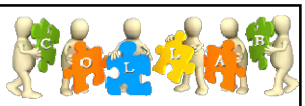
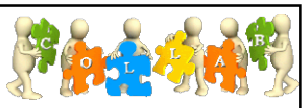


La collaborazione nello sviluppo del software open source

Prof. Filippo Lanubile

Laboratorio di Ricerca per la Collaborazione in Rete
(Collab)
Dipartimento di Informatica
Università degli Studi di Bari Aldo Moro

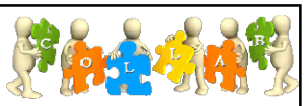
Research at COLLAB



http://collab.di.uniba.it

- Addressing the problem of
  distance in distributed workgroups
- Special focus on software development as an intense collaborative process
  collaborative software development
  - means for communication
  - shared information space
  - coordination

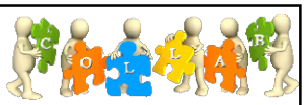
COLLAB Timeline



- 2003: COLLAB opening
- 2006: IBM Eclipse Innovation Award
  - Project “eConference over ECF”
- 2008: IBM Faculty Award
  - Project “Penelope Application Framework”
- 2011: Microsoft Research Software Engineering Innovation Foundation Award
  - Project “Augmenting Social Awareness in a Collaborative Development Environment”
- 2013: 8th IEEE Int. Conf. on Global Software Engineering, Bari, Italy
- 2015: National grant: Scientific Independence of young Researchers (SIR)
  - Project “Investigating the Role of Emotions in Online Question & Answer Sites”

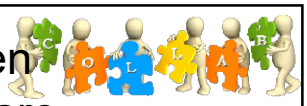
open source software

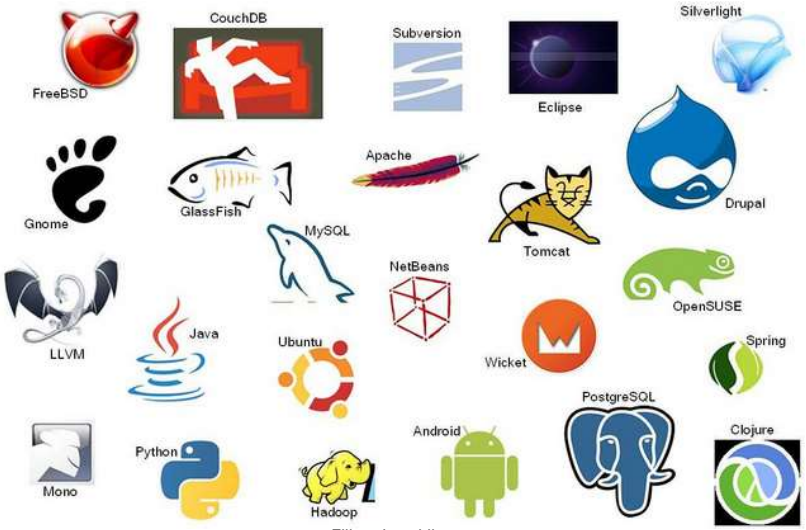
Comunità open source come caso di collaborazione in rete



- Sviluppo distribuito del software
  - Comunicazioni tutte online
- Organizzazione decentralizzata
  - I partecipanti conservano la loro indipendenza

Non solo Linux: comunità open source nate da progetti software

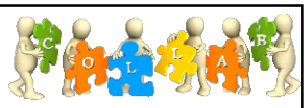




Filippo Lanubile

http://cloudramblings.me/2015/02/06/why-open-source-has-changed-from-the-cheapest-software-to-the-best-software/

Free/Libre Open Source Software (FLOSS)



Free/Libre Software

FREE SOFTWARE FOUNDATION

- Guidato da Free Software Foundation (1985)
- Enfasi sulla libertà dal controllo di altri e su questioni etiche:

Il software libero è una questione di libertà: ognuno deve essere libero di utilizzare il software in ogni modo che sia socialmente utile.

