

Software development

In general our O2A software is open source, free of charge and may be re-used in other contexts. Our software is administrated in the gitlab.awi.de repository and you can also find examples on github.com. To foster and streamline software development we want to have one central managed code basis. All splitted branches for new features and bug fixes shall be finally merged into the central *master* branch aligned with [Gitflow](#). The agile software development process for each O2A component is managed in 2-weekly sprints with jira-software.awi.de following [SCRUM](#)-based principles.

Licenses

Our O2A software is licensed under [BSD-3-Clause](#), if no other definition is given. To foster community value, we encourage you to use also [BSD-2-Clause](#), [BSD-3-Clause](#) or [MIT](#) licences for your derived works. We will only integrate your work into our managed code basis, if it is licensed the way described before. 3rd-party libraries keep their licences and must be compatible with named licenses.

BSD-3-Clause License

Copyright 2010-2020- Alfred Wegener Institute, Helmholtz Centre for Polar and Marine Research

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Standards

Our software development follows simple rules as best practices. Use the following topics and tasks as guideline and checklist.

Version control system

- ☐ Use Git. gitlab.awi.de
- ☐ Commit and push your daily work.
- ☐ Use [Gitflow](#) semantics to structure contributions aligned with tickets.
- ☐ Each repository must have a `README.md` markdown file in its root directory describing the project (see template below).
- ☐ Each repository must have a `.gitignore` file in its root directory defining files to exclude or to force inclusion.

README.md template

Name of the project and repository content

- * Short description of the project and repository content.
- * Provide relative links to demonstrators in this repository.
- * Provide an overview of the architecture and relation to other projects.
- * Use diagrams.net for your sketches and provide a editable link here.

Getting started

- * Describe how to start with this project.

Prerequisites

- * Describe requirements to fulfill before starting.

Installing

- * Describe how to install required software and this software.

Tests

- * Describe how to test this software.

Deployment

- * Describe how to deploy this software including required prerequisites and configuration.

Contributing

- * Describe how to contribute to this project. At best you link to general [documentation](https://spaces.awi.de/x/bu7FEw) and improve it if necessary.

Versioning

- * Describe shortly the versioning approach.

Authors

- * List the authors and emails here.

License

We have a BSD-3-Clause license and refer to [documentation](https://spaces.awi.de/x/bu7FEw).

Acknowledgments

- * Any acknowledgements?

Notes

- * Any additional notes?

.gitignore

```
### Additional Files ###
.env

# Created by https://www.gitignore.io/api/qt,vim,c++,java,linux,macos,python,windows,eclipse,netbeans,qtcreator,
jetbrains+all,visualstudiocode
# Edit at https://www.gitignore.io/?templates=qt,vim,c++,java,linux,macos,python,windows,eclipse,netbeans,
qtcreator,jetbrains+all,visualstudiocode

### C++ ###
# Prerequisites
*.d

# Compiled Object files
*.slo
*.lo
*.o
*.obj

# Precompiled Headers
*.gch
*.pch

# Compiled Dynamic libraries
*.so
*.dylib
*.dll

# Fortran module files
*.mod
*.smod

# Compiled Static libraries
*.lai
*.la
*.a
*.lib

# Executables
*.exe
*.out
*.app

### Eclipse ###

.metadata
bin/
tmp/
*.tmp
*.bak
*.swp
*~.nib
local.properties
.settings/
.loadpath
.recommenders

# External tool builders
.externalToolBuilders/

# Locally stored "Eclipse launch configurations"
*.launch

# PyDev specific (Python IDE for Eclipse)
*.pydevproject

# CDT-specific (C/C++ Development Tooling)
.cproject
```

```
# CDT- autotools
.autotools

# Java annotation processor (APT)
.factorypath

# PDT-specific (PHP Development Tools)
.buildpath

# sbteclipse plugin
.target

# Tern plugin
.tern-project

# TeXlipse plugin
.texlipse

# STS (Spring Tool Suite)
.springBeans

# Code Recommenders
.recommenders/

# Annotation Processing
.appt_generated/

# Scala IDE specific (Scala & Java development for Eclipse)
.cache-main
.scala_dependencies
.worksheet

### Eclipse Patch ###
# Eclipse Core
.project

# JDT-specific (Eclipse Java Development Tools)
.classpath

# Annotation Processing
.appt_generated

.sts4-cache/

### Java ###
# Compiled class file
*.class

# Log file
*.log

# BlueJ files
*.ctxt

# Mobile Tools for Java (J2ME)
.mtj.tmp/

# Package Files #
*.jar
*.war
*.nar
*.ear
*.zip
*.tar.gz
*.rar

# virtual machine crash logs, see http://www.java.com/en/download/help/error\_hotspot.xml
hs_err_pid*

### JetBrains+all ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio and WebStorm
```

