Employing Industrial Standards in Software Engineering for W7X

Georg Kühner[a,*], Torsten Bluhm[a], Peter Heimann[b], Christine Hennig[a], Hugo Kroiss[b], Alexander Krüger[c], Heike Laqua[a], Marc Lewerentz[a], Josef Maier[b], Heike Riemann[a], Jörg Schacht[a], Anett Spring[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*

[c] *University of Applied Sciences, Schwedenschanze 135, 18435 Stralsund, Germany*

[*] *Corresponding author; Tel.: +49-3834-88-2211; Fax: +49-3834-88-2509; E-mail address: kuehner@ipp.mpg.de*

Abstract

The stellarator W7-X is a large complex experiment designed for continuous operation and planned to be operated for about 20 years. Software support is highly demanded for experiment preparation, operation and data analysis which in turn induces serious non-functional requirements on the software quality like e.g.:

- high availability, stability, maintainability vs.
- high flexibility concerning change of functionality, technology, personnel
- high versatility concerning the scale of system size and performance

These challenges are best met by exploiting industrial experience in quality management and assurance (QM/QA), e.g. focusing on top-down development methods, developing an integral functional system model, using UML as a diagramming standard, building vertical prototypes, support for distributed development, etc., which have been used for W7X, however on an 'as necessary' basis. Proceeding in this manner gave significant results for control, data acquisition, corresponding database-structures and user applications over many years.

As soon as production systems started using the software in the labs or on a prototype the development activity demanded to be organized in a more rigorous process mainly to provide stable operation conditions. Thus a process improvement activity was started for stepwise introduction of quality assuring processes with tool support taking standards like CMMI, ISO-15504 (SPICE) as a guideline. Experiences obtained so far will be reported.

We conclude software engineering and quality assurance has to be an integral part of systems engineering right from the beginning of projects and be organized according to industrial standards to be prepared for the challenges of nuclear fusion research.

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

1 Introduction

High software quality means from the customer perspective that a software product fulfills his requirements to a sufficiently high degree. Translated to fusion research this means on the one hand that the software necessary for experiment planning, operation, simulation, analysis and publication supports the scientific staff significantly in its work as to 'increase productivity', on the other hand it satisfies scientific requirements for documentation and reproducibility.

As fusion experiments grow in size, complexity and lifetime and realistic reactor designs do show up, they become comparable with large industrial multi-technology projects and have to master serious non-functional requirements like:

- high availability, stability, maintainability
- high flexibility (change of functionality, technology, personnel)
- high versatility (scale of system size, performance)

Furthermore one has to take care of knowledge transfer to coming generations of scientists and engineers and finally to the commercial industry.

Besides a vast amount of special knowledge there is a generic part of domain knowledge comprising common sense about typical structures, functions, behaviours, workflows, etc. This consensus does definitely exist in brains but has astonishingly not been made explicit for a long time e.g. in the form of domain models (Fig. 1, and [1]) serving as a reference for terminology, as architectural driver for software developments or as a basis for standard software solutions hiding the influence of the above mentioned changes from its users.

This might be a consequence of the history of fusion research which has combined a large number of diverse technologies and research fields (especially plasma diagnostics) where the scientific success was strongly depending on highly specialized experience of few or single individuals developing especially diagnostics in a (cutting-edge!) technology driven bottom-up manner. Software support was added lately and developments took place often on an individual and short-term problem driven, incremental basis leading to additional diversity and parallel work.

For W7X software development was intended to keep up with up-to-date technology. Besides the challenging non-functional requirements a number of scientific goals increased the demand for higher quality of software:

- continuous experiment operation representing a change of paradigm.
- an integrated ansatz including reuse of physics models in the experimental context [2].
- a system of statistical evaluation for plasma diagnostic measurements to achieve uniform quality [3].

These requirements gain importance in reactor relevant fusion research, the strong dependence on models in the last two goals reflects a property specific to fusion plasmas.

2 Industry standards

Industrial standards for software engineering [4] were driven by quality requirements of public orders and were initially focused on product quality. More recent experience has shown that especially in case of serious non-functional requirements the quality of the development process becomes an essential prerequisite for product quality. Consequently standards like CMMI (US) [5] and ISO/IEC 15504 (Europe) [6] emerged which include process quality and process improvement activities as an integral part of quality management. These standards introduce metrics for assessment and prescribe ‘what’ has to be achieved to reach the proposed levels and indicate possibilities for improvement. The developing organization is free to chose the methods ‘how’ this is to be achieved. Several reference models have emerged (e.g. RUP [7], agile [8], etc.) offering comprehensive sets of 'best practices' which can be tailored to the situation and necessities of the corresponding software project and thus avoid the introduction of formal regulations ending in itself.