Open Source Software

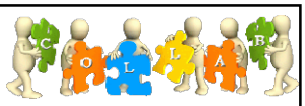
open source initiative

- Guidato da Open Source Initiative (1998)
- Enfasi sui vantaggi tecnici:

il software non libero è una soluzione non ottimale

Filippo Lanubile

Definizione ufficiale di Free Software

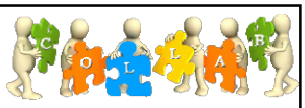


Da http://www.gnu.org/philosophy/free-sw.html

L'espressione "**software libero**" si riferisce alla libertà dell'utente di eseguire, copiare, distribuire, studiare, cambiare e migliorare il software

- ***Libertà 0:*** libertà di eseguire il programma, per qualsiasi scopo
- ***Libertà 1:*** libertà di studiare come funziona il programma e adattarlo alle proprie necessità
- **Libertà 2:** libertà di ridistribuire copie in modo da aiutare il prossimo
- **Libertà 3**: libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio

Definizione ufficiale di Open Source

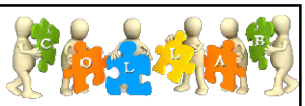


Da http://www.opensource.org/docs/definition.php

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

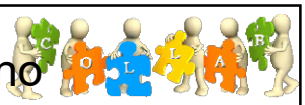
1. Free Redistribution
2. Source Code
3. Derived Works
4. Integrity of The Author's Source Code
5. No Discrimination Against Persons or Groups
6. No Discrimination Against Fields of Endeavor
7. Distribution of License
8. License Must Not Be Specific to a Product
9. License Must Not Restrict Other Software
10. License Must Be Technology-Neutral

Progetti FLOSS



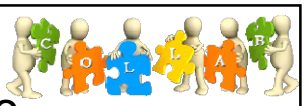
- Un progetto FLOSS produce software la cui licenza dà agli utenti la libertà di:
  - eseguire il programma per qualsiasi scopo
  - studiare e modificare il programma
  - ridistribuire copie del programma originale o del programma modificato
- L'opposto di FLOSS è software "chiuso" o "proprietario"
- Circa 70 licenze approvate da OSI
  - GNU General Public License (GPL). GNU Library or "Lesser" General Public License (LGPL), BSD 3-Clause "New" or "Revised" license, BSD 2-Clause "Simplified" or "FreeBSD" license, MIT license, Apache License 2.0, Mozilla Public License 2.0, Eclipse Public License, ecc.
  - Tutte le licenze permettono la vendita commerciale e l'uso commerciale del software

Quali tecniche e strumenti sono utilizzati per i seguenti problemi?

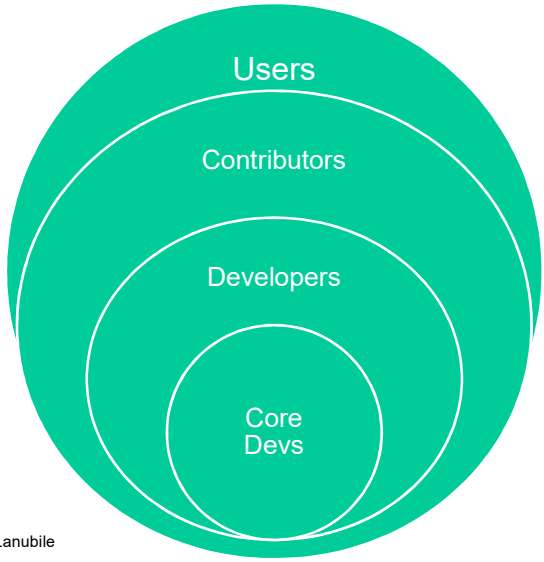


- Processo decisionale
- Gestione delle Release
- Manuali
- Punto di contatto
- Comunicazione
- Coordinamento
- Condivisione codice
- Assicurazione della qualità
- Hosting e integrazione degli strumenti

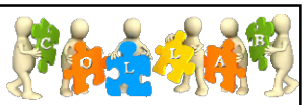
Processo decisionale: Come sono prese le decisoni?



Struttura gerarchica (onion model)



Processo decisionale: Come sono prese le decisoni?



