Software Engineering Initiative of DLR – Supporting Small Development Teams in Science and Engineering

ESA Software Product Assurance Workshop 2017 (September 19-22, 2017)
ESA/ESOC, Darmstadt, Germany

Tobias Schlauch <Tobias.Schlauch@DLR.de>
German Aerospace Center (DLR),
Cologne / Braunschweig / Berlin
http://www.DLR.de/sc

German Aerospace Center (DLR)

Approx. 8000 employees across
33 institutes and facilities at 20 sites.

Offices in Brussels, Paris,
Tokyo and Washington.

The three pillars of DLR

- Space agency
- Project management agency
- Research institution

Observations concerning Software Development at DLR

- **About 20% of DLR employees from the research institutes** are involved in software development.
- Typical examples of developed software include **simulation and modeling, flight control, signal and data processing, knowledge and data management, visualization**, and others.
- Software maturity ranges from **small research software**, to **large long-term maintained scientific frameworks**, up to **product-like software**.
- **A variety of programming languages is used** including Python, R, Perl, C, C++, Fortran, IDL, Matlab, LabView, Ada, Java, and others.
- Typical development **team sizes range from one up to 20 persons.** But usually there is **one main developer** supported by **students**.
- Developers are mostly **domain scientists without specific background in software engineering**.

We started the Software Engineering Initiative at DLR because we want to improve the overall quality of developed software at DLR.

