Software Engineering Guidelines

From Theory to Practice

Tobias Schlauch (@TobiasSchlauch)
Carina Haupt (@caha42)
Michael Meinel (@led02)

German Aerospace Center (DLR)

Simulation and Software Technology

Research Software Engineering Conference 2018

Software development at the German Aerospace Center (DLR)

Numbers

- More than 8000 employees
- ~20% of DLR employees involved in software development

Some Characteristics

- Variety of
  - Fields
  - Maturity
  - Software technologies
  - Team sizes
  - Backgrounds

We started a Software Engineering Initiative for DLR because we believe that our research results profit from better software!

DLRs central RSE group



Software Engineering Initiative for DLR

- Networking and Collaboration
- Guidelines and Tools
- Trainings and Consulting
- Knowledge and Experience Exchange

DLR Software Engineering Guidelines

Guidelines to help **research software developers to assess their software**

- Focus is on **good practice** and **documentation**
- Guidelines are available as **checklists in different formats**

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:

One size does not fit all!

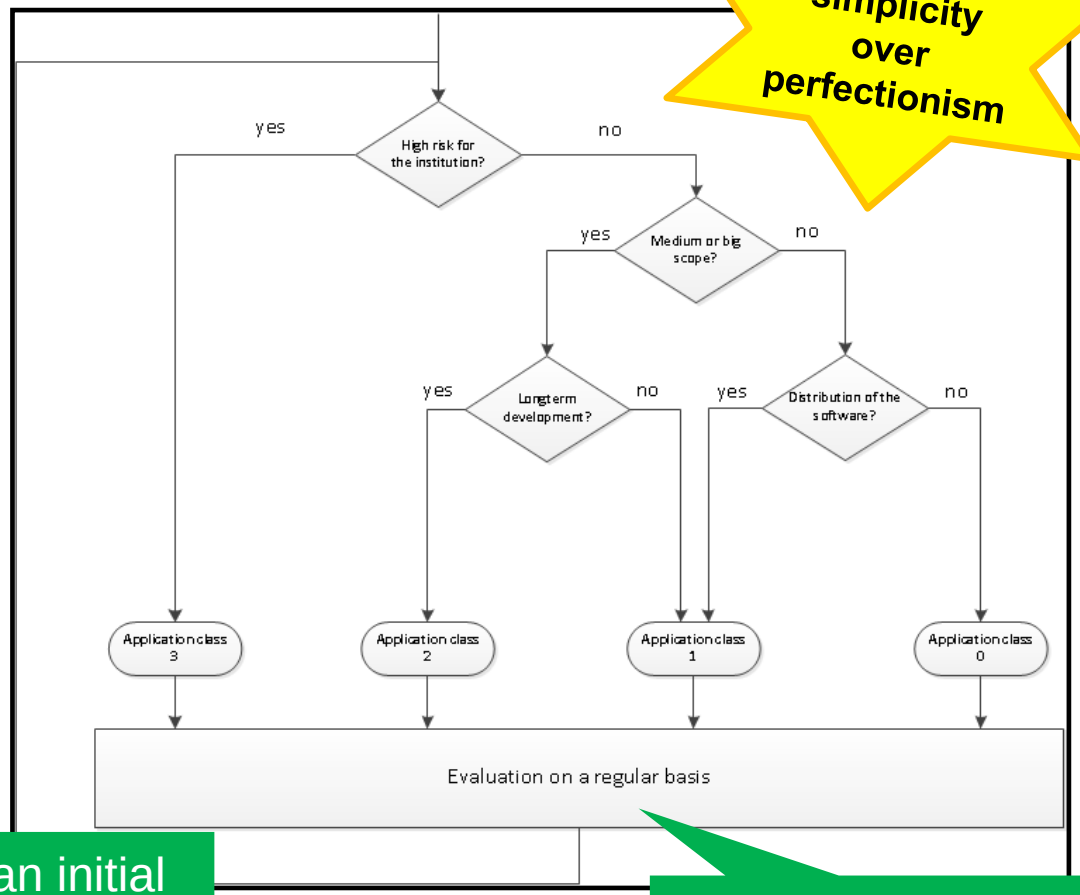
simplicity over perfectionism

Application class 1
- „small“, but other use it

Application class 2
- „medium – large“, other use it, long-term support

Application class 3
- „products“, critical for success of department or institute

Application class 0
- Personal „use“ (intentionally left blank)



An application class provides an initial starting point. Recommendations can be added and removed to fit the context.

