How to manage software development project

Valdis Vītoliņš

FuturICT 2.0

January 11, 2019

FuturICT 2.0

FLAG-ERA

About me

- I'm IT freelancer and work as developer and systems administrator.
I conduct Java Bootcamps for Accenture Latvia and maintain e-government portal pakalpojumi.carnikava.lv and websites: odo.lv, ante.lv, silvita.lv, dudajevagatve.lv
- I have worked for international IT companies Exigen, Accenture, E-Global Trade & Finance Group as developer, project manager, tester and configuration manager.
- I have contributed to the following open source projects:
  - As a developer to: eSpeak NG, MBROLA
  - As a translator to: XWiki, GnuCash, Vim, Openbravo

Don't manage

- Just write damn code
- A.k.a “Code-and-Fix”, a.k.a. “Cowboy coding”, a.k.a. “Hacking”, a.k.a “I don't lose time for planning”
- This model starts with an informal general product idea and just develops code until a product is ”ready” (or money, time or interest runs out). Work is in random order.
- Works for solo projects
- ***Doesn't work*** non-trivial collaborative software development projects

Why this happens?

Dealing with Brook's law

- "Adding manpower to a late software project makes it later"

–Fred Brooks, 1975 "The Mythical Man-Month"

- It takes some time for the people added to a project to become productive;

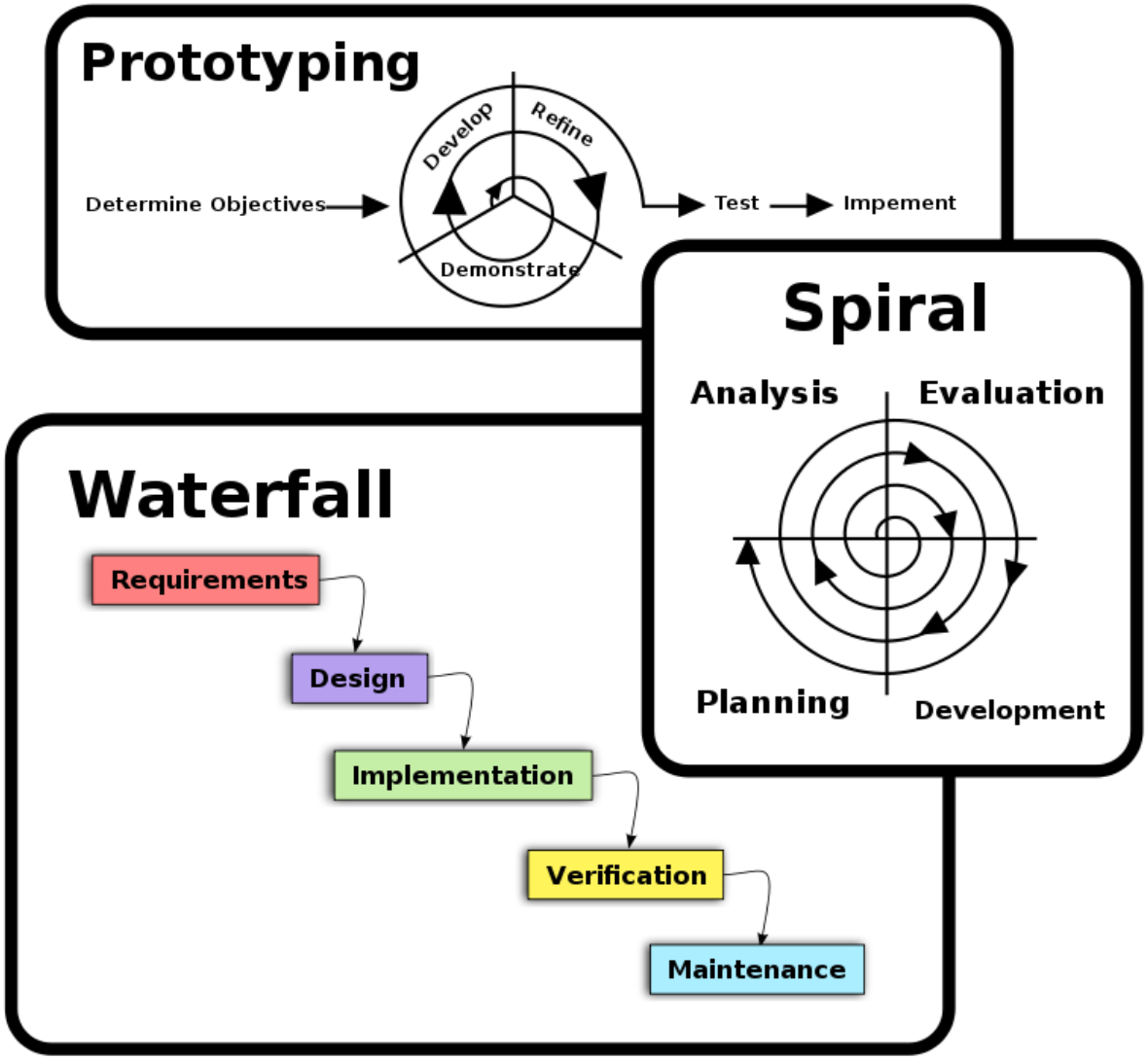
- As more people work on project:

–it is **hard to split up tasks** in parallel, independent streams,

–**communication and management overheads** increase by ~N^2, but productivity only by ≤N.

–strict **configuration management** is necessary to avoid “It works on my machine!” issues

Main software development methodologies



Prototyping

- Requirement gathering is difficult

–software is developed because the present situation is unsatisfactory

–however, the desirable new situation is as yet unknown

- Prototyping is used to obtain the requirements of some aspects of the system

- Prototyping should be a relatively cheap process

–use rapid prototyping languages and tools

–not all functionality needs to be implemented

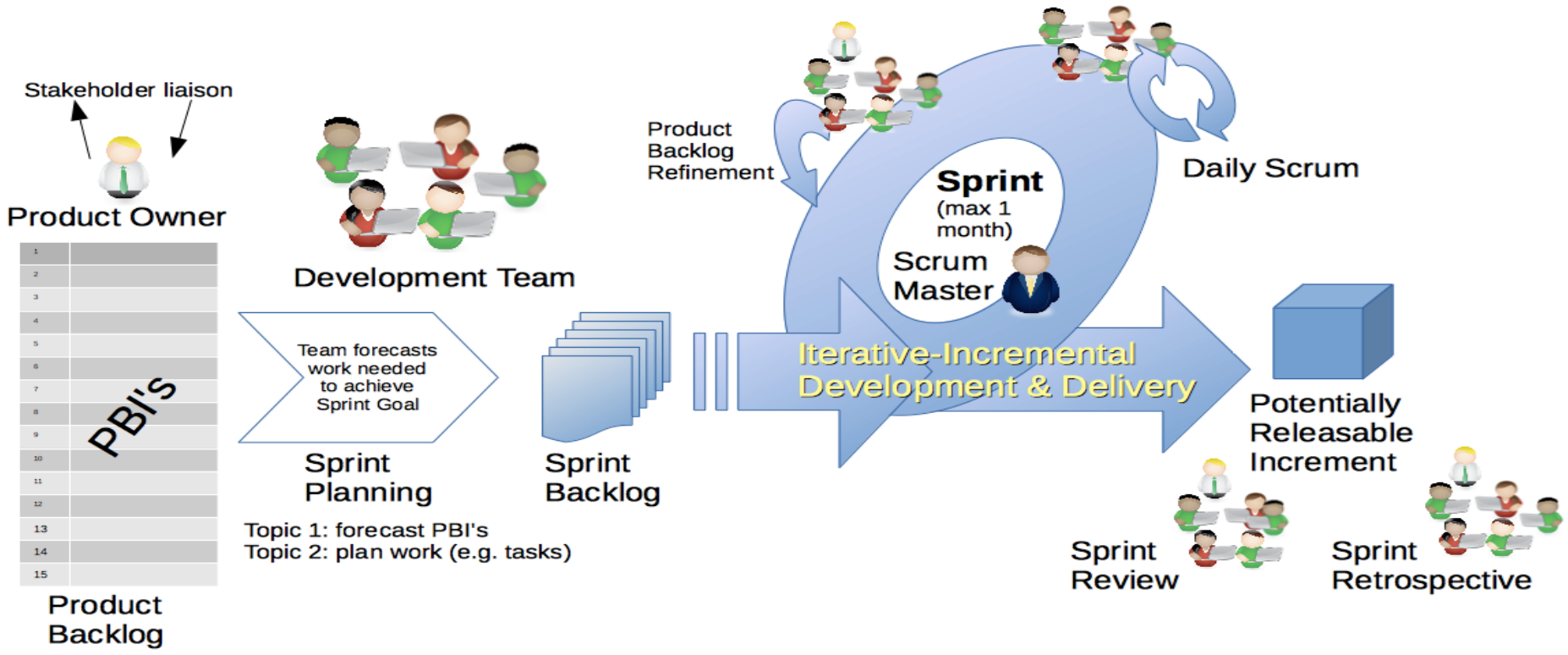
–production quality is not required

Agile principles

- Active user involvement is imperative
- The team must be empowered to make decisions
- Requirements evolve but the timescale is fixed
- Capture requirements at a high level; lightweight & visual
- Develop small, incremental releases and iterate
- Focus on frequent delivery of products
- Complete each feature before moving on to the next
- Apply the 80/20 rule
- Testing is integrated throughout the project life-cycle –

Scrum

