

Groupware and Collaborative Interaction
Collaborative Software Development

M2R Interaction / Université Paris-Sud / 2019 - 2020

Cédric Fleury (cedric.fleury@lri.fr)

Software development

Several users work on a same project

- Remote or collocated users

- Each one works on its own computer (asynchronous)

 - Work on different tasks

 - Work at different times

Collaboration is hard to organize

- Versioning, synchronization between users

- Tasks distribution, social aspects

Outline

Collaborative software development

Version control

Continuous integration

Software development methods

Outline

Collaborative software development

Version control

Continuous integration

Software development methods

Version control

Problematic

We want to avoid:

Manually share the files (USB key, email, Dropbox)

Delete or overwrite the files of other users

Broke all the project by making a mistake

We want to able to:

Edit the project at the same time

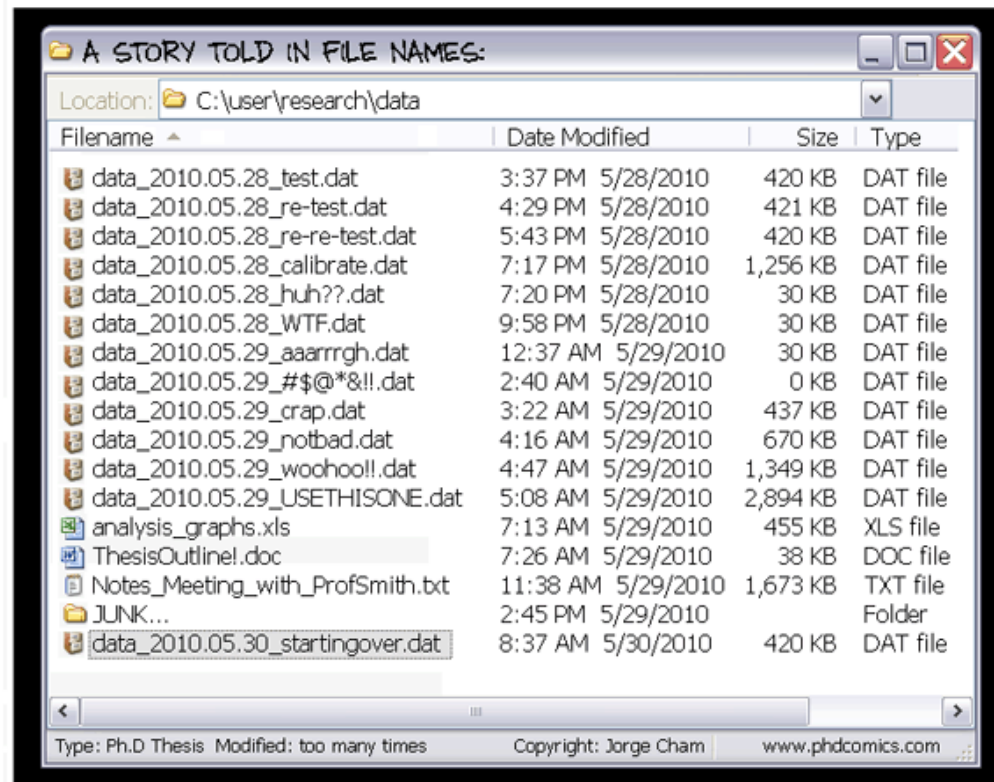
Keep an history of the modification

Keep the older version of the files + hierarchy

Version control

Problematic

We want to avoid this:



["Piled Higher and Deeper" by Jorge Cham: www.phdcomics.com]

Version control

Version control software

- Save & restore different versions of the files

- Synchronize users' versions

- Keep track of modifications and their authors

- Manage Branching and merging

Not only for software development

- Report, images, data from experiments

Version control

2 kinds of architecture

Centralized

CVS, SVN, ...

Decentralized (peer-to-peer):

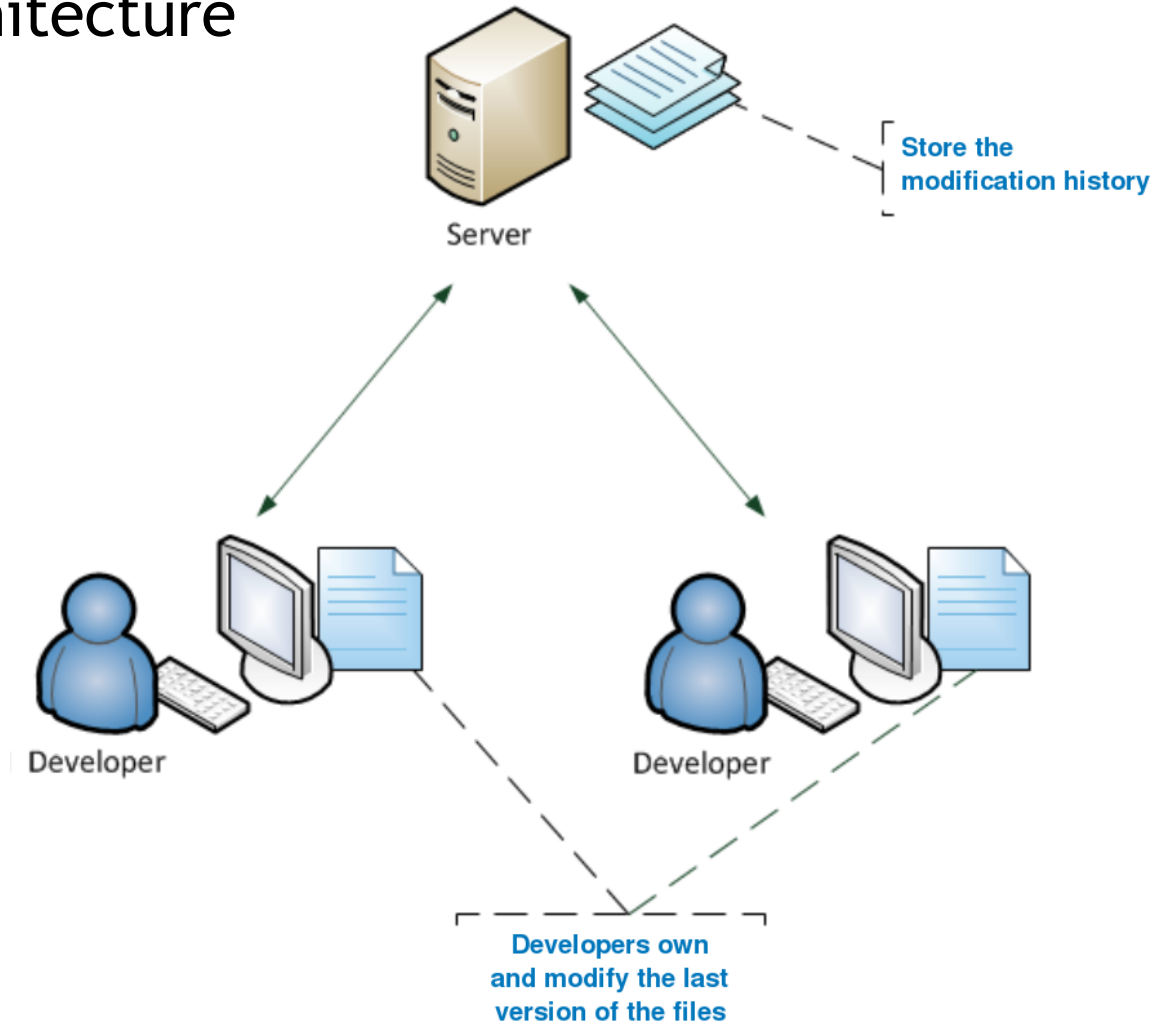
GNU Arch, Mercurial, Bazaar, Git,...

Decentralized can be used as a Hybrid Architecture

One peer can be a central server

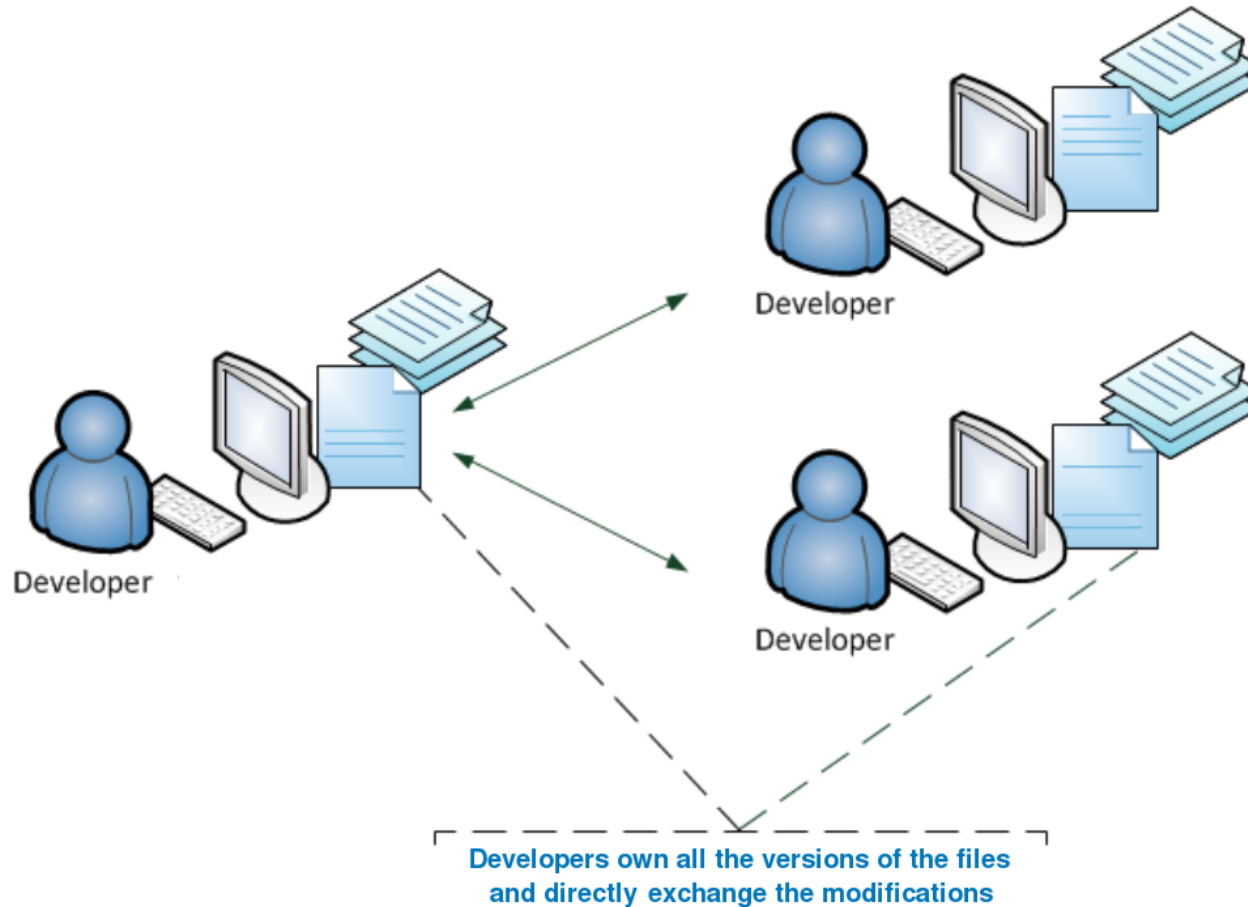
Version control

Centralized architecture



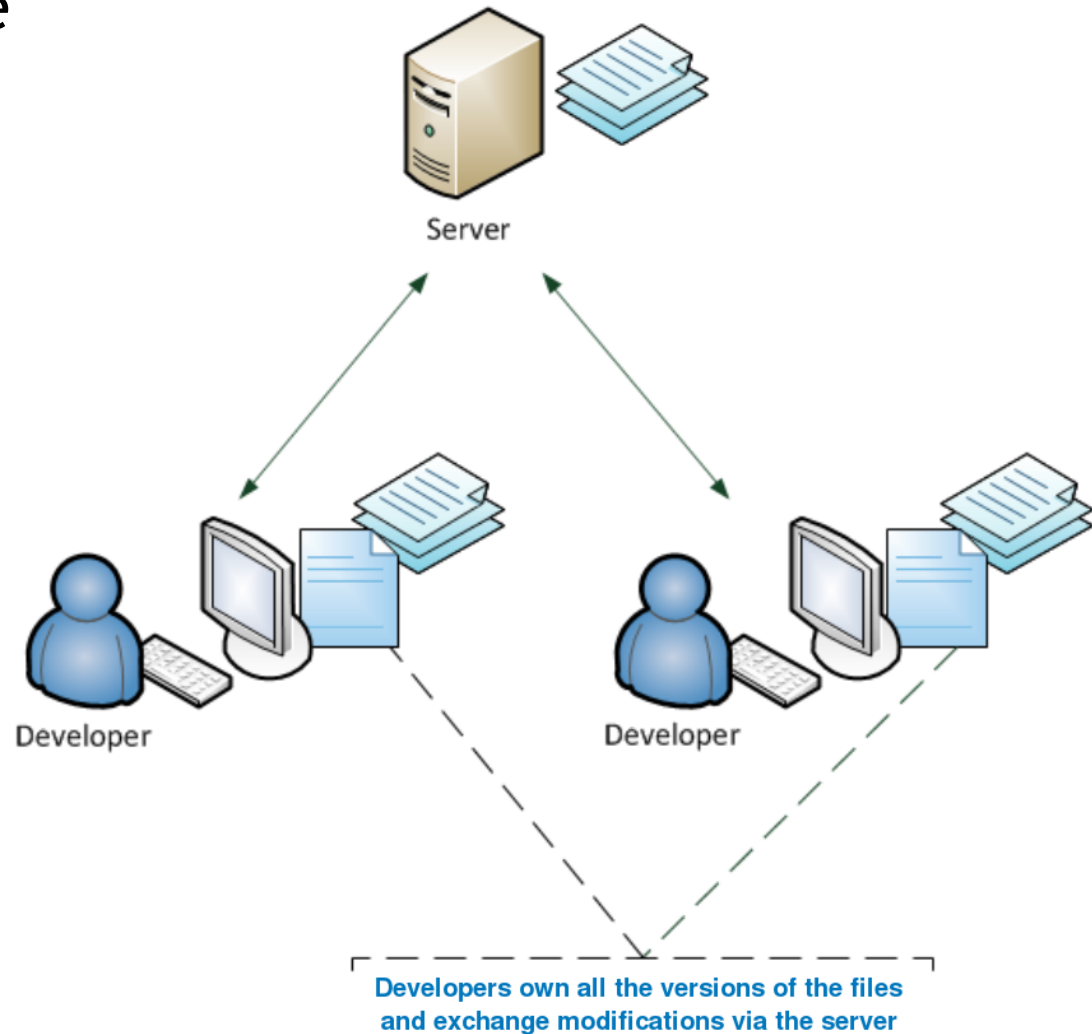
Version control

Decentralized architecture



Version control

Hybrid architecture



Version control

Vocabulary (SVN)

