**GESTIONE DEI CAMBIAMENTI, IL CONTROLLO DI VERSIONE**

*Lo scopo di questo capitolo è quello di motivare l’uso di software per il controllo di versione, descrivendo le modalità di applicazione.*

**Problematica**

Quando si realizza del software, vi sono sempre dei cambiamenti e poiché tali cambiamenti si verificano, occorre gestirli in maniera efficace. La *gestione dei cambiamenti* (change management), talvolta detta anche configurazione del software (software configuration management), è un insieme di attività progettate con lo scopo di tenere sotto controllo i cambiamenti tramite l’identificazione dei prodotti che possono cambiare, la definizione delle relazioni fra di essi, la definizione dei meccanismi per la gestione delle differenti versioni di questi prodotti, il controllo delle modifiche imposte e producendo versioni e report sulle modifiche eseguite. All’interno di un corso universitario di Architetture Software, in cui gli studenti sono chiamati a realizzare progetti che utilizzano tecnologie di frontiera e dove spesso tali tecnologie devono essere integrate le une con le altre, è possibile pensare che lo sviluppo di un elaborato passi attraverso diverse versioni. Spesso l’attività di progettazione non è eseguita da un singolo studente ma, per iniziare ad abituare il discente ad un contesto industriale, in team. Per tali ragioni è bene indurre gli studenti ad utilizzare un sistema per il controllo delle versioni in modo che la loro attenzione sia rivolta maggiormente sull’aspetto sperimentale piuttosto che su quello organizzativo.

**Gestione dei cambiamenti**

I cambiamenti sono inevitabili nella realizzazione di software ed accrescono la confusione in chi prende parte ad un progetto. La confusione nasce quando i cambiamenti non vengono analizzati preliminarmente, registrati prima di essere effettuati, riferiti a tutti gli interessati e controllati in modo da migliorare la qualità e ridurre gli errori. A questo proposito W.A. **Babich** nel testo “*Software Configuration Management*” edito da Addison-Wesley ha affermato

“*L’arte di coordinare lo sviluppo del software al fine di minimizzare […] la confusione si dice* ***gestione delle configurazioni****. La gestione delle configurazioni è l’arte di individuare, organizzare e controllare le modifiche al software in corso di realizzazione da parte di un team. Lo scopo è quello di massimizzare la produttività minimizzando gli errori.*”

La gestione delle configurazioni software (SCM, Software Configuration Management o CM, Change Management) è un’attività ombrello, che copre tutto il processo software. Un cambiamento, in generale, può avvenire in qualsiasi momento, quindi si sviluppano attività di SCM per

* riconoscere i cambiamenti;
* controllarli;
* garantire che siano opportunamente implementati;
* riferire a tutti gli interessati l’avvenuto cambiamento.

Uno degli obiettivi principali dell’ingegneria del software è quello di accrescere la facilità con cui trattare i cambiamenti e ridurre il carico di lavoro necessario in occasione dei cambiamenti stessi.

**Importanza della gestione dei cambiamenti**

“*Se non controllate i cambiamenti, saranno i cambiamenti a controllare voi.*” Tale frase, molto utilizzata nei libri di testo che trattano di ingegneria del software, sintetizza con estrema efficacia quale sia l’importanza della gestione dei cambiamenti. E’ molto facile che un insieme di modifiche incontrollate trasformi un progetto software ben realizzato in un caos. Per questo motivo, un appropiato sistema per la gestione delle configurazioni del software è una parte essenziale delle tecniche di buona gestione dei progetti e di solida ingegnerizzazione del software.

**Gestione delle configurazioni software**

Il risultato del processo software è una serie di informazioni che si possono suddividere in tre grandi categorie:

* Programmi (nelle forme sorgente ed eseguibile)
* Elaborati che descrivono i programmi (sia per i tecnici, sia per gli utenti)
* Strutture dati (contenute nei programmi o esterne).

Gli elementi che compongono l’informazione prodotta in ogni sua parte dal processo software sono globalmente chiamati *configurazione software*. Se ciascun elemento della configurazione si limitasse a generare altri elementi, non si creerebbe molta confusione. Ma, sfortunatamente, nel processo entra in gioco un’altra variabile: le modifiche.

La **gestione delle configurazioni** software è un insieme di attività ideate per gestire le modifiche per tutto il ciclo di vita del software; la si può anche considerare una attività di gestione della qualità del software, applicata in tutte le fasi del processo software.