Reference: <https://intellij-support.jetbrains.com/hc/en-us/articles/206544839>

User-specific stuff

.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

Generated files

.idea/**/contentModel.xml

Sensitive or high-churn files

.idea/**/dataSources/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
.idea/**/sqlDataSources.xml
.idea/**/dynamic.xml
.idea/**/uiDesigner.xml
.idea/**/dbnavigator.xml

Gradle

.idea/**/gradle.xml
.idea/**/libraries

Gradle and Maven with auto-import

When using Gradle or Maven with auto-import, you should exclude module files,
since they will be recreated, and may cause churn. Uncomment if using
auto-import.

.idea/modules.xml
.idea/*.iml
.idea/modules

CMake

cmake-build-*/

Mongo Explorer plugin

.idea/**/mongoSettings.xml

File-based project format

*.iws

IntelliJ

out/

mpeltonen/sbt-idea plugin

.idea_modules/

JIRA plugin

atlassian-ide-plugin.xml

Cursive Clojure plugin

.idea/replstate.xml

Crashlytics plugin (for Android Studio and IntelliJ)

com_crashlytics_export_strings.xml
crashlytics.properties
crashlytics-build.properties
fabric.properties

Editor-based Rest Client

.idea/httpRequests

Android studio 3.1+ serialized cache file

.idea/caches/build_file_checksums.ser

JetBrains+all Patch

Ignores the whole .idea folder and all .iml files

See <https://github.com/joeblau/gitignore.io/issues/186> and <https://github.com/joeblau/gitignore.io/issues/360>