Software Engineering Initiative of DLR

Software Engineering Initiative of DLR

Software Engineering Initiative of DLR

- Network
- Guidelines
- Tools
- Trainings
- Knowledge and Experience Exchange

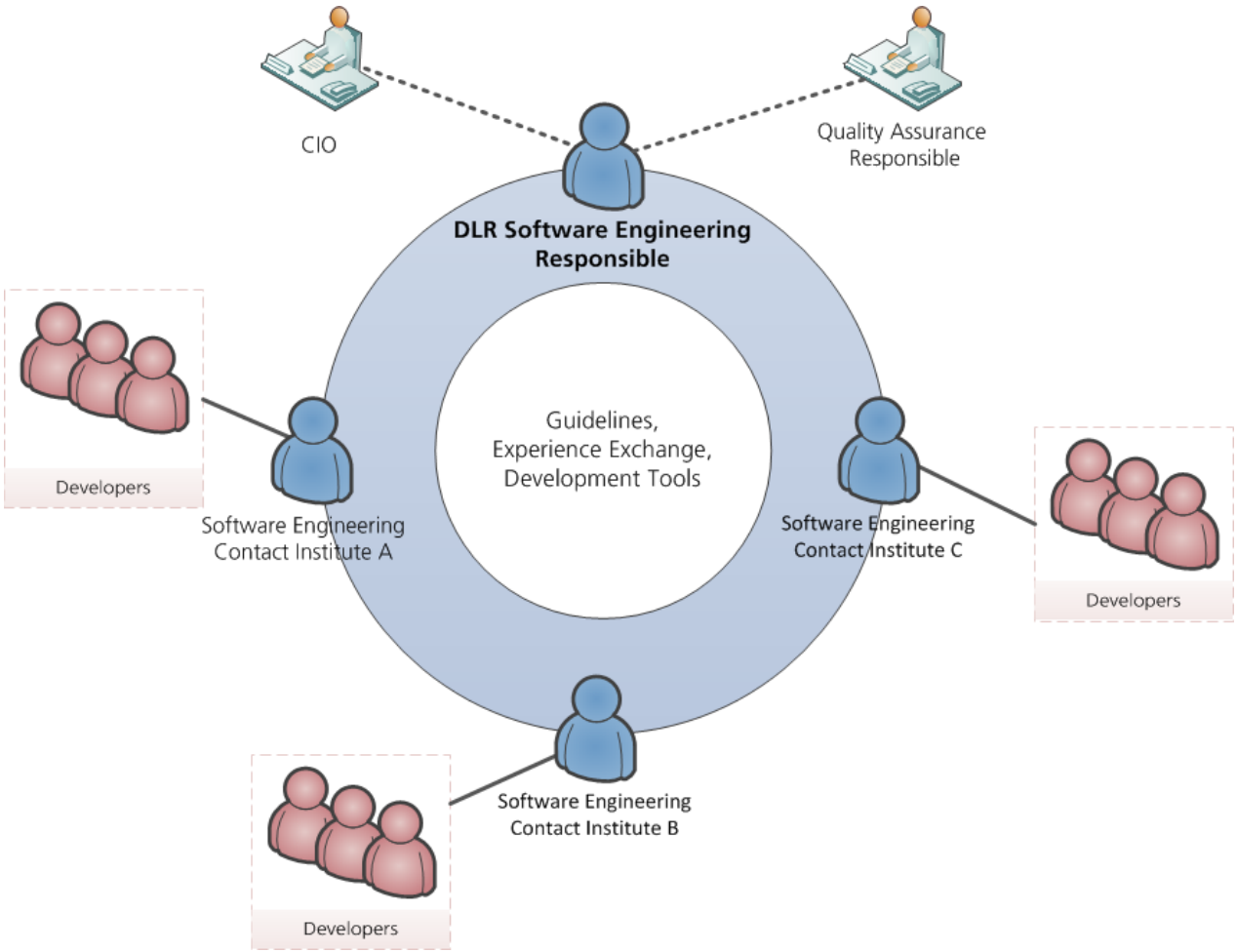
Software Engineering Network

Network consists of representatives from all DLR institutes concerned with software development.

Main responsibilities

- Development of DLR`s software engineering guidelines
- Improvement of central supporting activities such as development tools and trainings
- Supporting application of the guidelines within the institutes

Representatives further organize software engineering activities in their institutes.



Software Engineering Guidelines

Guidelines support developers to self-assess their software concerning good development practices.

- Joint development with focus on **good practices**, **tools**, and **essential documentation**
- **77 recommendations** give advice in different fields of software engineering:

Requirements Management

Software Architecture

Design & Implementation

Change Management

Software Testing

Release Management

Automation & Dependencies

Software Engineering Guidelines (cont.)

Guidelines are tailored into **three maturity level** to offer a **suitable initial set of recommendations** and are available as **checklists in different formats** (Markdown, Wiki, Word) to ease practical usage.

Checklists for different maturity levels

Change Management

Recommendation	Comment	Status
EÄM.2: The most important information describing how to contribute to development are stored in a central location. *(from application class 1)*	Build steps are missing	todo
EÄM.5: Known bugs, important unresolved tasks and ideas are at least noted in bullet point form and stored centrally. *(from application class 1)*		ok
EÄM.7: A repository is set up in a version control system. The repository is adequately structured and ideally contains all artifacts for building a usable software version and for testing it. *(from application class 1)*		ok
EÄM.8: Every change of the repository ideally serves a specific purpose, contains an understandable description and leaves the software in a consistent, working state. *(from application class 1)*		ok

Reasoning and further advice

The repository is the central entry point for development. All main artifacts are stored in a safe way and are available at a single location. Each change is comprehensible and can be traced back to the originator. In addition, the version control system ensures the consistency of all changes.

The repository directory structure should be aligned with established conventions. References are usually the version control system, the build tool (see the Automation and Dependency Management section) or the community of the used programming language or framework. Two examples:

Tools

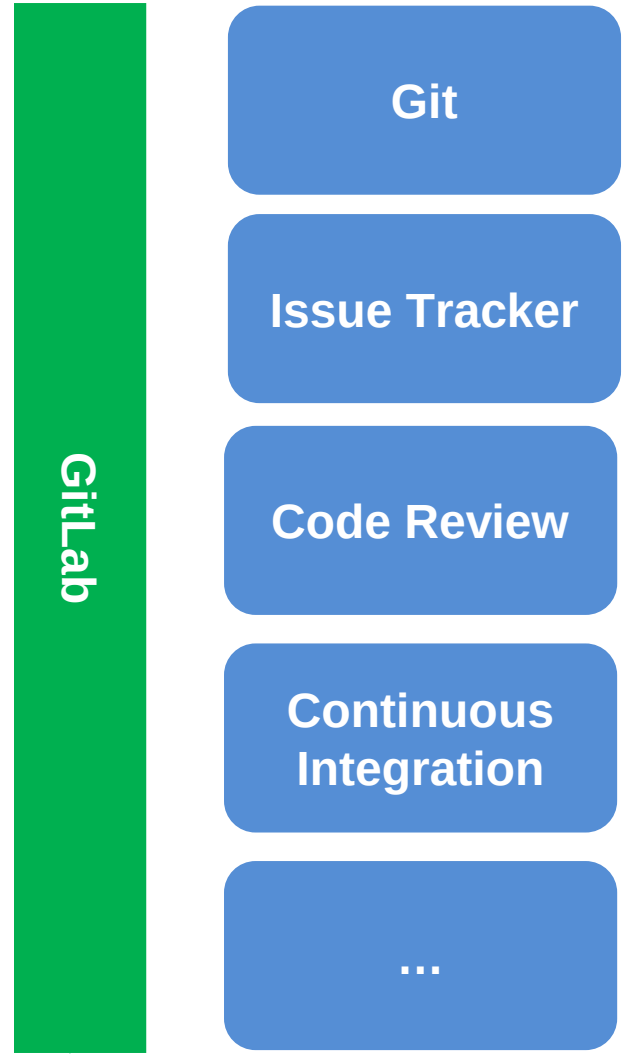
Essential development tools are centrally offered to support DLR developers.

