GIT refresh

Version control in Districon Solutions

Share best practices of working with GIT

1. Explain the story of GIT
2. Describe guidelines/principles for Districon Solutions
3. Use offline
4. Review code changes
5. Show examples of GIT trees

1. Explain the story of GIT

- … a system that records changes to a file over time so you can recall specific versions later
- Instead of saving hundreds of copies like v1, v1a, _old, 181022_old_, etc.
- Districon first used GIT in the Nampak GOBSS project (Q1-2015)

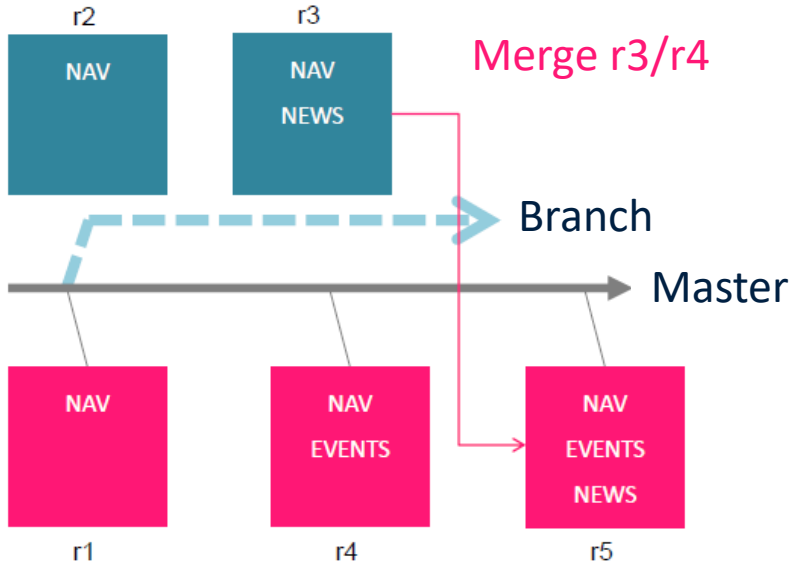
Main benefits

- Backup and restore
- Sync with multiple computers
- Work in a team
- Safely create and test new features

1. Explain the story of GIT

Vocabulary

- GIT = version control software (=> within Districon often used as synonym for version control)
- Repository = database where files are stored (=> Districon09, Martijn/Erik/Imke have admin access rights)
- Master = main trunk of GIT tree
- Head = latest revision in the repository
- Origin = refers to a version pulled from the central (server) repository
- Pull = retrieve all commits from central repository
- Push = store local changes to central repository
- Revert/stash = throw away your working copy and restore last revision
- Branch = duplicate copy of code used for a feature/development
- Merge = integrate changes from two different branches
- Resolve = manual fix conflicts in a merge action



2. Describe guidelines/principles for Districon Solutions

- Fix two bugs => two separate commits
- Make small commits
  - Easier to understand
  - Easier to roll back
  - Less conflicts
- Commit/update often
- Write **good** commit messages (see next slide)

But… do not commit

- Broken code
- Something that is half done (split in smaller chunks)
- Untested work

2. Describe guidelines/principles for Districon Solutions

- "A diff will tell you *what* changed, but only the commit message can properly tell you *why*."
  - (http://who-t.blogspot.co.at/2009/12/on-commit-messages.html)

Answer three questions in a **good** commit message:

- Why it is necessary? E.g. bug-fix, add feature, improve performance, etc.
- How does it address the issue? Explain the approach.
- What effects does the patch have? E.g. side effects, etc.

Proposed 'standard' way of working for Districon Solutions:

- [Prefix] to denote type of commit; where we distinguish 'bugfix', 'new feature', 'restructure', 'merge'
- After prefix describe the subject in the first line
- In the body <after new line> write the approach and effects
- Use tags to refer to deployed versions at the customer, like **GOBSS 1.8.3**

3. Use offline

- Check out same (master) version before starting separate developments
- Copy to a USB stick and paste on local laptops

