Pair Programming sotto 4 lenti

...ovvero tutto quello che è possibile sapere sul PP (e che magari manco vi interessava)

C’est moi

Marcello Missiroli

Prof. di Info: IIS Corni, Unimore, Unibo

Di C++ so il minimo. Per cui parlerò di altro!

> Every teaching activity should be observed by four lenses: the autobiographical, the students' eyes, our colleagues' experiences, and theoretical literature.
>
> – S. D. Brookfield

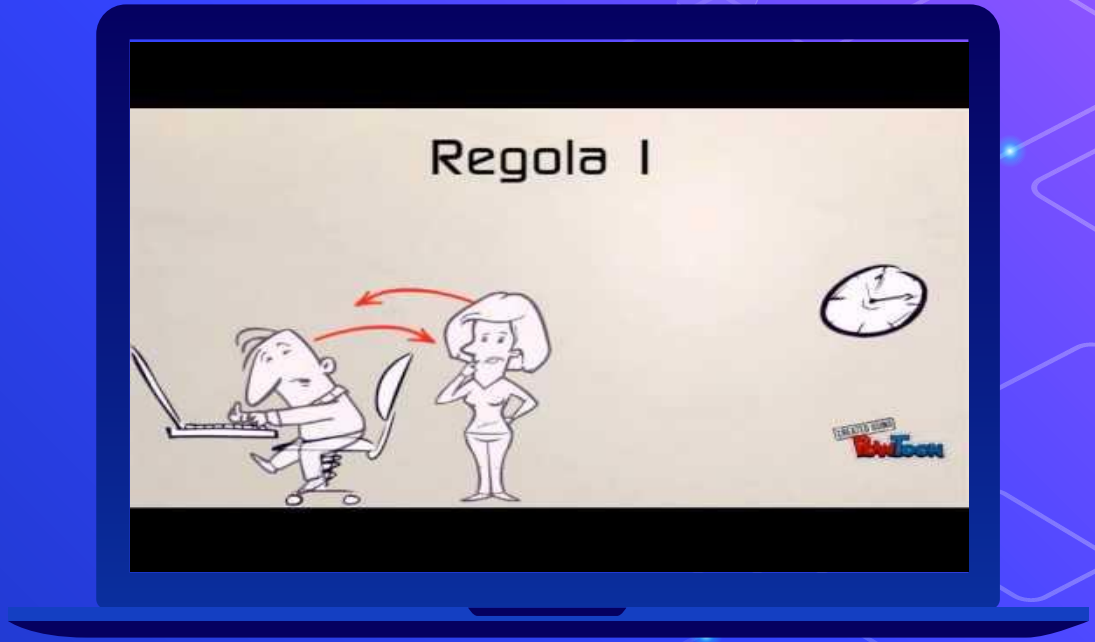
> Guarderemo il Pair Programming attraverso quattro lenti: lente autobiografica, della letteratura, dei studenti e dei peer.

Agenda

1. Introduzione
2. Lente 1 - Academia
3. Lente 2 - Personale
4. Lente 3 - Studenti
5. Lente 4 - Peers
6. Tricks & Tools

1. PP:
di che si tratta?

Microvideo

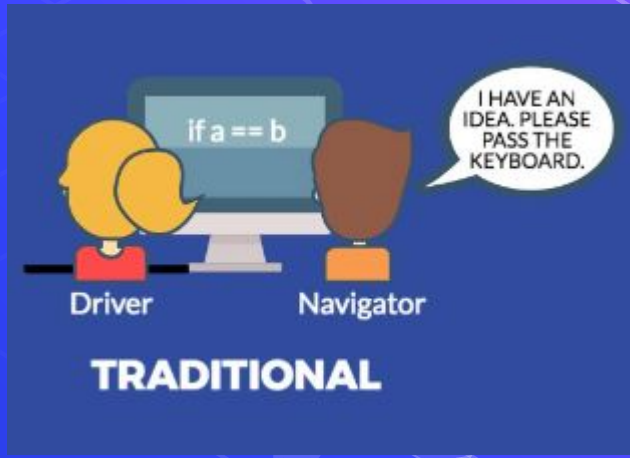


Cos'è

- È una tecnica di programmazione in cui di lavora in due, seguendo particolari regole
- È considerata una delle core practices dei metodi agili (specificamente in XP)
- Disponibile in diversi stili

