

# Corso Web MVC

## Introduzione

Emanuele Galli

[www.linkedin.com/in/egalli/](http://www.linkedin.com/in/egalli/)



# Informatica

- Informatique: information automatique
  - Trattamento automatico dell'informazione
- Computer Science
  - Studio dei computer e come usarli per risolvere problemi in maniera corretta ed efficiente

# Computer

- Processa informazioni
- Accetta input
- Genera output
- Programmabile
- Non è limitato a uno specifico tipo di problemi

# Hardware, Software, Firmware

- Hardware
  - Componenti elettroniche usate nel computer
  - Disco fisso, mouse, ...
- Software 
  - Programma
    - Algoritmo scritto usando un linguaggio di programmazione
    - Codice utilizzabile dall'hardware
  - Processo
    - Programma in esecuzione
  - Word processor, editor, browser, ...
- Firmware 
  - Programma integrato in componenti elettroniche del computer (ROM, EEPROM)
    - UEFI / BIOS: avvio del computer
    - Avvio e interfaccia tra componenti e computer

# Sistema Operativo





- Insieme di programmi di base
  - Rende disponibile le risorse del computer
    - All'utente finale mediante interfacce
      - CLI (Command Line Interface) / GUI (Graphic User Interface)
    - Agli applicativi
  - Facilità d'uso vs efficienza
- Gestione delle risorse:
  - Sono presentate per mezzo di astrazioni
    - File System
  - Ne controlla e coordina l'uso da parte dei programmi
- Semplifica la gestione del computer, lo sviluppo e l'uso dei programmi



# Problem solving



- Definire chiaramente le **specifiche** del problema
  -  Es: calcolo della radice quadrata. Input? Output?
  - Vanno eliminate le possibili ambiguità
- Trovare un **algoritmo** che lo risolva 
- Implementare correttamente la soluzione con un linguaggio di programmazione
- Eseguire il programma con l'input corretto, in modo da ottenere l'output corretto



# Algoritmo

- Sequenza di istruzioni che garantisce di dare il risultato di un certo problema
  - Ordinata, esecuzione sequenziale (con ripetizioni)
  - Operazioni ben definite ed effettivamente eseguibili
  - Completabile in tempo finito
- Definito in linguaggio umano ma artificiale
  - Non può contenere ambiguità
  - Deve essere traducibile in un linguaggio comprensibile dalla macchina

# Le basi dell'informatica

- Matematica

- L'algebra di George Boole ~1850

- Notazione binaria



- La macchina di Alan Turing ~1930

- Risposta all'Entscheidungsproblem (problema della decisione) posto da David Hilbert
    - Linguaggi di programmazione Turing-completi

- Ingegneria

- La macchina di John von Neumann ~1940

- Descrizione dell'architettura tuttora usata nei computer: Input, Output, Memoria, CPU






# Algebra Booleana

- Due valori
  - false (0)
  - true (1)
- Tre operazioni fondamentali
  - AND (congiunzione)
  - OR (disgiunzione inclusiva)
  - NOT (negazione)



A	B	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

A	NOT
0	1
1	0

# Linguaggi di programmazione

- Linguaggio macchina
  - È il linguaggio naturale di un dato computer
  - Ogni hardware può averne uno suo specifico
  - Istruzioni e dati sono espressi con sequenze di 0 e 1
  - Estremamente difficili per l'uso umano
- Linguaggi Assembly 
  - Si usano abbreviazioni in inglese per le istruzioni
  - Più comprensibile agli umani, incomprensibile alle macchine
  - Appositi programmi (assembler) li convertono in linguaggio macchina




# Variabile

- Locazione di memoria associata a un nome, contiene un valore
- Costante: non può essere modificata dopo la sua inizializzazione
- Una singola locazione di memoria può essere associata a diverse variabili (alias)
- Supporto a tipi di variabili da linguaggi di:
  - “basso livello” → legati all’architettura della macchina 
  - “alto livello” → tipi complessi 
  - script → runtime

# Array

- Struttura dati comune a molti linguaggi di programmazione
- Basata sul concetto matematico di vettore, nel senso di matrice monodimensionale
- Collezione di elementi (dello stesso tipo) identificati da un indice
  - Il primo elemento ha indice 0 in alcuni linguaggi, 1 in altri (e anche n in altri ancora)
- Gli elementi sono allocati in un blocco contiguo di memoria, il che permette accesso immediato via indice ai suoi elementi