One user should execute all GIT actions

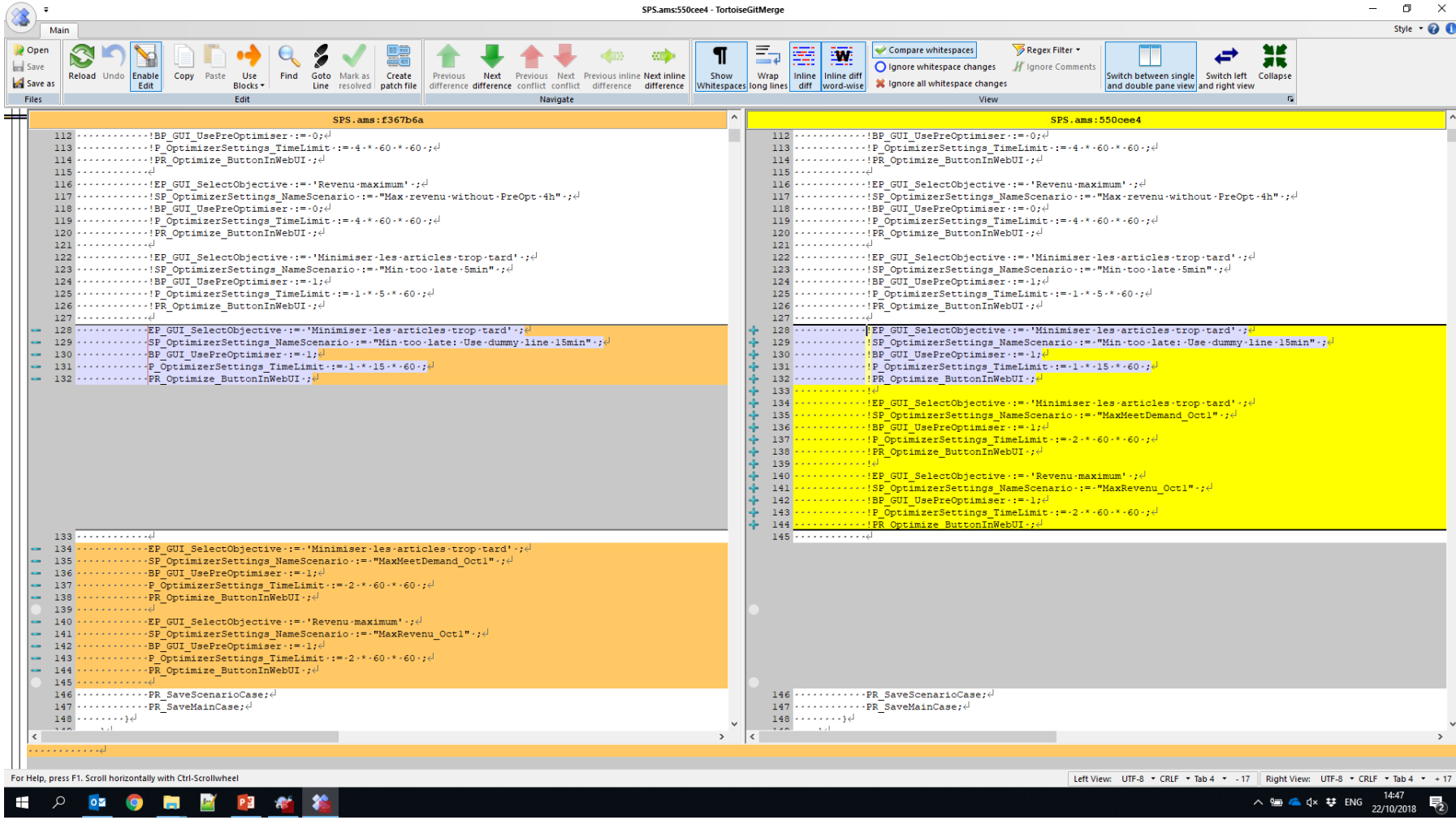
- Go back to start version, create branches for all developers, e.g. their 3-digit abbreviations TBU, IGE, KHU, EHO
- If someone does a ‘commit’, copy/paste using USB in the developers’ branch
- Redo until all features of all developers are part of the local GIT tree
- Merge all commits to a new master version (to be pushed if network access is available again)
- All local developments can be removed/deleted as they are part of the pushed revisions in the master commit

Warning: Please make sure all USB sticks are formatted after the offline work cycle

