Appunti oop 21 novembre 2022

DVCS Workflow (slide 09)

Lavoro di squadra con i dvcs

Obiettivo: disciplinare il workflow

Che modello adottiamo?

Dipende dai trade-off: team molto grande? Software molto complesso? Quanti sviluppatori interagiscono su parti simili, condivise, ecc?

Nell’immagine: più a destra c’è il branch master, ed è sempre stabile a livello di funzionalità software; ha pochi commit, e corrispondono ai rilasci stabili del software. I commit sono correlati da dei Tag.

In hotfixes si risolvono i bug del master.

Il branch develop, quello giallo al centro, è la linea principale dello sviluppo.

In develop vogliamo gestire features applicative (es gestione della configurazione utente, gestione dei boundaries del livello del gioco, ecc.) I features branches possono essere infiniti e ci ficchiamo i features. Ogni volta che vogliamo farle interagire facciamo il merge su develop, non le incolliamo fra di loro.

Quando il develop è pronto si può aprire la release branch dove fare vari commit che agiscano sulla versione del software o su alcuni bug, poi facciamo i merge su master e si fa il tag. Questo si chiama **git-flow** e ci sono molti dettagli nel link delle slide.

Per convenzione i branch si chiamano ad esempio *feature-level-boundaries*; *bug-nome-bug-da-risolvere*; *wip* (work in progress, attività molto complessa che sappiamo che non finirà presto); a volte si usa / per separare i nomi dei branches, usabile anche da certi strumenti per la rappresentazione ad albero del progetto. (es. *feature/nome-feature*)

Esistono supporti alla scrittura di messaggi di commit efficaci, i **conventional commit** (probabilmente c’è un link sulle slide) → commit limp? (nome di software utile forse, non ho capito il nome)

L’utilizzo corretto dello strumento git è valutato nel progetto oop.

È importante mantenere tutte le linee di sviluppo secondarie il più brevi possibili, e fare spesso sincronizzazione, per avere eventuali merge conflicts non troppo complessi da risolvere.

In progetti complicati e con team molto eterogenei, dove la gente deve occuparsi di controllare il codice scritto dagli altri, c’è un solo maintainer del repository che avrà i diritti di scrittura, e tutti gli altri fanno un **fork**. Poi si aprono i pull request dalle varie fork. Non è un servizio built-in in git ma c’è su GitHub.

Ci viene consigliato dai prof per il progetto di usare il workflow semplice, ma possiamo fare anche la modalità avanzata con i fork e le pull request.

**Profiling di applicazioni con VisualVM**

Valutazione delle performance di un progetto (tempo di esecuzione e risorse occupate, tipo memoria).

Approcci invasivi: significa quelle tecniche che includono la strumentazione del software (es. l’aggiunta di alcune operazioni in corrispondenza dell’ingresso in un metodo o dell’uscita da un metodo per tenere traccia di alcune informazioni).

**Startup Profiler –** plugin di VisualVM (per installarlo: Tools→ si apre la finestra con la barra di ricerca dove scrivere il nome del plugin; permette di monitorare il programma fin dal momento della partenza).