ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOMPR004

CONTROL AND ANALYSIS SOFTWARE DEVELOPMENT AT THE EUROPEAN XFEL

S. Brockhauser†[1], M. Beg, M. Bergemann, V. Bondar, C. Carinan, R. Costa, F. Dall'Antonia, C. Danilevski, W. Ehsan, S.G. Esenov, R. Fabbri, G. Flucke, D. Fulla Marsa, G. Giovanetti, D. Goeries, S. Hauf, D.G. Hickin, T. Jarosiewicz, E. Kamil, Y. Kirienko, A. Klimovskaia, T.A. Kluyver, D. Mamchyk, T. Michelat, I. Mohacsi, A. Parenti, R. Rosca, D.B. Rück, R. Schaffer, A. Silenzi, M. Spirzewski, S. Trojanowski, C. Youngman, J. Zhu, H. Santos, H. Fangohr[2]
European XFEL, Schenefeld, Germany

[1]also at University of Szeged, Szeged, Hungary, and Biological Research Center, Szeged, Hungary
[2]also at University of Southampton, Southampton, U.K.

Abstract

Agile Project Management (Agile PM), coupled with the DevOps concept, has been worked out as a fundamental approach in a highly uncertain and unpredictable environment to achieve mature software development and to efficiently support concurrent operation [1]. At the European XFEL[2], Agile PM and DevOps have been applied to provide adaptability and efficiency in the development and operation of its control system: Karabo[3,4]. In this context, the Control and Analysis Software Group (CAS) has developed in-house a management platform composed of the following macro-artefacts: (1) Agile Process; (2) Release Planning; (3) Testing Infrastructure; (4) Roll-out and Deployment Strategy; (5) Automated tools for Monitoring Control Points (i.e. Configuration Items[5]) and; (6) Incident Management[6]. The software engineering management platform is also integrated with User Relationship Management to establish and maintain a proper feedback loop with our scientists who set up the requirements. This article aims to briefly describe the above points and show how agile project management has guided the software strategy, development and operation of the Karabo control system at the European XFEL.

INTRODUCTION

European XFEL is a new facility with six scientific instruments as experimental stations at the end of its three SASE beamlines[2] producing trains of very short X-ray pulses. While the electrons are accelerated along a 2.1 km long tunnel, the SASE photon beams are created and transported in an additional tunnel system with the total length of 3.6 km. The accelerator machine is operated by DESY[7] and is controlled by the DOOCS[8] control system which is also supporting the FLASH facility. In contrast, EuXFEL's photon transport and its experiments at the scientific instruments are controlled by a novel in-house developed control system Karabo[3,4].

Karabo version 2.1.5 has been released in March 2017 and has been used to start the commissioning of the first beamline SASE1. Although this version could already support basic demonstration setups properly, it was lacking a huge list of essential control system features and was still suffering from reliability and stability issues. Hence, control software framework development had to be continued parallel to providing 24/7 on-call support for the commissioning and later even user operation activities (see the applied priority pyramid on Fig. 1). Next to on-call support, the parallel commissioning of continuously upcoming instrumentation set an additional challenge because of the conflicting and frequently changing priorities which was heavily influencing the development roadmap.

Another challenge was that only a few developers knew the Karabo system and even less the code itself. Progress required the introduction of new personnel while keeping and spreading the know-how, as well as avoiding single-point of failures and technical debts.

With the support of our management, we have introduced an efficient group structure, and took appropriate management, software engineering and technology choices for addressing these issues and have successfully driven the facility through its commissioning phase and started its operation.

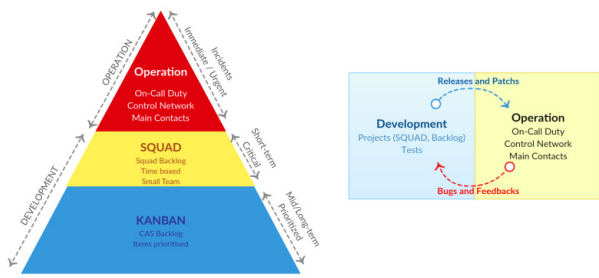

Figure 1: Priority of Operation vs. Development.

SOFTWARE ENGINEERING PLATFORM

The software engineering management platform has changed according to the different phases it had to support:

- *software framework development before commissioning*. In a short period before commissioning, we had the chance to implement major (and incompatible) changes in the framework which would have been very difficult during operation. In this limited period, we

† sandor.brockhauser@xfel.eu

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOMPR004

have focused on the implementation and consolidation of these fundamental steps.