# Linguaggi di alto livello

- Molto più comprensibili degli assembly
- Termini inglesi e notazioni matematiche
- Possono essere espressi in forma
  - **imperativa**: si indica cosa deve fare la macchina
  - **dichiarativa**: si indica quale risultato si vuole ottenere   
HTML non è un linguaggio di programmazione perché non è Turing completo. è del tipo dichiarativo.
- A seconda di come avviene l'esecuzione si parla di linguaggi
  - **compilati**: conversione del codice in linguaggio macchina, ottenendo un programma eseguibile   
Il linguaggio compilato è affidato al compilatore o al linker che trasforma il nostro file in linguaggio c (x.c) ad esempio in linguaggio macchina per un determinato tipo di macchina. è un linguaggio più lento rispetto al secondo. per essere letto altrove deve essere decompilato e non sempre quest'attività riesce bene.
  - **interpretati**: il codice viene eseguito da appositi programmi   
Javascript fa parte di questa famiglia. Questo linguaggio può invece correre sui browser che permette di leggerlo su diversi dispositivi. in questo caso è il browser ad essere il compilatore.

# Istruzioni

- Operazioni **sequenziali** I valori booleani di vero e falso servono per decidere le operazioni sequenziali e iterative. Restano fuori quelle sequenziali, sequenze di operazioni che la mia CPU deve eseguire.
  - Chiedono al computer di eseguire un compito ben definito, poi si passa all'operazione successiva
- Operazioni **condizionali** If (C>0) then c=c\*2 else c=0 questa è la logica delle operazioni condizionali. cioè eseguire un codice piuttosto che un altro in base all condizione di verità o falsità della premessa.
  - Si valuta una condizione, il risultato determina quale operazione seguente verrà eseguita
- Operazioni **iterative** eseguire una stessa operazione su tutti gli elementi dell'array o su un certo intervallo. per esempio for (i=1...10) print (i)
  - Richiede di ripetere un blocco di operazioni finché non si verifica una certa condizione – se ciò non accade: loop infinito

# Flow chart vs Pseudo codice

- Diagrammi a blocchi – flow chart
  - L'algoritmo viene rappresentato con un grafo orientato dove i nodi sono le istruzioni
  - Inizio e fine con ellissi
  - Rettangoli per le operazioni sequenziali (o blocchi)
  - Esagoni o rombi per condizioni
- Pseudo codice Rappresenta l'architettura approssimativa.
  - L'algoritmo viene descritto usando l'approssimazione un linguaggio ad alto livello, si trascurano i dettagli, ci si focalizza sulla logica da implementare

# Complessità degli algoritmi


- “O grande”, limite superiore della funzione asintotica Dalla complessità minore a quella maggiore. I computer quantistici dovrebbero eseguire operazioni  $O(n^2)$  in  $O(n \log n)$ .
  - Costante  $O(1)$  Il rapporto tempo/n (numero di input array) è lineare. L'input può crescere di dimensione ma il tempo di operazione su di esso rimane costante.
  - Logaritmica  $O(\log n)$  In questo caso il tempo aumenta come in una curva logaritmica ma non in maniera sensibile. Il costo logaritmico è ricavabile dagli alberi BST. Dimezzo il tempo di ricerca perchè l'albero già di per sé mi divide i valori in minori e maggiori rispetto alla radice. Si usa soprattutto del sorting.
  - Lineare  $O(n)$  Data un array con n elementi devo trovare un det valore. più è lungo l'array più mi posso aspettare che il costo di ricerca aumenti. in questo caso l'algoritmo cerca elemento per elemento.
  - Linearitmico  $O(n \log n)$  il costo è un pò peggio della lineare. complessità di questo tipo hanno a che fare con il sorting, il mettere in ordine. Quindi devo ordinare i valori come se dovessero essere messi su un albero.
  - Quadratica  $O(n^2)$  – Polinomiale  $O(n^c)$  Da questo punto la complessità aumenta di livello per cui o rinunciamo all'impresa, o attendiamo tempi molto lunghi o usiamo pochi dati.
  - Esponenziale  $O(c^n)$
  - Fattoriale  $O(n!)$
- Tempo e spazio Quanto tempo ci mette il mio algoritmo per essere eseguito e quanto spazio mi occupa. Queste sono le due variabili che più mi interessano. Tipicamente al crescere di una variabile corrisponde il diminuire dell'altro.
- Caso migliore, peggiore, medio



# Algoritmi di ordinamento

- Applicazione di una relazione d'ordine a una lista di dati
  - Naturale → crescente (alfabetico, numerico)
- Utile per migliorare
  - l'efficienza di altri algoritmi
  - La leggibilità (per gli umani) dei dati
- Complessità temporale
  - $O(n^2)$ : algoritmi naive
  - $O(n \log n)$ : dimostrato ottimale per algoritmi basati su confronto
  - $O(n)$ : casi o uso di tecniche particolari

