The DLR Software Engineering Guidelines

An approach to support sustainable software development in research

Carina Haupt
Institute for Softwaretechnology
German Aerospace Center (DLR)

DLR
Some Facts around Software Development

Some numbers…

- More than 1500 employees develop software

Characteristics

- „Developer“ often do not have any training in software development
- Huge amount of software projects
- Variety of used software technologies

How to support scientists to develop sustainable software?

Aspects of modern software development

- Distributed development processes via git, subversion,…
- Community software (github, bitbucket,…)
- Issue tracking systems
- Open source licensing (BSD, MIT, (L)GPL,…)
- Software architecture
- Build systems (CMake, Autotools,…)
- Meta build systems (Spack, EasyBuild, Conda)
- Test frameworks (GoogleTest, PyTest, jUnit,…)
- Continuous integration testing (Jenkins, gitlab-ci,…)
- Integrated development environments (IDEs, e.g. Eclipse, QtCreator, MS Visual Studio)
- .etc.

Do I need all that???

The DLR Software Engineering Guidelines
Topics

https://rse.dlr.de

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

- Guideline document & Checklists
- 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

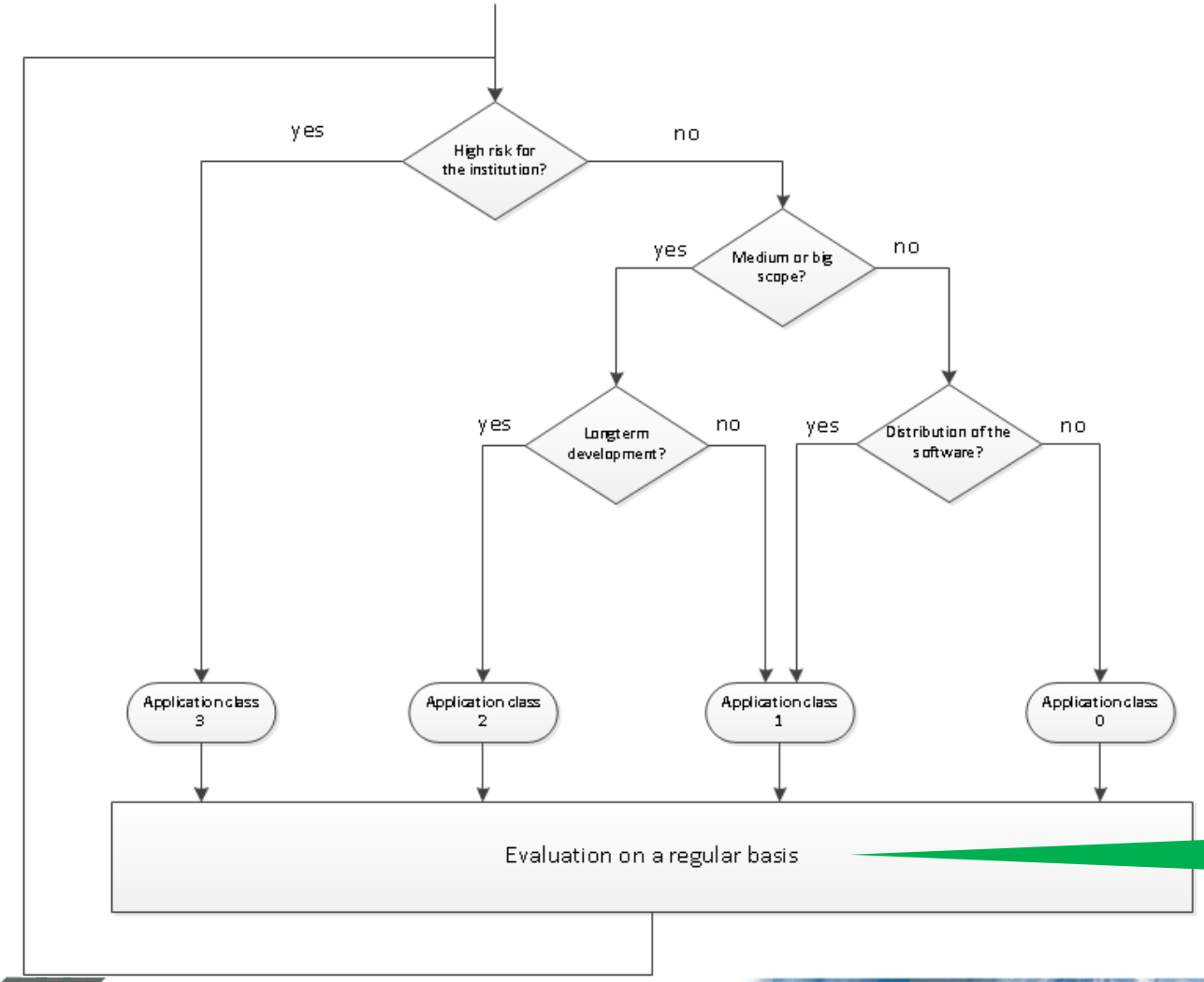
Release Management

Automation & Dependencies

https://zenodo.org/record/1344612 (EN)
https://zenodo.org/record/1344608 (DE)

The DLR Software Engineering Guidelines
Tailoring Checklists

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



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)

Classification may change over time!

The DLR Software Engineering Guidelines

Using Checklists

Check List

Änderungsmanagement		
EÄM.2: Die wichtigsten Informationen, um zur Entwicklung beitragen zu können, sind an einer zentralen Stelle abgelegt.		