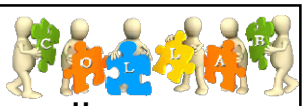
Struttura gerarchica con leader di progetto

- Esempio: Linux kernel
- Il leader ha autorità ultima su tutte le decisioni
- Può delegare a suoi fiduciari

Struttura gerarchica meritocratica

- Es.: Apache HTTP Server
- Comitato di developers, tra loro pari
- Un contributor può diventare developer per cooptazione, sulla base dei contributi offerti e della fiducia generata

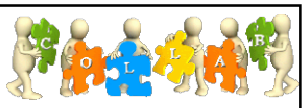
Gestione delle Release: come fanno gli utenti a prendere il risultato finale di un progetto?



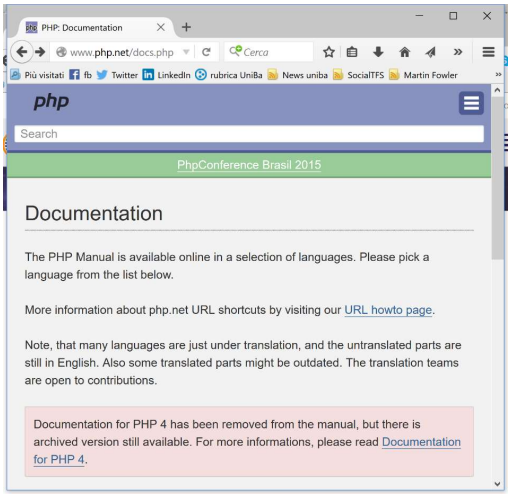
- Download di release già compilate o da compilare da forges, web servers o ftp servers
- Mirroring
  - Esempio: Debian Mirrors,
- Etichettatura delle versioni: stable, testing, unstable
  - Esempio: Debian Releases

Filippo Lanubile

Manuali: come imparano gli utenti a usare il sistema?

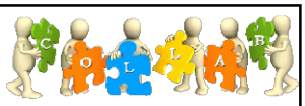


- Sottoprogetti dedicati alla redazione di documentazione
  - Esempio: PHP
- Possibilità di aggiungere commenti da parte degli utenti



Filippo Lanubile

Punto di contatto:
a chi si può rivolgere un utente per risolvere un problema?



- Azienda che fornisce supporto a pagamento
  - Esempi: PostgreSQL, (Oracle) MySQL
- Fondazioni no-profit
  - Gestiscono l'infrastruttura del progetto (es. sito web, mailing list)
  - Esempi: FSF, FreeBSD Foundation, Apache Software Foundation, Linux Foundation
- Sviluppatori
  - Possibilità di *follow* nei forge moderni (es. GitHub)

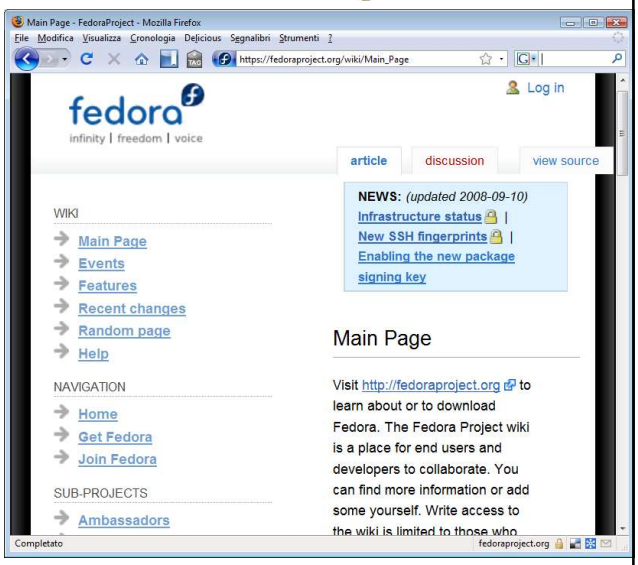
Comunicazione:
come sono scambiate le informazioni?

