

Refactoring in LabVIEW.

Category: LabVIEW Talk

scritto da admin | Novembre 9, 2022



Il refactoring in LabVIEW è un processo di ristrutturazione di codice, funzionante correttamente, con lo scopo di renderlo leggibile, manutenibile ed espandibile.

Viene anche definito come un processo di pulizia, non ha come obiettivo l'ottimizzazione del codice e non è specifico per labVIEW.

Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

Martin Flower – [Refactoring: Improving the design of existing code](#)

Refactoring o Rebuilding

Il refactoring è necessario quando:

- Si devono inserire caratteristiche o funzioni
- Implementazione/sostituzione di hardware
- Si sta facendo debug sui VI's
- Codice con valore intellettuale da non cestinare.
- Raggruppare il codice riusabile in librerie.

La prima cosa che viene in mente quando si deve lavorare sul codice esistente, è quello di **rifarlo da capo**.

Rifare o Ristrutturare, la scelta dipende dalla complessità, dal valore, dal rischio e da altre condizioni.

Bisogna anche tenere conto che il processo di refactoring può introdurre nuovi bachi.

Non è escluso che l'attività di Refactoring possa essere costosa.

Alcune regole di valutazione potrebbero essere:

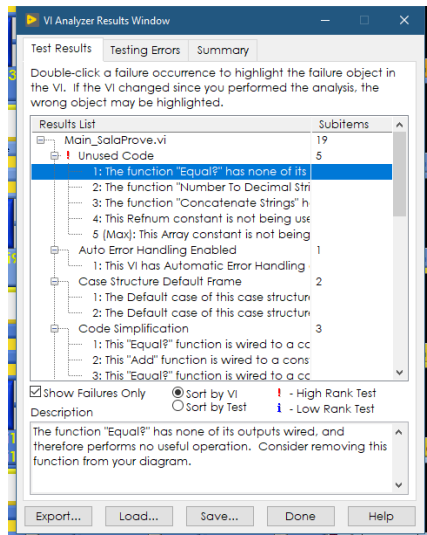
- Quanto è illeggibile e complesso il software
- Si può accedere a chi lo ha fatto per recuperare informazioni?
- Esiste Documentazione o persone in grado di dare spiegazioni sulle funzionalità e specifiche?
- Budget disponibile
- Hardware obsoleto che richiederebbe un ulteriore refactoring a breve

Preparazione

- Fare una copia di backup del codice esistente, possibilmente in un repository di versionamento (es. GIT)
- Studiare il codice e comprenderlo al meglio.
- Creare un piano di test e di azione
- Riordinare il più possibile il codice da un punto di vista estetico e di ordine, aggiungendo descrizioni e quanto necessario per poterlo interpretare al meglio.
- Rinominare Controlli, indicatori, costanti e collegamenti, con nomi **logici e comprensibili** e corti.
- Fare modifiche leggere completarle e verificarle sempre.
- Verificare la possibilità di raggruppare in cluster **type def** dati logicamente coerenti tra loro.
- Provare a incapsulare parti del codice in subVI.
- Ridurre quanto possibile, Sequenze e [local variable](#).

VI Analyzer

Il VI Analyzer può essere lanciato per verificare il codice con vari tipi di test.

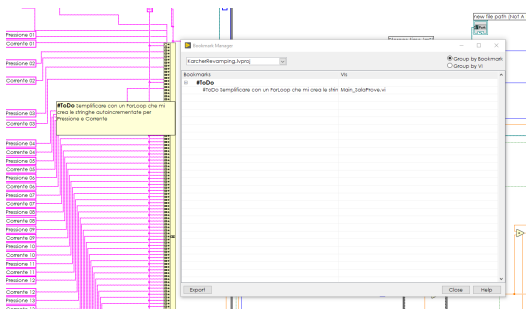


Vi Analyzer

Bookmark

I bookmark possono essere utilizzati per facilitare l'esplorazione dei task da eseguire e come piano di azione.

Per creare un bookmark basta segnarlo con un hashtag davanti al nome es. **#ToDo** Qui devo sostituire la variabile locale.



Esempio Bookmark

Approfondimenti

[Refactoring](#) Martin Fowler