Architecture

Repository

Working copy

Actions

Checkout

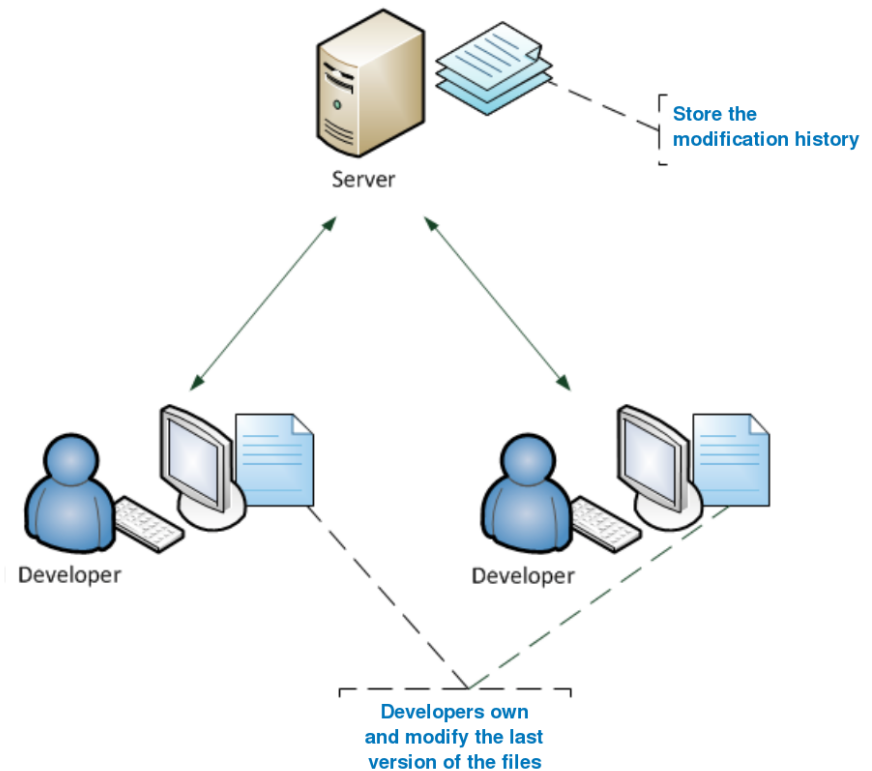
Update

Commit

Revert

Diff, log, status

Centralized Architecture



Version control

Drawbacks of the centralized architecture

- Just one access point to the data

- Just one communication point between users

- Just one historic of the files

- Versioning and sharing are the same operation

 - Need to have a stable state before "committing"

Version control

Vocabulary (Git)

Architecture

Remote and local repository

Working copy

Actions

Clone

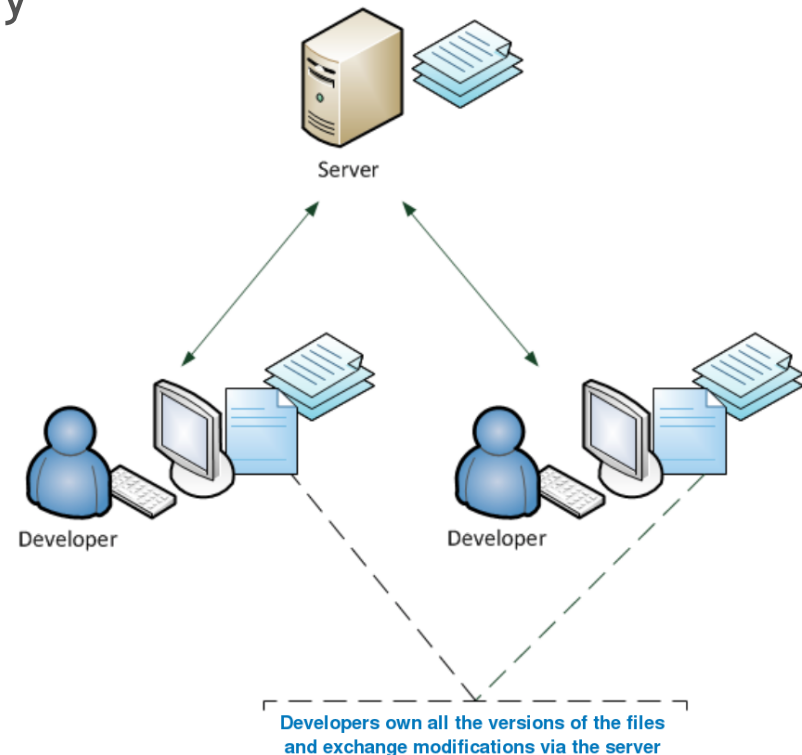
Pull, Push

Commit

Reset

Diff, log, status

Hybrid Architecture



Version control

Good practices

Work on the local copy

Send the modification

Check if the code compiles locally

Check for updates from the other users

Manage conflicts if there are some

Check if the code compiles with the updates

Commit the code on the shared version (server)

Version control

Users can modify the same file

But at different part/section of the files

If they modify the same part of a file

A conflict appends

Usually, it cannot be resolve automatically

Users have to fix the conflit

By telling to the system, which version is correct

By merging the modifications of the users

Version control

Conflicts management

```
C:\workspace\test>svn up
Conflict discovered in 'test.txt'.
Select: (p) postpone, (df) diff-full, (e) edit, (r) resolved,
        (mc) mine-conflict, (tc) theirs-conflict,
        (s) show all options: p
C      test.txt
Updated to revision 3.
Summary of conflicts:
  Text conflicts: 1
```

Version control

Conflicts management

```
08/10/2010  11:44 AM          94 test.txt
08/10/2010  11:44 AM          26 test.txt.mine
08/10/2010  11:44 AM          27 test.txt.r2
08/10/2010  11:44 AM          31 test.txt.r3
```

test.txt

```
<<<<<<< .mine
test User2 making conflict
=====
User1 am making a conflict test
>>>>>>> .r3
```

Version control

Tools for conflict management (TortoiseMerge)

```
SysImageList.cpp - TortoiseMerge
File Edit Navigate View Help
SysImageList.cpp
56 SHGetFileInfo(
57   T("Doesn't matter"),
58   FILE_ATTRIBUTE_DIRECTORY,
59   &sfi, sizeof sfi,
60   SHGFI_SYSICONINDEX | SHGFI_SMALLICON | SHGFI_USEFILEATTRIB
61   return sfi.iIcon;
62 }
63 }
64 }
65 }

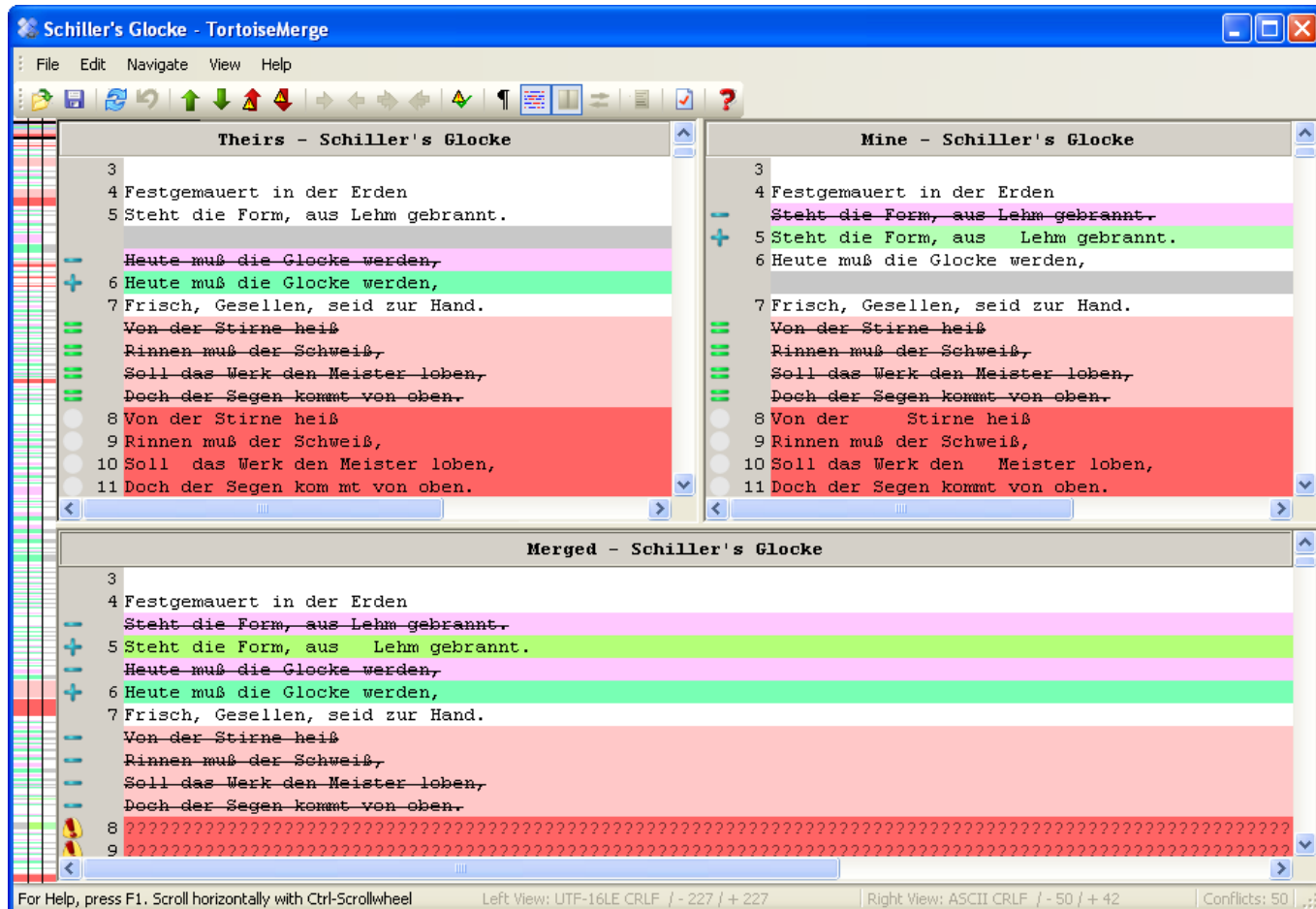
66 int CSysImageList::GetDefaultIconIndex() const
67 {
68     SHFILEINFO sfi;
69     // clear the struct
70     ZeroMemory(&sfi, sizeof sfi);
71
72     SHGetFileInfo(
73         T(""),
74         FILE_ATTRIBUTE_NORMAL,
75         &sfi, sizeof sfi,
76         SHGFI_SYSICONINDEX | SHGFI_SMALLICON | SHGFI_USEFILEATTRIB
77         FILE_ATTRIBUTE_DIRECTORY,
78         FILE_ATTRIBUTE_DIRECTORY,
79         FILE_ATTRIBUTE_DIRECTORY,
80         FILE_ATTRIBUTE_DIRECTORY);
81 }

SysImageList.cpp
55
56 SHGetFileInfo(
57   T("blablah"),
58   FILE_ATTRIBUTE_DIRECTORY,
59   &sfi, sizeof sfi,
60   SHGFI_SYSICONINDEX | SHGFI_USEFILEATTRIB
61   return sfi.iIcon;
62 }
63 }
64 void CSysImageList::Test()
65 {
66     RunTests();
67 }
68 }
69 int CSysImageList::GetDefaultIconIndex() const
70 {
71     SHFILEINFO sfi;
72     ZeroMemory(&sfi, sizeof sfi);
73
74     SHGetFileInfo(T(""), FILE_ATTRIBUTE_NORMAL,
75         FILE_ATTRIBUTE_DIRECTORY,
76         FILE_ATTRIBUTE_DIRECTORY,
77         FILE_ATTRIBUTE_DIRECTORY);
78 }

For Help, press F1. Scroll horizontally with Ctrl-Scrollwheel
Left View: ASCII CRLF / - 16 | Right View: ASCII CRLF / + 15 | Conflicts: 0 | CAP NUM SCRL ...
```