Classification may change over time!

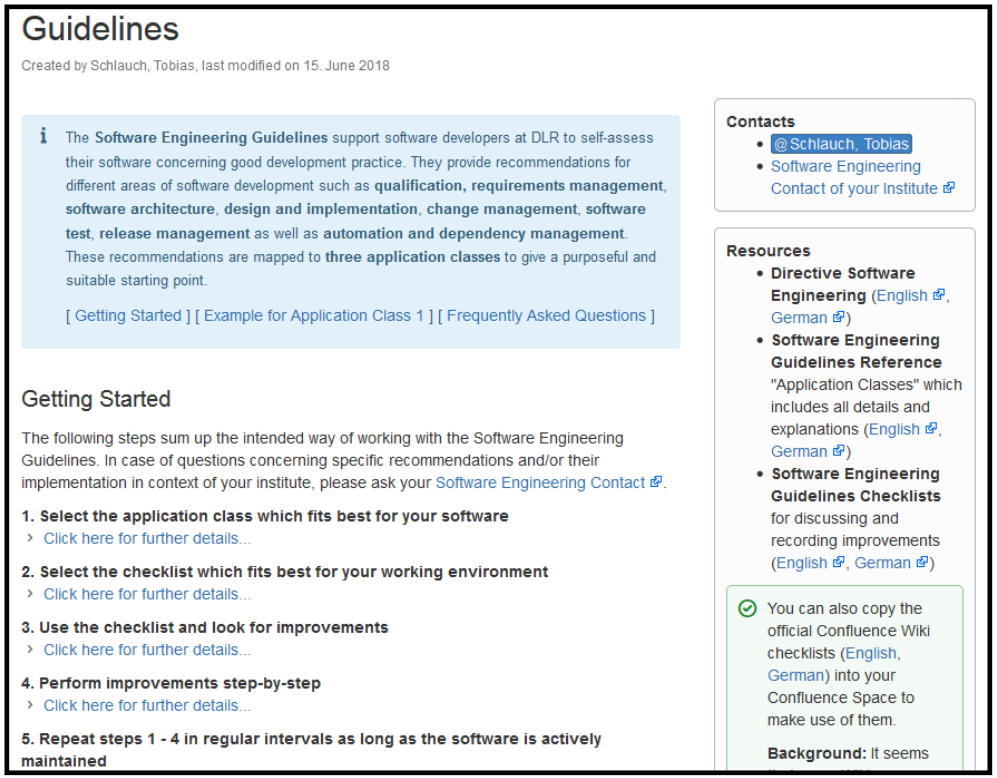
How do we use the guidelines?

Main use cases:

- Find out about the current status
- Identify improvements

Example situations:

- Find initial start point in new or legacy software projects
- Ongoing improvement
- Supporting hand-over
- Convince others
- Indicate applied practices to a third-party



Guidelines

Created by Schlauch, Tobias, last modified on 15. June 2018

The **Software Engineering Guidelines** support software developers at DLR to self-assess their software concerning good development practice. They provide recommendations for different areas of software development such as **qualification, requirements management**, **software architecture**, **design and implementation**, **change management**, **software test**, **release management** as well as **automation and dependency management**. These recommendations are mapped to **three application classes** to give a purposeful and suitable starting point.

[Getting Started] [Example for Application Class 1] [Frequently Asked Questions]

Getting Started

The following steps sum up the intended way of working with the Software Engineering Guidelines. In case of questions concerning specific recommendations and/or their implementation in context of your institute, please ask your Software Engineering Contact.

1. **Select the application class which fits best for your software**
   › Click here for further details...
2. **Select the checklist which fits best for your working environment**
   › Click here for further details...
3. **Use the checklist and look for improvements**
   › Click here for further details...
4. **Perform improvements step-by-step**
   › Click here for further details...
5. **Repeat steps 1 - 4 in regular intervals as long as the software is actively maintained**

Contacts
- @Schlauch, Tobias
- Software Engineering Contact of your Institute

Resources
- **Directive Software Engineering** (English, German)
- **Software Engineering Guidelines Reference** "Application Classes" which includes all details and explanations (English, German)
- **Software Engineering Guidelines Checklists** for discussing and recording improvements (English, German)

You can also copy the official Confluence Wiki checklists (English, German) into your Confluence Space to make use of them.