EÄM.5: Bekannte Fehler, wichtige ausstehende Aufgaben und Ideen sind zumindest stichpunktartig in einer Liste festgehalten und zentral abgelegt.		
EÄM.7: Ein Repository ist in einem Versionskontrollsystem eingerichtet. Das Repository ist angemessen strukturiert und enthält möglichst alle Artefakte, die zum Erstellen einer nutzbaren Version der Software und deren Test erforderlich sind.		
EÄM.8: Jede Änderung des Repository dient möglichst einem spezifischen Zweck, enthält eine verständliche Beschreibung und hinterlässt die Software möglichst in einem konsistenten, funktionierenden Zustand.		

Concrete Guideline

EÄM.7 Ein Repository ist in einem Versionskontrollsystem eingerichtet. Das Repository ist angemessen strukturiert und enthält möglichst alle Artefakte, die zum Erstellen einer nutzbaren Version der Software und deren Test erforderlich sind.	**ab 1**	Das Repository ist der zentrale Einstiegspunkt in die Entwicklung. Dadurch sind alle wesentlichen Artefakte sicher gespeichert und an einer Stelle auffindbar. Einzelne Änderungen können nachvollzogen und dem jeweiligen Urheber zugeordnet werden. Darüber hinaus stellt das Versionskontrollsystem die Konsistenz aller Änderungen sicher. Die Verzeichnisstruktur des Repository sollte man anhand bestehender Konventionen ausrichten. Quellen dafür sind typischerweise das Versionskontrollsystem, das Build-Werkzeug (vgl. Abschnitt 4.8 Automatisierung und Abhängigkeitsmanagement) oder die Community der eingesetzten Programmiersprache bzw. des verwendeten Frameworks. Dazu zwei

Topic Overview

4.4 Änderungsmanagement

Gegenstand des Änderungsmanagements[11] ist, systematisch und nachvollziehbar Änderungen an der Software durchzuführen. Ursachen für Änderungen sind beispielsweise Anforderungen, Fehler oder Optimierungen. Das Änderungsmanagement unterstützt dabei, den Überblick über den Entwicklungsstand zu behalten und die verschiedenen Entwicklungsaufgaben zu koordinieren.