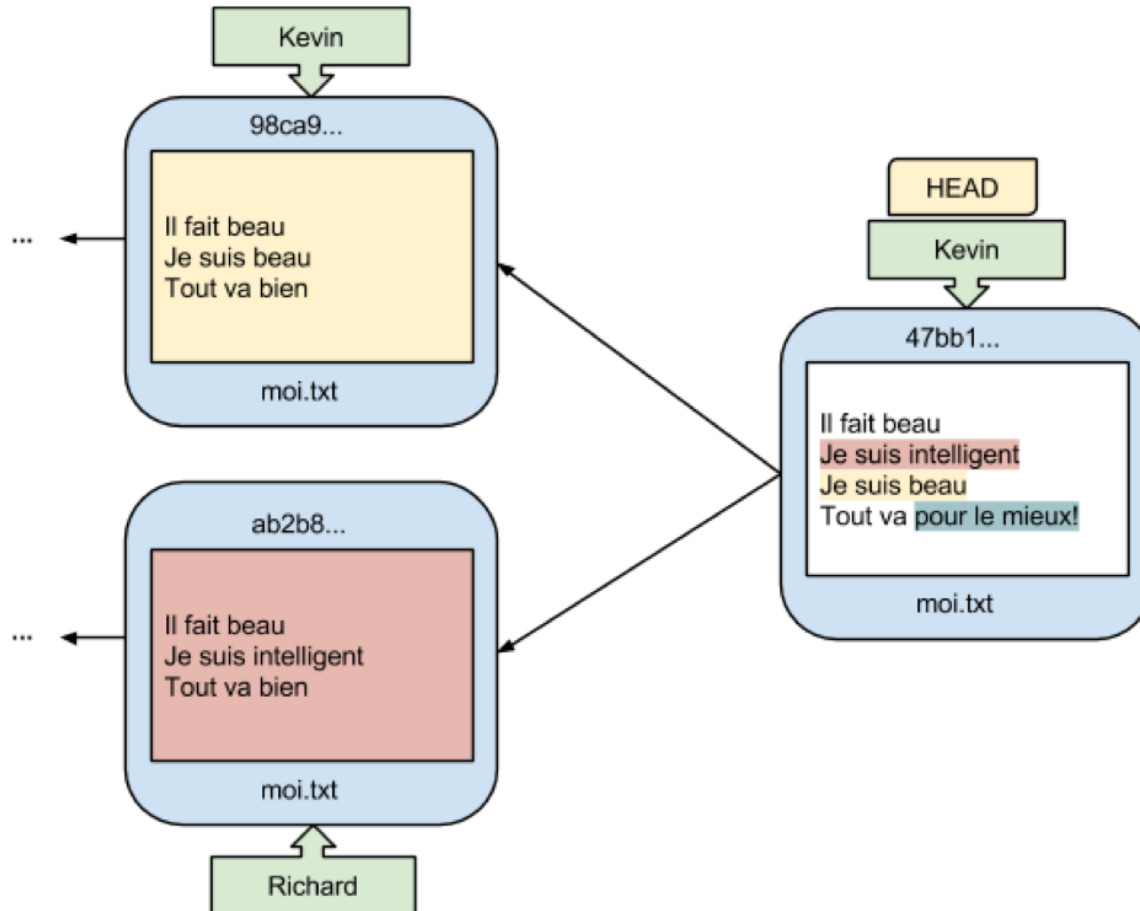
Version control

Tools for conflict management (TortoiseMerge)



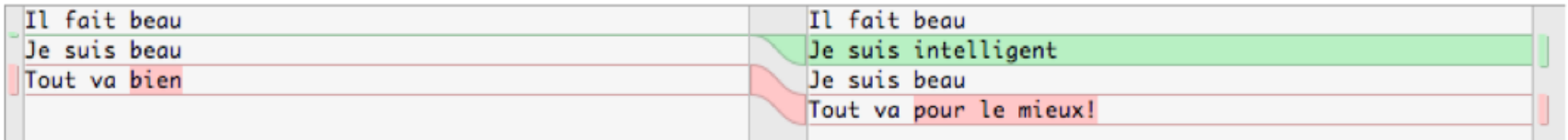
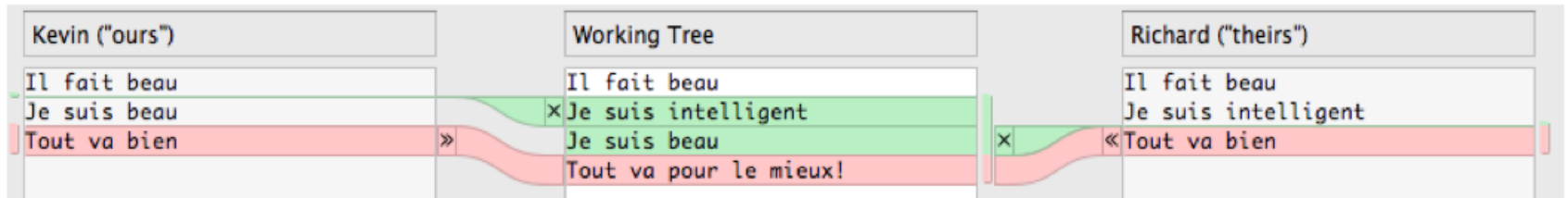
Version control

Tools for conflict management (SmartGit)



Version control

Tools for conflict management (SmartGit)



Version control

Conflicts management

To avoid conflict:

Users are able to "lock" a file

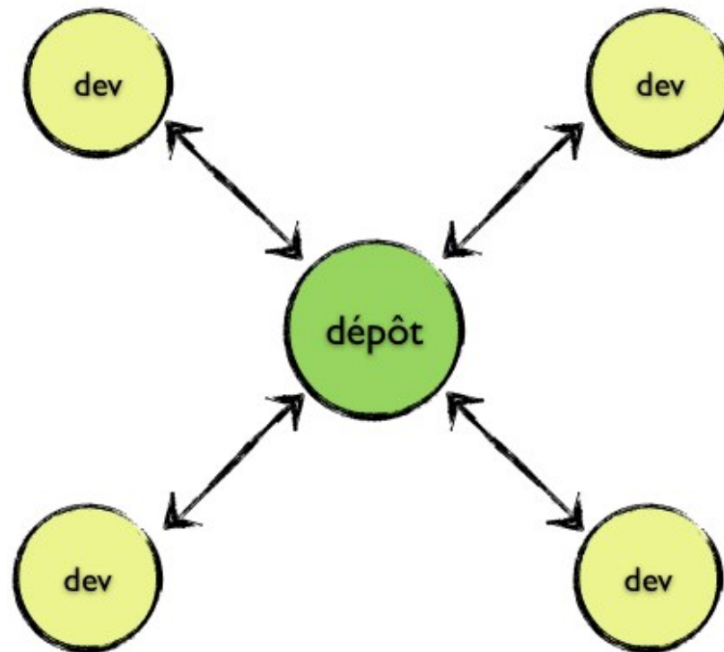
Only the user who locks the file can modify it

If another user try to lock a file while it is locked by another user, he receives an error message

Users have to manually unlock the file when they have finished to work on it.

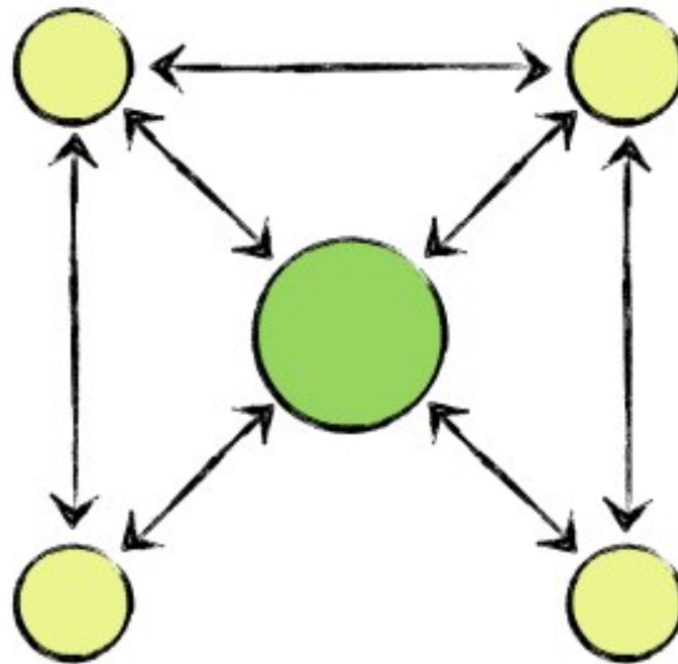
Version control

Collaboration scenario : centralized (SVN)



Version control

Collaboration scenario : decentralized (Git)



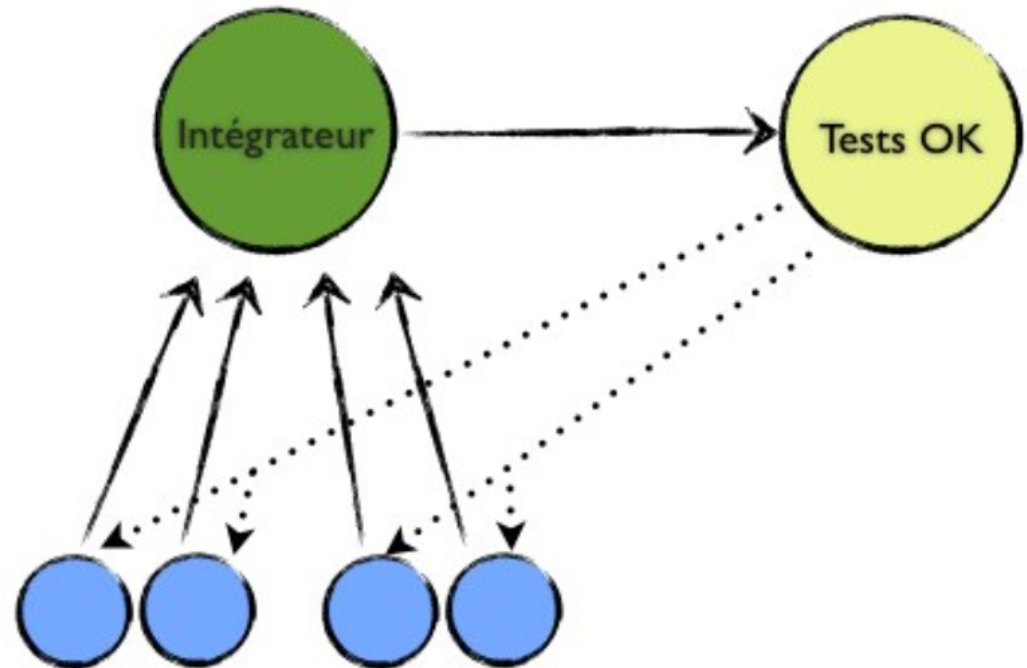
Inter-personal communications

Version control

Collaboration scenario : decentralized (Git)

Integrator mode

A repository
is in charge
of the test



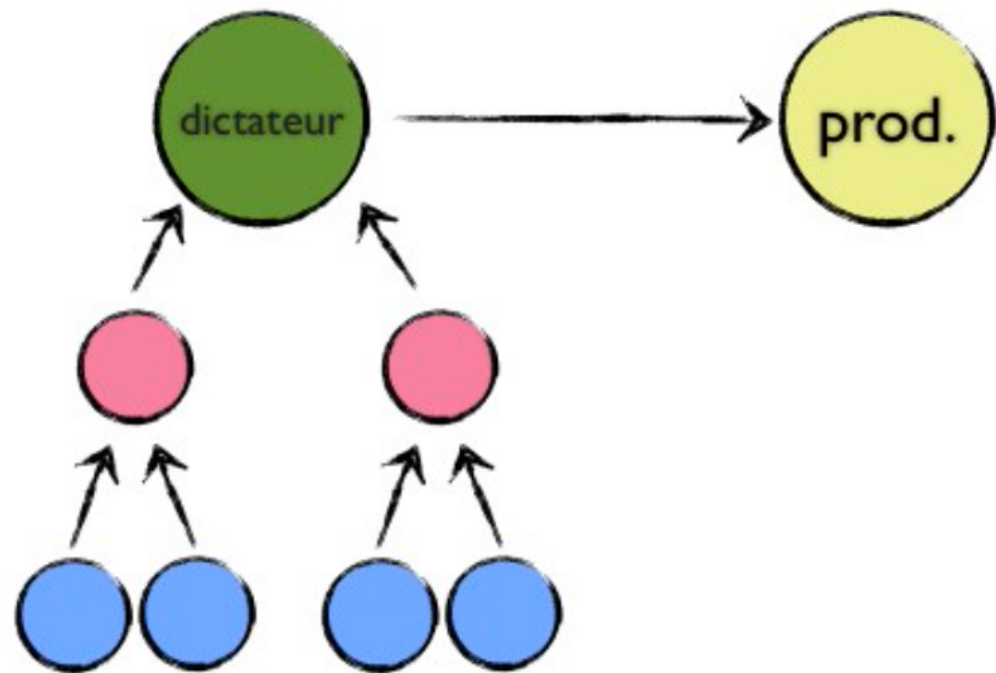
Version control

Collaboration scenario : decentralized (Git)

Dictator mode

Open-source projects

"Lieutenants" make a first check before sending to the "dictator"