Initial focus has been on software configuration management

- Version control system Subversion: 1.175 active projects
- Issue tracker Mantis: 246 active projects

Focus moved to collaboration and automation support

- Collaborative development supported by a software forge like GitLab
- Common infrastructure for automated testing and continuous integration

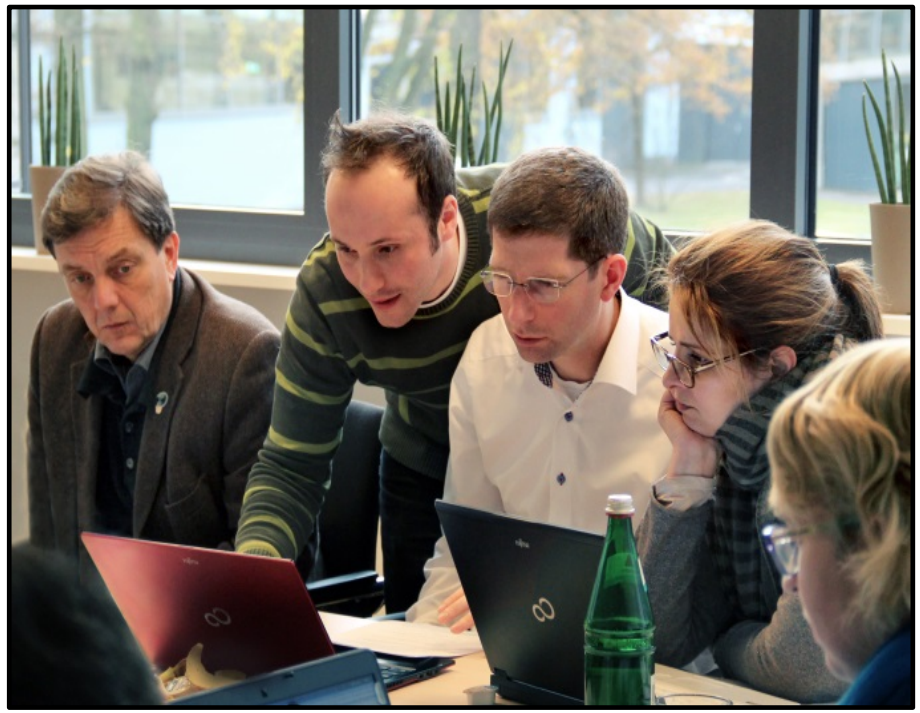


Trainings

Regular trainings are offered to provide hands-on experience in applying the guidelines and the DLR development tools.

Concept

- Intensive two-day course
- Small groups with up to 15 participants
- Hands-on experience on the basis of a complete example project
- Trainings are offered on a yearly basis at different DLR locations across Germany



Additional trainings are offered on request for specific topics such as unit testing, open source, and others.

