Version Control Systems: Subversion

Miguel A. Figueroa Villanueva
Xabriel J. Collazo Mojica

Outline

- Introduction
  - Document management
    - CMS
    - Wiki
    - Aigaion
    - Code and Document Repositories
- Version Control Systems
  - Centralized
  - Distributed
- Unison Synchronization

Document Management

- Content Management System (CMS)
  - Manage content:
    - Documents
    - Multimedia files: images, videos, etc.
    - Web content in general
  - Basic principles:
    - Separation of content and presentation
    - Intuitive web-based interface for uploading, editing, and managing content
    - Foster collaboration among teams
- Wikipedia
  - Articles written collaboratively by volunteers around the world.
  - Most content can be edited by anyone.
- Difference between wiki and blog?
- Aigaion - http://www.aigaion.nl
  - A Web based bibliography management system.

Document Management

- Code and Document Repositories: Version Control Systems (VCS)
  - You might already have it:
    - Resume200708.doc
    - Resume200801.doc
    - Etc.
  - Or maybe something more sophisticated:
    - Proyecto_ver1
    - Proyecto_final
    - Proyecto_final2
  - This doesn't work!! Not for software projects, at least…

Version Control Systems (VCS)

- **Backup and Restore** – You backup your files in a repository and checkout a working copy.
- **Synchronization** – Share files with team members; by more sophisticated methods than e-mail and a pen drive!
- **Track ownership** – Files are tagged with log messages and author/developer IDs.
- **Track changes** – You have a history of your files with log messages.
- **Undo changes** – Go back in time to any point!
- **Branch and merge** – Create a sandbox to work in without interrupting your teams work.

Version Control Systems (VCS)

- Centralized VCS
  - CVS
  - SVN – Subversion: “CVS done right”.
  - Perforce
  - etc.
- Distributed VCS
  - SVK
  - Git
  - Mercurial
  - etc.

Centralized VCS – Basic Interaction

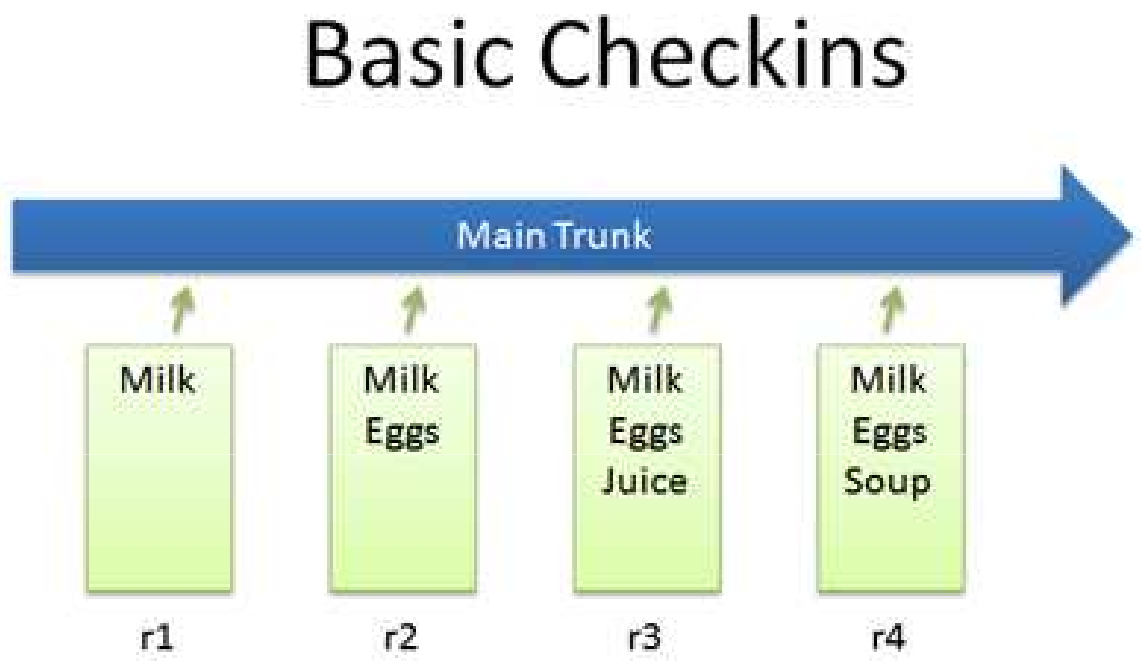
- Juan and Elena want to work on the database project, but they don't want to e-mail / pen drive each other every change.
- They decide to use a Centralized VCS.