PP Tradizionale

- 2 ruoli mutuati dal rally: Driver e Navigator.
- Il Driver controlla la tastiera e scrive il codice
- Il Navigator controlla e offre una prospettiva a lungo periodo
- Role switch periodico (minuti, ore, feature, richiesta)



Strong style PP

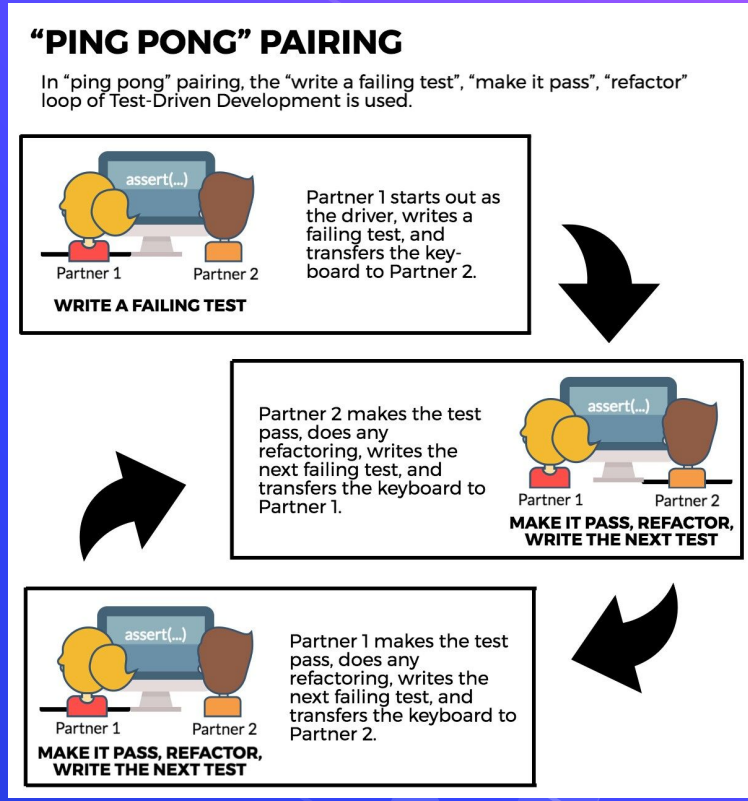
- Il Navigator decide cosa fare (alto livello)
- Il Driver decide come farlo (basso livello) ma non prende iniziative
- *"For an idea to go from your head into the computer it MUST go through someone else's hands"*

Tour Guide Style PP

- Il Driver decide, scrive, e descrive quello che sta facendo
- Il Navigator ha un ruolo più passivo e interviene solo se non capisce o non è d’accordo.
- Rischia di degenerare per eccessiva passività

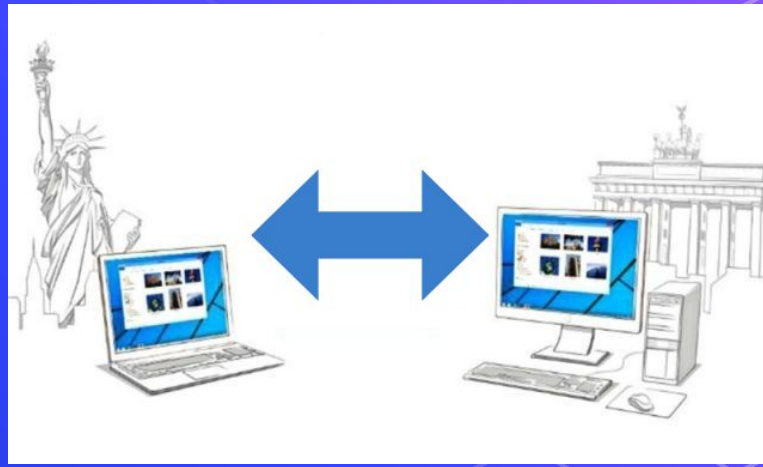
Ping-pong PP

- Tipicamente usato assieme al TDD
- Il Navigator scrive il test che fallisce
- Il Driver risolve il test, quindi si scambia di posto.



