**Git – The Clone Wars**

**Prerequisites**

* Install git - <https://git-scm.com/downloads>
* Install DiffMerge - <https://sourcegear.com/diffmerge/downloads.php>
* Create GitHub account - <https://github.com/>
* Clone repository

Clone the repository using the following git command

git clone <https://github.com/molenici/TheCloneWars.git>

**Resources**

Here you can find the resources we will use during this hands-on session

****

**Introduction**

In this session we will simulate an Agile methodology. It consists of 4 sprints during which we will use a git flow to simulate a project.

**Git Flow**

The main branch will be the **master** branch. During each sprint’s development period we will work on the **develop** branch. The changes on the **develop** branch will be merge at the **end** of each sprint.

* **Creating task branch**

For each task, a new branch will be create from the **develop** branch.

For this, first we have to move from the **master** branch onto the **develop** branch

1. **git checkout develop**

**Name convention**

The branches will be named ts\_<drawing\_name>\_<drawing\_number>

1. **git checkout –b <task\_branch\_name>**

* **Commiting changes**

After finishing the task, a commit will be made with the changes

1. To add the changed files for the commit we will use the **git add** command

**git add .**  – this command adds all the changed files

Afterwards we will create a commit with the changed files using the following command

**git commit –m “commit message”**

To push the commit to the remote server the next command will be used

**git push origin <task\_branch\_name>**

* **Merging the commit into the develop branch**

In order to merge the task branch into develop, we must be sure that the possible conflicts are resolved. To do this, we will have to pull the new changes that may have been made on the develop side, bring them on our branch and resolve the possible conflicts

1. **git checkout develop**  (this command will move you on the **develop** branch)
2. **git pull --rebase** (pull the new changes that have been made on the develop branch)
3. **git checkout <task\_branch\_name>** (will move you back on the **task** branch)
4. **git rebase develop** (sync with the new changes on the **develop** brach)

During this step we will have to fix the conflicts if there are any

After fixing the conflicts, use the command **git rebase --continue** to finish the rebase

To push the updates to the remote server use **git push –f origin <task\_branch\_name>** (be wary the –f / --force flag can have destructive consequences if someone else made changes on the branch meanwhile)

1. **git checkout develop** (move back on the **develop** branch)

**git merge --no-ff <task\_branch\_name>** (merge the **task** branch into the **develop** branch)

1. **git push** (push the changes to the remote server)

* **Merging the changes made during the sprint into the master branch**

At the end of the sprint, we will merge the changes made on the **develop** branch into the **master** branch

1. **git checkout master** (move on the **master** branch)

**git merge --no-ff develop** (merge the **task** branch into the **develop** branch)

1. **git push** (push the changes to the remote server)