- Juan "imports" the first version. Now a common copy is available for all users.
- Elena "checks out" a working copy. She makes changes and "commits".
- Juan makes changes too, but before "committing", he "updates" his working copy.
- Bosses (aka Nayda and Fernando) don't do work (as usual ☺), but they are always watching progress. So, he makes a "checkout" and "updates" frequently.

Centralized VCS – Basic Interaction

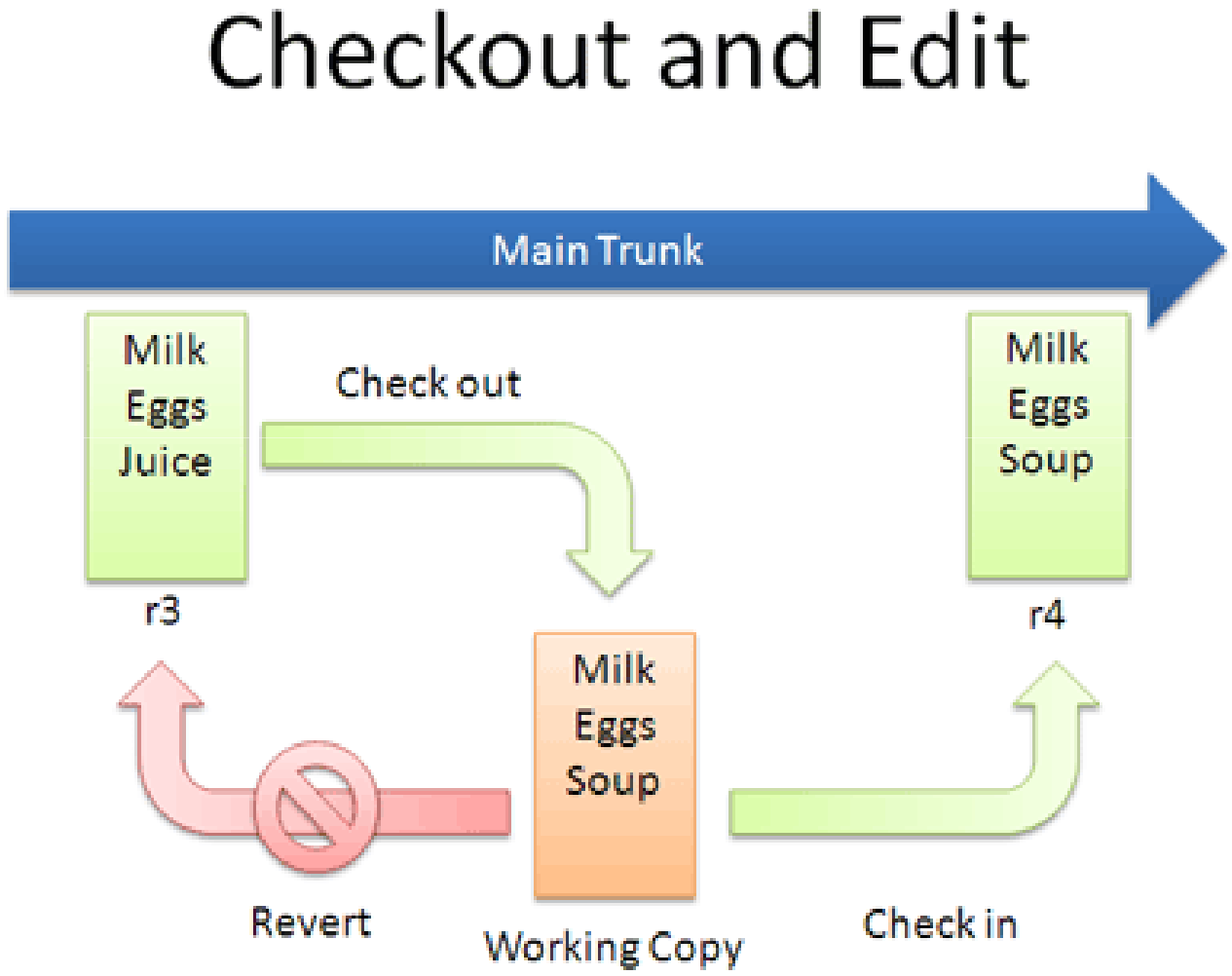
- Import – *Put a project in Version Control for the first time.*
- Check-out – *Get a "working copy" from the repo.*
- Commit = Check-in – *Put changes on the repo.*
- Update – *Get latest changes from the repo.*