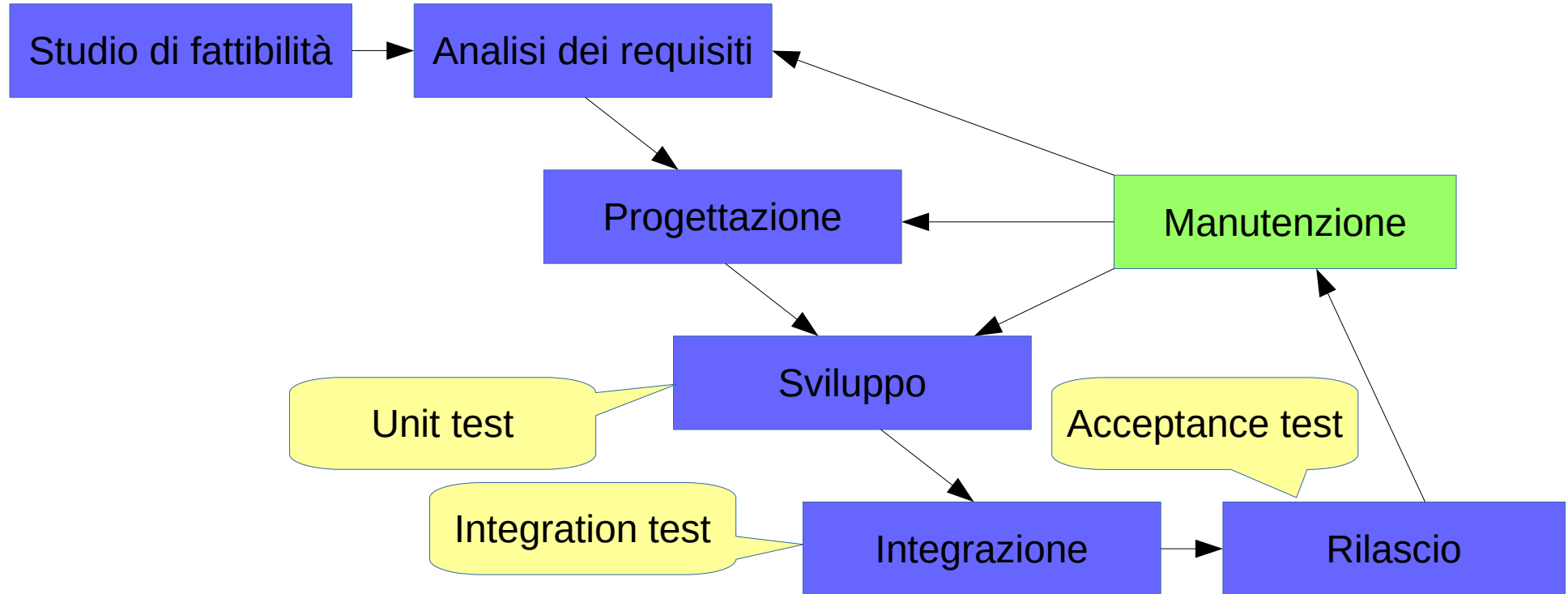
# Ingegneria del software

- Approccio sistematico alla creazione del software
  - Struttura, documentazione, milestones, comunicazione e interazione tra partecipanti
- Analisi dei requisiti
  - Formalizzazione dell'idea di partenza, analisi costi e usabilità del prodotto atteso
- Progettazione
  - Struttura complessiva del codice, definizione architetturale
  - Progetto di dettaglio, più vicino alla codifica ma usando pseudo codice o flow chart
- Sviluppo
  - Scrittura effettiva del codice, e verifica del suo funzionamento via **unit test** 
- Manutenzione
  - Modifica dei requisiti esistenti, bug fixing

# Unit Test

- Verificano la correttezza di una singola “unità” di codice
  - Mostrano che i requisiti sono rispettati
- Verifica
  - Casi base (positivi e negativi)
  - Casi limite
- Ci si aspetta che siano
  - Ripetibili: non ci devono essere variazioni nei risultati
  - Semplici: facile comprensione ed esecuzione
  - E che offrano una elevata copertura del codice

# Modello a cascata (waterfall)



# Modello agile



# Software Developer

- Front End Developer
  - Pagine web, interazione con l'utente
    - HTML, CSS, JavaScript
    - User Experience (UX)
- Back End Developer
  - Logica applicativa
    - Java, C/C++, Python, JavaScript, SQL, ...
    - JavaEE, Spring, Node, DBMS, ...
- Full Stack Developer
  - Sintesi delle due figure precedenti