- Scrum is an iterative and incremental agile software development framework
- A flexible, holistic product development strategy
- Development team works as an atomic unit



Scrum roles

- Product Owner — holds the vision for the product
- Scrum Master — helps the team best use Scrum to build the product
  - In discussion with team prepare Sprint backlog
  - Provides resources needed to complete tasks
- Development team — builds the product
  - Developer
  - Tester/quality assurer
  - Configuration manager

Scrum artifacts

- Product increment — an integrated, shippable subset of the product
- Product backlog — the list of ideas for the product, in order of priority
- Sprint backlog — (i.e. issue list) the detailed plan for development during the next sprint

–team members “pull” (i.e. take) new issue, when previous issue is done

–nobody “pushes” (i.e. orders to do) any task to the members

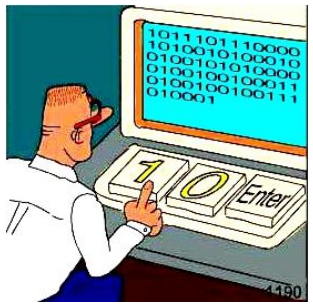
Scrum activities

- At the start of scrum:
  - Product backlog refinement
  - Sprint planning
- During scrum:
  - Daily Scrum
- There are no other interruptions during day for team members
- At the end of scrum:
  - Sprint review
  - Sprint retrospective

How to manage shared code?