Remote PP

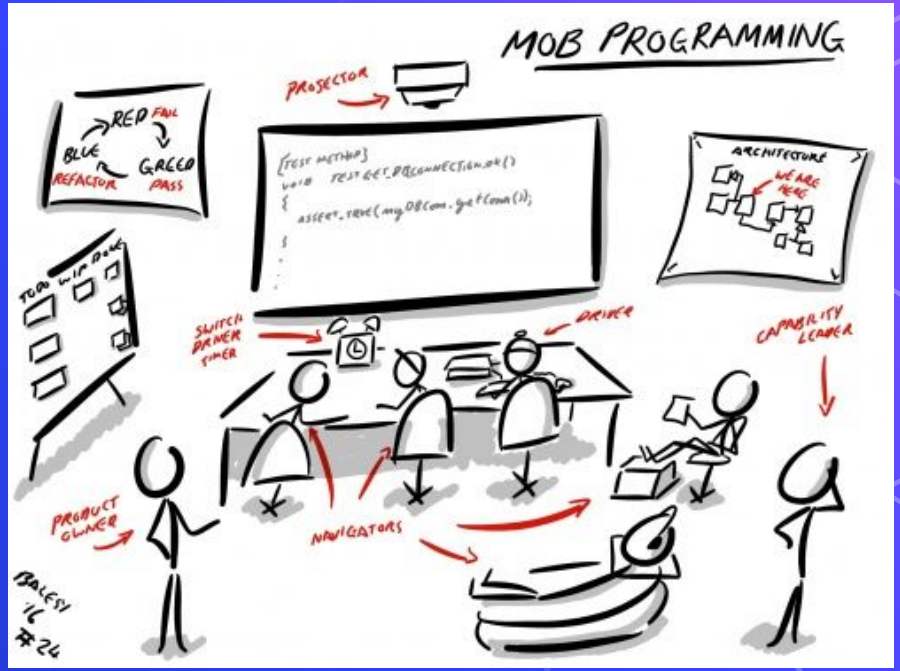
- Pair programming con partner non presenti fisicamente.
- Introduce una serie di problematiche complesse che tratteremo alla fine



Mob Programming

Variante in cui molti (tanti, tutti) i membri del team partecipano alla scrittura del codice.

Un driver, tanti navigator, possibili “consiglieri” ed “esperti”



2.
Letteratura

Cosa dicono gli studi universitari sull’argomento?

Definiamo il contesto

Anedottica

Risultati di esperienze personale, sporadiche, racconti, post su forum o social.

Esperimenti

Risultati di situazioni controllate, ripetibili, con gruppi di controllo, con valutazioni statistiche molto forti.

Situazione attuale

- A tutt’oggi, non esiste alcuno s**tudio scientifico** che dimostri che il Pair Programming porti complessivamente vantaggi (o svantaggi) rispetto alla programmazione in solitario.