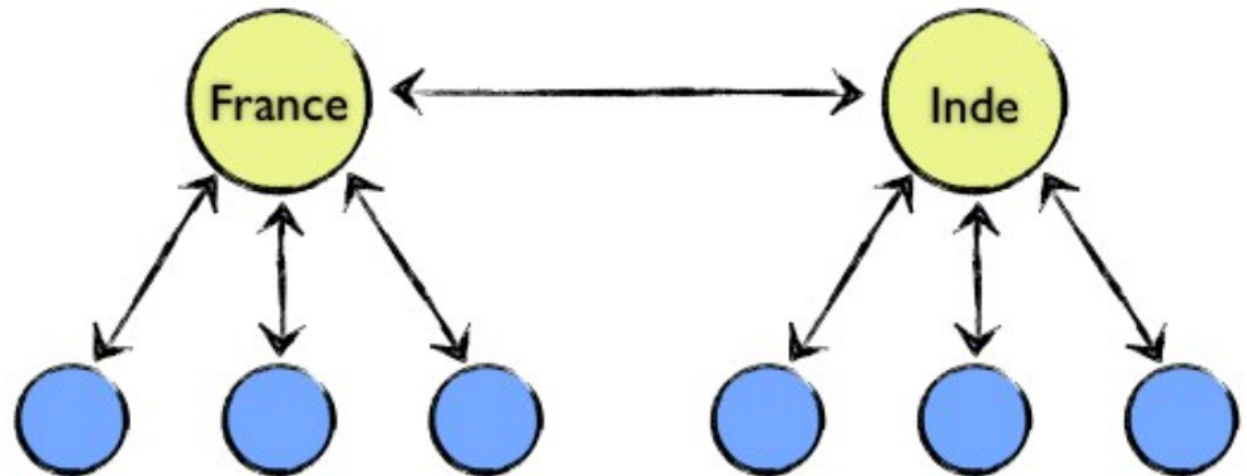
Version control

Collaboration scenario : decentralized (Git)

Multi-location team

Each team can work independently

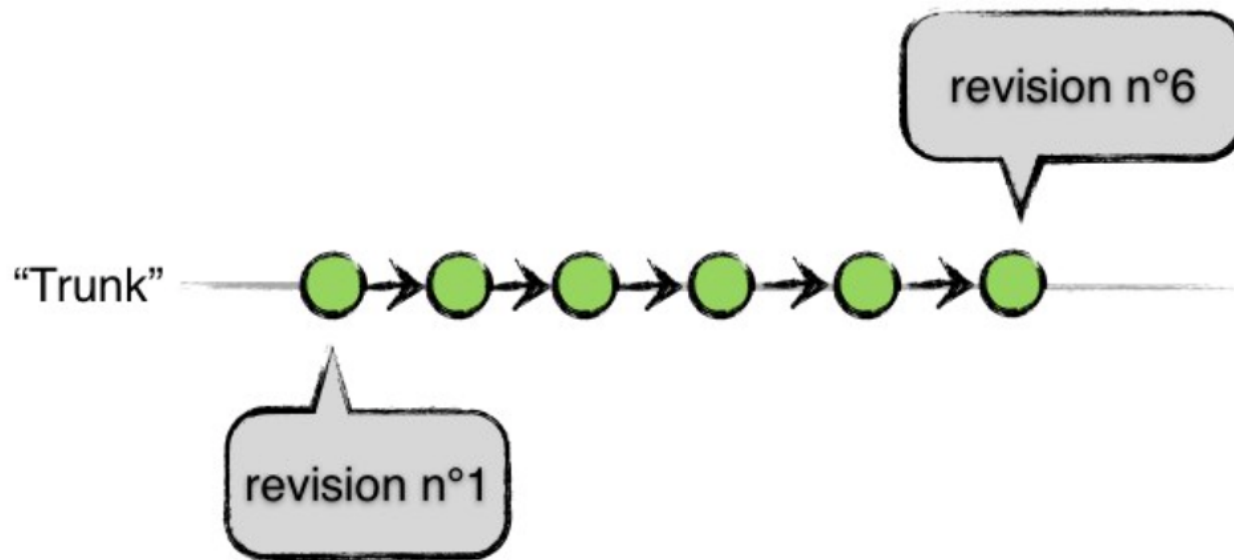
Regular integration of the work of each team
can be done



Version control

Historic management

Computation of the historic is linear according to the "commit" order

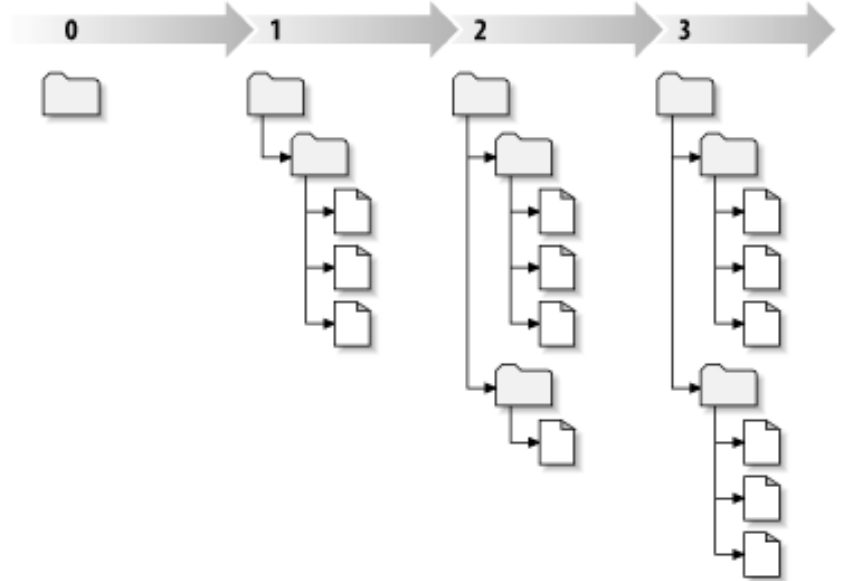


Version control

Historic management

SVN assigns a revision number to all the project

Git assigns a revision number per file

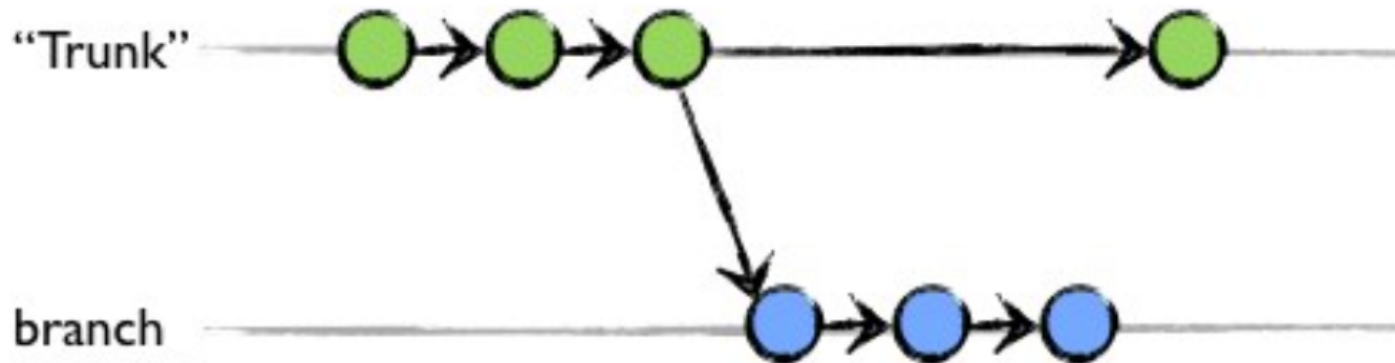


This difference has a strong impact on collaboration

Using branch for collaboration is easier with Git

Version Control

Branch management



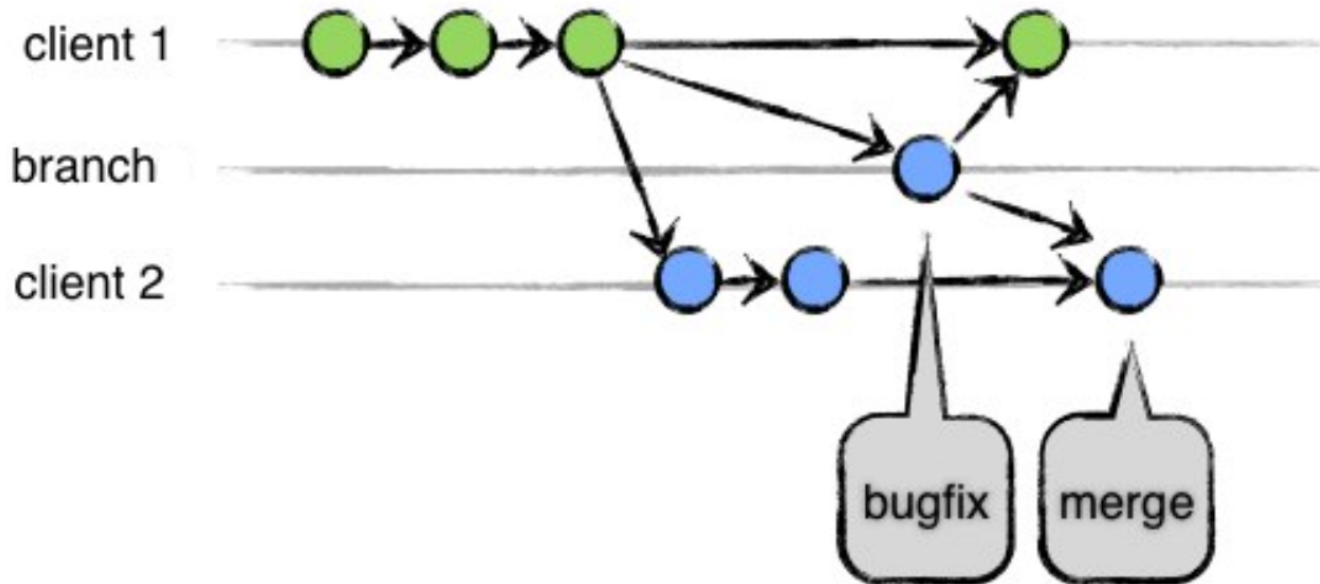
SVN make a copy of the all repository

Git make a link to a particular state of the files

Version Control

Branch management

Merging branch (very complex to achieve with SVN)

