# Nachhilfebörse HTL-Leonding Configuration Management

## Versioning Methods and Tools

For sharing our files, we use GitHub.

## Documents under version control

We plan to put all project artefacts under version control. The only exceptions will Word documents be because of their difficult handling in case of needing to be merged. Furthermore we estimate that the joint production of such documents will be easier done in a shared environment. Therefore we will use Google docs for the preparation of these documents. However, as soon as they are in a stable state, they will be added to the version control too.

## Organization of artefacts

We expect the following artefacts to be produced in our project.

- Planning: Some Excel sheets covering estimations, progress monitoring, etc.
- Requirements documents
- Source code: All source code files plus other resources which will go directly into our software
- Testing documents
- Release documents
- Protocols from important conversations (Mrs Keck)
- Mics Documents, such as Project Proposal, System Specification etc.
- Other files

Therefore it makes sense to establish the following top level folder structure in our project.

- 01_Planning
- 02_Requirements
- 03_SourceCode
- 04_Testing
- 05_ReleaseDocs
- 06_MeetingMinutes
- 07_MicsDocuments
- 08_Others

## Variant Management

*Organization of different versions and branches*

The different versions will be stored on GitHub. That way we do not have to back our project up regularly and in case of emergency (bugs, project cannot build anymore) we can revert to an older version and start again.
For branching we thought it would be convenient to use the standard branching plan, which includes:

- Master branch: This is the main branch of the project. It is just edited by merging.
- Development branch: This branch represents the progress of the project. It can be edited by directly committing of small changes or merging feature branches.

- Feature branches: Small features (usually done by one person) will be done in extra feature branches. They will be merged to the development branch when they are finished.
- Release branch: When a new release is planned, a new release branch will be added. It will be created from the development branch and when the release is done, the release branch will be merged to the master branch and the development branch.
- Bugfix branch: If there is a need to fix bugs after the project has been merged from the development branch to the master branch, we will create a bugfix branch from the master branch. After the changes are done, the bugfix branch will be merged to the master branch again and to the development branch, so that the bug fixes will go in the next release too.

*Naming of new project versions*

For the name-giving of the versions of our project we will stick to the easiest way: The structured way (Code_v1.2.4). The first number, e.g. 1, describes the major version (Version 2 could be completely different than version 1). The second one is the minor version (Interfaces from v1.1 are still available in v1.2, but new interfaces are added) and the third number describes the bugs fixed (v1.2.4 has less bugs than v1.2.3).