Workaround method if Districon server is not accessible or if pull/push is too slow

4. Review of code changes

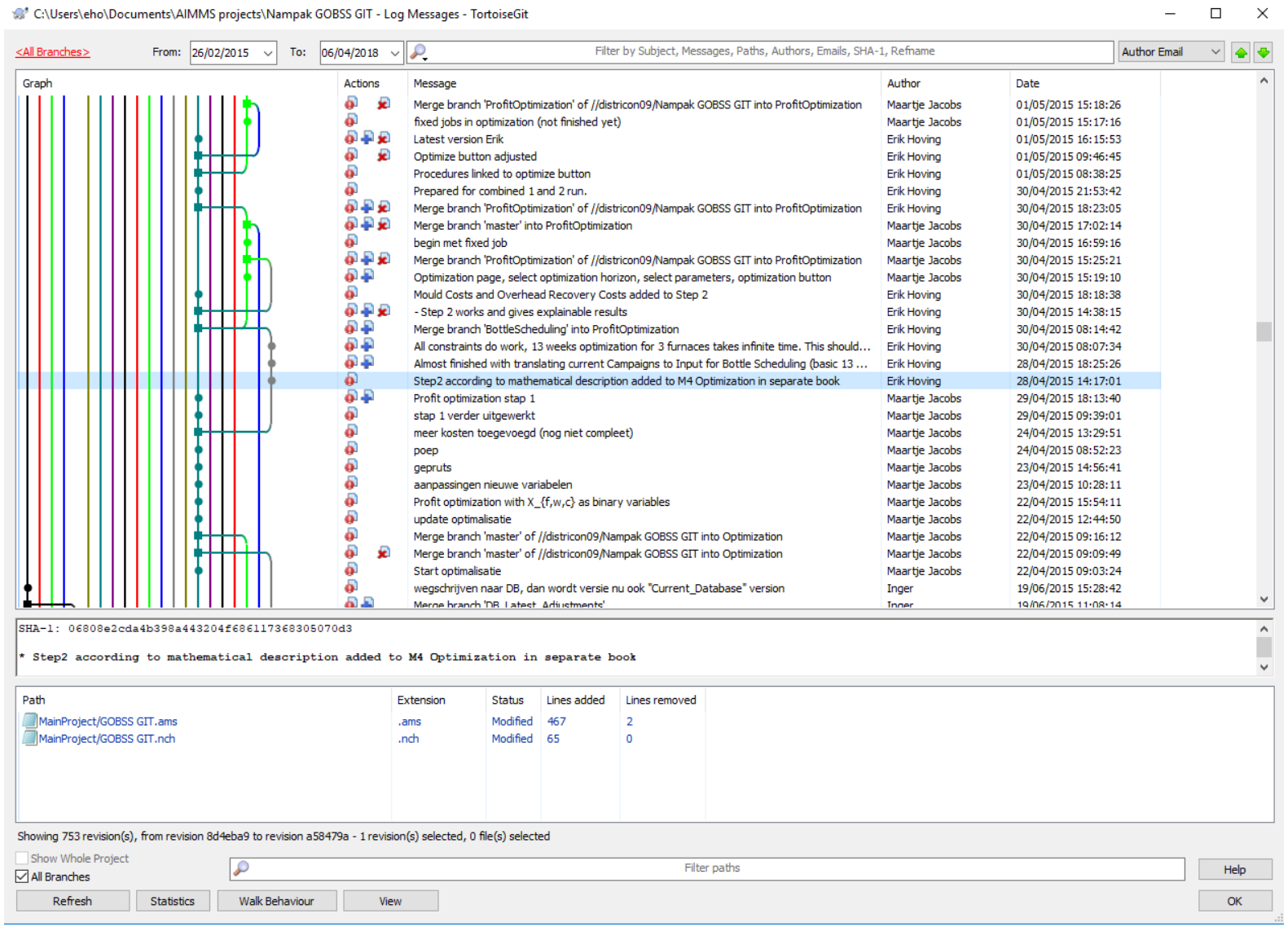
- Open **GIT Show Log**
- Select a revision/commit
- Check below which files are added/deleted/modified
- Double-click a modified file to review code changes