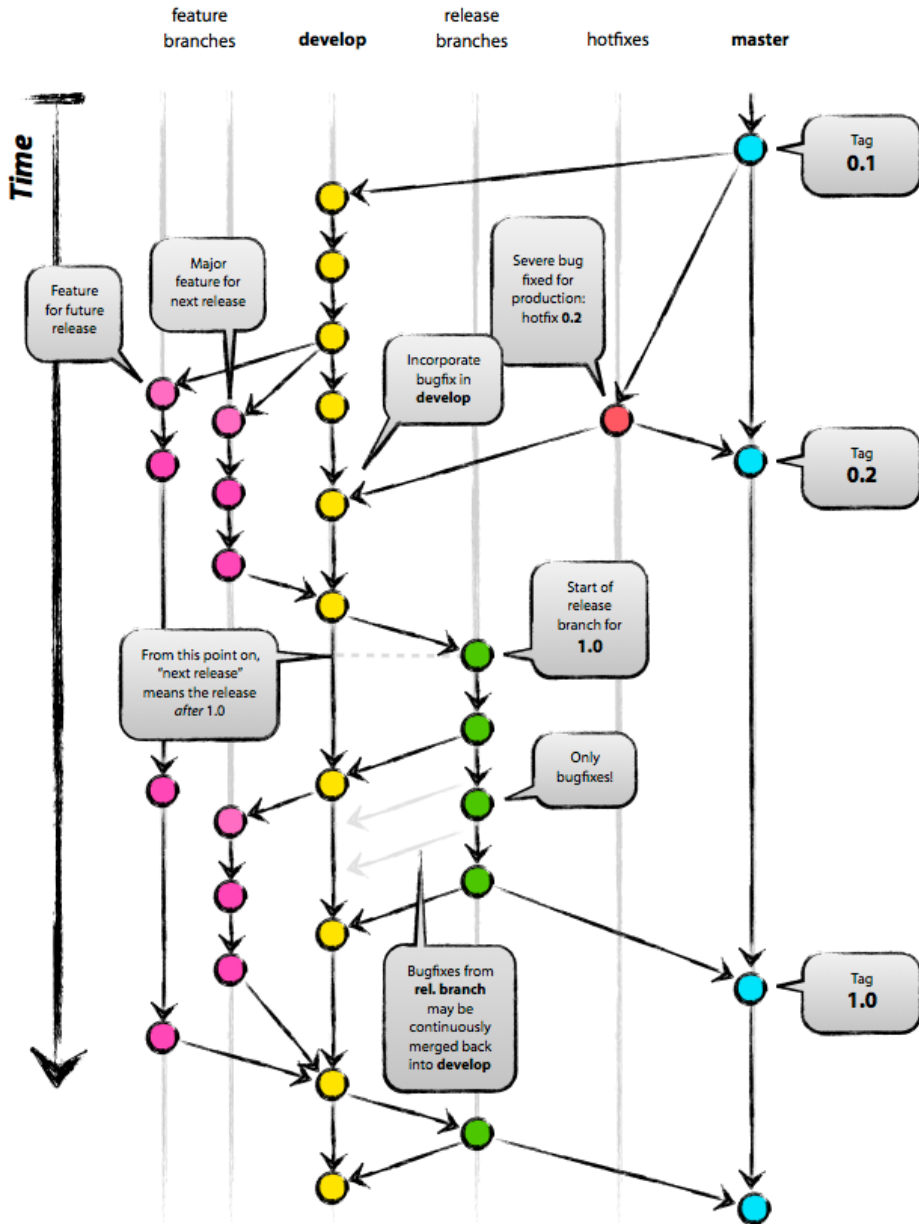


Version Control

Branch management

Classical organisation of a project into branches

<http://nvie.com/posts/a-successful-git-branching-model/>



Version Control

Web interfaces

Manage user access rights

Manage branches and access to branches

Review modifications and different versions

Track bugs

Create wiki / web pages for projects

Add social network functionalities



Outline

Collaborative software development

Version control

Continuous integration

Software development methods

Continuous integration

Integration

Merging the work of several developers

Goals

Test modifications from the beginning

Detect integration problems at an early stage

Avoid fastidious integration phases

Always have the system running

Tests, demos, discussion with the customers

<http://martinfowler.com/articles/continuousIntegration.html>

Continuous integration

Principles

Version control on a repository

Automatic and fast build

Auto-testing

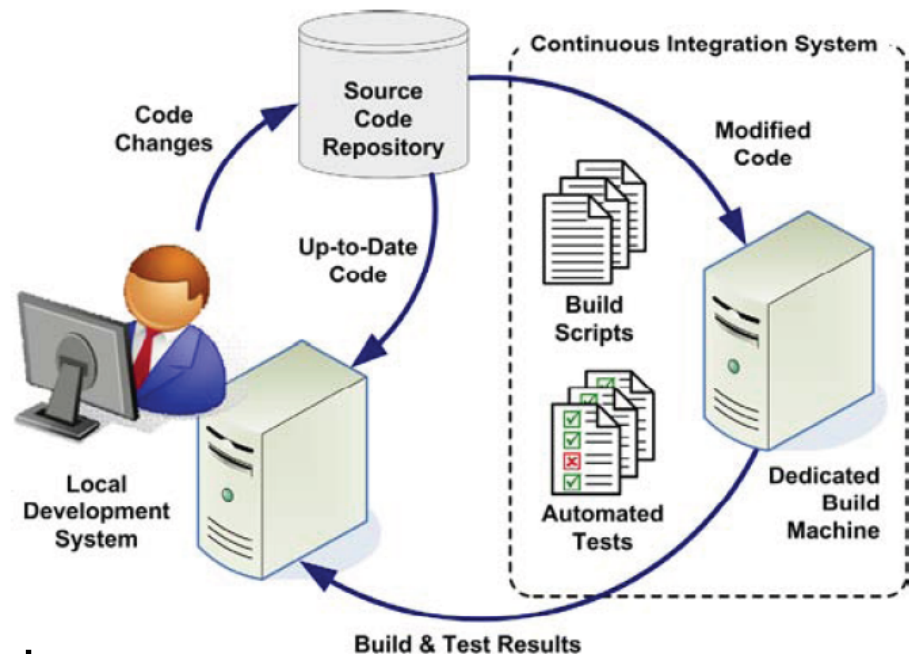
"Commit" every day

Deployment on an integration computer after each "commit"

Automatic deployment

Executable always available

Every body know the build state



Continuous Integration

Feedbacks for collaboration

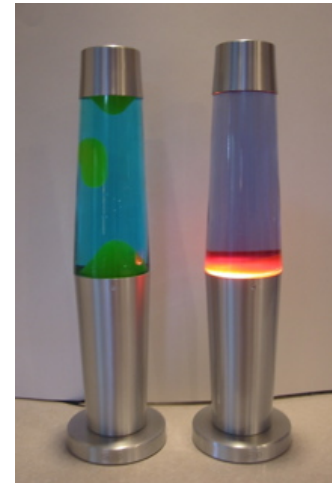
Token on the desk of the person who builds

Make a sound when a build is valid

Web page of the integration server

Bubble light

Wallboard



Outline

Collaborative software development

Version control

Continuous integration

Software development methods

Software development

Methods for software development

No methods: "Code and fix"

Efficient for small project

Difficult to add new features or to find bugs

Engineering / plan-driven methodologies

Come from civil or mechanical engineering

Drawing / construction plan / task distribution / construction

Agile methodologies

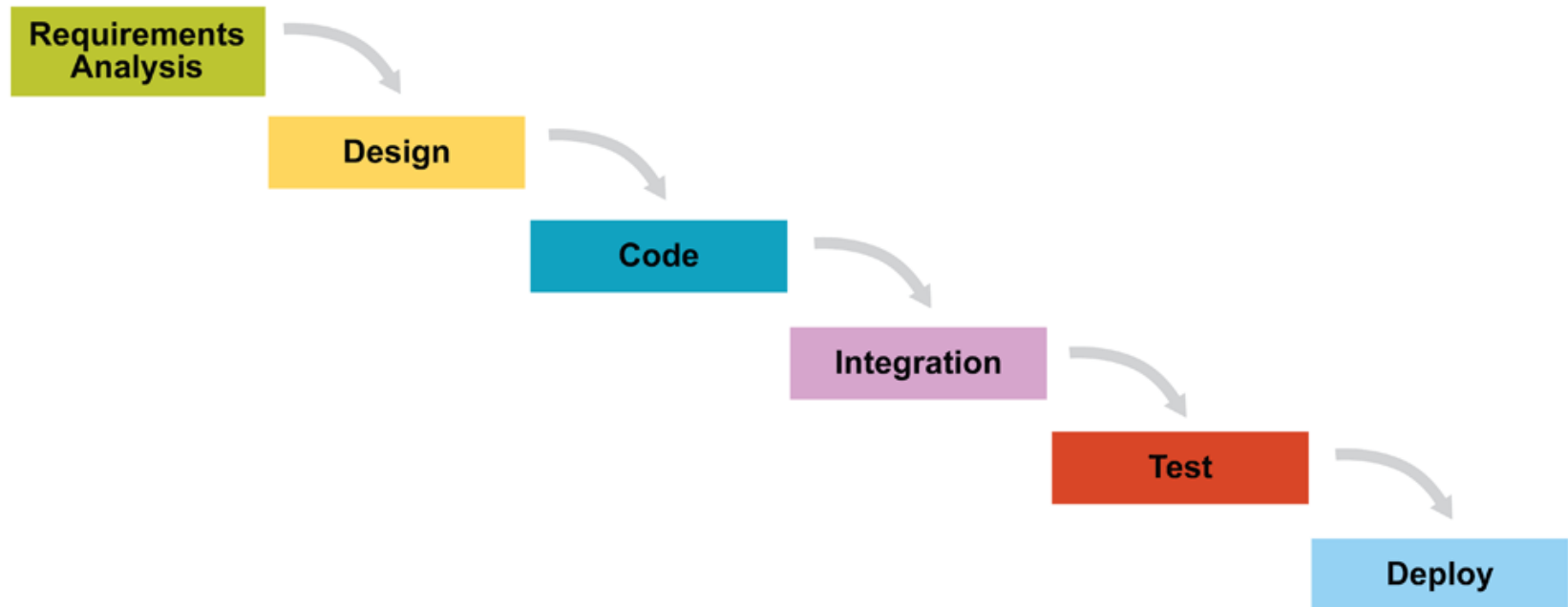
Adaptive rather than predictive

People-oriented rather than process-oriented

Software development

Engineering methodologies

Example: Waterfall



Software development

Engineering methodologies

Separation of design and construction

Design

Unpredictable

Require creative people

Construction

Predictable

Require people with lower skill

Example: civil engineering

construction is bigger in cost and time than design

Software development

Design and Construction for software?

UML = *Design*, coding = *Construction*?

Source code = *Design*, compilation = *Construction*

Construction is quick and cheap

Source code requires creative and talented people

Creative processes are unpredictable

Are the engineering methodologies well adapted ?

[Jack Reeves, C++ Journal, 1992]

<http://www.bleeding-edge.com/Publications/C++Journal/Cpjour2.htm>

Software development

Is software development predictable?

Yes in some cases...

NASA programs

Usually, requirements are unpredictable

(especially for software involving interactions with users)

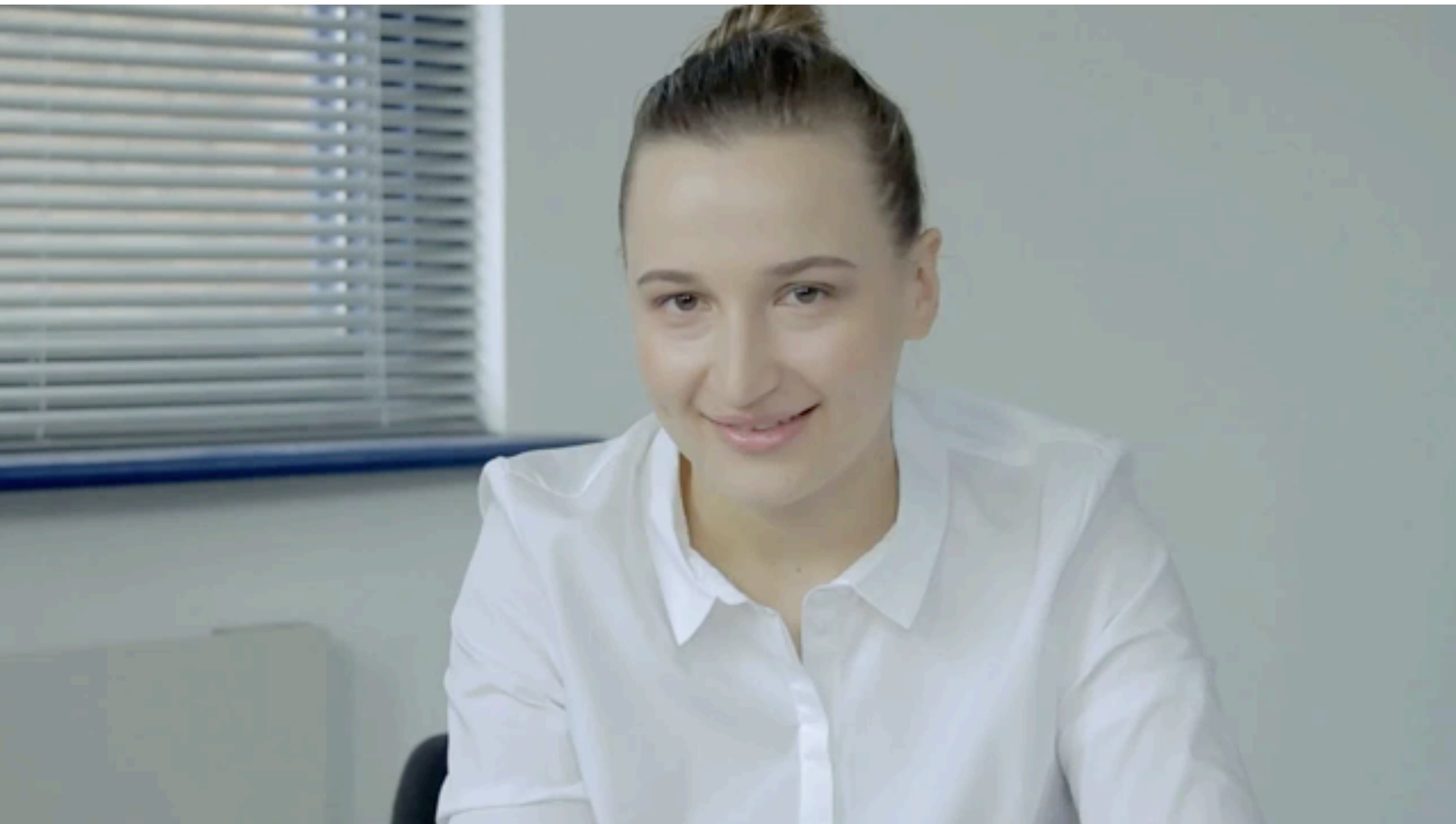
Customers don't precisely know what they want

Hard to evaluate the cost of different options

Hard to estimate which features are useful

⇒ Requirements should be flexible in these cases

Software development



Agile methods

Deal with unpredictable requirements

Iterative development

Involve the customers at each iteration

Improve the team organization (self-adaptive process)

Effective team of developers (people first)

