

5. Software Engineering best practices



Version Control: git

What is this?

A tool to allow you to track and revert changes, and collaborate with others (change management).

- Allow versioning of the code and continuing functionality.
- Allow simultaneous changes to the same files.
- Fundamental for reproducing historic states in the line of development.
- Development follows a story and allows other users to build confidence in your work.



Version Control in practice: git

- Create a repo on GitHub
- Clone the repo to your local machine
- Checkout a branch and make changes
- git add, git commit and git push: IDEs such as VSCode make it easy for you!
- Observe how your repo changes on GitHub
- GitHub offers great learning labs: GitHub skills https://skills.github.com/
- roadmap.sh offers a git roadmap: <u>https://roadmap.sh/git-github</u>





Development workflows: GitHub-flow

What is this?

GitHub-flow is a lightweight workflow with creating branches, making changes, opening Pull Requests, running Continuous Integration, and requesting code review, together with Issues and Kanban project boards.

- Manage how changes are incorporated in the software.
- Track progress in your project and highlight bottlenecks.
- Adhere to development guidelines, ensuring the implementation follows defined rules to ensure software quality.

Development workflows in practice: GitHub-flow

- After you made changes in your git branch, open a Pull Request on GitHub
- Observe how the PR highlights the changes in the line of development
- You can link issues, comment on the PR and run automated checks
- See GitHub skills https://skills.github.com/
- See roadmap.sh https://roadmap.sh/git-github



SCIENTIFIC

CENTER

UNIVERSITÄT HEIDELBERG

ZUKUNF SEIT 1386



55

What is this?

Requirements engineering is the process of translating stakeholder requirements on the research software into defined tasks. Early delivery and iteration over it allows refinement of the requirements and tasks.

- Ensure that the software fulfills its purpose. In research software, requirements engineering is closely intertwined with the research process and subject to frequent changes.
- Understand the problem the software should solve and map this onto an efficient technically feasible solution considering all constraints.
- Allows prioritization of requirements/tasks and decision-making. Decisions should be documented together with the requirements.



Requirements engineering in practice

- Functional requirements (what the software should do: features)
- Non-functional requirements (how the software performs a task: ie performance, security)
- Domain requirements (specific to the domain: ie. Healthcare)
- Use tools such as draft.io or miro to gather requirements

As a <type of user>, I want <some goal> so that <some reason>.

```
As a <researcher>, I want
<to obtain the
probability of a class>
so that <I can classify
incoming images>.
```

Continuous delivery in practice

- Deliver early to find out if your software is fulfilling its purpose/moving in the right direction
- Use quality control to allow early delivery through the main branch in your GitHub-flow (*keep your main branch operational at all times*)
- Following agile / lean principles
- Use git to allow consistent usability of your software





Project management: Kanban boards

What is this?

Project management is used to track progress, identify intertwined or dependent processes, and allows visual access to the project's status.

- Ensure that the software development moves in the right direction.
- Increase the flow of ongoing work.
- Allow prioritization of tasks and understand interdependencies and bottlenecks in the development.



Project management in practice

- Use a Kanban board to organize tasks
- Separate tasks into backlog/todo, in progress, done
- Prioritize tasks and assign contributors/identify necessities
- For example, GitHub projects

O No Status ③ (Estimate: 0) ···· Entry point for all new initiatives	Backlog 9) (Estimate: 0) ···· Initiatives waiting assignation to SSC members	In Progress (3) (Estimate: 26) ···· Initiatives actively being worked by the SSC.	Done 0 Estimate: 0 ···· Initiatives successfully completed, awaiting closing ceremony (Feedback)	Closed (3) (Estimate: 3) Here is when the project is archived once has ended.	Cancelled/Stopped
() Draft MatlabWebApp	Oraft predicTCR Funded project Medical 2024	sc-projects #29 Establishing a knowledge graph community in biomedical science		© ssc-projects #27 VASP: Voxel Attributes and Statistics for Point Clouds	initiatives cancelled or stopped for any administrative reason.
ssc-projects #32 Onboarding SSC PROJECT	Draft Control Dr	Po 21 AL Funded project Bioinformatics 2024 PROJECT O: ssc-projects #28 Image: Comparison of the second		(P2) 3 (S) Free consultation (O) ssc-projects #35 (D)	
Oraft 🔮 project-w Small-scale 2024	Oraft Sci-rocket	Project management tools for SSC P2 5 M internal SSC 2024 PROJECT Orac projects #54		Hamming distance project Small-scale Maths 2020	
	Small-scale Bioinformatics 2024	ParzivAl Small-scale 2024		Neuroscience data processing and analysis Open call Biology 2021 PROJECT	



Project planning: Architecture and design

What is this?