Centralized VCS – Basic Scenario

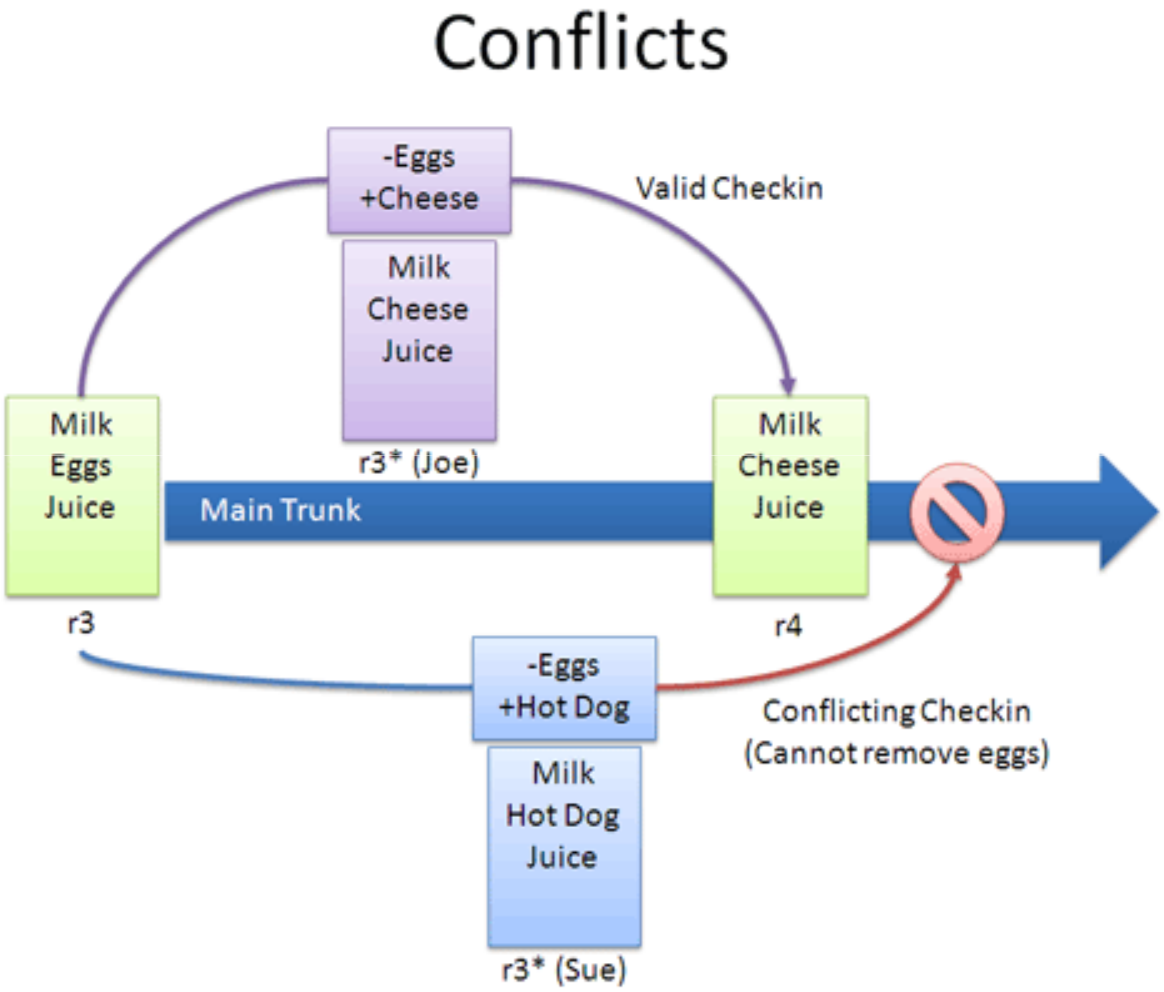


- Suppose we have a file list.txt
- The image represents four revisions checked in.