Knowledge and Experience Exchange
SoftwareEngineering.Wiki

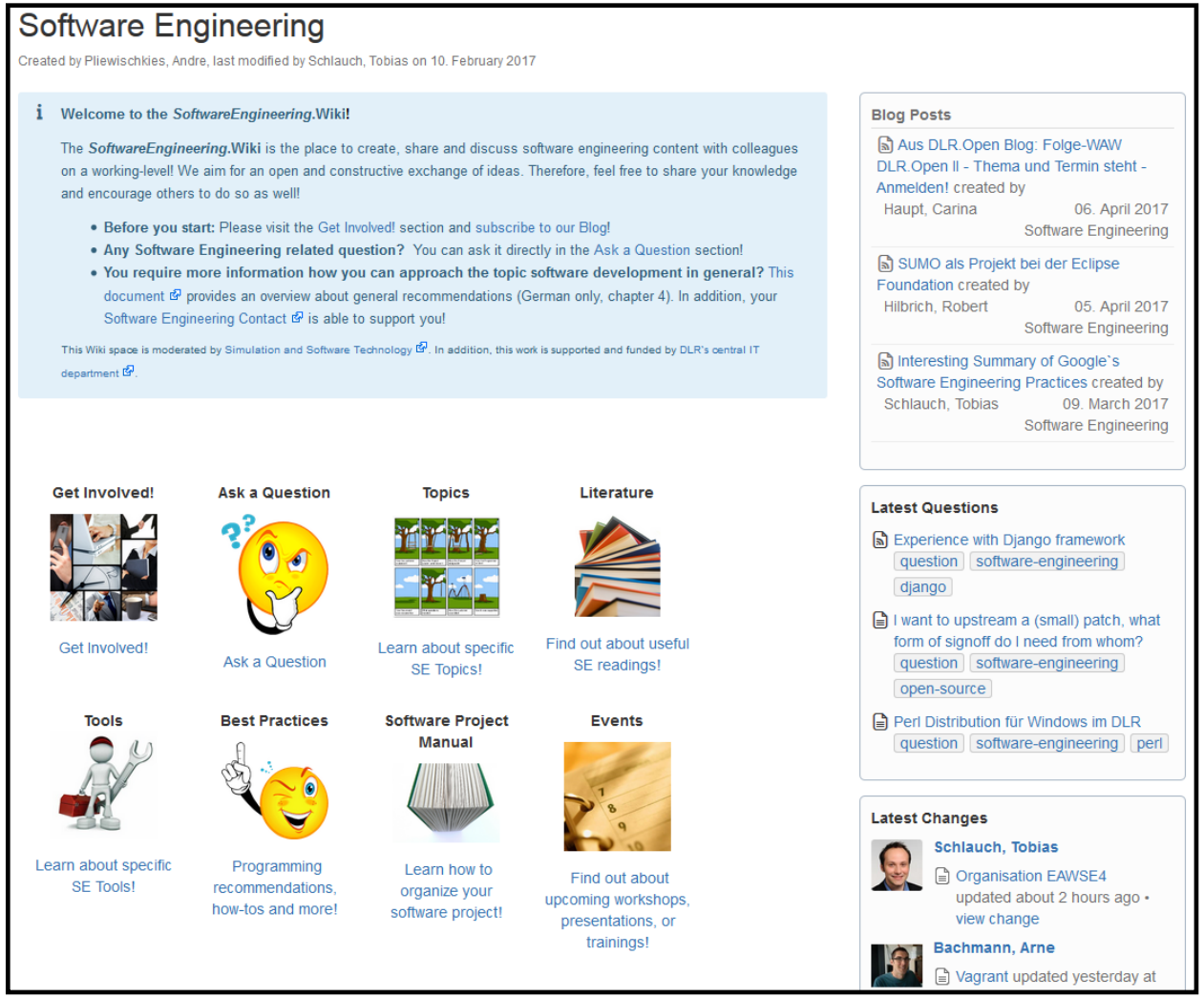
Internal Wiki space to share software engineering knowledge and experiences.

Concept

- Open to contributions of all DLR employees
- Moderation by a small central group

Main content categories

- News
- Questions & answers
- Information about topics like architecture, testing, etc.
- Experiences concerning development tools
- Official programming guides