5. Show examples of GIT trees

Nampak GOBSS

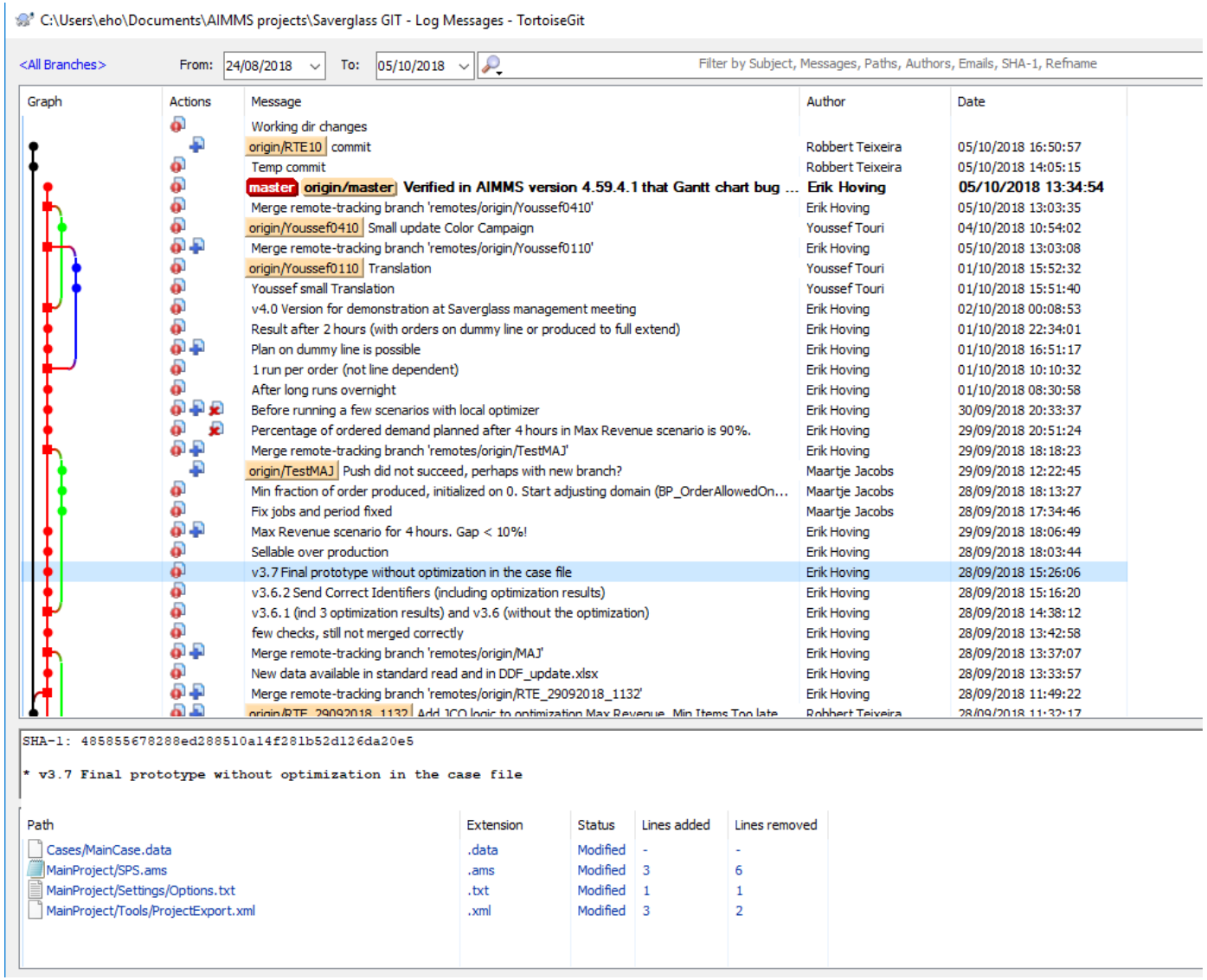
- First Districon GIT project
- Large commits
- Too many commits
- Messages not useful for colleagues



5. Show examples of GIT trees

Saverglass

- Version names in message
- Commit messages can still be improved



5. Show examples of GIT trees

Aurubis

- Use of tags
- Refer to issue in branch names ‘SES-211’
- Frequent, small commits