



European Master in
Building Information Modelling

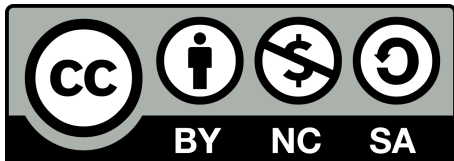
Introduction to Git and GitHub

Topic 2: Fundamentals of programming

BIM A+3: Parametric Modelling in BIM

Matevž Dolenc

Univerza v Ljubljani



© 2019 by authors

This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

"FINAL".doc



FINAL.doc!



FINAL_rev.2.doc



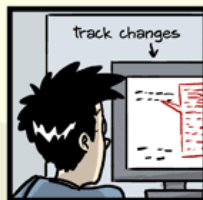
FINAL_rev.6.COMMENTS.doc



FINAL_rev.8.comments5.
CORRECTIONS.doc



JORGE CHAM © 2012



FINAL_rev.18.comments7.
corrections9.MORE.30.doc

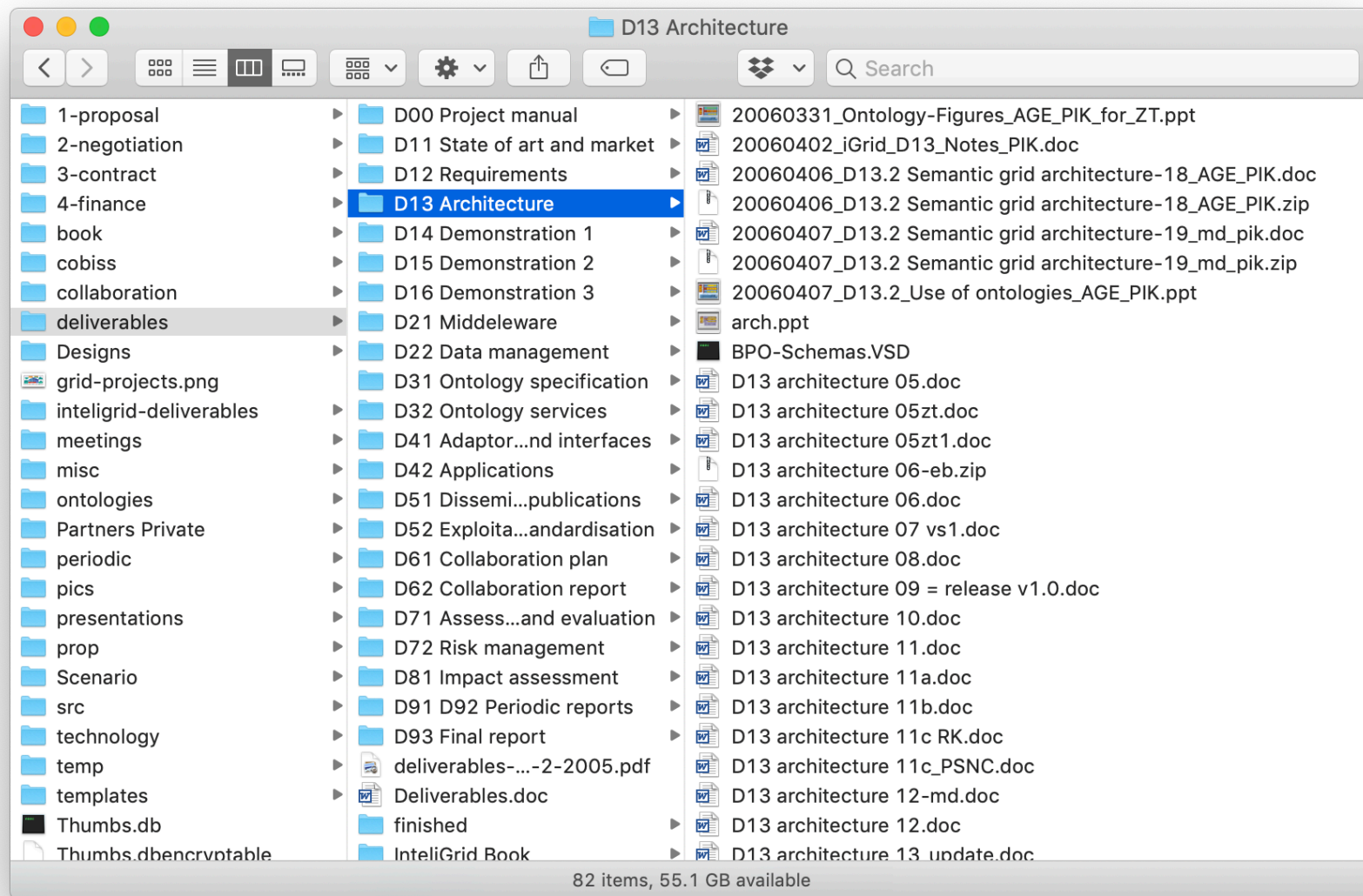


FINAL_rev.22.comments49.
corrections.10.##\$%WHYDID
ICOMETOGRADSCHOOL????.doc



WWW.PHDCOMICS.COM

<http://phdcomics.com/comics/archive.php?comicid=1531>



- A way to keep track of changes to files & folders
- Between multiple authors (developers)
 - A record of who did what, when
 - Why is provided by commit messages!
 - Non-distributed (Subversion, CVS) – Server has the master repo, all commits go to the server
 - Distributed (Git, Mercurial) – Server has the master repo, but you have a copy (clone) of the repo on your machine

- An open source VCS designed for speed and efficiency
- Created by Linus Torvalds (for managing Linux kernel)
- Your best insurance policy against:
 - Accidental mistakes like deleting work
 - Remembering what you changed, when, why
- Hosted solutions include Github or Bitbucket ...
... or running your own Git server



Tech Talk: Linus Torvalds on git

Google ✓ 2.1M views • 12 years ago

Linus Torvalds visits Google to share his thoughts on git, the source control management system he created two years ago.

CC

- Download the software - it's free
 - <http://git-scm.com/downloads> or on Mac (homebrew), \$ brew install git
- Download a GUI, optional
 - <http://git-scm.com/downloads/guis>
- Read the manual and/or the book
 - <http://git-scm.com/docs>, <http://git-scm.com/book>

- Obtain a repository
 - Either via git init, or git clone, or if you already have the repo, pull changes!
- Make some edits
 - Use your favorite text editor or source code IDE - Most IDEs have Git integration, including VSCode
 - Git tracks changes to binary files too: images, pdf, etc. – Less useful though, than text-based files
- Stage your changes
 - using git add
- Commit your work
 - git commit -m "Always write clear commit messages!"