Do not consider developers are replaceable parts

Analysts, coders, testers, managers

Developers are responsible professionals

Make the technical decisions

Evaluate the time required to perform the tasks

Agile methods

Manifesto for Agile Software Development

Individuals and interactions over processes and tools

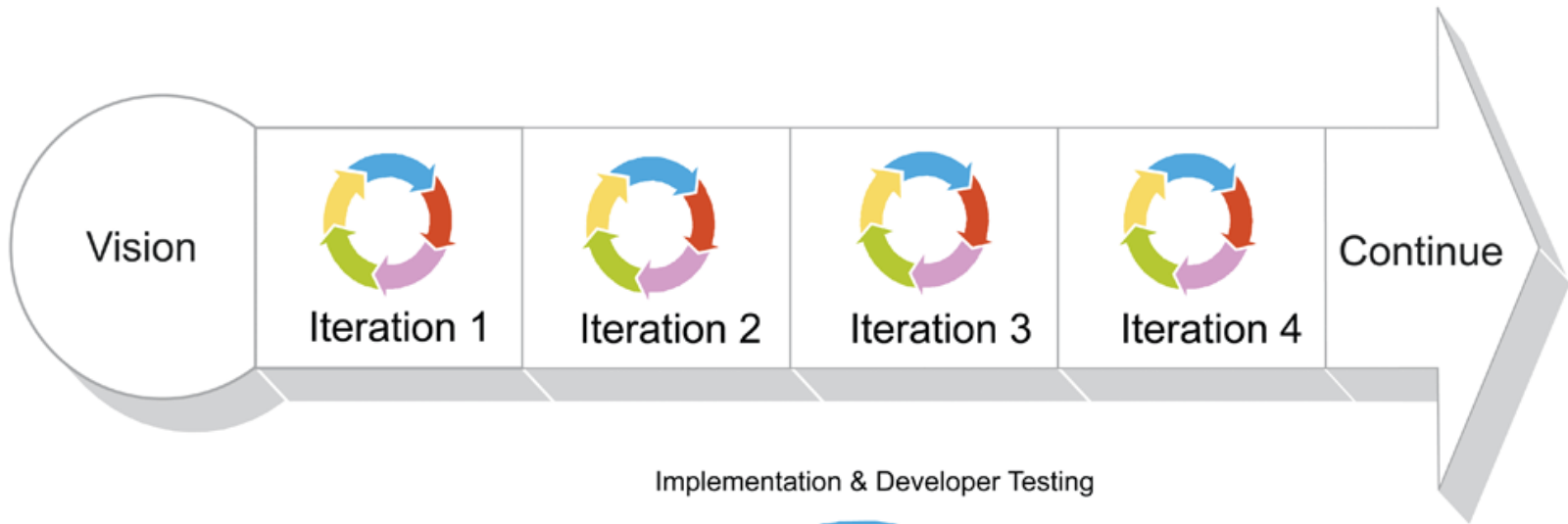
Working software over comprehensive documentation

Customer collaboration over contract negotiation

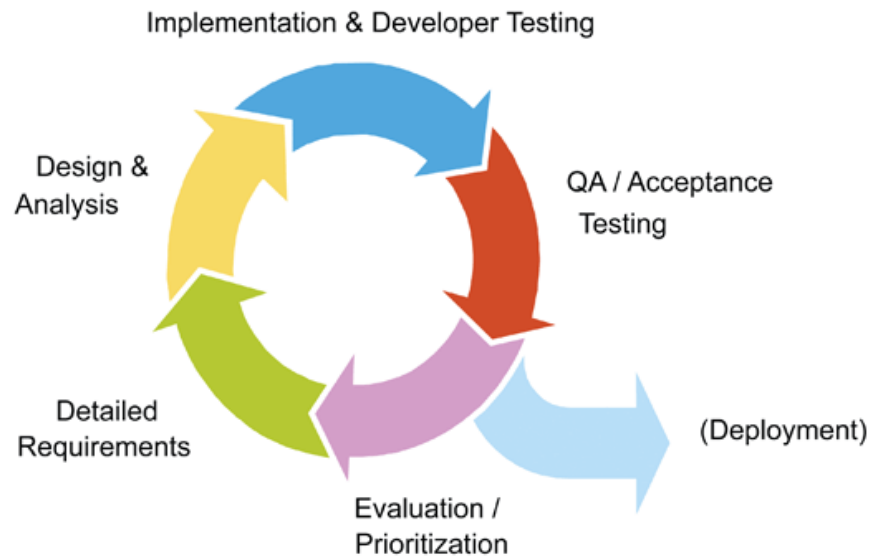
Responding to change over following plan

<http://agilemanifesto.org>

Agile methods



Iteration Detail



Agile methods

Examples

XP (Extreme Programming)

Test driven development, pair programming

Scrum

Crystal

Safety, efficiency, habitability (less discipline than XP)

Open source process

Distributed contributors, parallelized debugging

Lean software development (Lean development @ Toyota)

Just in time, Jidoka ("automation with a human touch")

RUP (Rational Unified Process)

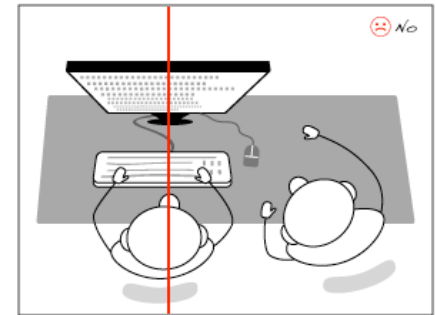
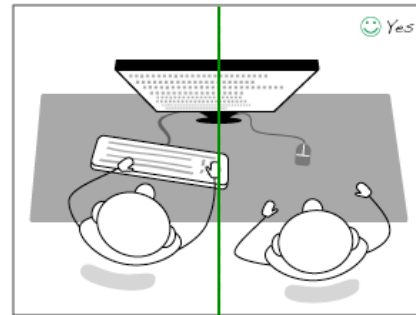
Use case driven, iterative, architecture centric

Pair programming

Two programmers

One computer

Roles



One "drives": operating mouse and keyboard

Code: syntax, semantics, algorithm

One "navigates": watches, learns, asks, talks, makes suggestions

Higher level of abstraction

Test, technical task, time since the last commit,

Quality of the overall design

Pair programming

Advantages

Code quality

- Better designs

- Fewer bugs

Spreading Knowledge

- Pairs have to switch off regularly

- Technical and conceptual knowledge

Social aspects

- No loneliness, conviviality, better motivation

Pair programming

Productivity

(it depends on how you measure productivity : lines of code VS running and tested features)

Short-term productivity might decrease slightly (about 15%)

Long-term productivity goes up

Because the code is better

Even better if you consider staff turnover

Truck number in XP

Close as possible to the team size

Pair programming

Pairing strategies

In XP, all production code is written by pairs

In non-XP agile teams, usually pairing is not used at all

A trade-off can be found

For some particular tasks

Mentoring new hires

Extremely high-risk tasks

Start of a new project when the design is new

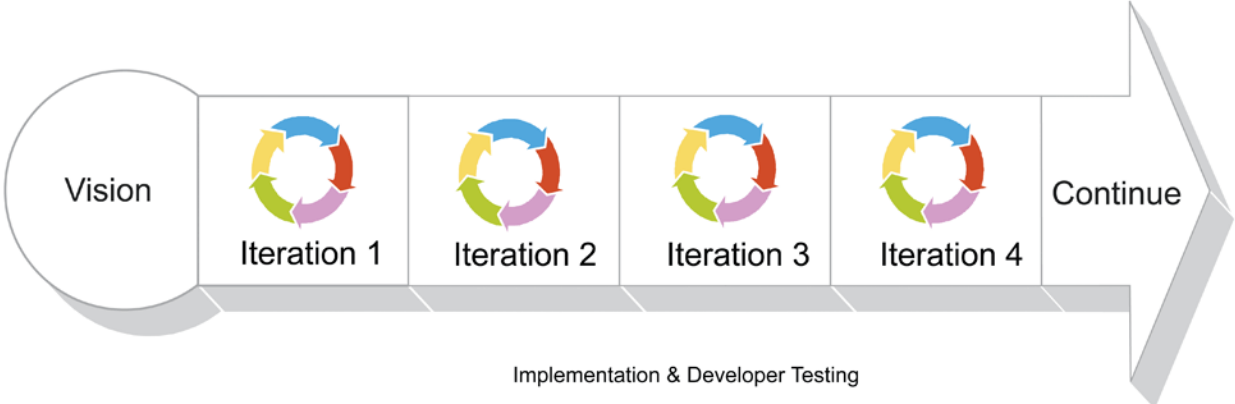
When adopting a new technology

On a rotating monthly or weekly basis

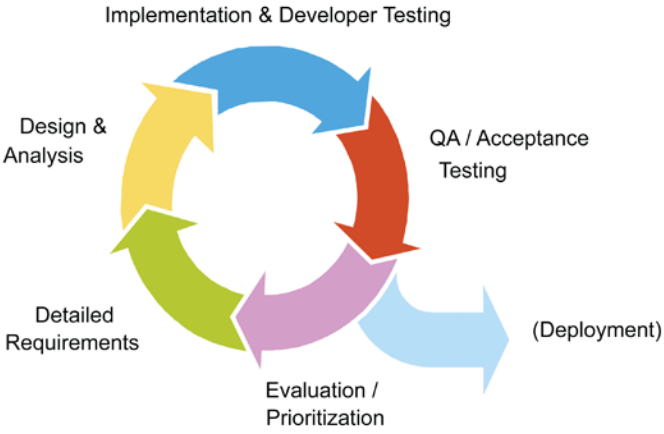
Developers who prefers to pair

Scrum

Iterations called Sprint (about 1 month)



Iteration Detail



Scrum

Roles

Product Owner

Single person

Responsible for products vision

Constantly re-prioritizes the Product Backlog

Accepts or rejects product increment

Development team

Self-organized

Negotiate commitments with the Product Owner

Has autonomy regarding how to reach commitments

Intensely collaborative

Master

Facilitates the Scrum process

Helps resolve issues

Shields the team from external inferences and distractions

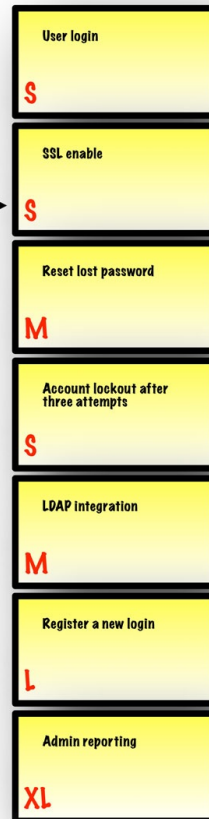
Has no management authority



Scrum

Product Backlog

top items
are more
granular



only one item
at a time
is top priority

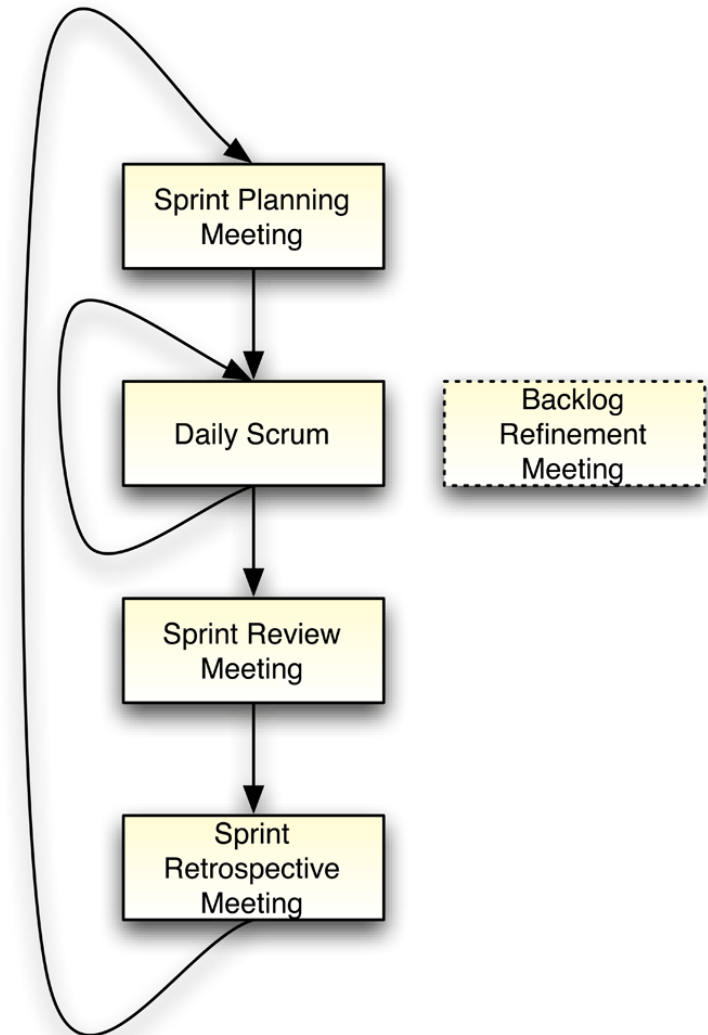
Scrum

Sprint

Planning Meeting

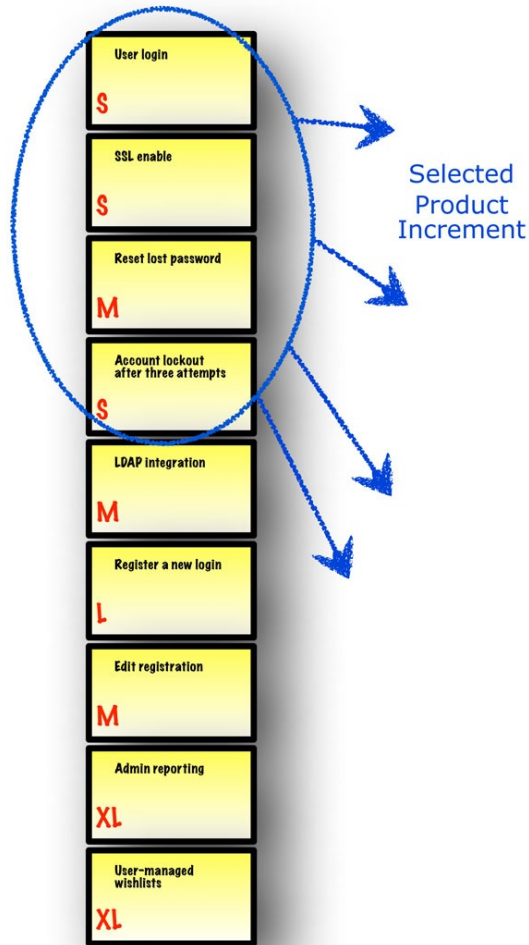
Negotiate which Product Backlog items will be processed