Software architecture describes how the system is composed of different pieces, and the interplay of the components. Design refers to the actual implementation of the requirements in the system as a whole and the different components.

- Makes the software efficient and allow re-use of functionalities.
- Allows extensions and additions of features at a later stage without major refactoring.
- Makes the software maintainable.



Architecture in practice

- Use (black/white) box diagrams to identify components and their interactions
- <u>https://roadmap.sh/software-design-architecture</u> / miro / draft.io / draw.io





Design in practice

- Map the input/output, data formats, transformations / logic / processes
- <u>https://roadmap.sh/software-design-architecture</u> / miro / draft.io / draw.io





Quality management: Testing and continuous Integration

What is this?

GitHub-flow is a lightweight workflow with creating branches, making changes, opening Pull Requests, running Continuous Integration, and requesting code review, together with Issues and Kanban project boards.

- Manage how changes are incorporated in the software.
- Track progress in your project and highlight bottlenecks.
- Adhere to development guidelines, ensuring the implementation follows defined rules to ensure software quality.



Testing in practice

- Use testing frameworks such as pytest
- Write tests in a tests/ folder: unit tests, integration tests, system tests, compatibility tests, ...
- To learn how to use pytest: https://docs.pytest.org/en/stable/,

	26 27	@pytest.fixture
∽ 📀 ammico	28	<pre>def accepted(monkeypatch):</pre>
∨ 💿 test	29	<pre>monkeypatch.setenv("OTHER VAR", "True")</pre>
> <pre>O test_faces.py</pre>	30	<pre>tt.TextDetector({}, accept privacy="OTHER VAR")</pre>
> 💿 test_multimodal_search.py	31	return "OTHER_VAR"
> 0 test_summary.py	32	
∽ ⊘ test_text.py	33	
 test privacy statement 	▷ 34	<pre>def test_privacy_statement(monkeypatch):</pre>
	35	<pre># test pre-set variables: privacy</pre>
		<pre>monkeypatch.delattr("builtins.input", raising=False)</pre>
o test_run_spacy	37	<pre>monkeypatch.setenv("OTHER VAR", "something")</pre>
o test_clean_text	38	with pytest.raises(ValueError):
 test_init_revision_numbers 		<pre>tt.TextDetector({}, accept_privacy="OTHER_VAR")</pre>
test_check_add_space_aft		<pre>monkeypatch.setenv("OTHER_VAR", "False")</pre>
⊘ test_truncate_text	41	with pytest.raises(ValueError):
o test analyse image	42	<pre>tt.TextDetector({}, accept_privacy="OTHER_VAR")</pre>



Continuous integration in practice

- Set up your tests to be automatically run by GitHub actions
- Include code linter and quality control in your actions
- These should be set up to run automatically when you open a Pull Request
- GH actions, codecov, sonarcloud, snyk, pre-commit, code formatting (black), GitHub Guardian, dependabot





Software Management Plans

What is this?

Software Management Plans (SMPs) help to identify goals and the means required to pursue the goals in practice.

- Identify criticality and required maturity of your software.
- Identify which measures are needed to ensure compliance of your software with the intended goals.
- Quantify milestones and tools for the intended purpose.

SMPs in practice

RDMO for MPG Back to project

Software Project Schedule

When does the software project start?

When does the software project end?

Software Project Management

Which software development process is defined? How will process roles be assigned? How do you track the different tasks and use cases? Will there be a specification document (briefly) outlining the most important requirements?

Software Development Requirements

Are there institutional requirements for software development? Are there requirements regarding the software development form other parties?