- Push to remote
 - git push remote_name local_branch

GitHub

GIT CHEAT SHEET

Git is the open source distributed version control system that facilitates GitHub activities on your laptop or desktop. This cheat sheet summarizes commonly used Git command line instructions for quick reference.

INSTALL GIT

GitHub provides desktop clients that include a graphical user interface for the most common repository actions and an automatically updating command line edition of Git for advanced scenarios.

GitHub for Windows
<https://windows.github.com>

GitHub for Mac
<https://mac.github.com>

Git distributions for Linux and POSIX systems are available on the official Git SCM web site.

Git for All Platforms
<http://git-scm.com>

CONFIGURE TOOLING

Configure user information for all local repositories

```
$ git config --global user.name "[name]"
```

Sets the name you want attached to your commit transactions

```
$ git config --global user.email "[email address]"
```

Sets the email you want attached to your commit transactions

```
$ git config --global color.ui auto
```

Enables helpful colorization of command line output

CREATE REPOSITORIES

Start a new repository or obtain one from an existing URL

```
$ git init [project-name]
```

Creates a new local repository with the specified name

```
$ git clone [url]
```

Downloads a project and its entire version history

MAKE CHANGES

Review edits and craft a commit transaction

```
$ git status
```

Lists all new or modified files to be committed

```
$ git diff
```

Shows file differences not yet staged

```
$ git add [file]
```

Snapshots the file in preparation for versioning

```
$ git diff --staged
```

Shows file differences between staging and the last file version

```
$ git reset [file]
```

Unstages the file, but preserve its contents

```
$ git commit -m "[descriptive message]"
```

Records file snapshots permanently in version history

GROUP CHANGES

Name a series of commits and combine completed efforts

```
$ git branch
```

Lists all local branches in the current repository

```
$ git branch [branch-name]
```

Creates a new branch

```
$ git checkout [branch-name]
```

Switches to the specified branch and updates the working directory

```
$ git merge [branch]
```

Combines the specified branch's history into the current branch

```
$ git branch -d [branch-name]
```