In diesem Zusammenhang beschreibt der **Änderungsprozess**, wie **Änderungswünsche** (z.B. Anforderungen, Fehler, Optimierungen) prinzipiell auf Entwicklerseite abgearbeitet werden und anschließend ggf. in Form einer neuen Software-Version zur Verfügung stehen. Dieser Prozess ist im Detail in jedem Entwicklungskontext unterschiedlich. Daher ist es wichtig, diesen im Entwicklungsteam abzustimmen und kontinuierlich zu verbessern. In der Praxis ist darauf zu achten, dass sich die Abläufe effizient umsetzen lassen. Daher ist auf angemessenen Einsatz von Werkzeugen und Automatisierung zu achten.

The DLR Software Engineering Guidelines
An Example

Python script for calculation of characteristics of a set of sample values

- Software is a small tool and used by other internally.

The software fits well into the application class 1.

Summary of the generic recommendations:

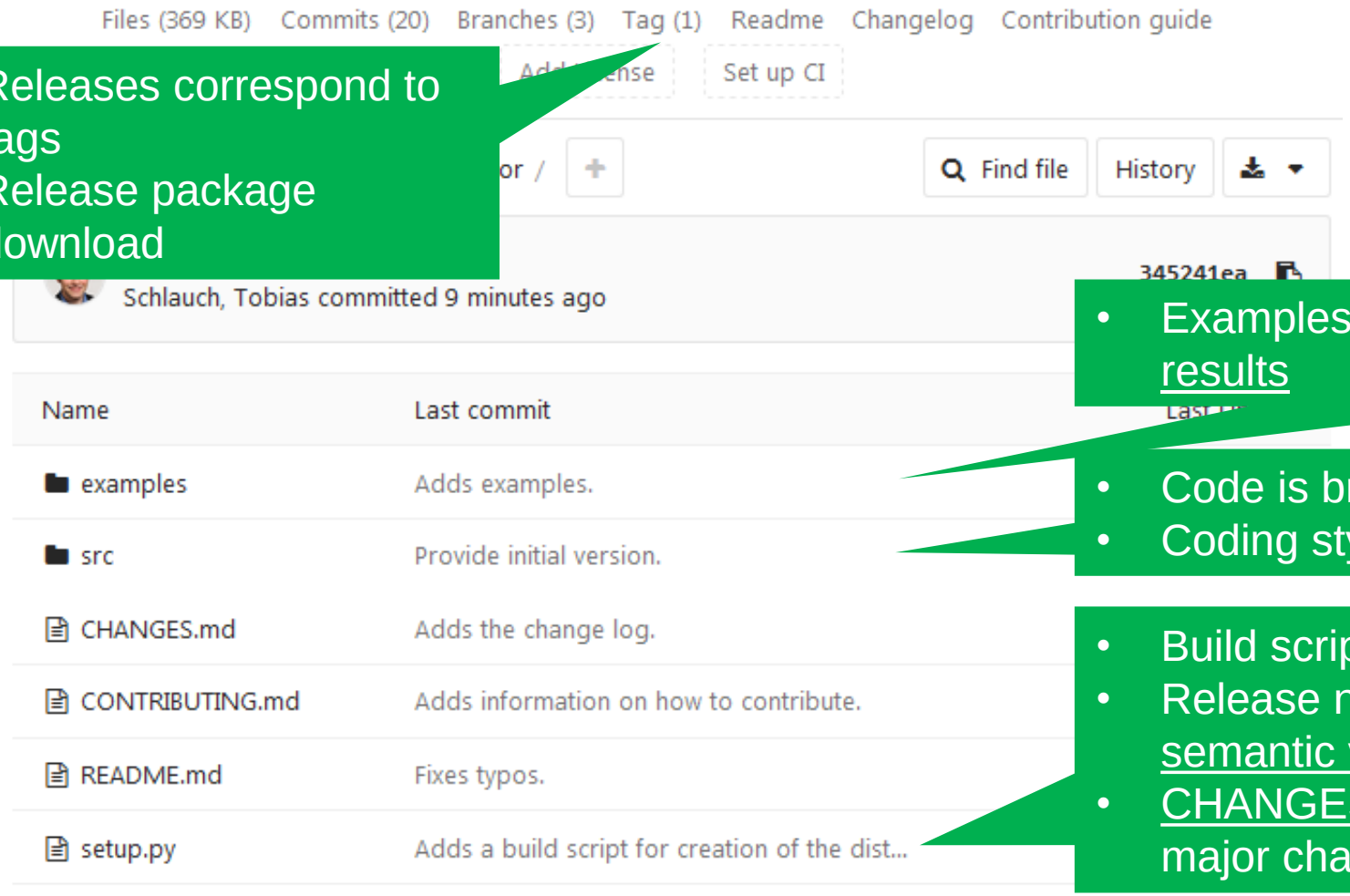
- Manage your code using a **version control system**
- Apply a **basic coding style**, strive for a **modular design**, **avoid code duplication** and **over-engineering**
- **Automate creation of an executable, usable version**
- **Provide essential documentation:** software purpose, user and developer information, constraints and central concepts, known problems and ideas
- **Internal release: test** your software and assign a proper **release number**
- **Public release:** check the **open source guidelines**