Germany

USA



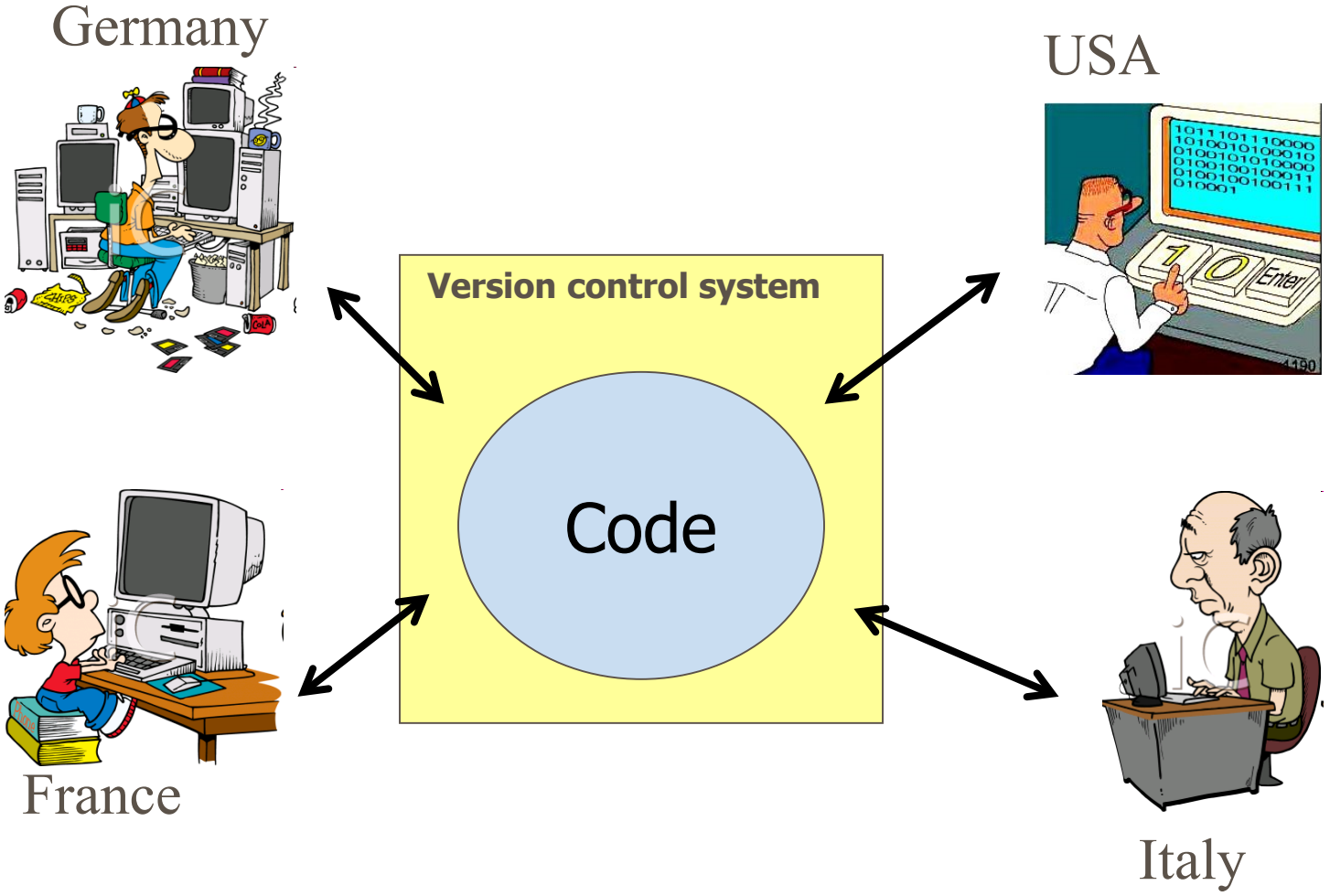
DropBox?
Google drive?
ShareFile?