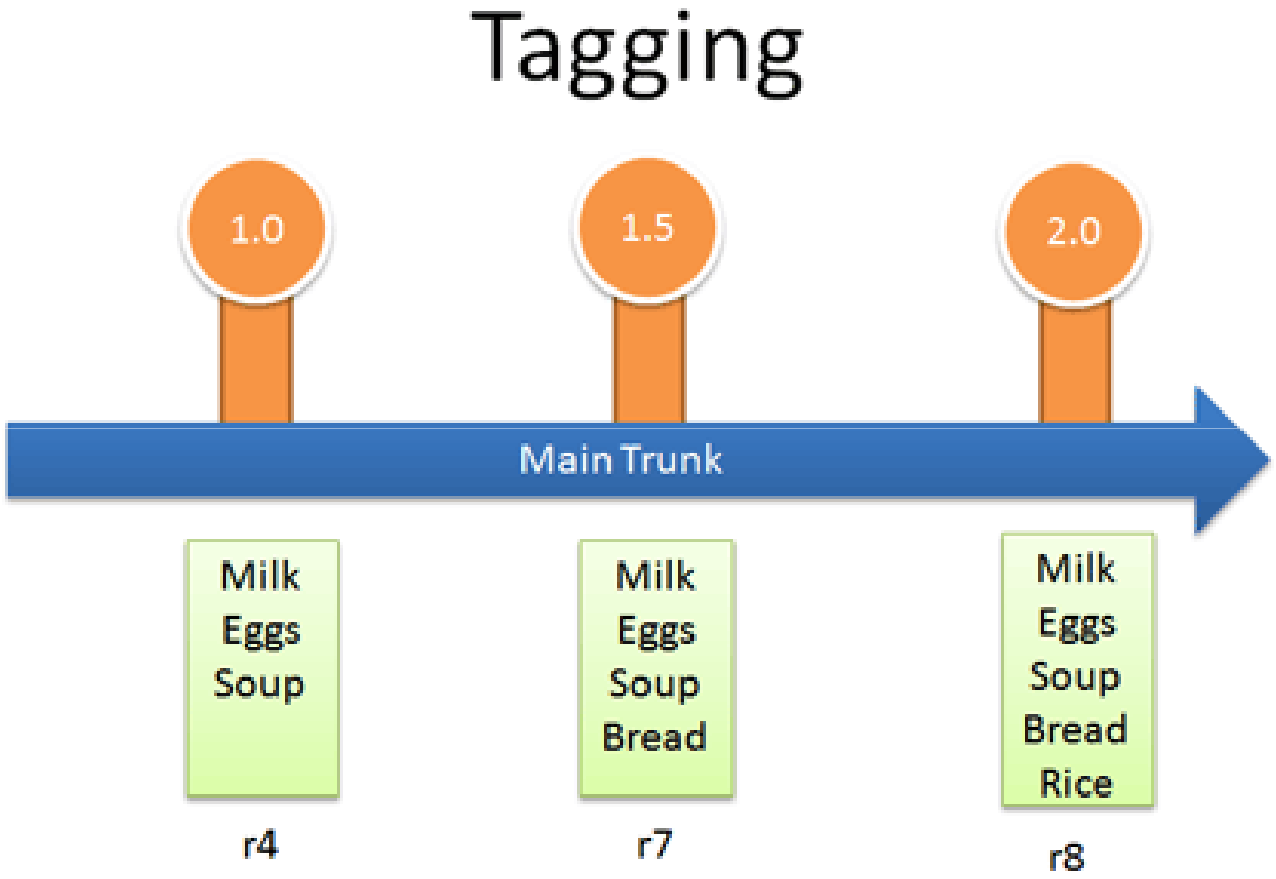
Centralized VCS – Basic Work Cycle



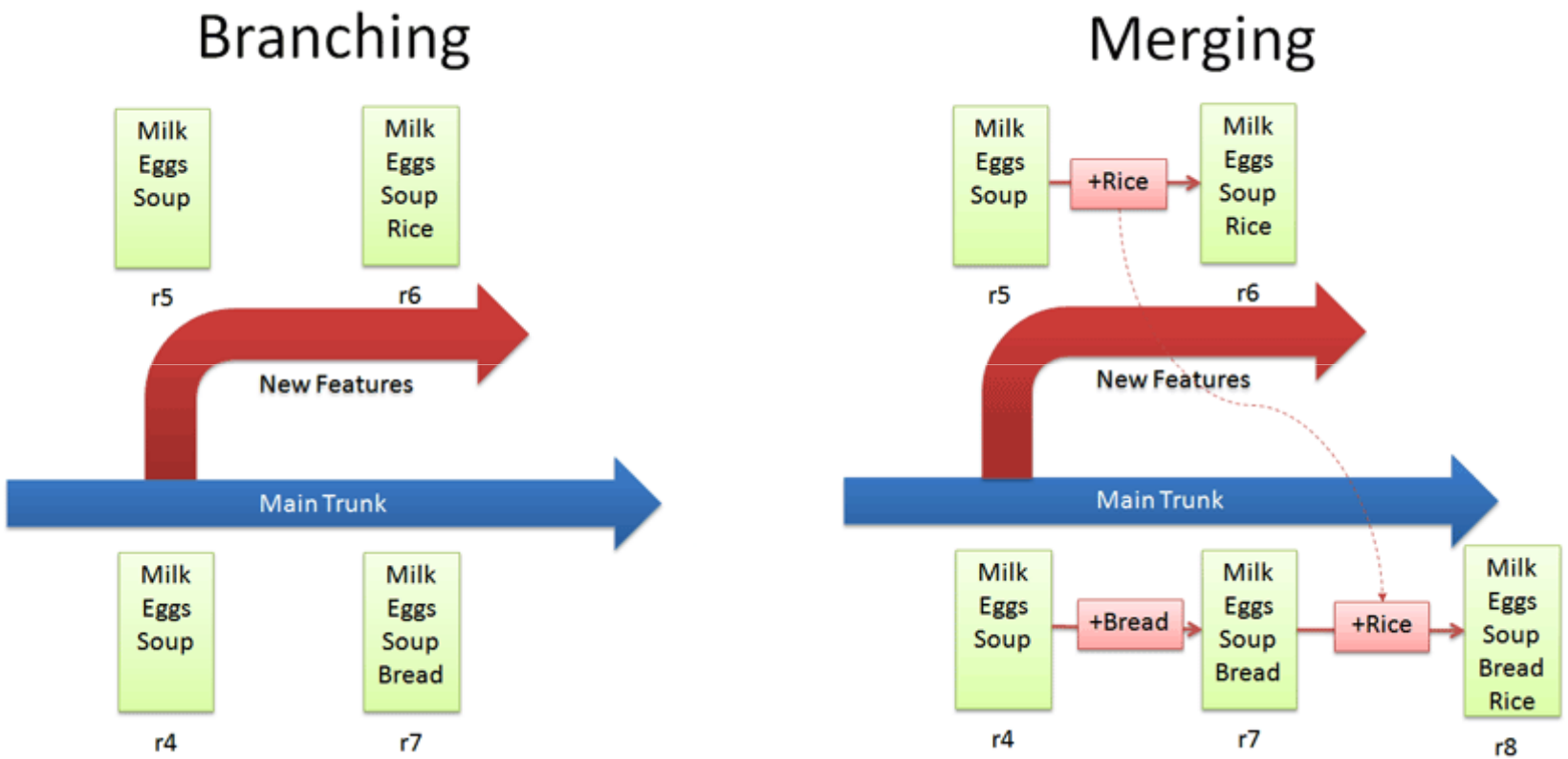
Centralized VCS – Basic Work Cycle



Centralized VCS – Basic Work Cycle



Centralized VCS – Advanced Work Cycle



Subversion: Tools and Resources

- TortoiseSVN - http://tortoisesvn.tigris.org
  - Easy to use svn GUI software for windoze.
- Subclipse - http://subclipse.tigris.org/install.html
  - Eclipse SVN Plugin by developers of subversion.

- Version Control with Subversion - http://svnbook.red-bean.com
  - Free on-line book by O'Reilly

Subversion: Experimental Repo Setup

- ICOM 5047 Subversion access (Remote URL):

  `https://capstone.ece.uprm.edu/svnroot/icom5047/ss08/<project>`

- Will follow conventional structure:

```
/project/trunk
        /branches
        /tags
```

- Go to the following address to make the request:

  `https://cga.ece.uprm.edu/admin/send_password_form.html`

  - Requested resource field:

    "capstone repo: <desired_repo_name>"

  - Reference field:

    e.g., "Nayda Santiago" or "Fernando Vega" or "Manuel Rodríguez"

Subversion: Best Practices

- Always update BEFORE committing.
- If after updating there are conflicts, its YOUR duty to solve them.
- Always commit WORKING code.
  - NEVER break the code in the trunk.
  - Use branches for long term development cycles that will break the code.
- Always append COMMENTS (log messages) to your commits.