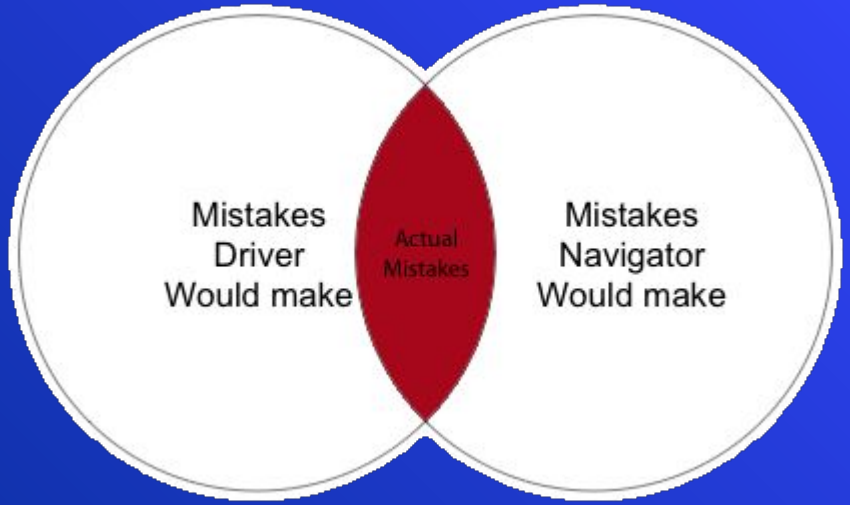
Tuttavia..

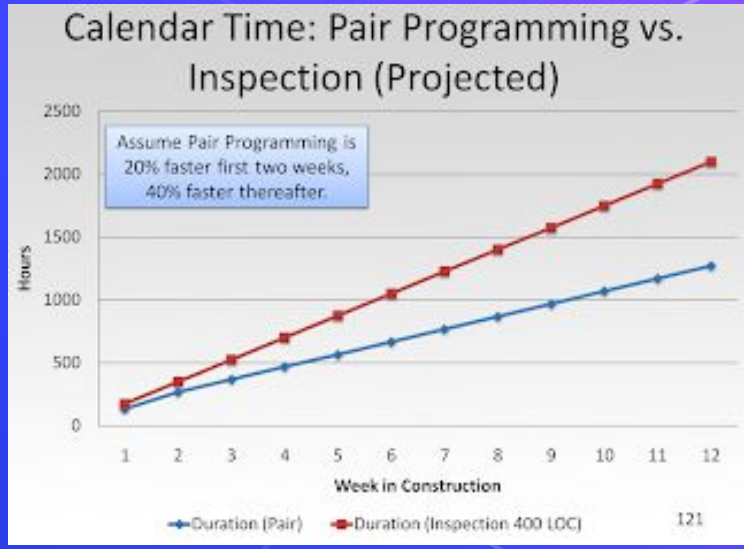
- Vi sono moltissime prove anedottiche sui vantaggi del PP, meno sui suoi svantaggi (possibile **bias**)
- Esistono studi che provano vantaggi o svantaggi in particolari ambienti/situazioni

Saggezza anedottica

- PP produce meno codice (-15% rispetto) ma mediamente più corretto (-50% errori)
- Un buon sistema per condividere informazioni, conoscenze, cultura aziendale (pairing junior/senior)
- Problemi di personalità e con il management

Saggezza anedottica





Studio 1

- Dati raccolti in 14 mesi in un team di sviluppo.
- Risultati inconsistenti. : minor numero di difetti quando il codice è modificato

Table 10. The summary of Q1-Q5, presenting the situations in which pair programming has been effectively used to reduce defects on the basis of t-test.

Id	The situation in which pair programming was used	Have the defects been decreasing?
Q1	For implementing changes in general	yes
Q2	Prior to the defect detection	no
Q3	For the defect correction	yes
Q4	Prior to the user story implementation	yes
Q5	For the user story implementation	no

Pair Programming and Software Defects – an Industrial Case Study Nattakarn Phaphoom, Alberto Sillitti, Giancarlo Succi

Studio 2

- Dati raccolti confrontando le test suite realizzate da Solo & Pair programmers
- Contraddice la versione anedottica: PP non produce differenze significative

Impact of pair programming on thoroughness and fault detection effectiveness of unit test suites - Lech Madewski

Studio 3 (review)

- 73% degli impiegati notano una accelerazione dello sviluppo
- Discussioni costruttive
- Studenti: utile ma talvolta problematico
- Positivo per l'apprendimento
- Generalmente inefficiente
- Stancante

Empirical studies of agile software development: A systematic review - Tore Dybå˚, Torgeir Dingsøyr

2. Esperienza personale

Chennesò io sul Pair Programming?