- *essential development cycles with active deployment support during commissioning*. During commissioning, the highest priority became the operation, so the development had to follow it and focus on improving the reliability of the system.
- *feature developments with stable deployment supporting the operation*. Achieving the state when the deployed system became reliable and more stable, the focus could move from firefighting to a roadmap based feature implementation. The roadmap still has to be continuously adjusted to the actual requirements of the scheduled high impact experiments.

Below, we detail the artefacts as we have applied then in the last phase when moving from commissioning to operation.

Agile Management

To be able to respond to dynamically changing requirements and/or priorities, we have decided to follow agile principles [9]. In an agile environment, people are prepared for changes and are ready to refine the goals as they become relevant and gain priorities. But the efficient execution of such process requires maturity at company level. It is very important that software development team understands and appreciates the process to be followed, but it is equally important that the whole company also understands and buys-in this process and especially the interfaces of the developers with the scientists as customers, and with upper management who can change the priorities. For this purpose, the CAS Agile Process (as well as its modifications, see version 3 as Fig. 2 /note that better resolution figure is available on the corresponding poster/) has been compiled into a figure which has been announced and explained on different forums. This clarifies interfaces, but also defines roles and responsibilities:

- *CAS Contacts.* Main and deputy contact persons are assigned to each customer group (e.g. scientific instrument). While the main contact has the responsibility to keep the connection, collect requirements and suggest priorities for the backlog, with their deputies they also form a network which allows keeping the know-how spread and avoid isolated solutions.
- *Chapters.* To support the highest priority instrument groups, we could allocate dedicated people (e.g. main and deputy contacts and a data analysis scientist) who are closely working together with the representative of the instrument who plays the product owner (PO) role and sets the priorities for the backlog items. Chapters supporting instruments in the same phase have their Daily Standup Meetings together to ensure communication and help finding coherent solutions.
- *Squad.* For the implementation of a high priority strategic deliverable, we can form a group of developers with required skillset who can work in sprint(s) as a SCRUM [10] team. An external responsible is assigned as PO who follows the development and can make immediate decisions.
- *Task Forces.* For longer projects which people cannot be assigned exclusively for (like in case of a squad), a group of assigned people with relevant experience and interests (including external responsible) work together to align long term visions and build mid-term goals to be prioritized in the backlog.
- *Executive.* While the Group Leader is the overall responsible who reviews the priorities, it is the Project Manager who ensures that the processes are properly followed and resources are optimally allocated. Technical Coordination Team (TCT) and the Chief Data Analysis (DA) scientist are responsible for the design of control [11] and data analysis [12] tasks respectively.

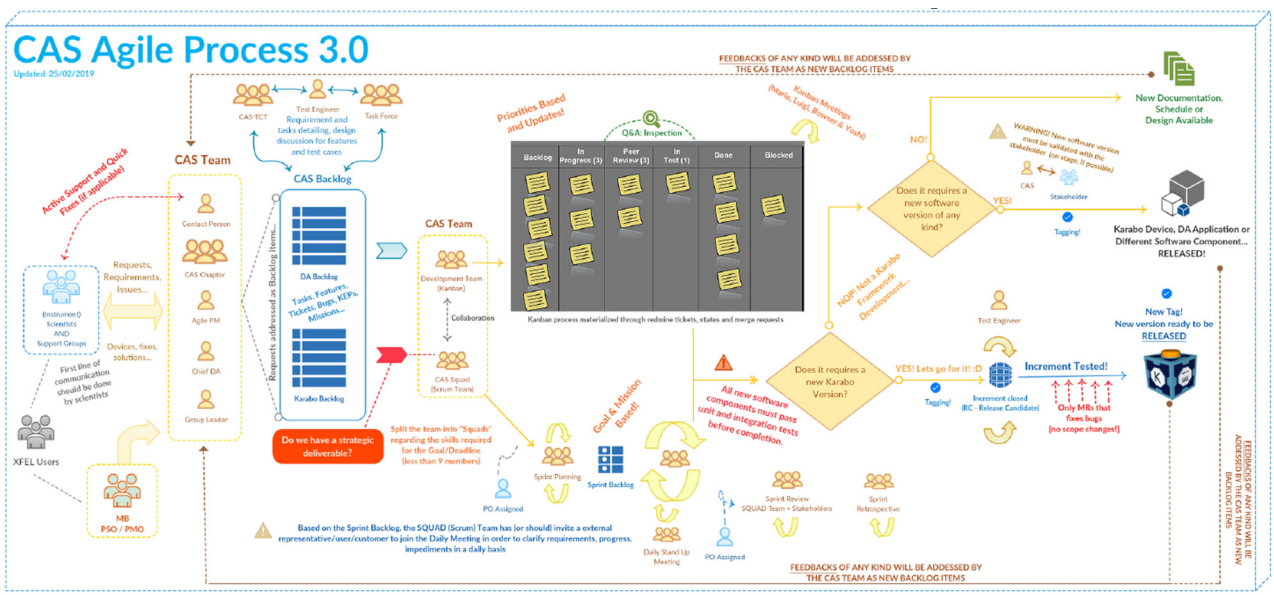

Figure 2: Agile Process v3.0.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOMPR004