- Strumenti
  - Mailing list
  - Newsgroup
  - Internet Relay Chat (IRC)
  - Wiki
  - Blogs
  - Microblogs
  - Question&Answer sites

- Prevalenza di comunicazione asincrona
- Diffusione crescente di social software moderno (web 2.0)

Wikis

Sharing of explicit knowledge

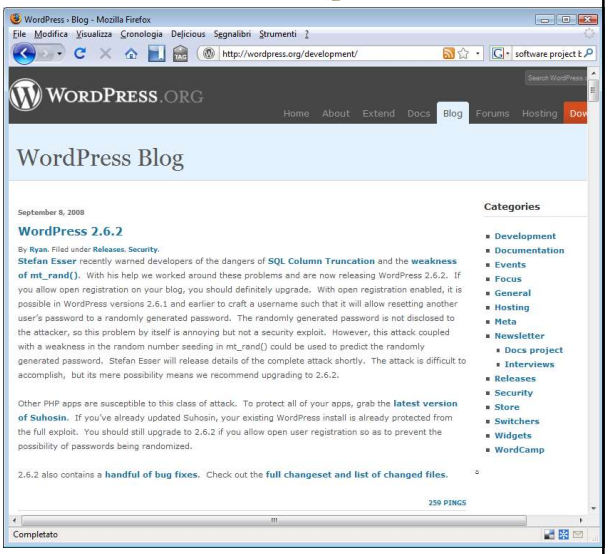


Filippo Lanubile

https://fedoraproject.org/wiki/Main_Page

Blogs

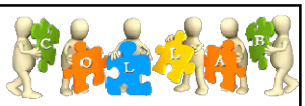
Promote sharing of tacit knowledge by narratively recording projects' events



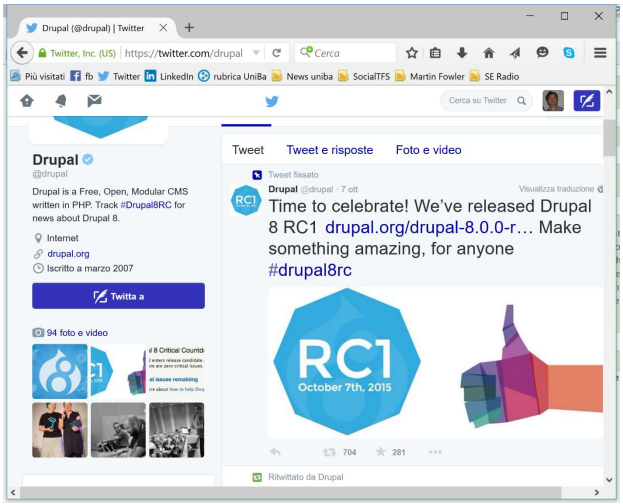
Filippo Lanubile

http://wordpress.org/development/

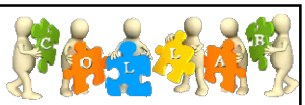
Microblogs

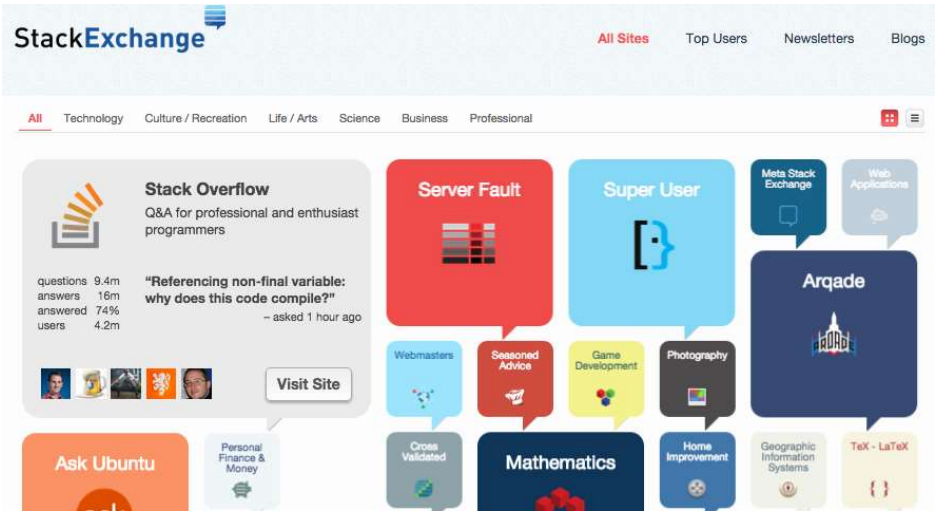


- Tweets to promote sharing of status updates
- Twitter-like interactions within GitHub (@mentions)