Technical

Code

Which programming language(s) do you plan to use? Which technology or process is used for versioning?

Third Party Components and Libraries

Which external software components will be used? What dependencies on software libraries do exist? How do you document this? What licences are on the third-party software components? What is the process to keep track of the external software components? Can critical dependencies be eliminated or mitigated? Do you plan to use third party web services? Does the software refer to other software projects or objects?

Infrastructure





- Use the SMPs provided by the Max Planck digital library
- Helps you with your requirements and project management
- https://rdmo.mpdl.mpg.de/



Documentation

What is this?

Documentation can be comments, docstrings, readme's, tutorials, demonstration notebooks, and contains technical and domain-specific / application-specific descriptions of the software.

- Document what the software can and cannot do, and parameter ranges.
- Allow others to install and use your software (or yourself, at a later time).
- Allow others to contribute to your software.

Documentation in practice

text module

class text.PostprocessText(mydict: dict | None = None, use_csv: bool = False, csv_path: str | None = None, analyze_text: str = 'text_english')

Bases: object

analyse_topic(return_topics: int = 3)→ tuple

Performs topic analysis using BERTopic.

 Parameters:
 return_topics (int, optional) - Number of topics to return. Defaults to 3.

 Returns:
 tuple - A tuple containing the topic model, topic dataframe, and most frequent topics.

 $get_text_df(analyze_text: str) \rightarrow list$

Extracts text from the provided dataframe.

Parameters:analyze_text (str) - Column name for the text field to analyze.Returns:list - A list of text extracted from the dataframe.

get_text_dict(analyze_text: str)→ list

Extracts text from the provided dictionary.

Parameters:analyze_text (str) - Key for the text field to analyze.Returns:list - A list of text extracted from the dictionary.



SCIENTIFIC SOFTWARE

CENTER

UNIVERSITÄT HEIDELBERG

ZUKUNFT SEIT 1386

- Include jupyter notebooks that showcase use of your software - these can be run on google colab
- Document dependencies in a requirements file and provide installation instructions



Deployment: Runtime environment / containerisation

What is this?

Deployment information such as runtime environments or containers allow easy adaption as they provide direct access to running the software without installation and dependency conflicts.

- A big step towards reproducibility and transferability of your approach.
- The software ecosystem changes quickly, and this allows to preserve a snapshot that can be shared and run easily.



Containerisation in practice

- Use docker to provide build instructions for containers, and possibly deploy the containers on Dockerhub for anyone to download and use
- Docker roadmap <u>https://roadmap.sh/docker</u>, official tutorial <u>https://docs.docker.com/</u>

#	erfile > 🏵 FROM	
	FROM jupyter/base-notebook	
	# Install system dependencies for computer vision packages	
	USER root	
	RUN apt update && apt install -y build-essential libgl1 libglib2.0-0 libsm6 libxrender1 lib	oxext6
	USER \$NB_USER	
	# Copy the repository into the container	
	COPYchown=\${NB_UID} . /opt/ammico	
	# Install the Python package	
	RUN python -m pip install /opt/ammico	
	# Make JupyterLab the default for this application	
	ENV JUPYTER_ENABLE_LAB=yes	
	# Export where the data is located	
	ENV XDG_DATA_HOME=/opt/ammico/data	



Software Licensing

What is this?

A software license states the terms of use, re-use and distribution, among others, without violating copyrights, and defines responsibilities.

- So that others may use your code, and to prevent misuse.
- So that others may contribute to your code.
- So that the responsibilities for how the software is used are clear.
- Establishes the rights of all parties involved with the software.



Software licensing in practice

- Use the provided templates from GitHub: Either at repository creation or when adding a new file called LICENSE
- Permissive open-source license: BSD 2-Clause, MIT, Apache License 2.0
- Copyleft open-source license: GNU version 3, LGPL
- Proprietary licenses: Do not only keep your project close-source and potentially less visible, but also carry responsibilities for contract fulfillment
- <u>https://opensource.org/licenses</u>, <u>https://choosealicense.com/</u>
- When using/incorporating third-party software: ie. use of open-source libraries is the third-party code distributed with your software? If so, compatibility needs to be confirmed!