**Il processo di gestione delle configurazioni**

Il processo di gestione della configurazione del software definisce una serie di task che hanno quattro obiettivi principali:

* identificare tutti gli elementi che collettivamente definiscono la configurazione del software;
* gestire le modifiche ad uno o più di questi elementi;
* facilitare la costruzione di versioni differenti di un’applicazione
* garantire che venga mantenuta la qualità del software a mano a mano che la configurazione evolve nel corso del tempo.

All’interno di tale processo, un ruolo importante è ricoperto dai meccanismi di controllo delle versioni.

**Controllo delle versioni**

Il controllo delle versioni combina procedure e strumenti al fine di gestire versioni diverse degli oggetti della configurazione, prodotti nel corso del processo software.

Un sistema di controllo delle versioni implementa o viene integrato direttamente con quattro funzionalità principali:

* un database del progetto (repository) che memorizza tutti gli oggetti rilevanti della configurazione;
* una *funzionalità di gestione della versione* che memorizza tutte le versioni di un oggetto di configurazione (o che consente di costruire una qualsiasi versione utilizzando le differenze rispetto alle versioni precedenti);
* una *funzionalità make* che consente di raccogliere tutti gli oggetti di configurazione rilevanti e costruire una specifica versione del software.

Inoltre, i sistemi di controllo delle versioni e delle modifiche spesso implementano un monitoraggio dei problemi (chiamato anche monitoraggio dei bug) che consente al team di registrare e monitorare lo stato di tutti i problemi associati a ciascun oggetto della configurazione.

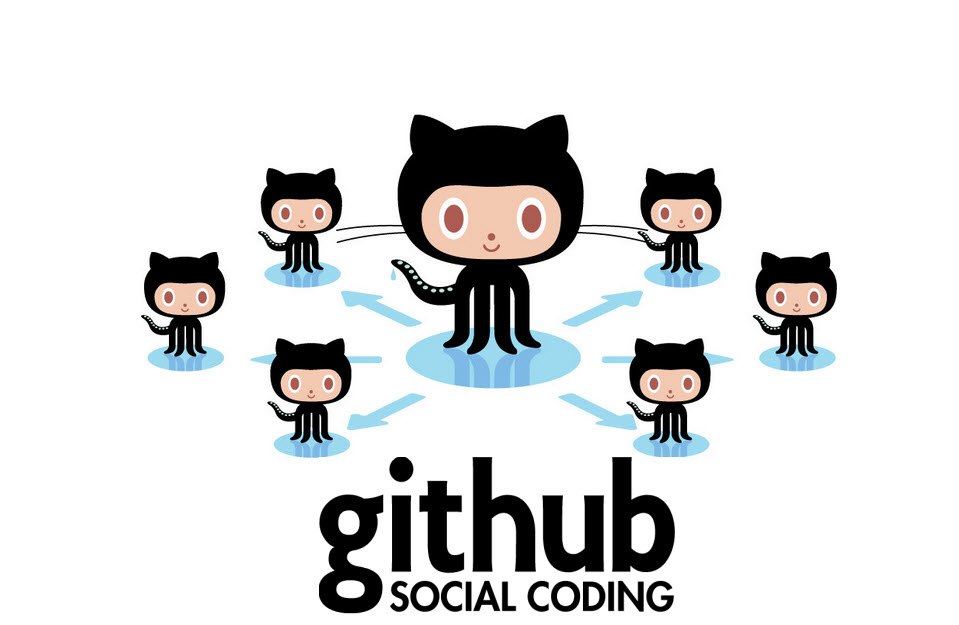
Molti sistemi di controllo della versione definiscono il concetto di *insieme di modifiche* (*change set*), una raccolta di tutte le modifiche (rispetto ad una qualche configurazione stabilizzata) necessarie per creare una specifica versione del software.

Per un’applicazione od un sistema possono essere identificati vari insiemi di modifiche. Questo consente di costruire una versione del software specificando gli insiemi di modifiche (usandone il nome) che devono essere applicate alla configurazione stabilizzata. Per ottenere ciò viene applicato un approccio a modellazione del sistema. Il modello del sistema contiene:

* uno schema (template) che include una gerarchia di componenti ed un “ordine di costruzione” per i componenti che descrive il modo in cui il sistema deve essere costruiti
* le regole di costruzione
* le regole di verifica (spesso è possibile interrogare il modello del sistema per valutare quale impatto avrà un cambiamento in un componente sugli altri componenti)

Negli ultimi venti anni sono stati proposti diversi strumenti automatici di supporto al controllo di versione. La differenza fra le soluzioni sta soprattutto nel grado di sofisticazione degli attributi utilizzati per costruire le diverse varianti e versioni di un sistema e nel meccanismo del processo di costruzione.