France

Italy

Professional code sharing approach

Version Control (VC) system



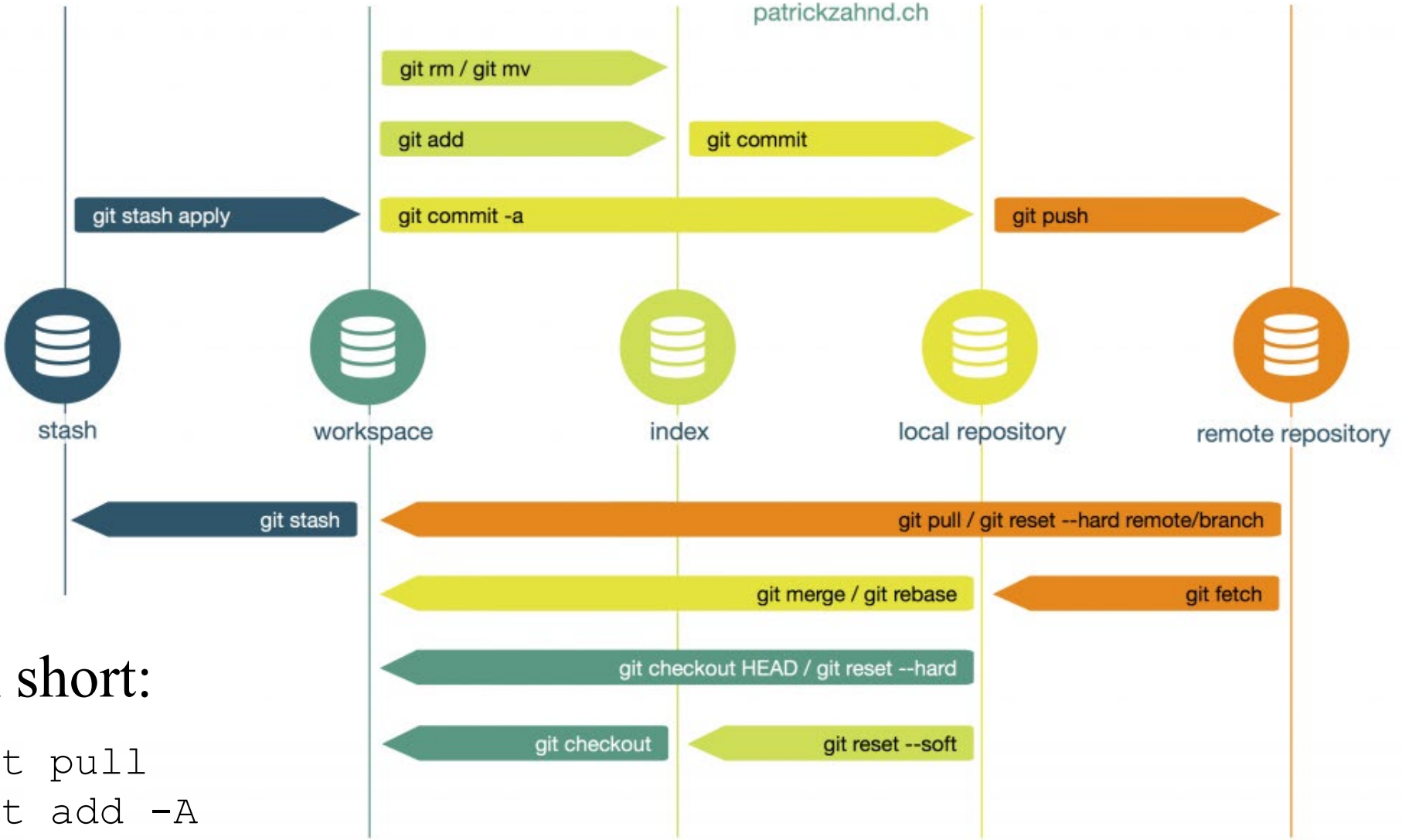
Why version control

- Need to develop non-trivial applications
- More than one developer work with the same file set
  - (***But usually not with the same file!***)
- Automated "snapshots/backups" (revisions)
- Can "move in time" (versions)
- Can do "parralel universes" (branches)

What is Git and what is GitHub

- Git is the leading open source distributed version control system
  - Developed by Linus Torvalds (a.k.a. developer of Linux kernel)
  - Strongly oriented on fast, parallel and non-linear development
  - Is used for largest open- and closed-source software projects in the world:
- Microsoft Windows operating system
- Linux kernel
- GNU Emacs text editor
- GitHub is the most popular web site to host open source

Most important Git commands



In short:

```
git pull
git add -A
git commit -m "Your message here"
git push
```

