****

# ISW – Gruppo 01 – TeamSoftwareRevolution

Metrica

# Sommario

1. Introduzione
2. Complessità dei prodotti software
   1. Linguaggi imperativi
   2. Linguaggi Object-Oriented
3. Metriche
   1. Tempi di risposta
   2. Modularità
   3. Velocità di completamento
4. Errori
   1. Definizione
   2. Classificazione
   3. Testing
5. Disciplime e fondamenti scientifici
6. Conclusioni

# Introduzione

Il presente elaborato tratta alcuni aspetti fondamentali dell’ingegneria del software.

Inoltre si è descritta una disciplina, che come l’ISW, sta cercando di trovare un proprio fondamento scientifico.

Attraverso un costruttivo lavoro di gruppo, partendo dal lavoro e dalle conoscenze individuali, sono stati argomentati gli aspetti sopra riportati.

Nei paragrafi seguenti, vengono trattati temi quali la complessità dei prodotti software, le metriche di valutazione e tipologie di errori, evidenziando difetti e mancanze.

# Complessità dei prodotti software

Un prodotto software è rappresentato dall'insieme di programmi, procedure di realizzazione, documentazione di sistema, i dati di configurazione e il manuale di riferimento.

Un prodotto software può essere di due distinte tipologie:

* Generico: destinato all’uso di massa (genarl porpose)
* Specifico: realizzato per le esigenze di uno specifico utente/azienda (specific porpose)

La realizzazione di un prodotto software è un processo lungo il quale richiedere la collaborazione di diverse figure professionali specializzate (sviluppatori, team leader, analisti ecc) i quali, insieme agli investitori, utenti ed altre figure interessate ,rappresentano gli stakeholders del prodotto in questione.

Il prodotto software nella generalità dei casi risulta molto complesso, pertanto necessita di una suddivisione in moduli, ciascuno dei quali viene implementato e curato da un diverso team di sviluppo.

Si nota che un possibile criterio per valutare la complessità di un prodotto software è rappresentato appunto dal numero di moduli e sotto-moduli che lo compongono, pertanto la complessità cresce all’aumentare dei moduli.

La dimensione di un programma, inteso come numero di righe di codice, pur avendo un impatto visivo notevole, non rappresenta sicuramente un indice di valutazione della complessità.

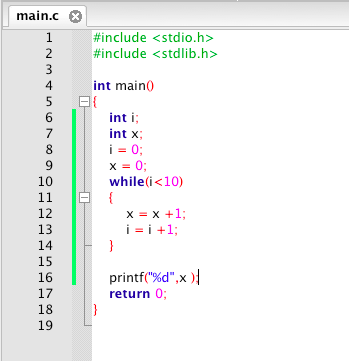
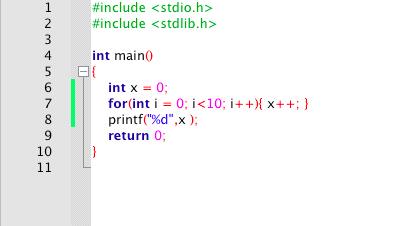
Due codici differenziati dal numero di righe infatti, a parità di dati inseriti in input, possono fornire entrambe la stessa soluzione.

A conferma di questa tesi, esistono diversi stili di programmazione che si differenziano per la compattezza del codice.

Diversi programmatori preferiscono un codice più “esteso”, e di conseguenza più leggibile, piuttosto che un codice più compatto e di difficile interpretazione.

## 2.1 Linguaggi imperativi

La seguente figura fornisce un esempio di due codici con risolvono lo stesso problema con scelte sintattiche e stilistiche differenti:



1.a 1.b

bla bla bla

## 2.2 Linguaggi Objcet-Oriented

Vediamo ora un esempio legato alla programmazione ad oggetti.

Bla bla bla

# Metriche

Nel capitolo 2 abbiamo introdotto il concetto di valutazione della complessità di un prodotto software, ma come possiamo valutarne la qualità?

Prima di elencare varie alternative è doveroso introdurre il concetto di **metrica.**

Una metrica è uno standard per la misurazione di alcune proprietà legata al prodotto software.

Queste metriche possono essere suddivise in due categorie:

* Metriche esterne
* Metriche interne

Le metriche esterne o di prodotto stabiliscono le caratteristiche principali per la produzione di un software:

* Costi
* Tempi
* Produttività

Le metriche interne o di progetto permettono la valutazione qualitativa del software:

* Funzionabilità
* Manutenibilità
* Usabilità

Introdotto il concetto di metrica, possiamo ora elencarne alcune.

## 3.1 Tempi di risposta

Una buona metrica per giudicare la qualità di un SW è la valutazione della complessità computazionale e quindi dei tempi di risposta.

Questa metrica però, ha il difetto che non sempre i tempi di risposta sono assoluti ma possono essere influenzati dall’ hardware della macchina sulla quale sono installati.

## 3.2 Modularità

La modularità, concetto parzialmente introdotto nel capitolo 2, consiste nel suddividere il problema in sotto-problemi e quindi il prodotto software il moduli secondo il principio divide et impera.

Il deficit della modularità è dato dal fatto che a volte, visto che i diversi moduli possono essere sviluppati da programmatori diversi, è possibile che si verifichino errori dovuti alla scarsa comunicazione tra i componenti del team.

## 3.3 Velocità di completamento

Per determinare se un prodotto è di buona qualità, possiamo anche valutare la velocità di completamento di un’azione intesa come numero di task.

Questa metrica da sola però risulta inefficace poiché non tutti gli utenti che utilizzano un dato prodotto software possiedono le stesse caratteristiche fisiche, logiche e informatiche.

# Errori

Dopo aver introdotto le metriche, abbiamo visto alcuni casi per valutare un prodotto software e i difetti ad essi legati.

Ogni problema nel software rappresenta una mancanza di valore e una opportunità per migliorarlo.

Classificare i problemi del software permette di affrontarli e risolverli.

Tra i tanti problemi che possiamo riscontrare durante il processo di “creazione” di un prodotto software ci sono gli errori.

## 4.1 Definizione

Un errore o bacco (in inglese bug) rappresenta un "difetto" che si ha nell'esecuzione di un programma o di un prodotto software, quando, per certi tipi di dati in ingresso, si ha un funzionamento diverso da quello voluto.

Gli errori di differenziano per casi e tipologie.

## 4.2 Classificazione

* Requisiti difettosi
* Errati o incompleti
* Incomprensione cliente-sviluppatore
* Errori progettuali
* Mancanza di specifiche, errori algoritmici
* Errori durante il testing
* Testing affrettati, correzione parziale degli errori
* Errori di documentazione
* Documentazione parziale o inesistente