Recommendations have to be mapped into the concrete development context.

The DLR Software Engineering Guidelines
Possible Implementation

Git repository which contains code, examples, build script, and documentation

- Releases correspond to tags
- Release package download



- Examples provide reference input values and results

- Code is broken into small functions
- Coding style recommendations applied

- Build script for packaging and installing
- Release numbers follow semantic versioning approach
- CHANGES.md explains user-understandably major changes

The DLR Software Engineering Guidelines
Possible Implementation (cont.)

What is SampleCalculator?

SampleCalculator is a command line tool to calculate characteristic values of a sample.

It provides the following features:

- Reading sample values from command line and CSV (Colon Separated Values) files.
- Calculation of average, variance, and standard deviation.
- Configurable logging of results and interim results.
- Easy integration of new input sources
- Extensible by easily adding new calculations

SampleCalculator targets **scientists** who want to easily perform such calculations as part of their workflow and **Python developers** who want to integrate the functionalities into their software. We implemented as we have not found a suitable, zero-dependency alternative.

> The current version is only an initial alpha version which is **NOT** suited for production use. Particularily, it is not sufficiently tested with large data sets. It requires **Python** >= **3.4** and has been only tested on **Windows 7** so far. However, it should basically work on operating system.

How can I install it?

- Make sure that you use Python >= 3.4
- Download the latest package
- Extract it to a directory