Project Hosting Sites

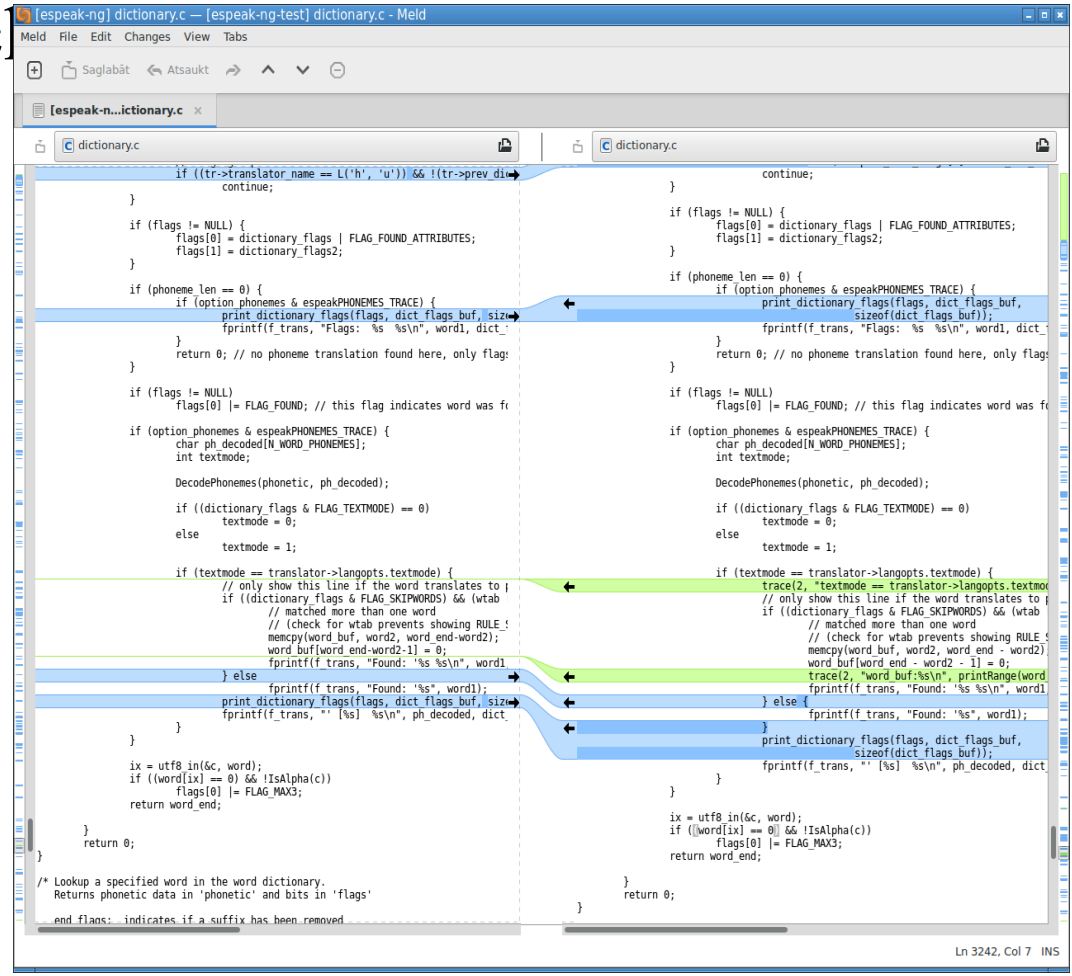
- GitHub
  - The #1 project hosting site in the world
  - Git repositories, issue management, wiki, on-line documentation in MD syntax
  - Free for public projects
  - Paid service for private projects
- GitLab – if you don't like GitHub
  - Git repositories, issue management, wiki, on-line documentation in MD syntax, built in CI
  - Free for both — public and private projects
- Sourceforge – “GitHub for old people”

Dealing with conflicts by avoiding them

- Conflicts happen, when:
  - **The same file is changed by several persons** *paralelly in the same branch* (e.g. by applying different formatting to the file)
  - **The same file is changed by several persons** in *different branches* and somebody has to *merge branches*
- Don't allow conflicts to appear:
  - **Do file dividing**
    agree who do changes on what files (e.g. split by functionality) and don't touch files of others
  - **Do time dividing**

If conflicts appeared, do...

- Create two (or even three) copies of the project, e.g.:
  - Checkout different commits
  - Checkout different branch
- Compare copies with comparison tools, e.g. *meld, diffuse, WinDiff or WinMerge*
- Apply differences in comparison tool and add them as new



…or simple approach

https://m.xkcd.com/1597/

Issue management and information sharing

- Use web based solutions whenever possible
  - Solution should follow web standards
  - Import/export to stand alone document format is important
- Free, simple solutions for online sharing:
  - Google Docs
    - Web based office documents
    - can do authentication and user access management
  - Etherpad
    - Simle wiki (rich text format) page
    - easy information sharing with anonymous users

How to start a project?

- Collect and categorize business requirements
- Determine system requirements for:
  - hardware (if necessary)
  - software
- Prepare development environment…
  - with needed hardware (e.g. mobile phone)
  - set up necessary IDE (Integrated Development Environment)
  - set up version control system
  - *…with the same settings for all team members!*
- Set up collaboration tools:
  - issue management system

Finally, some project lifetime metrics

Expectations

Feelings