Background: It seems

Two recent examples…

Example 1: Support researchers improving a legacy Matlab code

Context:

- Matlab toolbox for image processing and analysis
- Legacy code base, one researcher + students ► close to class 1

Involvement of our RSE group:

- Make it “production-ready”, team development ► class 2
- Consulting: Set up processes and tools, training, no feature development
- Challenges: (legacy) Matlab, RSEs and developers at different sites

Example 1: Support researchers improving a legacy Matlab code (cont.)

Approach:

- Moving to GitLab
- Iterative process refinement
- Improving documentation and testing

Experiences:

- Checklists worked pretty well ► focus, discussions, status
- Remote consulting ► hard to assess “real” status
- Legacy code ► harder to make the right judgement

Example 2: New development of a metrics calculation tool for satellite performance

Context:

- Originally: New development of a metrics tool in Python 3
- But: Python 3 legacy code re-use required

Involvement of our RSE group:

- Develop a “production-ready” tool ► class 2/3
- Development and consulting: Feature development, set up processes and tools, supporting individual developers
- Challenges: (legacy) Python, scattered development team, many partners, hard time constraints

Example 2: New development of a metrics calculation tool for satellite performance (cont.)

Approach:

- Involve all partners
- Establish professional development process and environment early
- Iterative process refinement
- Refactor legacy Python code as needed

Experiences:

- Checklists worked pretty well ► status, discussion
- Unforeseen effort ► “hidden” dependencies, environment
- Missing priority hints ► harder to set focus

Lessons learnt

Guidelines help to find out about the status, to discover improvements as well as to focus activities and discussions but …

… are no beginners tool and some details require improvements:

- Better indicate priorities
- Make dependencies more transparent
- Direct links to (more) practical examples

Supportive environment is key!

- Community, team culture, mentors
- Tools and trainings

Next steps

Improving guidelines

- Collect and analysis further feedback
- Content adaptations
- Clearer priorities and dependencies
- Stronger focus on open research software

Improving culture and environment

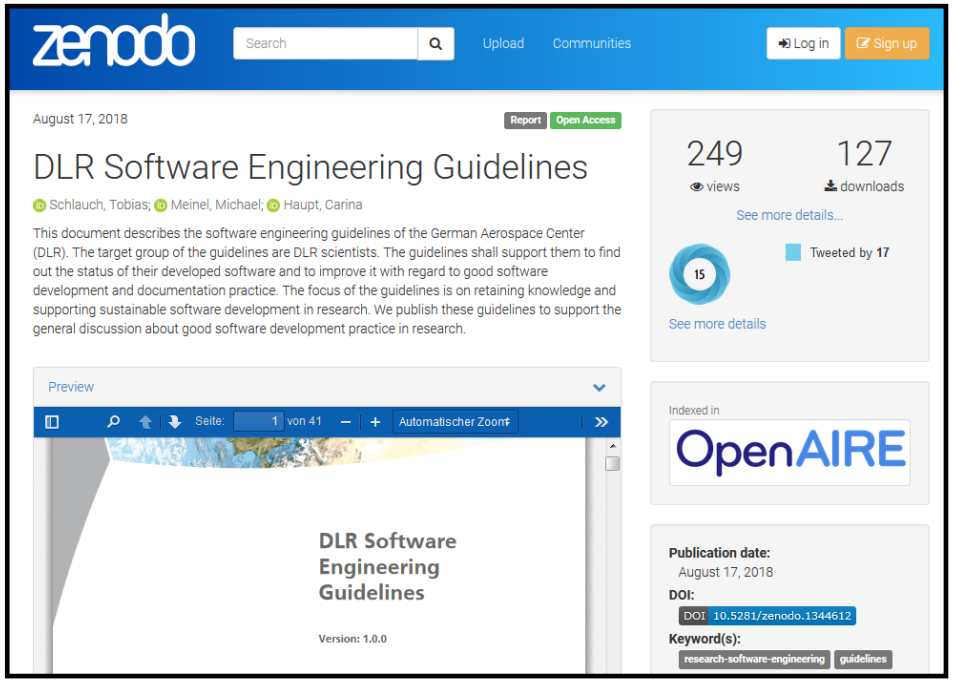
- Communities of practice
- Tooling
- Examples

Interested in sharing ideas about minimal practice for research software and making effects of best practice measurable? Please let us discuss!

Do you want to find out more?

- My RSE17 talk "Helping a friend out – Guidelines for better software"

- We published the reference guides:
  - German version: https://doi.org/10.5281/zenodo.1344608
  - English version: https://doi.org/10.5281/zenodo.1344612



Source: Zenodo, https://zenodo.org/record/1344612