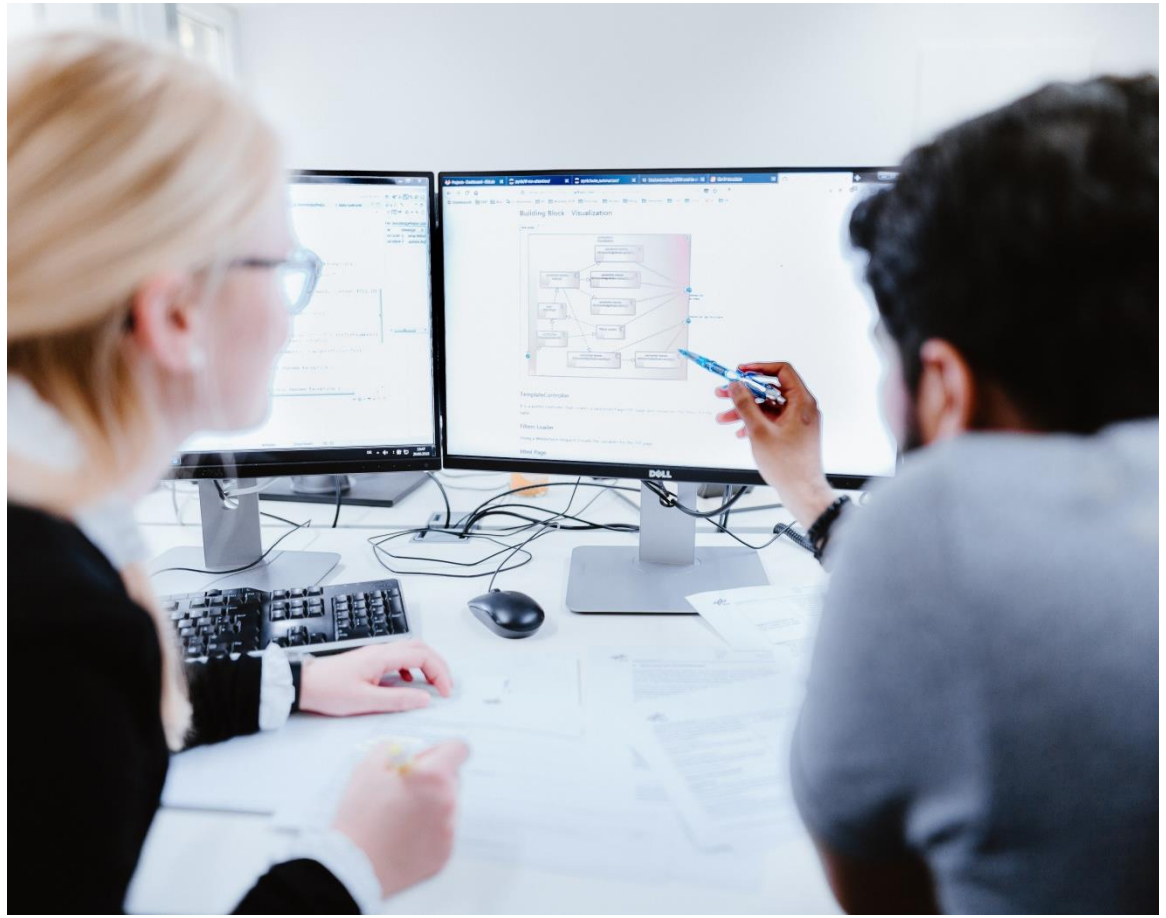
- README.md: main documentation
- CONTRIBUTING.md: contributor information

- Explanation of the software purpose (what?, for whom?, why?)
- Overview of the main features
- Important usage constraints and conditions

- Basic installation and usage information
- Future plans and ideas

The DLR Software Engineering Guidelines
Summary

- Checklists are project sensitive
- Generic recommendations → Solution Suggestions
- Covering the Basics
- Reevaluate regularly



DLR
Our approach

Software Engineering Initiative of DLR

- Guidelines
- Trainings
- Knowledge Provision
- Collaboration
- Experience Exchange

We are not the only ones…
eScienceCenter (NL)

- Similar to Application Class 1
- Split up by use case
  - Paper Publishing
  - Users
  - Contributors

Software checklist

Here we provide a short checklist for software projects, the rest of this chapter elaborates on the various point in this list.

The **bare minimum** that every software project should do, from the start, is:

- Pick & include an open source license
- Use version control
- Use a publicly accessible version control repository
- Add a readme describing the project

We recommend that you also do the following (from the start of the project):

- Use code quality tools
- Testing
- Use standards

Additional steps depend on the goal of the software (zero or more can apply):

- I'm publishing a paper
- I'm expecting users
- I'm expecting contributors

Quelle: eScienceCenter
https://guide.esciencecenter.nl/best_practices/checklist.html

There is a whole movement…

- RSE UK (http://rse.ac.uk/)
- de-RSE (http://www.de-rse.org/)
- Helmholtz Task Group „Wissenschaftliche Software“ (https://os.helmholtz.de)

Questions?

carina.haupt@dlr.de
@caha42

DLR