too, a possibility for release builds which have to be started by hand as a release number has to be specified. This number is used to tag code versions in SVN and produced artefacts. *Maven* release builds require all subproject dependencies to be in a released (i.e. stable) state. The release process is functional since some time and has become a routine activity. The separate release and deployment process has provided a stable operation basis for users in the labs.
Release building applies not only to code but also to databases schemas. This has not been taken sufficiently into account so far. Thus going back to an older release comprising an earlier schema version is hardly feasible as it would require a doubling of the complete database system.

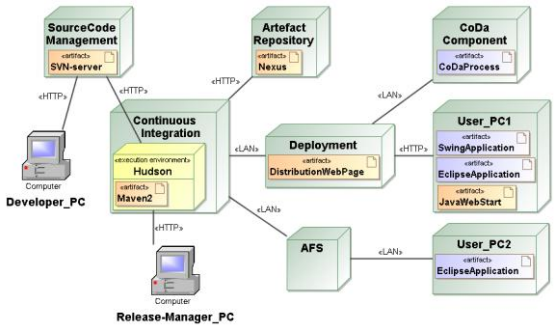


**Fig. 4:** The W7X build, deployment and provisioning system.

*5.8. Artefact management:*

For management of *Maven* build artefacts (jar-files) the repository system *Nexus* (25) has been introduced which is a proxy to the central one on the internet and thus provides independence of the availability of internet resources. Three repositories have been prepared to administrate snapshot, release and 3rd party products. The reposi-

tory structure is being changed to reflect the java-namespace and project structure.

*5.9. SPL.2 product release*

According to the W7X project plan stable software products are released to the general users at the WEGA (26) test-bed and in the labs.

*5.10. ENG.11 software installation*

Once the software is built it is stored on a deployment server from where it can be provided to the corresponding end users. There are three provisioning cases:

- The software on the data acquisition computers is updated by the *rsync* (27) mechanism.
- Java swing graphical applications (configuration and program editors) are installed and updated by the java web-start mechanism.
- Eclipse RCP (rich client platform) applications must still be installed manually as the deployment and update process are based on the Equinox-p2 (28) provisioning system which has not yet put into operation.

The first two cases cover the main activities necessary for experiment and lab-setup operations. The complete build, deployment and provisioning system infrastructure is shown in Fig. 4.

## 6. Experiences

The largest progress has been achieved in the build and deployment process. For users and developers there is a process without gaps from requirements to a usable product within less than two hours (process time without development time). All process steps are optimized such, that there is no need for cumbersome intermediate operations. The process is accepted and practised routinely. A significant part of time can now be used for stepwise improvement of code quality and construction of integration tests.

The agile management has made development discernible for developers and the W7X project management and increased the developers' awareness of the strategic goals. So this approach has shown its advantages and will be continued.

Work is continuing to increase the general software quality concerning detailed and architectural design, requirements traceability, testability, etc. This has become necessary as quality management has been introduced only in an advanced state of development.

## 7. Conclusion

For W7X software development the use of development standards has already shown enormous benefits. Adopting the quality level 1 ("performing") of ISO/IEC15504-5 provides the basis for detailed planning and prioritizing of software multi-projects in the context of a large system project (and subordinate multi-projects). Future experiments should start with quality management for software development from the very beginning.

Furthermore the fusion community needs a domain and process model capable of representing common sense domain knowledge as a basis for construction of appropriate software architectures. This is a necessary condition for acquiring the capability to delegate the development of software components to the industry (which is the usual procedure in machine construction).

## 1 References

1. Kühner, Georg. *Employing Industrial Standards in Software Engineering for W7X.* : Fusion Engineering and Design, 2009. pp. 1130-1135. Vol. 84.

2. F. Wagner, *The Wendelstein 7-X Project:* Europhys. News, 1995. Vol. 26, pp. 3-5.

3. For an introduction see e.g.: Sommerville, I. *Software Engineering.* Amsterdam : Addison-Wesley Longman, 2006.

4. ISO 9001:2008. [Online] http://www.iso.org/iso/home.htm.

5. ISO/IEC 15504 (SPICE). *(Part 5 has been taken from the older standard*

*ISO/IEC 12207).* [Online]
http://www.iso.org/iso/home.htm.
6. ISO/IEC 25000 (SQuaRE). [Online]
http://www.iso.org/iso/home.htm.
7. Evans, Eric, *Domain Driven Design* :
Addison-Wesley, 2004.
8. arc42. *Software architecture
documentation template.* [Online]
http://www.arc42.de/.
9. Unified Modelling Language. [Online]
http://www.omg.org/ , http://www.uml.org.
10. Subversion. [Online]
http://subversion.tigris.org/.
11. Ant. [Online] http://ant.apache.org/.
12. Hudson. [Online] http://hudson-ci.org/.
13. Agile Manifesto. [Online]
http://agilemanifesto.org.
14. Scrum. [Online]
http://en.wikipedia.org/wiki/Scrum.
15. Redmine. [Online]
http://www.redmine.org/.
16. Maven. [Online]
http://maven.apache.org/.
17. Eclipse-M2. [Online]
http://m2eclipse.sonatype.org/.
18. Checkstyle. [Online]
http://checkstyle.sourceforge.net/.
19. Javadoc. [Online]
http://java.sun.com/j2se/javadoc/.
20. PMD. [Online]
http://pmd.sourceforge.net/.
21. Findbugs. [Online]
http://findbugs.sourceforge.net/.
22. Junit. [Online] http://www.junit.org/.
23. Cobertura. [Online]
http://cobertura.sourceforge.net/.
24. OSGi Alliance. [Online]
http://www.osgi.org/.
25. Nexus. [Online]
http://nexus.sonatype.org/.
26. Schacht, J. and al. *Stellarator WEGA
as a test-bed for the WENDELSTEIN 7-X
control system concepts.* s.l. : Fusion
Engineering and Design, 83 (2008) 228-
235.
27. rsync. [Online]
http://de.wikipedia.org/wiki/Rsync.
28. Equinox-P2. [Online]
http://wiki.eclipse.org/Equinox/p2.