Helping a friend out

Guidelines for better software

Tobias Schlauch

German Aerospace Center (DLR)

Department Intelligent and Distributed Systems

Berlin / Braunschweig / Cologne

Research Software Engineering Conference 2017

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** are involved in software development.
- Typical examples of developed software include **simulation and modeling, flight control, signal and data processing,** 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 a Software Engineering Initiative at DLR because we believe that our research results profit from better software.

Software engineering initiative of DLR

Software Engineering Initiative of DLR

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

Software engineering guidelines

Guidelines support **research software 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** and are available as **checklists in different formats** (e.g., 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:

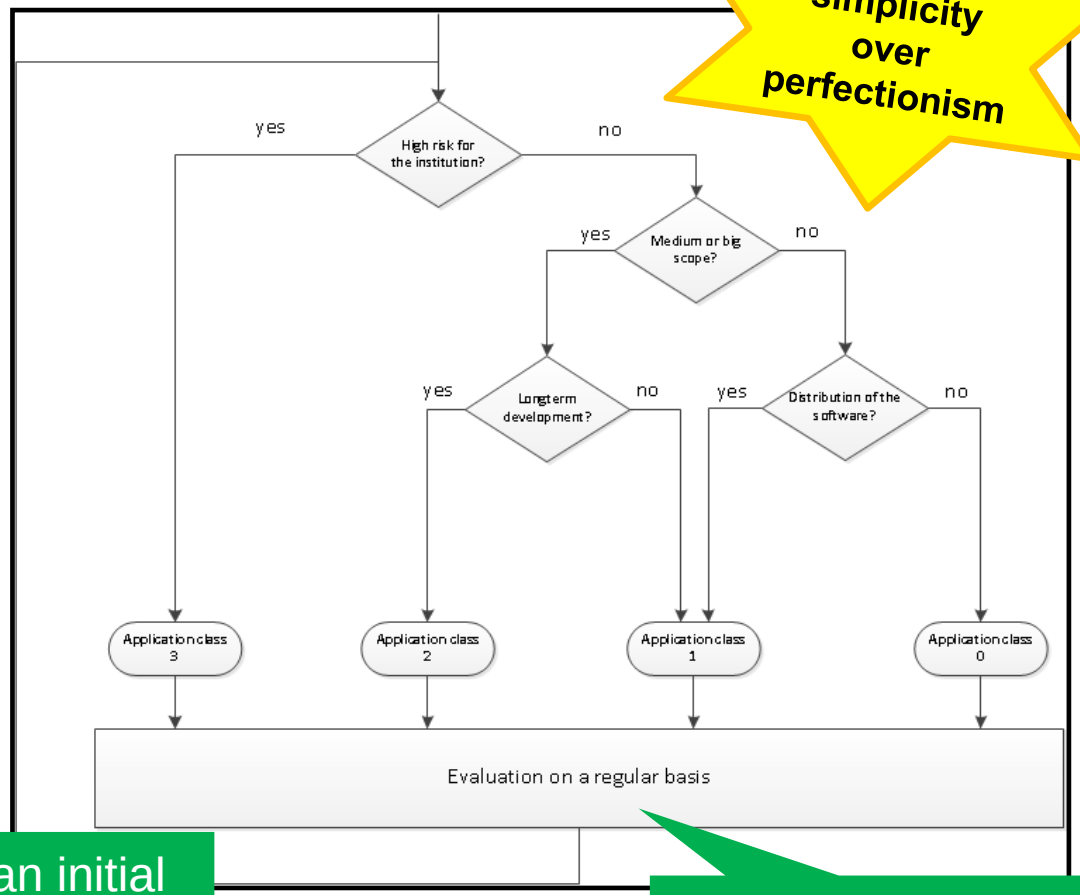
Tailoring scheme

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!

Example for application class 1

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.

Example for application class 1
Possible implementation

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

- Releases correspond to tags
- Release package download

Files (369 KB) Commits (20) Branches (3) Tag (1) Readme Changelog Contribution

Add license Set up CI

tor / + Find file History

minutes ago

Name	Last commit
examples	Adds examples.
src	Provide initial version.
CHANGES.md	Adds the change log.
CONTRIBUTING.md	Adds information on how to contribute.
README.md	Fixes typos.
setup.py	Adds a build script for creation of th

- Code is broken into small functions
- PEP8 coding style recommendations applied
- Examples provide reference input values and results

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

Example for application class 1
Possible implementation (cont.)

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

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
Python developers who want to integrate the functionalities into their software. We
not found a suitable, zero-dependency alternative.

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

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

How can I install it?

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

- Basic installation and usage information
- Future plans and ideas

Summary and experiences

Current status:

- Initial version of the guidelines published in March 2016 at DLR
- Promotion activities on institute and DLR level
- Additional supporting material has been provided (e.g., checklist formats)
- Institutes begin applying guidelines as part of their quality policy

Initial feedback from interviews with domain scientist is good but for certain topics more detailed solution approaches are demanded:

- Need for **clearer indication** and **provision of solution approaches** at DLR level (e.g., central software forge)
- Need for **domain-specific solution approaches** (e.g., testing approaches for real-time, embedded systems) which requires a stronger experience exchange

Summary and experiences (cont.)

Factors that helped us so far:

- Establishment of a vital core community across the DLR institutes
- Joint development of practical guidelines
- Raising management awareness of the topic
- Upper management support
- Initiating group is part of DLR`s research institutes

Guidelines are <u>important but not sufficient</u> for better research software:

- All activates of the initiative are equally important and build on each other
- Guidelines require regular updates on the basis of feedback from domain scientists