Esperienza diretta: no

- Non sono uno sviluppatore!
- Scrivo programmi didattici, toy projects, “spike” di specifiche librerie/frameworks/tecniche
- Già ho poco tempo per farlo, farlo con un collega è impossibile.

Esperimenti

- Primo esperimento: confronti tra studenti solitari e coppia, tenendo presente i diversi gradi di abilità e loro combinazioni. Singolo progetto con PP, TDD
- Secondo esperimento: confronto tra progetti sviluppati tramite waterfall e scrum (PP consigliato ma non imposto)

Missiroli, Marcello, Daniel Russo, and Paolo Ciancarini. "Learning Agile software development in high school: an investigation."ACM, 2016.
Missiroli, Marcello, Daniel Russo, and Paolo Ciancarini. "Agile for millennials: a comparative study." SECM. IEEE, 2017.

Impressioni

- Due fattori fortemente discriminanti
  - Abilità individuali
  - Personalità

Missiroli, Marcello, Daniel Russo, and Paolo Ciancarini. "Learning Agile software development in high school: an investigation."ACM, 2016.
Missiroli, Marcello, Daniel Russo, and Paolo Ciancarini. "Agile for millennials: a comparative study." SECM. IEEE, 2017.

Impressioni (abilità)

- Abilità individuali:
  - Professionisti: senior/junior
  - Studenti: capaci/incapaci
- Difficile fare tutte coppie funzionali - scarsi con scarsi non funziona!. L'ideale è avere un certo differenziale di abilità, né troppo piccolo né troppo grande
- Senior/senior o capaci/capaci può funzionare, ma non sempre.

Impressioni (personalità)

- Alcune persone sono inguaribili cowboy
- Tenere conto di amicizie e inimicizie (senza esagerare in nessuno dei due sensi)

3.
Opinione degli studenti

Che ne pensano gli studenti?

Come insegnare PP?

- Video
- Coding Dojo (Ping pong style)
- Mini code retreat
- Tipicamente usato lo stile tradizionale

PP & gli studenti: verifica

- Piccoli progetti, esperienze, lavoro in classe.
- Riscontro tramite osservazione diretta, relazioni (soprattutto all’università)
- Analisi di esperimenti accademici

Impressioni

- Funziona se imposto o proposto su progetti di breve durata
- Molto difficile da acquisire come pratica standard collaborativa nel mondo scolastico / universitario in cui la competizione è ancora il motore primario
- Buono per cercare di condividere le conoscenze, meno per la produttività (iper-mini toy projects)

3. Opinioni ed esperienze dei peer

Docenti

- Francamente, pochi lo usano / insegnano /propongono
- Proposto in forma blanda, spesso per necessità (es: 30 studenti e 20 PC).
- Spesso decade in YouTube programming: uno fa, l'altro guarda come se si trattasse di un video
- Apprezzato a livello teorico

Practitioners

- That’s you, baby
- Vediamo come la vedete

WWW.MENTI.COM

Big concept

Bring the attention of your audience over a key concept using icons or illustrations

Tricks & Tools

Pair Programming & Git
(Github/Bitbucket)

Riconoscimento

- Se si lavora in due, è giusto avere il riconoscimento
- Spesso Git è usato per analisi, profilazione, ecc. Come fare?

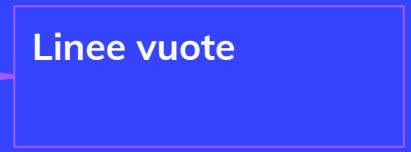
Co-authored tag.

- Quando Git è nato, il PP non era diffuso come ora.
- Recentemente è stato introdotto un nuovo tag “standard”.