.idea/

```
# Reason: https://github.com/joeblau/gitignore.io/issues/186#issuecomment-249601023

*.iml
modules.xml
.idea/misc.xml
*.ipr

### Linux ###
*~

# temporary files which can be created if a process still has a handle open of a deleted file
.fuse_hidden*

# KDE directory preferences
.directory

# Linux trash folder which might appear on any partition or disk
.Trash-*

# .nfs files are created when an open file is removed but is still being accessed
.nfs*

### macOS ###
# General
.DS_Store
.AppleDouble
.LSOVERRIDE

# Icon must end with two \r
Icon

# Thumbnails
.*_

# Files that might appear in the root of a volume
.DocumentRevisions-V100
.fsevents
.Spotlight-V100
.TemporaryItems
.Trashes
.VolumeIcon.icns
.com.apple.timemachine.donotpresent

# Directories potentially created on remote AFP share
.AppleDB
.AppleDesktop
Network Trash Folder
Temporary Items
.apdisk

### NetBeans ###
**/nbproject/private/
**/nbproject/Makefile-*.mk
**/nbproject/Package-*.bash
build/
nbbuild/
dist/
nbdist/
.nb-gradle/

### Python ###
# Byte-compiled / optimized / DLL files
__pycache__/*
*.py[cod]
*$py.class

# C extensions

# Distribution / packaging
.Python
```

```
develop-eggs/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
MANIFEST

# PyInstaller
# Usually these files are written by a python script from a template
# before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.nox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
.hypothesis/
.pytest_cache/

# Translations
*.mo
*.pot

# Django stuff:
local_settings.py
db.sqlite3

# Flask stuff:
instance/
.webassets-cache

# Scrapy stuff:
.scrapy

# Sphinx documentation
docs/_build/

# PyBuilder
target/

# Jupyter Notebook
.ipynb_checkpoints

# IPython
profile_default/
ipython_config.py

# pyenv
.python-version

# celery beat schedule file
```

```
celerybeat-schedule

# SageMath parsed files
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

### Python Patch ###
.venv/

### Qt ###
# C++ objects and libs

# Qt-es
object_script.*.Release
object_script.*.Debug
*_plugin_import.cpp
/.qmake.cache
/.qmake.stash
*.pro.user
*.pro.user.*
*.qbs.user
*.qbs.user.*
*.moc
moc_*.cpp
moc_*.h
qrc_*.cpp
ui_*.h
*.qmlc
*.jsc
Makefile*
*build-*

# Qt unit tests
target_wrapper.*

# QtCreator
*.autosave

# QtCreator Qml
*.qmlproject.user
*.qmlproject.user.*

# QtCreator CMake
CMakeLists.txt.user*
```



```
### QtCreator ###
# gitignore for Qt Creator like IDE for pure C/C++ project without Qt
#
# Reference: http://doc.qt.io/qtcreator/creator-project-generic.html

# Qt Creator autogenerated files

# A listing of all the files included in the project
*.files

# Include directories
*.includes

# Project configuration settings like predefined Macros
*.config

# Qt Creator settings
*.creator

# User project settings
*.creator.user*

# Qt Creator backups

### Vim ###
# Swap
[.]*.s[a-v][a-z]
[.]*.sw[a-p]
[.]*s[a-rt-v][a-z]
[.]*ss[a-gi-z]
[.]*sw[a-p]

# Session
Session.vim

# Temporary
.netrwhist
# Auto-generated tag files
tags
# Persistent undo
[.]*.un~

### VisualStudioCode ###
.vscode/*
!.vscode/settings.json
!.vscode/tasks.json
!.vscode/launch.json
!.vscode/extensions.json

### VisualStudioCode Patch ###
# Ignore all local history of files
.history

### Windows ###
# Windows thumbnail cache files
Thumbs.db
ehthumbs.db
ehthumbs_vista.db

# Dump file
*.stackdump

# Folder config file
[Dd]esktop.ini

# Recycle Bin used on file shares
$RECYCLE.BIN/
```

```
# Windows Installer files
*.cab
*.msi
*.msix
*.msm
*.msp

# Windows shortcuts
*.lnk

# End of https://www.gitignore.io/api/qt,vim,c++,java,linux,macos,python,windows,eclipse,netbeans,qtcreator,
jetbrains+all,visualstudiocode
```

Ticketing - documentation of work

- ☐ Document and describe your work and requirements in stories, tasks and bugs. - jira-software.awi.de
- ☐ Define a definition of done.
- ☐ Put new things into the backlog.
- ☐ Define a definition of ready, when do you can start to work on a ticket.
- ☐ Estimate the effort for a ticket.
- ☐ Only ready tickets go into a sprint.
- ☐ Set the software version to a ticket.
- ☐ Keep track of progress and log your work.
- ☐ Documentation and tests are part of the work within a ticket.
- ☐ Check definition of done, then done.

Documentation

- ☐ Provide an up-to-date description of the project. - README.md
- ☐ Provide a "getting started" description for developers. - README.md
- ☐ Provide a "deployment" description for developers to setup. - README.md
- ☐ Create architecture and relationship diagrams where necessary to get an overview **for others**.
- ☐ Use simple diagrams - blocks and edges - where useful and provided editable formats. Best go for diagrams.net.
- ☐ For details use UML standardized diagrams. Best go for diagrams.net.
- ☐ Document and comment your code. It's part of your daily work.
- ☐ Provide admin documentation to setup the project including configuration description.
- ☐ Provide usage and API documentation for end users. Be clear, not technical.
- ☐ For web services use [Swagger](https://swagger.io) documentation.

Code style

- ☐ Align your code structure and documentation to [Google Style Guides](https://google.github.io/styleguide/).
- ☐ Document your code short and clear. Your (not involved) colleague should easily jump in.
- ☐ Use [Editorconfig](https://editorconfig.org) and place a .editorconfig file in your project root to force basic code styling.