For W7X software development the existing QM handbook is being extended under the influence of ISO/IEC 15504. On this basis the software development process is being formalized and standardized in a stepwise procedure.

3 First Experiences

At the very beginning of W7X software development (1997) the intent was definitely to employ a top-down object oriented development method based on RUP. The start was indeed the construction of an integral abstract system model as a reference for design and implementation using the UML diagramming standard [9]. This model turns out to be a kind of domain model for fusion experiments which had not existed before explicitly [1]. Initially the context was restricted to **Co**ntrol[10] and **D**ata **a**cquisition[11] (“*CoDa”*) systems development (Fig. 2**Fig. 2**). This work was supported by introducing IDEs (*interactive development environments*) as well as commercial CASE-tools (*computer aided software engineering*) separately for control (C++) and data acquisition (Java). In order to synchronize the development in (partially) distributed teams the code versioning system CVS [12] was introduced. This turned out to be of large help for securing daily work, supporting design and implementation alternatives and having stable releases for the running prototype systems at any time.

4 Team Integration

As soon as a significant part of software developers for W7X was present at the Greifswald site an activity began to coordinate work and introduce quality standards. The main driving force was to achieve a means for knowledge transfer (future and community) by providing well documented code having passed standard tests and quality criteria.
The inclusion of the scientific software developers made obvious that distributed work with international collaborators is common and would benefit largely from well organized repositories. As a consequence the chance was taken for a modernization step migrating all CVS repositories to the more modern SVN [13] system. This is hosted on a powerful server machine and is worldwide accessible via an IPP-*kerberos* account.
It became obvious, however, that the interoperability of the in-house community was even more demanding so the focus moved to standardization of integration builds [14]. This could be set into practice by installation of a dedicated integration server using the *Hudson*[15] software which allows arbitrary build scripts to be triggered automatically (continuous integration) as well as interactively (Fig. 3). The tool’s user interface provides access to generated reports like code documentation, test results, audits, etc. and offers error and audit statistics in graphical form.
The largest progress in this respect could be achieved with *CoDa* software written in Java where a standardized build process (based on Ant[16]) is employed to control the invocation of style checks [17], integration tests [18] and coverage audits [19] before a software unit is released. Besides the test and audit reports the generated artefacts comprise standard *javadoc*[20] code documentation, the target *jar*-files (*java archive*) and on demand READMEs, *zipped* distribution packages, etc. Presently this procedure has been applied successfully to some selected software units and is being extended stepwise to the remaining (sub-)projects. In the C++ and FORTRAN world some proof of principle integration build procedures exist using GNU Autotools[21] build scripts and code documentation by Doxygen [22].

5 The *CoDa* Testing Cycle

As soon as system testing on prototype *CoDa* systems became prominent the communication between testers and developers had to be improved drastically. This could be achieved by means of the TRAC bug tracking system [23]. Within the course of two years about 400 tickets were created. A large portion concerned the basic *CoDa* software and specific bugs of prototypes occurring during system tests. Another large portion was due to functional tests of applications (editors, browsers). Here the system was also used to communicate requests for small scale enhancements of functionality.

6 Influence of Production Systems

As production *CoDa* systems exist already for prototype and laboratory installations and new systems are added continuously the demand for stable operation conditions has arisen. Especially major releases are significant as the physical data model of the employed database system may be affected. Second the interaction with real end users became prominent. Initially developed applications were mainly basic tools for experts and lacked operational security and user friendliness. Third the integration as part of the W7X system was unsatisfactorily. Software engineering was not embedded into systems engineering so that requirements elicitation was not complete.

These issues made it necessary to introduce more rigour on the development process, i.e. improve project embedding, standardize analysis, design, implementation, integration and release activities in a manageable manner. This was begun under the influence of the ISO/IEC 12207 standard (now in ISO/IEC 15504/4) which provides a list of items to be organized (Tab. 1).
As a start the existing status was described by means of the engineering categories ENG.6 *Software construction* and ENG.7 *Software integration* and by the supporting categories SUP.7 *Documentation* and SUP.8 *Configuration management*. Tool use, tool configurations and resources were described, standardized and templates for documents and reports were allocated. A configuration management-handbook has been prepared but has not yet been approved.
In order to stabilize the functionality of the software in productive use and gain capability for synchronization with the main project a release cycle has to be introduced keeping changes and enhancements in a manageable size. Thus the analysis of the integral system (*CoDa*, applications and physics models) has been resumed to prepare a comprehensive requirements document, derive the relevant use cases (working packages) and arrange them in a long term work, release and resource plan.