- Inspect what you are committing. Please remember NEVER to include auto-generated files, such as *.class binaries.

Distributed VCS (DVCS)

- New trend in VCS
- Many distributed VCS server/clients.
- Linus Torvalds - http://www.youtube.com/watch?v=4XpnKHJAok8
  - Very biased, opinionated, and demeaning perspective about non-DVCS users, but useful information never the less.

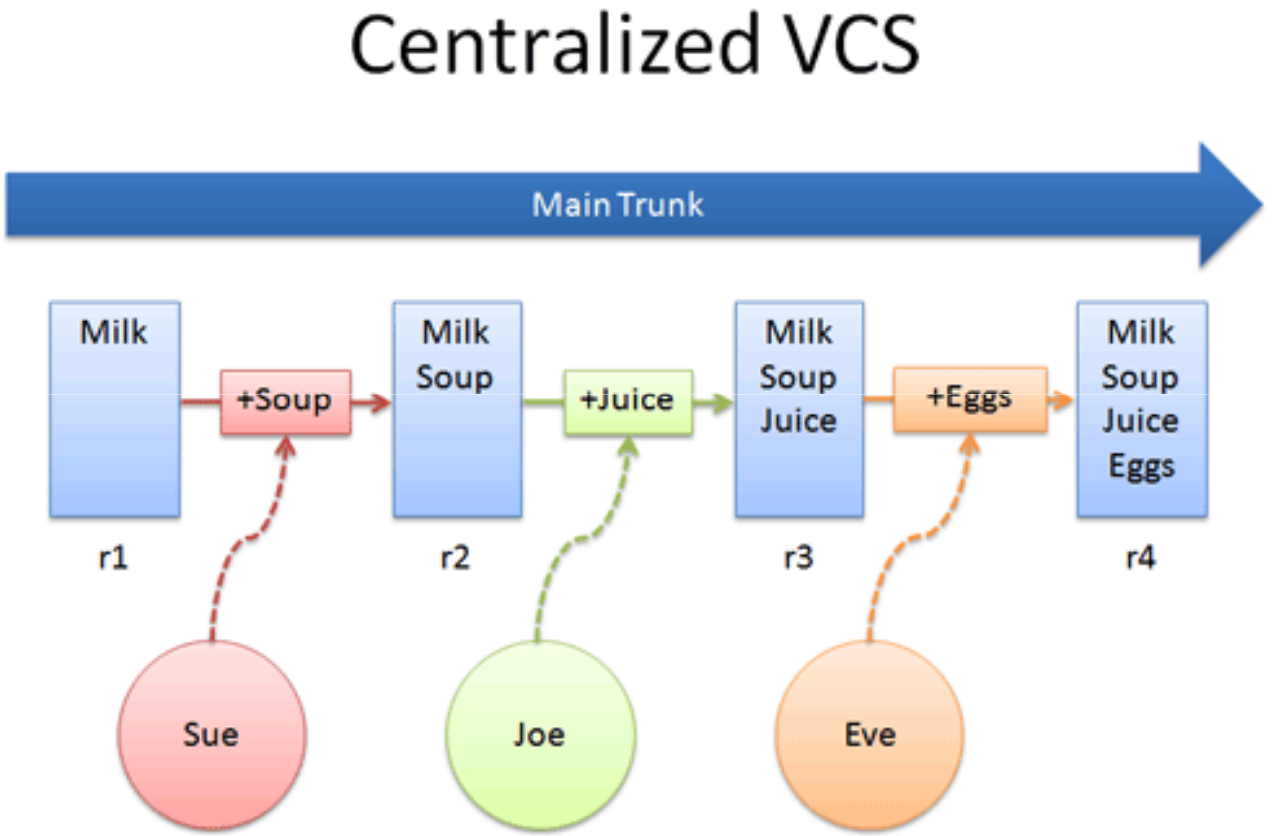
VCS vs DVCS – Differences

- Pros
  - Allows disconnected (off-line) operations with the repository, since every client contains a copy of the entire repository.
  - Branches are natural and don't have to be published.
  - Merge algorithms are more robust.