Deletes the specified branch

GIT CHEAT SHEET

REFACTOR FILENAMES

Relocate and remove versioned files

```
$ git rm [file]
```

Deletes the file from the working directory and stages the deletion

```
$ git rm --cached [file]
```

Removes the file from version control but preserves the file locally

```
$ git mv [file-original] [file-renamed]
```

Changes the file name and prepares it for commit

REVIEW HISTORY

Browse and inspect the evolution of project files

```
$ git log
```

Lists version history for the current branch

```
$ git log --follow [file]
```

Lists version history for a file, including renames

```
$ git diff [first-branch]...[second-branch]
```

Shows content differences between two branches

```
$ git show [commit]
```

Outputs metadata and content changes of the specified commit

SUPPRESS TRACKING

Exclude temporary files and paths

```
*.log
build/
temp/
```

A text file named .gitignore suppresses accidental versioning of files and paths matching the specified patterns

```
$ git ls-files --other --ignored --exclude-standard
```

Lists all ignored files in this project

SAVE FRAGMENTS

Shelve and restore incomplete changes

```
$ git stash
```

Temporarily stores all modified tracked files

```
$ git stash pop
```

Restores the most recently stashed files

```
$ git stash list
```

Lists all stashed changesets

```
$ git stash drop
```

Discards the most recently stashed changeset

REDO COMMITS

Erase mistakes and craft replacement history

```
$ git reset [commit]
```

Undoes all commits after [commit], preserving changes locally

```
$ git reset --hard [commit]
```

Discards all history and changes back to the specified commit

SYNCHRONIZE CHANGES

Register a repository bookmark and exchange version history

```
$ git fetch [bookmark]
```

Downloads all history from the repository bookmark

```
$ git merge [bookmark]/[branch]
```

Combines bookmark's branch into current local branch

```
$ git push [alias] [branch]
```

Uploads all local branch commits to GitHub

```
$ git pull
```

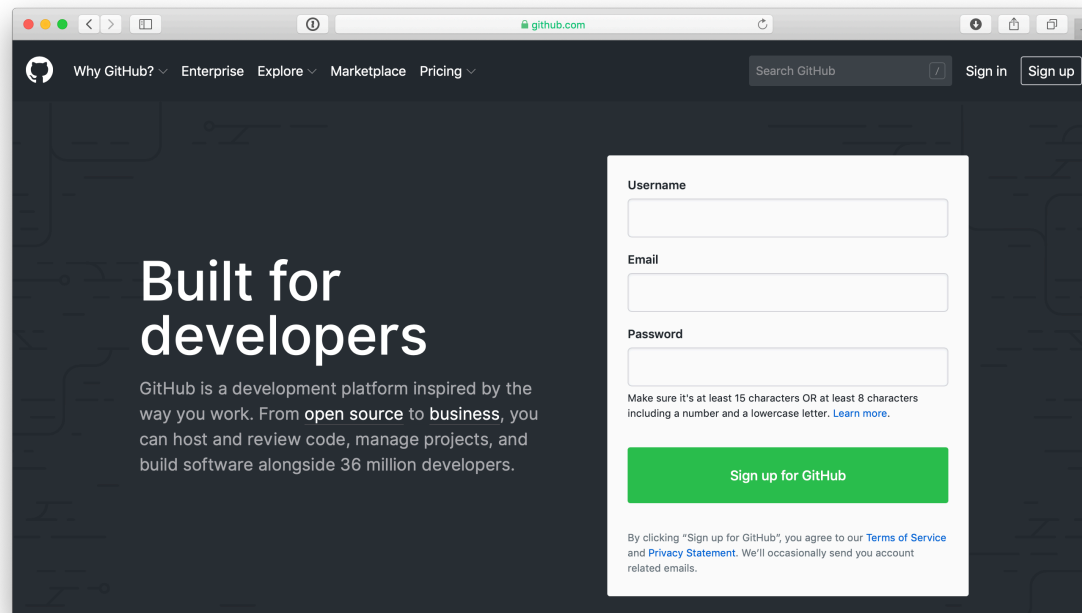
Downloads bookmark history and incorporates changes

GitHub Training

Learn more about using GitHub and Git. Email the Training Team or visit our website for learning event schedules and private class availability.

✉ training@github.com
 🌐 training.github.com

- Largest web-based git repository
- Hosting service
 - Aka, hosts 'remote repositories'
- Allows for code collaboration with anyone online
- Adds extra functionality on top of Git
 - UI, documentation, bug tracking, feature requests, pull requests, and more!



Demo