**GitHuB**



GitHub è un social-code, uno speciale social network dedicato ai programmatori. La piattaforma social dedicata alla programmazione è intersecata al controllo di versione dei file chiamato Git. Per capire cosa sia GitHub e come funzioni, è necessario capire innanzitutto cos'è Git, cuore pulsante della piattaforma web. Creato dal Linus Torvalds, Git è un **software di controllo di versione**: ciò vuol dire che controlla e gestisce gli aggiornamenti di un progetto senza sovrascrivere nessuna parte del progetto stesso. Venne creato da Torvalds e dai suoi collaboratori nel corso dello sviluppo del kernel Linux: nel caso in cui qualche aggiornamento non avesse dato gli effetti sperati, si poteva sempre tornare indietro e recuperare la versione funzionante senza troppi problemi. In GitHub questi concetti e queste pratiche sono state portate all'estremo, applicandole su un campione molto più ampio e in ambiti più disparati. Grazie a Git, gli utenti della piattaforma social creata da Tom Preston-Werner, Chris Wanstrath e PJ Hyett possono lavorare contemporaneamente sulla medesima versione dello stesso progetto senza timore di apportare modifiche sostanziali. Tutte le vecchie versioni saranno conservate nella propria *repository*, così da poterle recuperare in caso di necessità. Inoltre, per ogni utente al lavoro verrà creata una differente versione del progetto, così da non creare fastidiose sovrapposizioni o sovrascritture.

Grazie a Github è inoltre possibile gestire i progetti online, mantenere una copia sul server e visionarla online senza scaricarla in ssh. In dettaglio le caratteristiche di github sono:

• Mandare messaggi privati;

• Sistema follow/unfollow dei utenti (stile twitter) e dei progetti;

• Possibilità di gestire una wiki (per ogni repository);

• Possibilità di gestire un sistema di issue (per ogni repository);

• Possibilità di usare la maggior parte dei comandi del protocollo git tramite un pannello;

GitHub facilita pertanto la comunicazione tra sviluppatori di uno stesso team. Facilità di utilizzo e apprendimento immediato, nonché grande disponibili di documentazione e tutorial, sono altre caratteristiche fondamentali di tale piattaforma che ne hanno consentito un grande successo. Per tali motivi, GitHub, ben si presta ad essere usato per la gestione del controllo delle versioni nell’ambito di progetti di un corso universitario quale Architetture Software.

**Cooperative Learning**

L’idea di suddivedere gli studenti in piccoli team di sviluppo, e l’utilizzo di strumenti quali GitHub che facilitano la comunicazione e la condivisione di codice tra membri del gruppo, ha ripercussioni positive sull’apprendimento. Per spiegare brevemente tali vantaggi è possibile fare riferimento al “*Cooperative Learning*”.

Con **Cooperative Learning** (CL) si intende un insieme di principi, tecniche e metodi di conduzione della classe, in base ai quali gli studenti affrontano l’apprendimento delle discipline curriculari lavorando in piccoli gruppi in modo interattivo, responsabile, collaborativo, solidale e ricevendo valutazioni sulla base dei risultati ottenuti.

L’idea di lavorare insieme non è certo nuova nella storia dell’umanità, ma mai come in questo periodo l’arte della collaborazione appare indispensabile e difficile.

Lev Semenovic Vygotskij (1896 - 1934), brillante psicologo sovietico particolarmente impiegato nell’ambito educativo, ha sottolineato come un apprendimento basato su collaborazioni in gruppo abbia ripercussioni positive sull’alunno. Per Vygotskij le conoscenze di un umano sono divise in due parti:

* **zona di sviluppo attuale**, che riguarda le abilità già acquisite
* **zona di sviluppo potenziale**, dove si trovano le abilità non ancora presenti nella persona perché con ogni probabilità verranno acquisite in futuro.

Mediante un approccio di apprendimento basato sul lavoro di gruppo, grazie alla condivisione di esperienze, alla diversificazione delle conoscenze tipica di un team, alla responsabilità personale di ogni membro del gruppo, un alunno è stimolato a svolgere al meglio il proprio lavoro. Se qualche conoscenza richiesta per portare a termine un task del lavoro assegnato al gruppo non è presente in modo maturo all’interno della zona di sviluppo attuale di un discente, grazie alla collaborazione, lo studente in questione può arrivare ad avere una conoscenza migliore e piena di tale competenza.

Tale frase, in estrema sintesi, riassume il pensiero di Vygotskij.

“*Ciò che l’alunno riesce a fare in cooperazione oggi, potrà farlo da solo domani. Pertanto, l’unica buona forma di istruzione è quella che anticipa lo sviluppo e lo conduce; essa non dovrebbe essere indirizzata tanto alle forme mature quanto a quelle che stanno maturando.*”