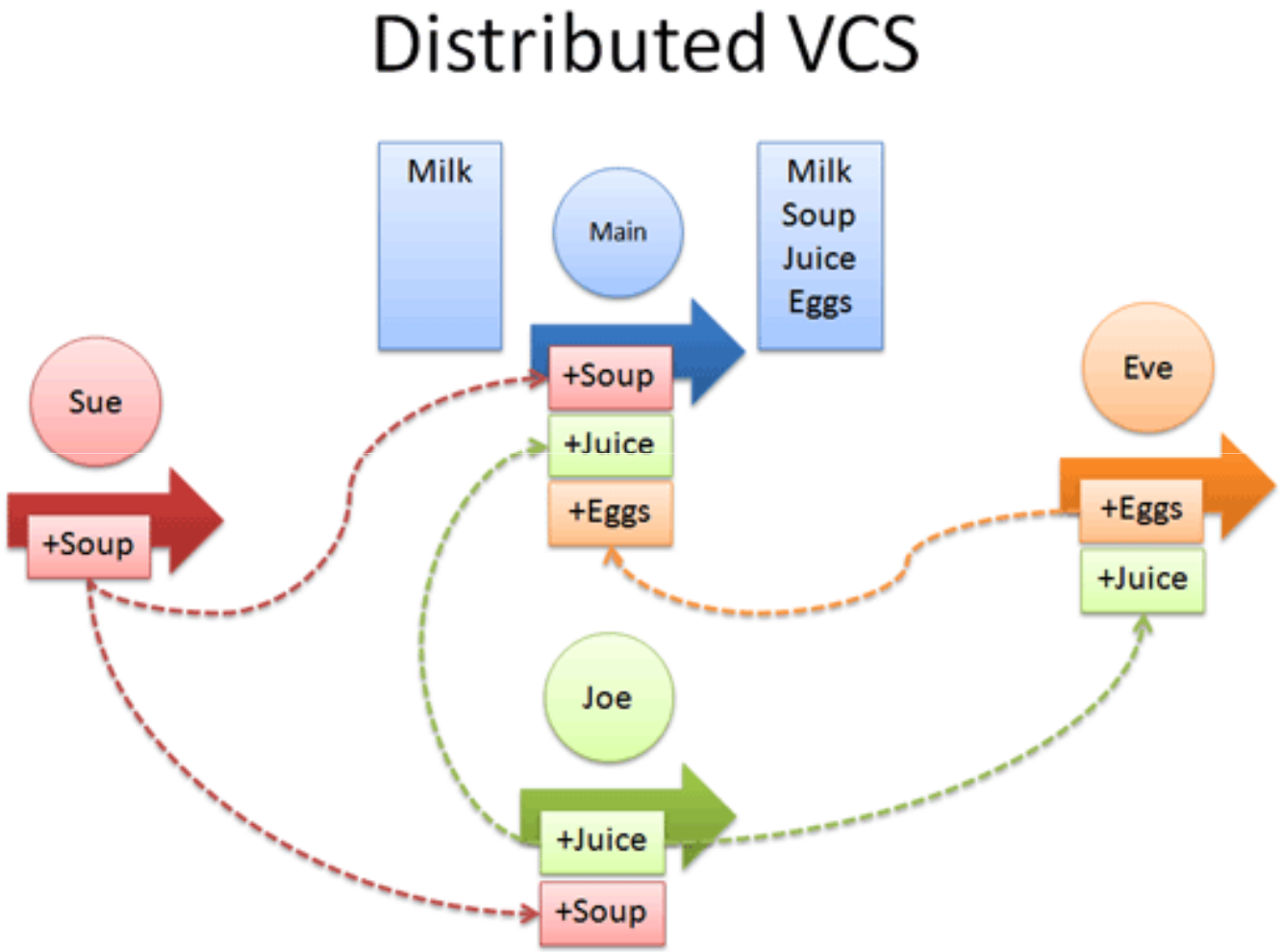
- Cons
  - Developers can run off and hide for too long…
    - then merging becomes overwhelmingly difficult.
  - It is not centralized! Who has the official version of the repository?
  - Graphical tools (GUIs) are still lacking a bit.