Social Q&A sites





35

2

Let's say I have a `DataTable` with a `Name` column. I want to have a collection of the unique names ordered alphabetically. The following query ignores the order by clause.

```
var names =
    (from DataRow dr in dataTable.Rows
    orderby (string)dr["Name"]
    select (string)dr["Name"]).Distinct();
```

Why does the `orderby` not get enforced?

c# linq .net-3.5

share | improve this question

edited Jan 1 at 16:43

l3arnon
14.7k • 4 • 30 • 51

asked Aug 1 '08 at 13:14

Bob
37.1k • 15 • 87 • 102

add a comment

5 Answers active oldest votes

7

To make it more readable and maintainable, you can also split it up into multiple LINQ statements.

1. First, select your data into a new list, let's call it `x1`, do a projection if desired
2. Next, create a distinct list, from `x1` into `x2`, using whatever distinction you require
3. Finally, create an ordered list, from `x2` into `x3`, sorting by whatever you desire

share | improve this answer

edited Oct 14 '12 at 12:22

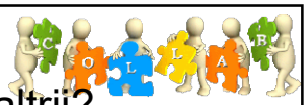
flem
13.3k • 4 • 22 • 59

answered Sep 4 '08 at 2:57

a7drew
4,195 • 3 • 22 • 34

2 @Bob's answer seem the best and uses the lease lines of code – CodeBlend Aug 6 '12 at 16:00

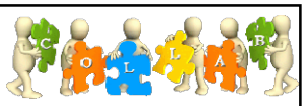
Coordinamento:



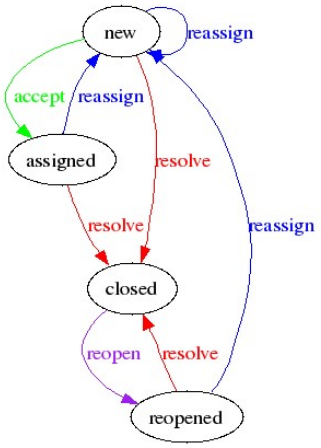
Come si scopre cosa fanno gli altri? Come viene incoraggiata e controllata la partecipazione?

- Pubblicazione di tutorial, linee guida, FAQs sul processo di sviluppo e gli strumenti utilizzati
  - Es: KDE TechBase
- Uso degli strumenti di comunicazione
- Pubblicazione dei problemi aperti e delle assegnazioni di lavoro
- Strumenti: Issue tracking system (es: Bugzilla, JIRA)

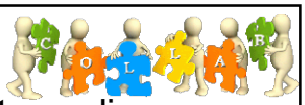
Issue tracking system



- Database with a web UI
- Issues (or tickets) as a generalization of
  - bugs/defects, enhancements, tasks
- Built-in life cycle to track the resolution



Condivisione codice:



Come lavorare contemporaneamente sugli stessi file di codice sorgente?

Gestione della configurazione e controllo delle versioni

- Presenza di un repository da cui prelevare codice (check-out) e riporre codice (check-in o commit)
- Gestione delle modifiche effettuate contemporaneamente a uno stesso file da parte di due membri differenti
- Gestione delle diverse versioni di un prodotto che può evolvere
- Ripristino di vecchie versioni di prodotti
- Individuazione di versioni del prodotto in cui sono stati risolti errori o introdotte nuove funzionalità

Condivisione codice:

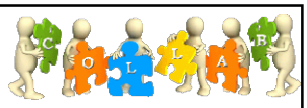
Come lavorare contemporaneamente sugli stessi file di codice sorgente?