Break items into a list of sprint tasks

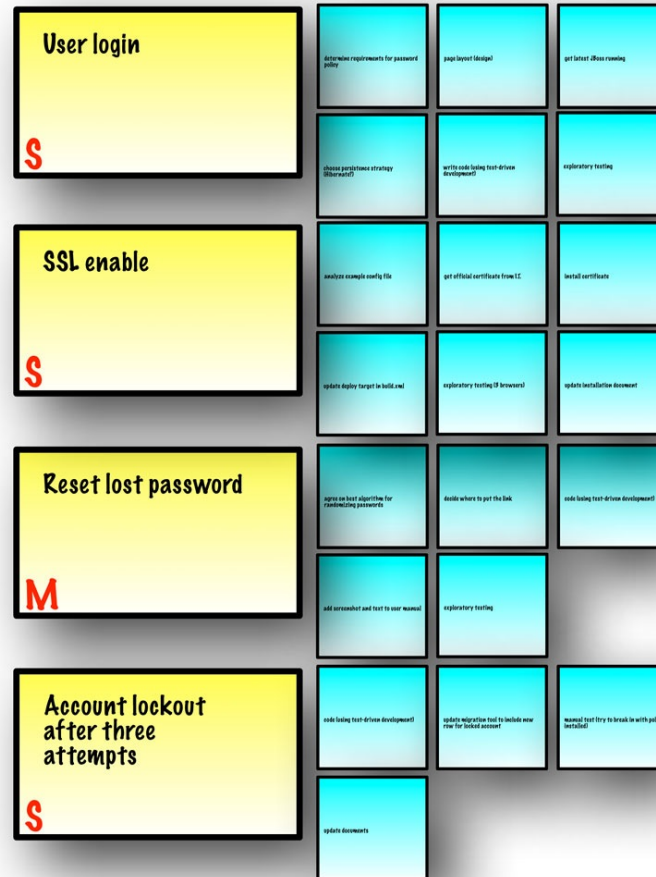


Scrum

Product Backlog



Sprint Backlog



Scrum

Sprint

Planning Meeting

Daily Meeting

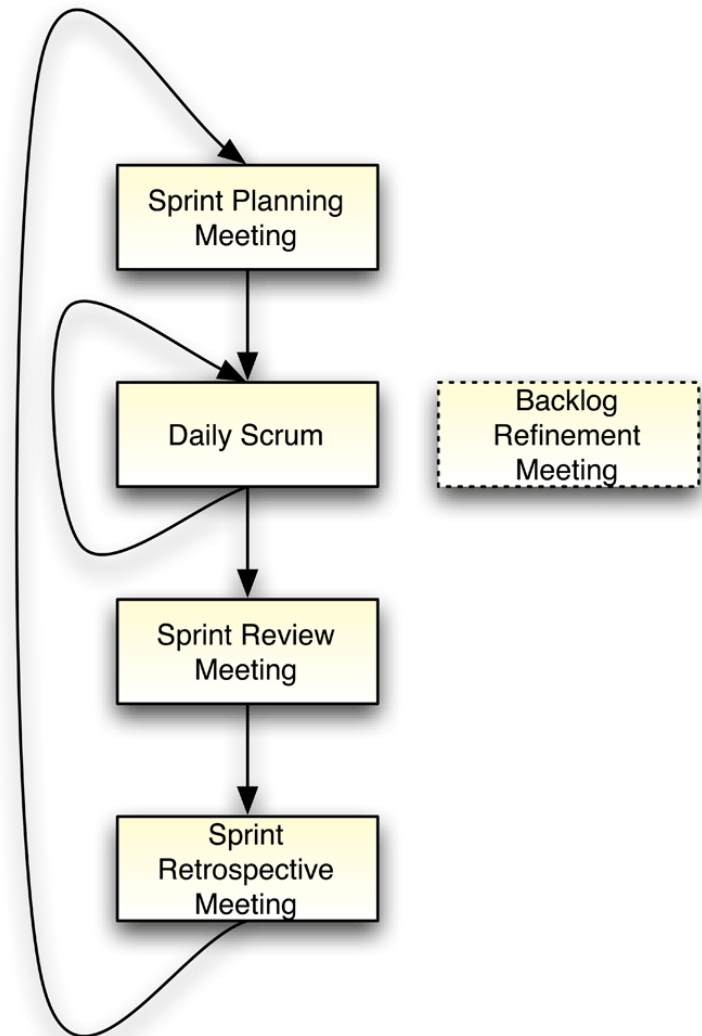
Same time and place

15 minutes, standing up






















Summarize work of previous day, work of today, issues

Maintain tasks list (not started, in progress, done), issues list and burn-down chart.

Product Owner may attend



Sprint Backlog

Committed Backlog Items	Tasks Not Started	Tasks In Progress	Tasks Completed
	  		 
	  		
	     		
			