- A compromise: Benefit from both models using DVCS tools.
  - You can implement a good policy that enforces a centralized model and frequent commits, while taking advantage of the DVCS distributed tools flexibility.

VCS vs DVCS – Differences



VCS vs DVCS – Differences

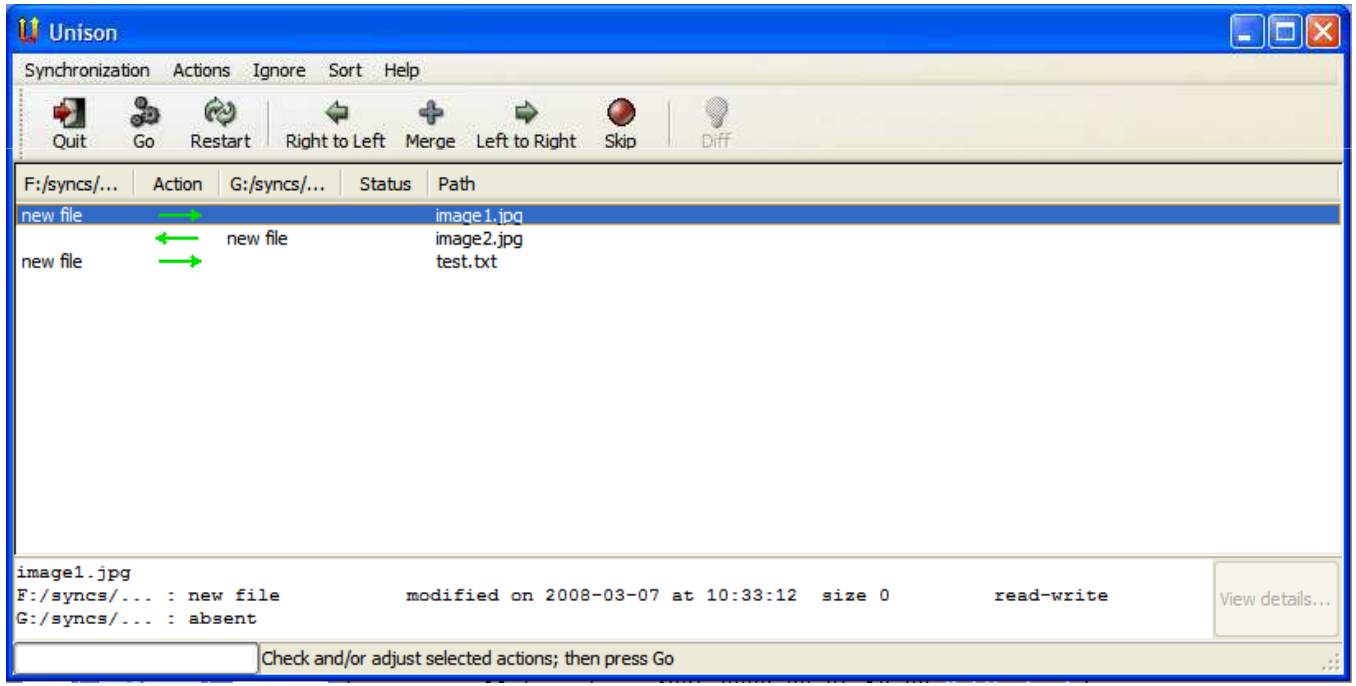


Miguel A. Figueroa Villanueva
Electrical and Computer Engineering Department
University of Puerto Rico – Mayagüez Campus

Photo taken from: Better Explained [1]

Unison Synchronization

- Not everything worth backing up is worth saving with history…
- Unison - http://www.cis.upenn.edu/~bcpierce/unison
  - File-synchronization tool for Unix and Windows.
  - Allows two replicas of a collection of files and directories to be stored on different hosts (or different disks on the same host), modified separately, and then brought up to date by propagating the changes in each replica to the other.



Conclusions

- Some tools where presented to help manage your documents and code files.
- You should start using software such as these to prevent chaos and disaster. It should be second nature after a while and very rewarding.
- Everyone here is required to have at least there code in a Subversion repository, which will be created when requested through the form presented previously.

References

[1] Better Explained

- http://betterexplained.com/articles/a-visual-guide-to-version-control
- http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated

Questions?