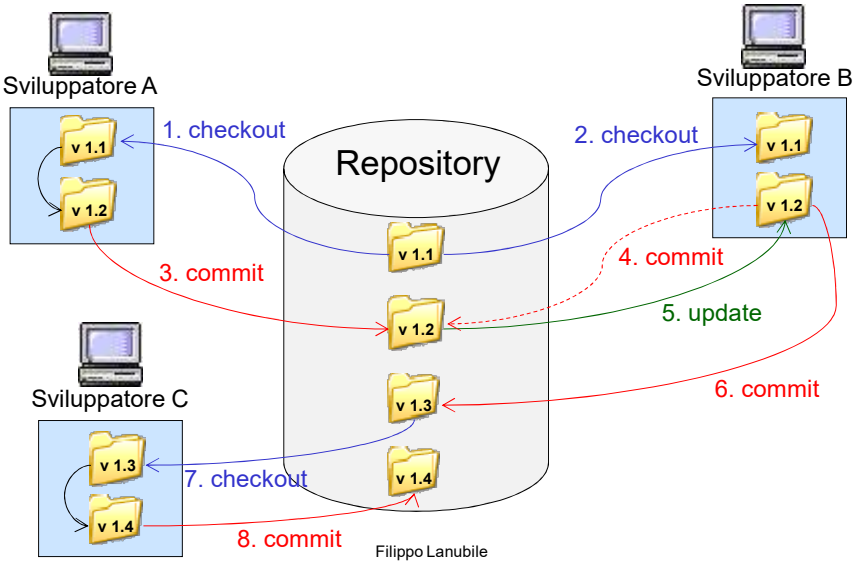
Version control systems (VCS)

- Pessimistic vs. ***Optimistic model***
  - lock-modify-unlock vs. copy-modify-merge
- Centralized vs. ***Distributed server***
  - Unique server (CVS, SVN) vs. multiple servers, one for each developer (Git)

Filippo Lanubile

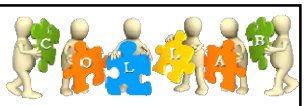
Modello copy-modify-merge





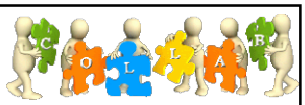
Filippo Lanubile

Distributed VCS



- Modello P2P
- Server multipli: ogni repository è creato (fork) come un clone completo
- Il commit è locale
- Per restituire la modifica bisogna renderlo pubblico (push o pull request)
- Es. Git
  - Sviluppato da Linus Torvalds per il progetto Linux

Common branching workflow in git



Main branch

aka long-running

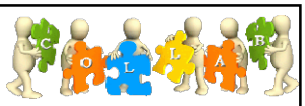
- master
  - codice stabile, pronto per andare in produzione
  - build stabili
- develop
  - cambiamenti al codice, pronti per essere integrati nella prossima release
  - build nightly

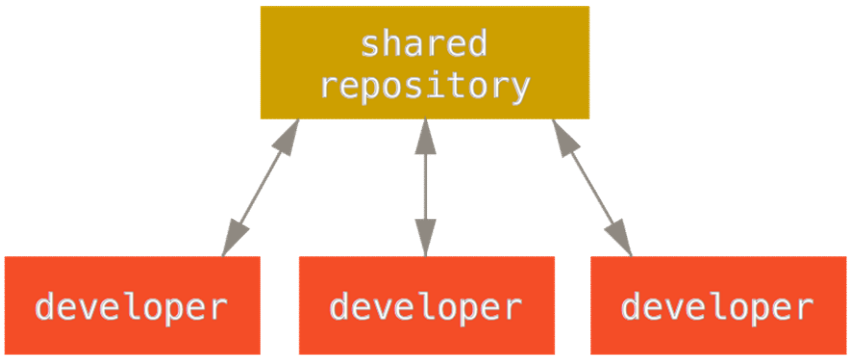
Topic branch

aka short-lived, supporting

- feature
  - aggiunta di nuove funzionalità
  - e.g., *filedownload*, *logging*
- issue
  - risoluzione di un issue specifico
  - e.g., issue3, iss91a, iss91v2
- hotfix
  - risoluzione di difetti gravi di codice in produzione