.editorconfig

```
root = true

[*]
charset = utf-8
end_of_line = lf
indent_size = 2
indent_style = space
insert_final_newline = true
trim_trailing_whitespace = true

[*.md]
trim_trailing_whitespace = false

[*.js,py]
charset = utf-8

[*.php,py]
indent_style = space
indent_size = 4

[*.yaml]
indent_size = 2

[Makefile]
indent_style = tab

[lib/**/*.js]
indent_style = space
indent_size = 2

[{package.json,.travis.yml}]
indent_style = space
indent_size = 2
```

Languages

We focus on specific languages for client-side and server-side developments and data processing. We try to keep dependencies to 3rd-party libraries as low as possible. These decisions are based on a trade-off estimate between effort and sustainability. So take the following points as guideline.

Client-side

- ☐ Use basic JavaScript supported features.
- ☐ Follow the [JavaScript](#), [HTML/CSS](#) code style guides.
- ☐ Don't get for 3rd-party libraries for simple and standard functions if not required. Discuss the trade-off.
- ☐ Use JavaScript modules where ever possible and useful. Keep an eye on re-usability and usage in other contexts.
- ☐ Provide an index.js in module directories to bundle useful module classes.
- ☐ Always provide [JSDoc](#) for classes, methods and integrations. Don't comment every line or simple getters and setters.
- ☐ Use primary used frameworks: [Bootstrap](#), [Leaflet](#), [D3.js](#), [Plotly](#), [jQuery](#). Discuss using frameworks like [React](#), [Vue](#) or similar.
- ☐ Use bundling tools for web-optimizing, not for primary development.

Server-side

- ☐ Use Java for middleware and backend.
- ☐ Follow the [Java](#) code style guide.
- ☐ Use [Maven](#) for dependency and build management.
- ☐ Always provide [Javadoc](#) for classes, methods and integrations. Don't comment every line or simple getters and setters.
- ☐ Use primary defined standards and long-term sustained libraries, e.g. [Java Persistence API](#) with [Hibernate](#) implementation.
- ☐ Use logging libraries for system outputs and write to files, e.g. [Log4j](#).

- ☐ Install snapshots and releases to [Nexus](#) repository manager (only AWI internal access).

Of course other languages are possible. But we clearly focus on Java. For example you can use PHP for simple web application or portal tasks. But don't develop heavy software. even if you go with [Laravel](#) or similar. Use [Composer](#) for dependency management.

Processing

- ☐ Use Python for data processing and automation tasks.
- ☐ Follow the [Python](#) code style guide.
- ☐ Use [virtual environments](#) for dependencies with pip or conda.
- ☐ Provide a `requirements.txt` file in the project repository root directory.
- ☐ Always provide comments for classes, methods and integrations. Don't comment every line or simple getters and setters.
- ☐ Use primary used libraries and sustained libraries, e.g. [pandas](#), [requests](#), [tensorflow](#).
- ☐ Use [logging](#) for system outputs and write to files.

Tests

- ☐ Tests exists for my commit?
- ☐ Write unit tests for methods and critical workflows.
- ☐ Write integration and system tests from frontend (grey) to backend (black).
- ☐ Check requirements for development, test and production environment.
- ☐ Use [Gitlab Continuous Integration](#) pipeline.

Conventions

Project and repository names

- ☐ Always define a namespace for the project like `de.awi.` for e.g. `de.awi.sensor`.

Software versions

- ☐ Make sure your version covers also documentation and tests.
- ☐ Use the [Semantic Versioning](#) paradigm.
- ☐ Use 0.x.y as major version during initial development phase.
- ☐ Use version 1.0.0 as the official first public release.
- ☐ Don't prepend "v" or similar to a version.

Commit messages

- ☐ Each commit must relate to a ticket.
- ☐ Use the ticket ID and short description syntax for commits. For example `DAS-134 - Fixed dragging of tiles`
- ☐ If multiple tickets need to get references, add the uppercase ticket IDs in the body of the commit message.

Branch names

- ☐ Use [Gitflow](#) syntax and semantics.
- ☐ Each feature, bugfix, hotfix branch requires a ticket.
- ☐ Prepare release branches with the defined software version.

```
master
develop
feature/[ticket ID]_[short_name] - Use underscores. For example: feature/SEN-312_create_versions
bugfix/[ticket ID]_[short_name] - Use underscores. For example: bugfix/DAS-134_tile_dragging
hotfix/[ticket ID]_[short_name] - Use underscores. For example: hotfix/DAS-134_tile_dragging
release/[version] - For example release/1.24.3
```