7 Introducing a process

A software development process is being established with focus on the items ENG.1-10 of the ISO/IEC 12207 engineering category which have the highest relevance in the short term.
ENG.1 Requirements elicitation for *CoDa* has been performed initially by documentation of experiences and by inquiries. Elicitation of detailed information is done by interviews which are communicated in the routine developer meetings and documented using the TRAC wiki or associated discussion groups [23].
ENG.2 System requirements analysis
Generally much more effort is spent on *requirements engineering* than in the past where requirements were often not made explicit as they were assumed to represent 'standard domain knowledge'. As client and contractor roles were often played by the same person or group requirements were biased (and expressed) by solutions and a creeping divergence between actual requirements and offered solution occurs. This can be avoided by rigorous separation of requirements and design documents. On the level of hardware subsystems integration this separation has already been successfully implemented.
For software development a requirements management and analysis tool is being employed where genuine requirements can be clearly separated from requirements induced by design decisions and the mutual dependencies can be made explicit. The general aim is to achieve a continuous chain of dependencies from requirements to design decisions and induced requirements down to implementation details (Tab. 2). This makes explicit an essential part of domain knowledge that can be used for impact analysis of requirements' changes and thus contribute substantially to long term project management.
ENG.3 System architectural design
The essential aspect of architecture, i.e.the assignment of required tasks to system elements, can be seen from table.2. Architecture design is rather complete, but no standardized documentation or even notation is used up to now.
A point of implicit knowledge is the border between systems and software engineering. For *CoDa* systems this is not a serious problem but it would be desirable to make design decisions explicit about what is to be solved in hardware and what in software.
ENG.4 Software requirements analysis
For *CoDa* software requirements are expressed by appropriate structural and behavioural models. For application development a use case driven approach makes communication with users easier as it allows to introduce continuously more workflow details by UML scenario specifications (Fig. 4). A comprehensive documentation of the users' view is underway; it has relevance as a reference model for future experiment software.
ENG.5 Software design models using UML have been used from the very beginning of W7X software development. Especially the discussion of design variants has benefited largely. The complete design history is still available but not under version control which has become available only recently with a more

modern CASE-tool. A lesson that had to be learned is that several levels of models are to be maintained in parallel reflecting abstraction levels and use of metamodels. Having recognized this, however, much of the corresponding work remains to be done.
Class diagrams were successfully used for design of data models and are even under version control. Behavioral models on the other hand were rarely used as development happened mostly code centered. It became important more recently since the complexity of the system had increased, application development became prominent and reference information for daily work was asked for.
ENG.6-8 Software construction, integration and testing have already reached some level of quality (chap. 3-5). A problematic point remains for application testing. Investigations are underway to employ a tool for GUI (graphical user interface) testing, mainly regression testing in order to avoid trivial traffic on the trouble ticket system.

ENG.9 System integration

A technical procedure has been described and established, but the process is neither standardized nor sufficient. This issue becomes relevant in cases where e.g. Java and C++ releases have to be synchronized due to their dependence on the data model.

ENG.10 System testing

A big success could be achieved here as an existing experiment (WEGA) became available as a testbed for a *CoDa* software vertical prototype [10].

8 Lessons learned

The largest progress has been achieved with ENG.6,7 *Software construction and integration*. Setting up sub-projects from zero takes still time, mainly due to complete code documentation and package descriptions and creation of integration test cases. Once this has been achieved development becomes much easier as changes are relatively small and easy to maintain.
The *design* artefacts (UML diagrams) are comprehensive but still insufficiently organized. Some subprojects have benefited largely during ‘hot design phases’ from the CASE-tool's teamwork server and its capability to administer design versions or alternatives. It became obvious that the software system’s architecture is still insufficient which led to the plan of a refactoring project within one year.
In a few cases requirements documents for sub-projects could be generated by a requirements management tool which showed up to provide a reasonable reference for development and validation.
Concerning the scientific goals: The challenges due to the continuous experiment operation requirement do clearly benefit from utilizing the described processes. Increased product quality means here solutions tailored to requirements, enhanced and available documentation, new concepts not known from existing solutions; increased process quality means formalization and automatization of the trivial, concentration on creative work, well defined collaboration interfaces.
The standardized integration of physics models and the use of validated model data for experiment control, however, is still in its infancy and few experience exists so far.