Command line

- Durante un commit, nel messaggio LASCIARE ALMENO UNA RIGA VUOTA nel commento poi aggiungere i tag necessari

```
$ git commit -m "Esempio di commit.
>
>
Co-authored-by: piffy
<piffy@gmail.com>
Co-authored-by: prof.missiroli
<prof.missiroli@corni.it>"
```



Command line: add-on

- Git pairing session fornisce una serie di bash script che rendono l'uso del pp con git molto naturale.
- Occorre configurazione.

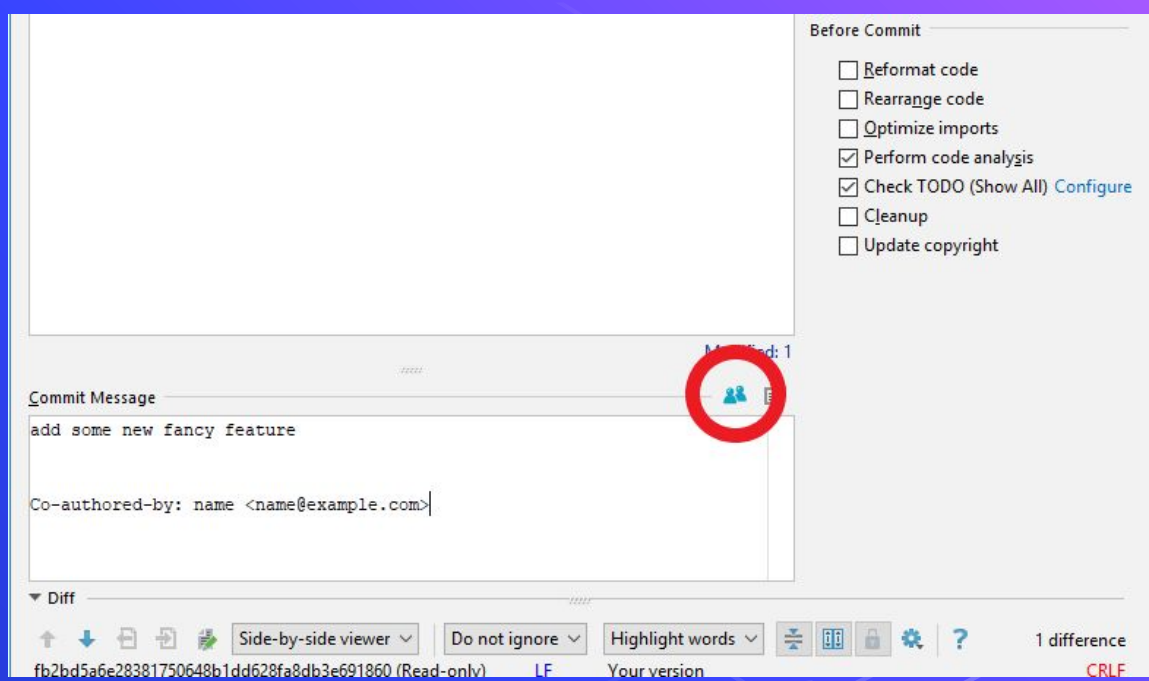
```
$ git commit -m "Esempio di commit.
>
>
Co-authored-by: piffy
<piffy@gmail.com>
Co-authored-by: prof.missiroli
<prof.missiroli@cerni.it>"
```

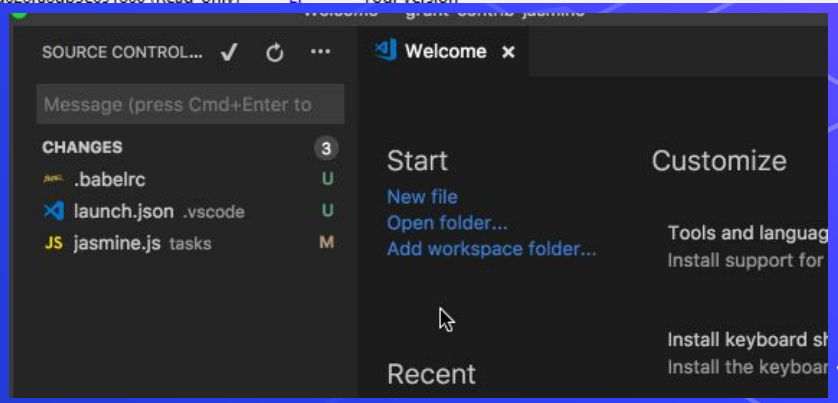
Su Github

- Github ha una interfaccia molto semplificata per farlo.