Scrum

Sprint

Planning Meeting

Daily Meeting

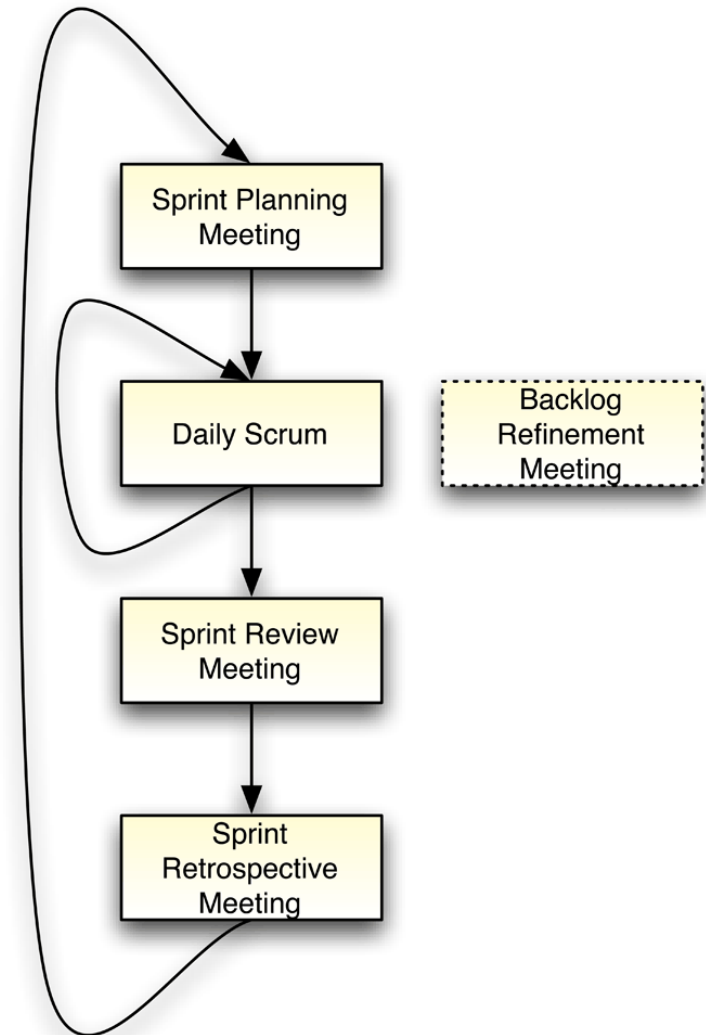
Review Meeting

Demonstrate the working product increment to the Product Owner

Product Owner declares which items are done

Unfinished items return to the Product Backlog

Master proposes new items for the Product Backlog



Scrum

Sprint

Planning Meeting

Daily Meeting

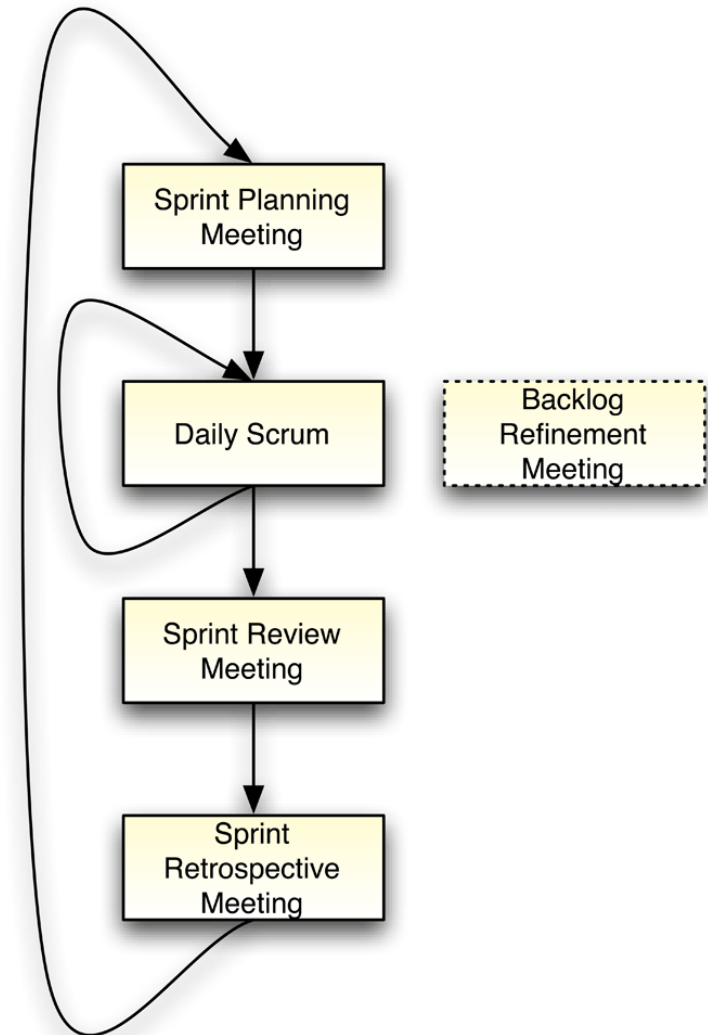
Review Meeting

Retrospective Meeting

Team reviews its own process

Team takes to adapt it for futur Sprints

Master have to manage the psychological safety of the meetings



Scrum

Sprint

Planning Meeting

Daily Meeting

Review Meeting

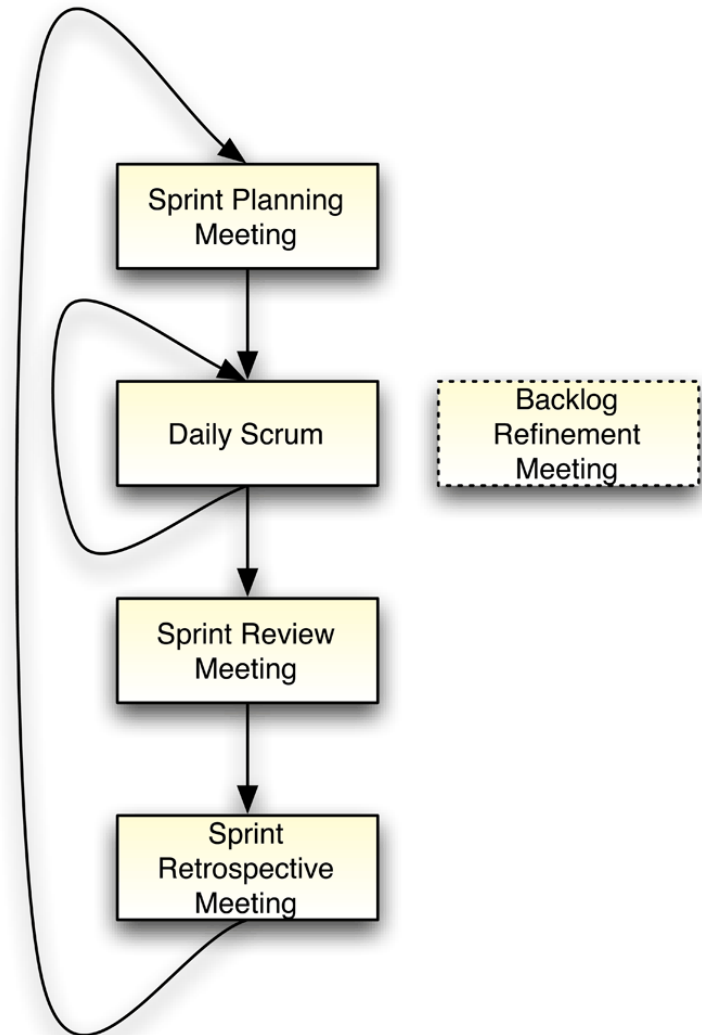
Retrospective Meeting

Backlog Refinement Meeting

Items are usually too large or poorly understood

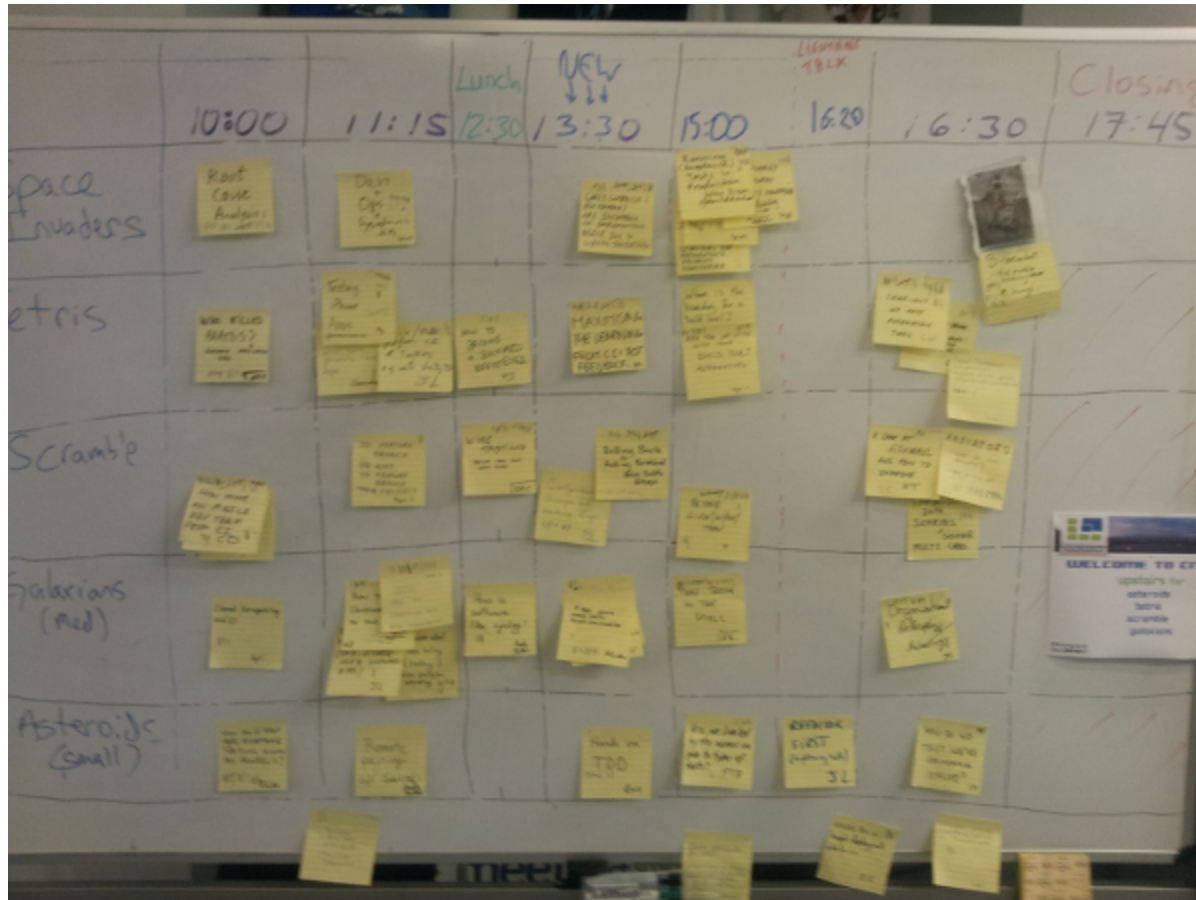
Refine these items into smaller one

Master can help



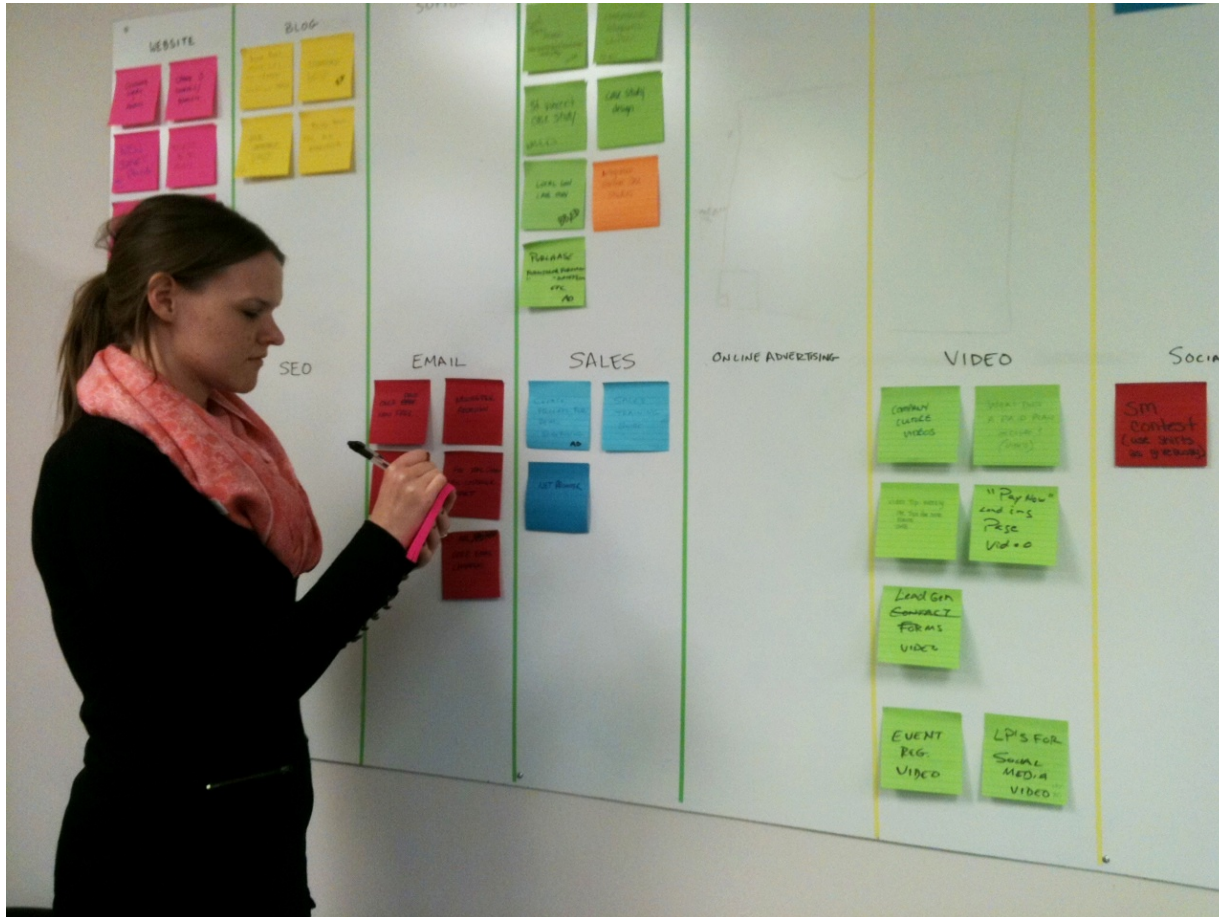
Scrum

Feedbacks to the team: wallboard



Scrum

Feedbacks to the team: wallboard



Feedbacks to the team: wallboard

JAVA 1 "OMEGA TEAM" BACKLOG

EZE STORY STATUS

EZE STORY STATUS

EZE STORY STATUS

Release Burndown chart

Team parking lot

Unplanned items (production tickets of the day)

"In progress"

DONE red star indicates story complete

Color magnet indicates team

All this up to here is now in scope for Sprint 9

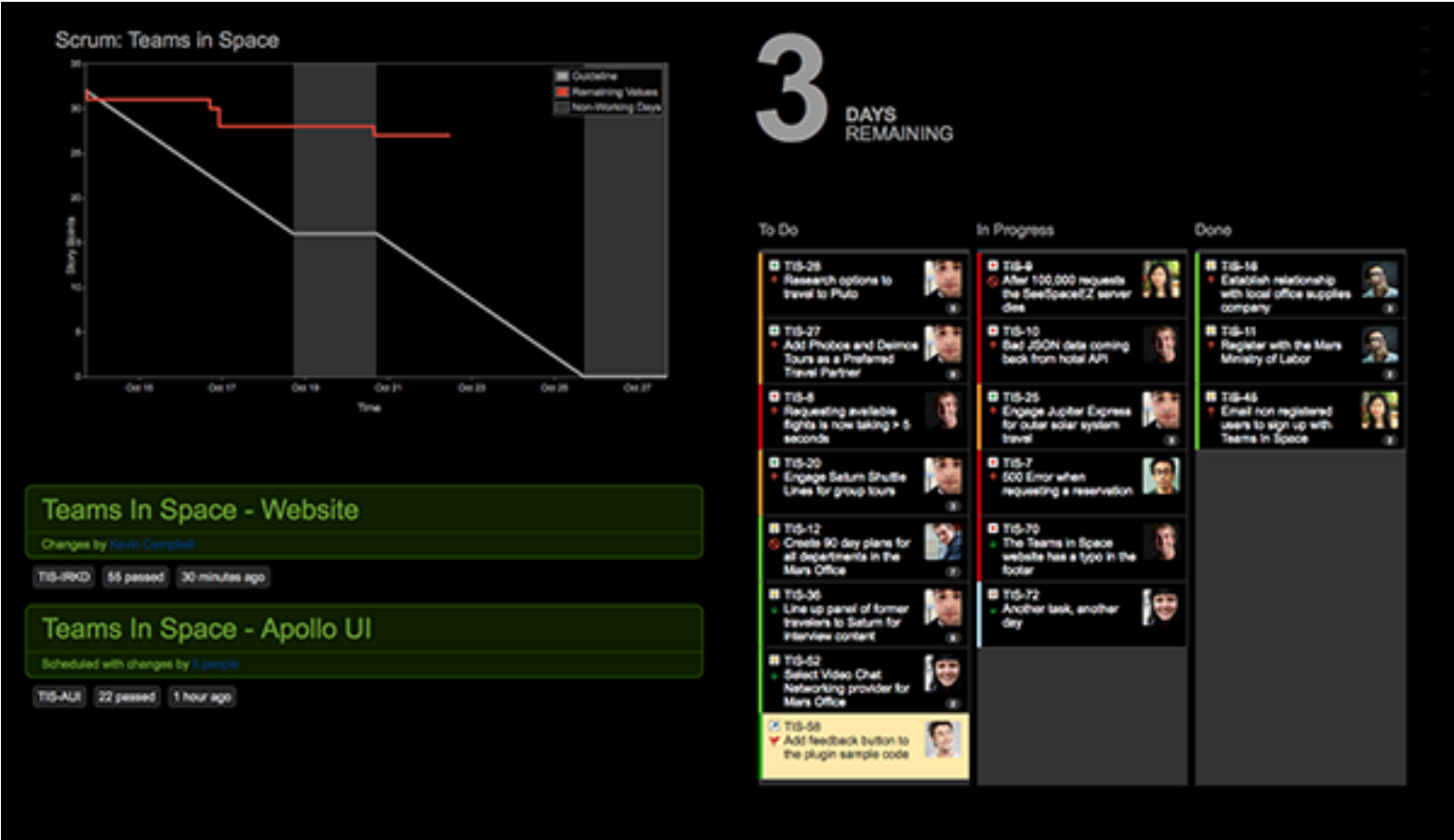
This should have been finished in Sprint 5...

This should have been finished by Sprint 8...

(c) 2009 Visual Management Blog

Scrum

Feedbacks to the team: wallboard



Scrum

Software to manage Scrum projects

The screenshot displays the JIRA Agile interface for a project named 'Angry Nerds'. At the top, there is a navigation bar with 'Dashboards', 'Projects', 'Issues', 'Agile', and 'Bonfire'. A search bar and 'Create Issue' button are also present. Below the navigation, the 'Angry Nerds' dashboard is shown with a progress bar indicating '8 days left' and '23/Jul/12 4:21 AM — 03/Aug/12 4:21 AM'. A list of issues is displayed, including 'NERD-4 As a Front-Ender I would like to stop supporting IE6 so I can enjoy my life' (3 points), 'NERD-2 As a Hacker I would like more Red Bull so I can work all night' (8 points), 'NERD-3 As the Dev Manager I would like to look busy so I can keep my job' (0 points), 'NERD-4 As an Outsourcerer I want to get paid for working in my pyjamas' (20 points), 'NERD-5 As an Agilista I want to play buzzword bingo so I can ban' (0 points), 'NERD-6 As the Founder I want to have the last say so I can get my way' (13 points), 'NERD-8 As a Bug I want to fly in the face of progress' (0 points), and 'NERD-7 As a Bug I want to make like hard for the Angry Nerds' (0 points). Below the list, there are sections for 'Upcoming Sprint 1' (5 issues, 29 estimate) and 'Upcoming Sprint 2' (3 issues, 9 estimate). On the right, the details for issue 'NERD-4' are shown, including its description, status ('Not Started'), and a session table with one session 'Confirm they are in pyjamas' in 'Created' status.

<http://youtu.be/KdyV9okLRlc>

Conclusion

Collaboration in software development

- Is necessary for big projects

- Is not obvious

 - Technical, organizational and social aspects

Version control

- Synchronization, versioning

- Branching: split work between users

Conclusion

Continuous integration

Improve safety and efficiency

Agile method

Organize the team

Propose an adaptative process to unpredictable requirements

References

Version control

<http://nvie.com/posts/a-successful-git-branching-model/>

<http://www-igm.univ-mlv.fr/~dr/XPOSE2010/gestiondeversiondecentralisee/dvcs-svn.html>

<http://www.infres.enst.fr/~bellot/java/poly/git.pdf>

<http://fr.openclassrooms.com/informatique/cours/gerez-vos-codes-source-avec-git/qu-est-ce-qu-un-logiciel-de-gestion-de-versions>

Continuous Integration

<http://martinfowler.com/articles/continuousIntegration.html>

Agile Models

<http://agilemanifesto.org>

<http://martinfowler.com/articles/newMethodology.html>

Pair Programming

http://www.versionone.com/Agile101/Pair_Programming.asp

Scrum

<http://scrumreferencecard.com>