Centralized workflow

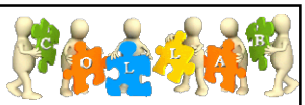


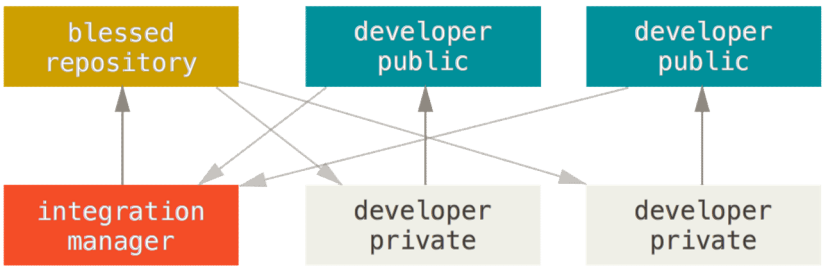


Possibile con VCS centralizzati e decentralizzati

Filippo Lanubile

Integration-Manager workflow

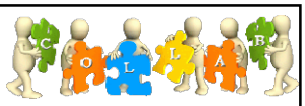


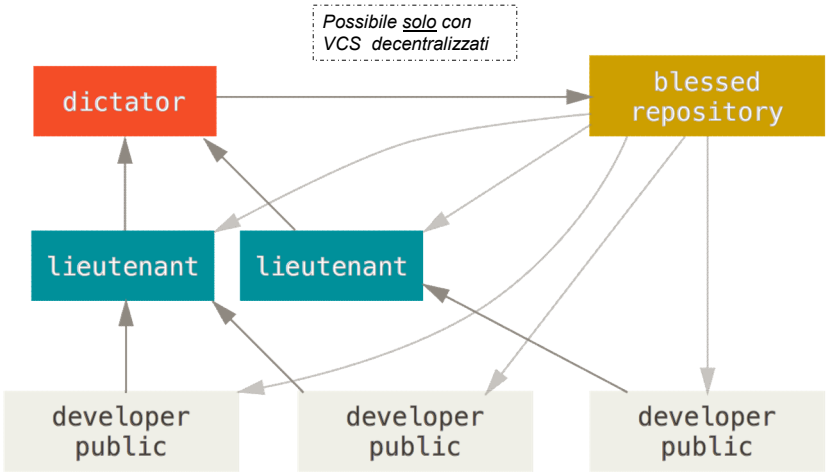


Possibile solo con VCS decentralizzati

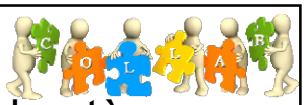
Filippo Lanubile

Dictator and Lieutenants Workflow



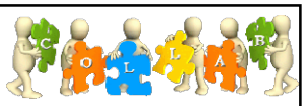


Assicurazione della qualità: come si acquista fiducia nella bontà del risultato finale?



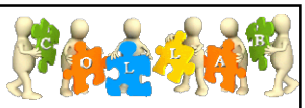
- Revisioni del codice applicate a cambiamenti autoconsistenti e frequenti
  - Patch via mailing lists
  - Pull request via DVCS
- Beta-test in larga scala
  - Linus Law: "*Given enough eyeballs, all bugs are shallow*" (cit. Eric Raymond)
    - *Dato un numero sufficiente di occhi, tutti i bug vengono a galla*

Hosting e integrazione degli strumenti



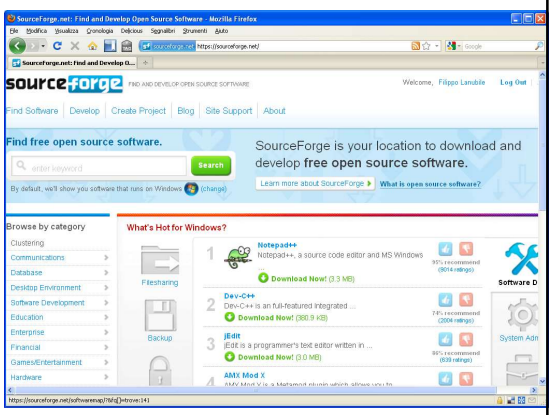
- Forge platforms
  - Host multiple independent projects
  - Collaborative Development Environments that integrate SE task-specific tools and generic communication tools

