**Manutenzione**

Creare un progetto software perfettamente funzionante non è impresa facile e può capitare di riscontrare errori/anomalie nella struttura o nel funzionamento del codice, solo a consegna effettuata; a questo proposito si sfrutta il processo definito **Manutenzione,** il cui scopo principale è correggere gli errori postumi consegna e adattare il software alle mutevoli esigenze.

Affinché l’attività di Manutenzione sia completa, occorre che risponda a diversi criteri e azioni:

* *Correttiva*: correggere gli errori
* *Adattiva*: deve adattarsi ai cambiamenti dell’ambiente, sia hardware che software;
* *Perfettiva*: deve adattarsi alle esigenze degli utenti (possono mutare in base alla singola persona)
* *Preventiva*: per aumentare il grado di manutenibilità del sistema

Qui sotto vi è riportato un grafico a torta che mostra l’importanza data ad ognuno dei tipi di manutenzione rispetto al nostro progetto.

Immagine che contiene schermata, diagramma, Policromia, Elementi grafici

Descrizione generata automaticamente

Come si può notare, la manutenzione correttiva occupa la maggior parte dello spazio dedicato, proprio per risolvere errori in seguito alla consegna; la manutenzione preventiva è importante affinché il grado di manutenibilità aumenti sempre di più e ciò porti ad un miglioramento di qualità. Siccome il nostro progetto non è ancora pronto per la commercializzazione, durante la manutenzione non si tiene molto conto dei cambiamenti dell’ambiente di sviluppo o delle richieste degli utenti, ma si possono fare delle ipotesi circa l’utilità complessiva.

L’attività di Manutenzione è iniziata subito dopo la consegna dell’elaborato e ogni membro del gruppo ha lavorato in maniera attiva sulle singole componenti, in particolare sull’analisi del codice e correzione di errori; le principali ricerche rispetto alla manutenzione riguardano:

* Utilizzo improprio di gerarchie di classi
* Assenza di commenti
* Utilizzo improprio di Tools o attività di Refactoring

L’obbiettivo finale è raggiungere un sistema non troppo complesso (a livello di struttura), che sia sufficiente a livello di documentazione e che non si verifichino fallimenti frequenti.

***Refactoring***

Il refactoring è un processo attraverso il quale il codice viene rivisto, senza aggiungere o modificare funzionalità, ma migliorandone la qualità interna eliminando i cosiddetti “bad smells”, come per esempio:

* Eliminare codice duplicato
* Evitare l’uso di variabili globali
* Evitare l’utilizzo di metodi troppo lunghi
* Evitare “classi pigre” (impiegano più tempo del dovuto a completare una funzione)
* Evitare “invidia tra le classi” (porterebbe a scontri tra più classi, compromettendo la qualità del sistema)

Il procedimento permette di diminuire il codice utilizzato, garantendo un aumento della qualità finale del prodotto e una riduzione di necessità di manutenzione; di conseguenza, si è più preparati ad anticipare nuovi cambiamenti e si evita di usare frammenti di codice, che altrimenti sarebbe “inutile”.