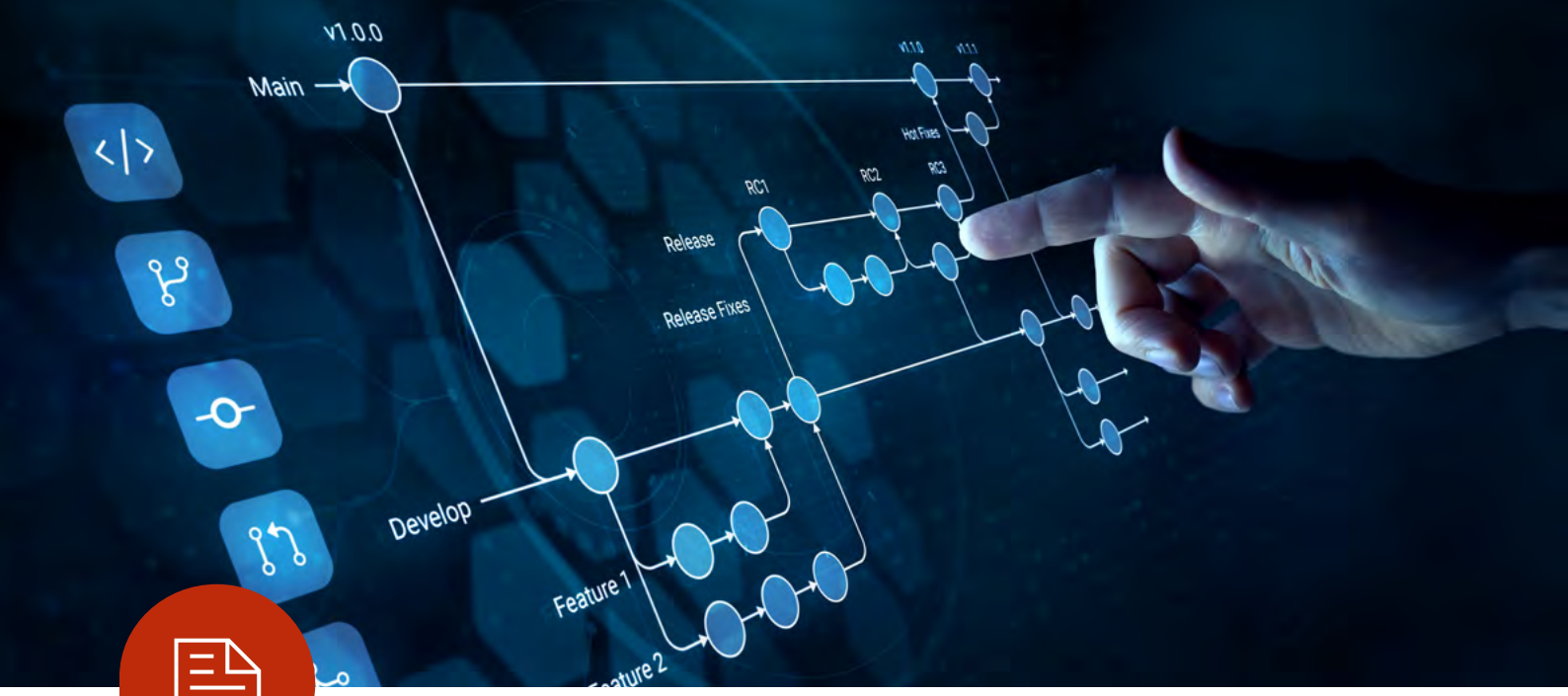

Managing software

Software is developed across a variety of disciplinary domains and should always be treated with care. Unlike data, software is executable, it continuously changes its form over time and hence requires specific steps to be managed properly.

Let us identify the steps and tools relevant to each phase of code development.

In the **planning** phase, roles and responsibilities for designing, developing and maintaining the code should be defined. From architecture and concept design, to development, up to revision and debugging.

During **development** it is highly recommended to use online platforms such as GitHub and GitLab. They are built to monitor development through version control and ensure effective collaboration thanks to Git's distributed system.

When your software reaches a satisfactory working version, it is also a good idea to **deposit** it in Zenodo or in another repository specific to this type of digital object, such as Software Heritage. They both allow the software to be semi-automatically updated in case of further development.

Deposit both **machine-readable and human-readable documentation** with your code **Metadata and documentation**. On the one hand, this could simply consist of metadata and a citation file – two examples specifically designed for software are CodeMeta and CITATION.cff. On the other hand, human-readable documentation usually includes a README file and inline documentation, i.e. comments within the code. Both make your code easier to understand for anyone else (or your future self!).

Once deposited, the software requires a licence to clarify which uses are permitted. The choice of a certain licence over another depends on your

specific situation – whether you need to work in a community, you want a simple and permissive licence, or you wish to share improvements.

After deposit in a repository such as Software Heritage, the code is assigned a Persistent Identifier (PID), a licence, and disciplinary metadata describing it. This means that it can be cited as a research output, in the same way as a publication.

Sometimes, however, archiving your software (even with the related input and output data) is not sufficient to ensure its **reusability** by contemporaries, and especially by the developers of the future. Programming languages, libraries and plug-ins change version often, and even code developed a few months prior may no longer be up to date. Special solutions, such as containers or cloud-based systems, can simulate the system environment variables in which the code was developed and to run it again.

In the field!

The ultimate research goal of my competitive project is to develop a data analysis software.
What workflow should I follow?

Arrange a kick-off meeting with the entire development team to identify roles and responsibilities, as well as setting a work schedule and deadlines.

Development can start directly in GitHub – creating a new repository shared by all developers facilitates collaborative work by keeping track of changes to the code, input data and other support material.

Some cloud-based tools, such as electronic notebooks, can help document every step of the code algorithm, keeping track of and visualising the input and output data of each function.

Linking the repository to Software Heritage before your code reaches the first operating version enables regular and automatic harvesting and ensures that the code is properly preserved (a feature not guaranteed by GitHub!) with an appropriate metadata schema, i.e. CodeMeta.

Various interactive online tools – such as Choose a Licence – summarise the entire documentation of the main licences in use for software, allowing you to choose a licence that suits your project requirements, which you can include in the GitHub repository.

To ensure proper recognition of your work, CITATION.cff files can also be generated semi-automatically, which saves a lot of time, and then uploaded directly to the GitHub repository and thus automatically saved to Software Heritage.

Useful links

Licence chooser tool https://choosealicense.com/

Citation file generator https://citation-file-format.github.io/cff-initializer-javascript

Examples of cloud notebooks for code development https://datasciencenotebook.org/

- Software Heritage https://www.softwareheritage.org/
- CodeMeta https://codemeta.github.io/codemeta-generator/
- GitHub https://github.com/