9 Conclusion

For W7X software development the increasing influence of non-functional requirements like availability (client demands), maintainability, flexibility, etc. can be met by the adoption of industry standards and practices. Already in the early planning phase the employment of industrial experience can be of large help which is easier than to rely solely on own experience or impose standards late in a running project. As large future experiments depend strongly on industrial or international collaboration contracting is easier on the basis of established experience and terminology than on ‘scientific way of its own’.
The work on W7X has shown that a large portion of domain knowledge is implicit (mainly about processes). In order to become future proof this knowledge has to be made explicit, i.e. it has to be put into domain specific models.
Recently the focus of proper embedding software engineering in systems engineering has gained importance for W7X. As fusion research is a ‘multi technology field’ an integrated view from the very begin-

ning is appropriate and allows for easier administration and maintenance structures in the operation phase. Work exploiting ISO15288 (system processes) and constructing SysML [24] models has begun.

10 Acknowledgements

Thanks go to the Open Source Community providing a large set of powerful tools which would be hardly accessible for scientific research if only commercially available.

11 References

[1] G. Kühner, P. Heimann, S. Heinzel, Ch. Hennig, H. Kühntopf, H.Kroiss, J.Maier, J. Reetz, M. Zilker, Editor for system configuration and experiment program specification, Fusion Eng. Design 71 (2004) 225-230
[2] Yu.A. Turkin, H. Maaßberg, C.D. Beidler, J. Geiger,N.B. Marushchenko, Predictive transport modelling for the W7X, Europhysics Conference Abstracts, vol. 28G, 2004, p. P1.198
[3] A. Dinklage, R. Fischer, J. Svensson, Topics and Methods for Data Validation by Means of Bayesian Probability Theory, Fusion Science and Technology, 46, 355,2004
[4] For an introduction see e.g.: I. Sommerville, Software Engineering, Addison-Wesley Longman, Amsterdam, 2006
[5] Capability Maturity Model Integration, Software Engineering Institute, Carnegie Mellon, US, http://www.sei.cmu.edu/cmmi
[6] ISU/IEC 15504 (SPICE), http://www.iso.org/iso/home.htm
[7] P. Kruchten, The Rational Unified Process: An Introduction, Addison-Wesley Longman (2004)
[8] Manifesto for Agile Software Development, http://agilemanifesto.org
[9] Unified Modelling Language, http://www.omg.org/, http://www.uml.org/
[10] J. Schacht, D. Assmus, T. Bluhm, A. Dinklage, S. Heinrich, C. Hennig, U. Herbst, R. König, H. Laqua, M. Lewerentz, I. Mueller, M. Otte, S. Pingel, J. Sachtleben A. Spring, A. Werner, A. Woelk, Stellarator WEGA as a test-bed for the WENDELSTEIN 7-X control system concepts, Fusion Eng. Design 83 (2008) 228-235
[11] P. Heimann, S. Heinzel, Ch. Hennig, H. Kühntopf, H.Kroiss, G. Kühner, J.Maier, J. Reetz, M. Zilker, Status report on the development of the data acquisition system of Wendelstein7-X, Fusion Eng. Design 71 (2004) 219-224
[12] Concurrent Versionss System, http://www.nongnu.org/cvs/
[13] http://subversion.tigris.org/cvs/
[14] A. Kus, A. Krüger, A. Dinklage, T. Bluhm, C.G. Hanke, G. Kühner, A. Werner, C. Hennig, J. Geiger, M. Lewerentz, Y. Turkin, Unified software repositories for Wendelstein 7-X: Workflow elements for fusion software development, Fusion Eng. Design 83 (2008) 410-412
[15] https://hudson.dev.java.net/
[16] http://ant.apache.org/
[17] http://checkstyle.sourceforge.net/
[18] http://www.junit.org/
[19] http://cobertura.sourceforge.net/index.html
[20] http://java.sun.com/j2se/javadoc/
[21] http://www.gnu.org/software/autoconf/
[22] http://www.stack.nl/~dimitri/doxygen/
[23] http://trac.edgewall.org
[24] Systems Modelling Language, http://www.omgsysml.org/

12 Figures

Fig. 1: First approach for a business model describing main activities of typical nuclear fusion experiments.

Fig. 2: UML context diagram for the control and data aqcuisition (*CoDa*) system

Fig. 3: W7X code repository and software integration server configuration.

Fig. 4: UML activity diagram describing the principal physics workflow (expresses the basic idea behind Fig. 1)

13 Tables

Table 1: ISO 12207 engineering category items

Table 2: Use case to application tracing. The use case ‘edit physics scenario’ has no solution yet.