IDE

- Vari IDE si stanno attrezzando in questo senso.
- Jetbrains ha un apposito plugin: Co-author
- Visual studio ha Git Co Authors





Tricks & Tools 2

Remote pairing

Casi d’uso

1. Progetto diviso in tanti luoghi fisici
2. Lavori per una “ditta virtuale”
3. Il progetto ha sviluppatori remoti
4. Utile per condivisione conoscenze
5. Serve la consulenza di un esperto lontano
6. Fate un’intervista/esame
7. I, vostro partner è assente

Consigli per vivere sereni

CERCARE DI

- Essere pazienti
- Fare pause
- Coinvolgere gli sviluppatori remoti
- Usare webcam

EVITARE DI

- Escludere i remoti da conversazioni o riunioni
- Parlare poco
- Appoggiare le cuffie

Come “sharearci”?

- Solo testo?
- Audio?
- Screen Sharing?
- Webcam?

Text-only

- Terminal (tmux)
- Old style chat (Jabber, irc)
- New style chat (Hipchat, Discord, Slack, Hangouts, Riot, Social)
- Shared online text (pirate pad)
- CVS (Git/Bitbucket)

+ Veloce, ubiquo
+ Zero costi

- Poca/no grafica
- Interazione difficile
- Unicast (in genere)
- Sicurezza

Text+Audio

- GP Tools (Skype [FB], Cellulare, Whatsapp, Teamspeak)
- Audio conferencing (meets, webex)
- New Style Chat + plugins

\+ Veloce
\+ Costi limitati
\+ Broadcast

\- Poca/no grafica
\- Interazione non ottimale

Text+Audio+Screensharing

- Conferencing software (meets, webex)
- New Style Chat + plugins

+ Abbastanza veloce
+ Costi limitati

- Richiede stesso tool
- Controllo

The Full Monty

- Video conferencing
- New Style Chat + plugins
- Tool specifici

+ Esperienza immersiva
+ Broadcast

- La banda è il limite
- Richiede stesso tool

Vincoli generali

- Cross platform?
- Usabile attraverso VPN/NAT ?
- Web browser?
- Controllo reciproco?

Generale o specifico?

Per tool specifici si intendono strumenti espressamente pensati per i programmatori, spesso con il PP in mente.

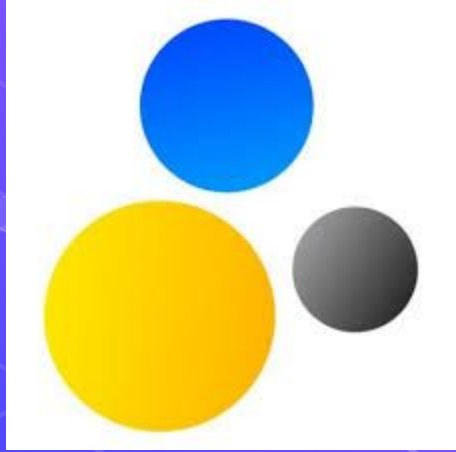
Ecco qualche esempio.