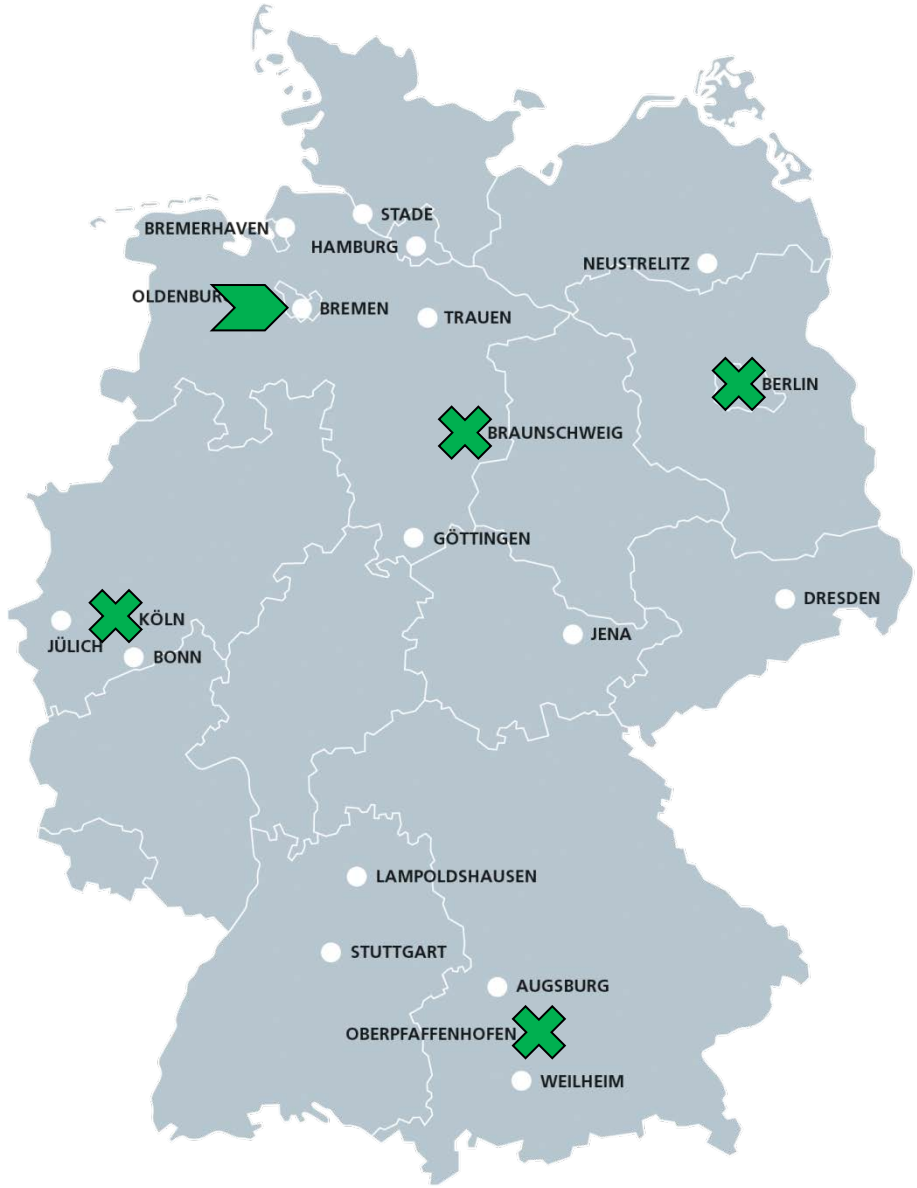
Knowledge and Experience Exchange
Workshops

Regular knowledge exchange workshops are held to actively involve DLR scientists and to foster exchange.

Concept

- Intensive 1.5-day workshop to provide *knowledge*, *experience exchange* and *networking opportunities*
- Dedicated main topic supported by keynotes of invited experts
- Active involvement of the participants through group work, experience reports, technical presentations, and lightning talks
- Results are shared via the *SoftwareEngineering.Wik*i

Since 2014, four knowledge exchange workshops have been organized at different locations across Germany. About 50 scientists participated in every workshop.



Summary and Outlook

Summary and Outlook

First steps have been taken to build a self-reliant software engineering community at DLR.

Key success factors

- Establishment of a vital software engineering core community
- Joint development of practical software development guidelines
- Active support of domain scientist and DLR institutes
- Raising management awareness and achieving management support

Next steps

- Provide further, community-driven solutions to ease implementation of guidelines
- Strengthen community (exchange, “inner source”)
- Addressing challenges of small teams in specific research domains

Summary and Outlook (cont.)
Challenges at the Example of the Space Domain

Common situation:

- Small teams are involved in space projects due to their expert knowledge
- Small teams struggle to comply with heavy-weight standards like ECSS

A solution could offer the ISO 29110 extended by a specific space profile:

- **Basic idea:** Start with an minimum set of processes and add additional processes in dependence of the actual project context
- ISO 29110 base profile requires a project management and a software implementation process
- Space profile could extend it by processes such as quality assurance, safety and dependability, etc.

DLR software engineering initiative and guidelines are complementary to ISO 29110. They focus on practical implementation of the base profile.