- *Test Engineers*. All new software components pass unit and integration tests. Additionally, release candidates are checked against regression tests. Test engineers feed then back to the developers directly.

The majority of the development work is done along a Kanban board which features the prioritized Backlog; In progress; (code) Peer Review; In Test; Done; and Blocked columns. The Kanban process supported by Daily Standups (in group of 6-8 developers) is materialized through redmine [13] tickets, states, and gitlab [14] merge requests.

User Relationship Management

An important element of the agile process is the management of the relationship with our users and scientists. Not only they should use the defined interfaces, but they shall also engage with the process and take specific responsible roles, e.g. being the PO of a Squad, or Chapter. Most importantly, regular feedback is needed. Hence, the summary of the weekly chapter review meetings (which also sets the priorities) is collected and announced to the whole facility. This provides visibility and also transparency among all scientific instruments as our customers. While bi-weekly '*Did you know?*' snippets are broadcasted after each Karabo release, a demonstration meeting is also held where users' feedback and suggestions are collected and channeled to the relevant Task Force.

Roadmap Management

Task Forces provide mid-term goals for the TCT and Chief DA who are responsible to check the design, optimize the dependencies, and regularly update the roadmap. Features from the backlog are organized in a matrix to see the affected severity groups, like stability, performance, robustness, usability, etc. This is used to help the prioritization and the finalization of the scheduling on the roadmap. The roadmap prepared for 1-1.5 years in advance with the details of the releases planned for the coming year is announced. Note that changes can occur in the roadmap, e.g. as a consequence of changing priorities which is also announced.

Software Quality Assurance

We apply some basic *programming and naming principles* for code cleanness (e.g. camelCase [15], loose Hungarian notation [16], PEP8 [17]), *Unit tests* for functions and modules (e.g. scan) and require *integration tests* with different modules (e.g. devices) for each sw component. While we manually monitor our *test-coverage*, automatic *CI* [18] tests are performed on the commits to provide immediate feedback to the developers. A manual process, Gitlab's *code review* is also applied to ensure a second eye quality check before merging to the master branch. While *acceptance tests* (e.g. Karabo device implementation [19]) is done with the scientists requested the development, other manual type of tests (e.g. *stress* and *performance tests*) are performed occasionally.

Smoke and subsequent *regression tests* are performed on all release candidates. These regression tests include several test suits and are performed on all operating systems Karabo is used on at EuXFEL. While some of the test suits contains local tests only, there are distributed system test too, some of which also involves *hardware-in-the-loop tests*. Note that hw interaction can be tested in our Test Laboratory using different firmware setups of the controllers (e.g. widely used Beckhoff PLCs).

While our *GUI tests* are automated using the Squish tool [20], they are integrated to our fully automated regression test system implemented in robotframework [21]. All logs and results are automatically collected and sorted to facilitate the communication between the test engineers and the developers until the release gets accepted and tagged.

Roll-out, Deployment Management

The roll-out of a freshly tested Karabo release is announced in advance and is planned together with the instrument responsibles. Typically, a shutdown period is selected to provide scope for performing some final tests with special hardware, too (e.g. DAQ and special detectors), before its real use. The deployment itself is automated as Ansible [22] playbooks which also supports an easy redeployment of the system if needed. Shutdown and the subsequent startup period also allows a sequential deployment (first at the tunnel systems, and then at the instruments) which allows an easier identification of any occurring glitches.

Between two deployments, only hotfixes are supported (see Fig .3) which can contain bugfixes and/or address specific issues.

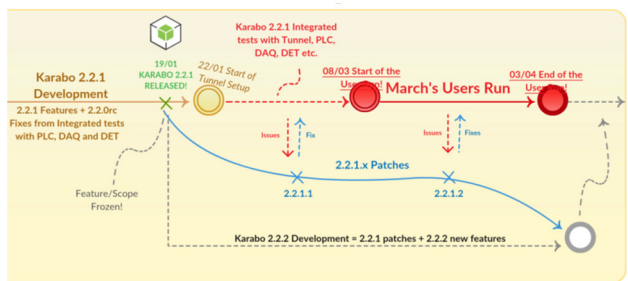

Figure 3: Karabo release and deployment cycle.