Floobits

- Open source, Freemium
- Richiede Github
- Integrazione diretta per IntelliJ, Atom, Neovim
- Usabile con varie chat

Saros

- Plugin per Eclipse e IntelliJ
- Richiede server Jabber
- Anche per IntelliJ
- Open source, Free
- Poco usable



Teletype

- Plugin per Atom (solo)
- Ancora in beta
- Richiede Github
- Broadcast

Microsoft Live Share

- Plugin per Visual Studio Code (solo)
- Freeware (ma in futuro Freemium)
- Debug condiviso
- Esplorazione indipendente
- Pare buono (però è Microsoft)

Cloud development

- Cloud9 IDE:
  - Totalmente online,
  - Ben fatto
  - Richiede account AWS
- Eclipse Che
  - Concepito per Kubernetes
  - Massicciamente multi user
  - Red hat

..ebbasta!!

Ho parlato troppo?

Conclusioni

1. Indimostrabile dire se in assoluto “conviene”
2. Tanti stili, derivanti da varie esigenze
3. Diffusione ignota
4. Utile per condivisione conoscenze
5. Codice più corretto (maybe)
6. Didatticamente interessante, ma poco applicato
7. So what?

Giudizio finale

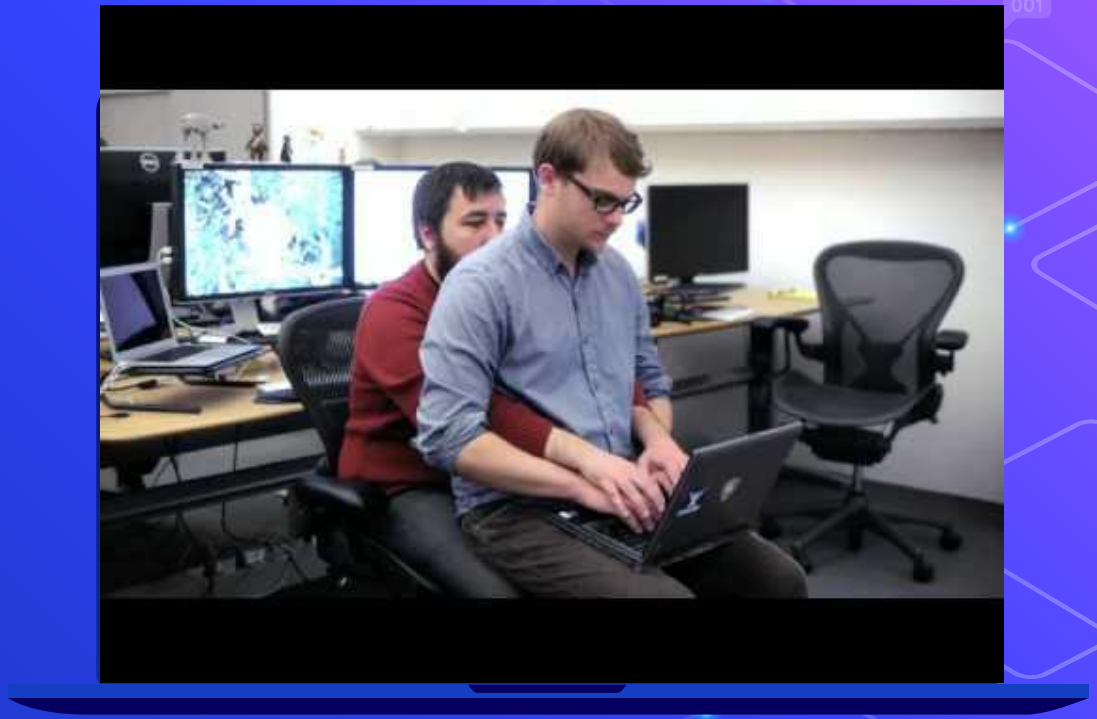
Dipende dai parametri scelti

1. Tempo di sviluppo
2. Tempo di debug/refactoring
3. Placere del programmatore
4. Soddisfazione cliente
5. Cultura aziendale
6. Soldi

Ma soprattutto... it's all about people.

Thanks!

Some fun on the right...



Credits

- Template by SlidesCarnival
- Video di Atlassian
- Mio video (Powtoon)

Per tutto il resto, questa presentazione è sottoposta a licenza CC: BY-SA