SourceForge

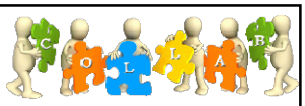


- Version-Control Systems
  - SVN, Git, Mercurial
- Trackers:
  - Bugs, feature requests, patches, support requests
- Communication tools
  - Mailing lists; forums
- Web 2.0 applications
  - Feeds; hosted applications for blogs, microblogs and wikis

https://sourceforge.net/



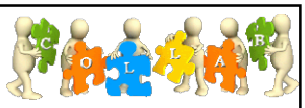
Launchpad



- Version-Control Systems
  - Bazaar
- Trackers:
  - Bugs; integrating with external trackers
- Communication tools
  - Mailing lists
- Web 2.0 applications
  - No

http://launchpad.net

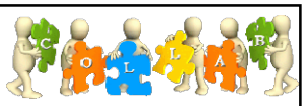
CodePlex



- Version-Control Systems
  - Git, Mercurial, Team Foundation Server
- Trackers:
  - Work items (features, issues, tasks)
- Communication tools
  - Mailing lists, discussions list
- Web 2.0 applications
  - Feeds, wiki

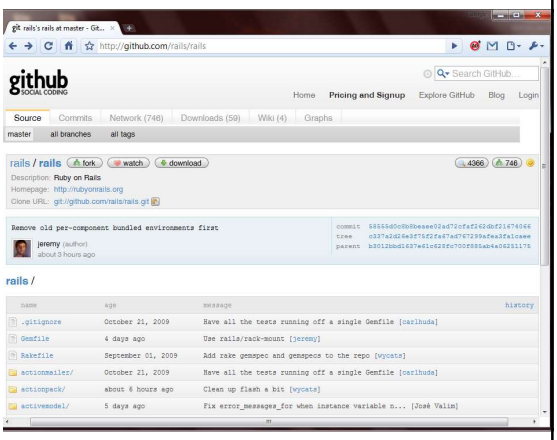
www.codeplex.com

GitHub

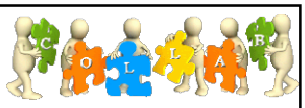


- Version-Control Systems
  - Git , SVN
- Trackers:
  - Issues
- Communication tools
  - No
- Web 2.0 applications
  - Feeds, wiki, social networks
- Collaborative code review
  - Pull requests

https://github.com

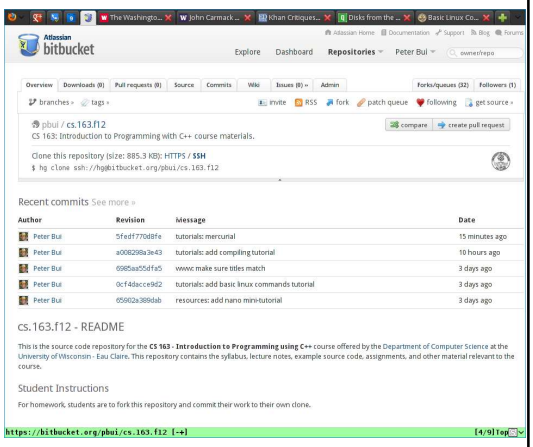


BitBucket

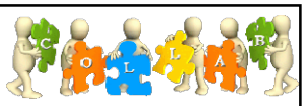


- Version-Control Systems
  - Git, Mercurial
- Trackers:
  - Issues (integrated with Jira)
    or via FogBugz, Lighthouse
- Communication tools
  - Chat
    (via HipChat or Campfire),
    IRC (via Groove)
- Web 2.0 applications
  - Feeds, wiki, social networks

https://bitbucket.org/



Stage e Tesi di laurea



- Contributi significativi a un progetto FLOSS possono coincidere con il lavoro di tesi
  - Mario Scalas: KDevelop
  - Myriam Leggieri: Hackystat
  - Pasquale Minervini: Apertium
- Lavorare a un progetto FLOSS è una forma di stage
  - Il riconoscimento formale dipende dal docente
- Google Summer of Code
  - *Global program that offers students stipends to write code for open source projects.*