Monitoring and Incident Management

While the orchestrated deployment of specific versions of Karabo device servers on the control network is managed by Ansible, an option for on-the-fly manual development is also available. While Karabo can show in its dynamic topology viewer the available servers within a section of the control system, in a so called Karabo Topic, a global overview of which version of servers are active or when they were up last time is not provided on the full control system level inside Karabo. Instead, this functionality is implemented in an external tool, the OCD Manager which provides a web interface for managing incident calls. Hence an integrated environment is provided to register the incident and to have a first glimpse. The OCD Manager is also providing a link to the electronic logbook used at EuXFEL where technical details of the problems as

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-MOMPR004

well as the solutions applied are provided in a collaborative environment. Incidents are carefully evaluated on a weekly manner when people scheduled for on-call duty on the week before are handing over to the OCD team of the subsequent week. Note that OCD Manager can also provide statistics on how many incidents have occured (and how long the interventions took) on different components and/or instruments in a selected period. This tool is a great help in identifying problems and how much they cost which is an important input for prioritizing future developments.

CONCLUSIONS

In the last three years, the CAS group has managed an increase of its size from 12 to 28 while efficiently integrating the newcomers to the group structure by applying mentorship and a welcome structure with a training program. A good and motivating atmosphere has been achieved where the group members are happily volunteering to help one another or even take on-call support duties. During this period, a workable system was delivered in spite of a difficult task prioritization management environment. The Karabo control system has been made stable and reliable, and is ready to integrate newer and more sophisticated features. As the stability and robustness of the system has increased, the amount of required support by on-call has measurably decreased; the execution of experiments became more mature. By the increased performance, GUI responsiveness, and added tools, like the scan-tool, the usability of the system has been radically improved. After cumbersome early experiences at SASE1 (FXE and SPB) in 2017, the improved Karabo has allowed smooth starting up of the new instruments at the end of 2018 (SCS and SQS at SASE3) and early 2019 (MID and HED at SASE2).

This achievement is the result of the whole of CAS group which executed the described software engineering processes, but it would not have been possible without the support of management and the continuous devoted and sometimes determined assistance of our scientists.

REFERENCES

[1] Levitt, R. E. "Toward Project Management 2.0", *Engineering Project Organization Journal*, vol. 1, no. 3, pp. 197-210, 2011.

[2] M. Altarelli *et al.*, XFEL: The European X-ray Free Electron Laser technical design report, DESY XFEL Project Group, 2006.

[3] B B. C. Heisen *et al.*, "Karabo: An Integrated Software Framework Combining Control, Data Management, and Scientific Computing Tasks", in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper FRCOAAB02, pp. 1465-1468.

[4] S. Hauf *et al.*, *J. Synchrotron Radiat.,* vol. 26, pp. 1448–1461, 2019.

[5] J. Dugmore and S. Taylor, ITIL® V3 and ISO/IEC 20000, The Stationery Office, 2008, pp. 2-5.

[6] J. J. Cusick, and G. H. K Ma. "Creating an ITIL inspired Incident Management approach: Roots, response, and results." 2010 IEEE/IFIP Network Operations and Management Symposium Workshops. IEEE, 2010. doi:10.1109/nomsw.2010.5486589

[7] https://desy.de/.

[8] https://confluence.desy.de/display/FLASHUSER/User+overview

[9] https://agilemanifesto.org

[10] https://www.scrumguides.org/.

[11] G. Flucke *et al.*, "Status of the Karabo Control and Data Processing Framework", presented at the 17th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper WECPR03, this conference.

[12] H. Fangohr *et al.*, "Data Exploration and Analysis with Jupyter Notebooks", presented at the 17th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper TUCPR02, this conference.

[13] https://www.redmine.org/.

[14] https://gitlab.com/.

[15] https://en.wikipedia.org/wiki/Camel_case

[16] https://en.wikipedia.org/wiki/Hungarian_notation

[17] https://www.python.org/dev/peps/pep-0008/.

[18] G. Booch, *Object Oriented Design: With Applications* , Benjamin Cummings, 1991, p. 209. ISBN 9780805300918.

[19] V. Bondar *et al.*, "Beam Position Feedback System Supported by Karabo at European XFEL", presented at the 17th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'19), New York, NY, USA, Oct. 2019, paper MOPHA040, this conference.

[20] https://www.froglogic.com/squish/.

[21] https://robotframework.org/.

[22] https://github.com/ansible