

# 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

è UNA MACCHINA CHE:





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

come una calcolatrice




# Hardware, Software, Firmware

- Hardware
  - Componenti elettroniche usate nel computer
  - Disco fisso, mouse, ...
- Software permette l'accesso all'utente alle risorse del computer ed agli applicativi.
  - 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 (è il primo programma che parte nel pc al momento dell'accensione)
    - Avvio e interfaccia tra componenti e computer

# Sistema Operativo

- Insieme di programmi di base
  - Rende disponibile le risorse del computer ed ai programmi applicativi
    - 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 l'istruzione è la componente minima dell'algoritmo
  - Ordinata, esecuzione sequenziale (con ripetizioni) quindi con dei loop
  - 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
      - ha la possibilità di gestire un input e dare un output
      - deve prendere delle decisioni: istruzioni condizionali ( con if e then) e delle variabili definite
      - deve poter fare ripetizioni /loop per eseguire più volte le stesse istruzioni (for/while)
      - devono esserci blocchi di istruzioni (block)






- 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





è un tipo di variabile che contiene più valori dello stesso tipo

- 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

Per identificare quale elemento devo indicarne la posizione. Devo sapere dove sono collocati.

# Linguaggi di alto livello

è qualcosa che si stacca dalla macchina e diventa più comprensibile agli umani. Se un linguaggio è Turing completo è equivalente agli altri, nel senso che è in grado di risolvere lo stesso problema, magari in modo diverso.

- 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 
- A seconda di come avviene l'esecuzione si parla di linguaggi
  - **compilati**: conversione del codice in linguaggio macchina, ottenendo un programma eseguibile 
  - **interpretati**: il codice viene eseguito da appositi programmi 

# Istruzioni

- Operazioni **sequenziali** tutte le normali operazioni eseguibili su una variabile (es. somma). Se si hanno più core posso fare più operazioni contemporaneamente
  - Chiedono al computer di eseguire un compito ben definito, poi si passa all'operazione successiva
- Operazioni **condizionali** (if ... then; else...) valore booleano  
Quindi si hanno due istruzioni possibili e voglio che ne venga eseguita solo una
  - Si valuta una condizione, il risultato determina quale operazione seguente verrà eseguita
- Operazioni **iterative** es. applica un'operazione su tutti gli elementi che hanno det. caratteristiche. I loop si chiamano while/for
  - 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

2 modi per commissionare il codice al programmatore, cioè per descriverlo.

- 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
  - 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

funzione che sta sempre al di sopra della mia funzione e le si avvicina il più possibile, perchè vogliamo dare una rappresentazione, e quindi una stima, di quanto costa il ns. algoritmo.

- “O grande”, limite superiore della funzione asintotica
  - Costante  $O(1)$  è una funzione parallela alla retta che passa da 1 (y) parallela all'asse x. Ci mette sempre lo stesso tempo a calcolare la risposta
  - Logaritmica  $O(\log n)$  curva che sale e cresce in modo esponenziale. tipicamente vuol dire che c'è dentro un albero.  Digita qui il testo
  - Lineare  $O(n)$  curva bisettrice del primo e del terzo quadrante
  - Linearitmico  $O(n \log n)$  la funzione si allontana in peggio rispetto alla lineare. Si parla di sorting cioè mettere in ordine.
  - Quadratica  $O(n^2)$  – Polinomiale  $O(n^c)$  aumento a picco
  - Esponenziale  $O(c^n)$  Digita qui il testo
  - Fattoriale  $O(n!)$
- Tempo e spazio sono i criteri sulla base dei quali valutare l'algoritmo, che di norma sono inversamente proporzionali.
- Caso migliore, peggiore, medio




# Algoritmi di ordinamento<sub>(o sorting)</sub>


se sono sicuro di lavorare su dati ordinati posso applicare algoritmi che mi permettono di lavorare più velocemente.

- 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

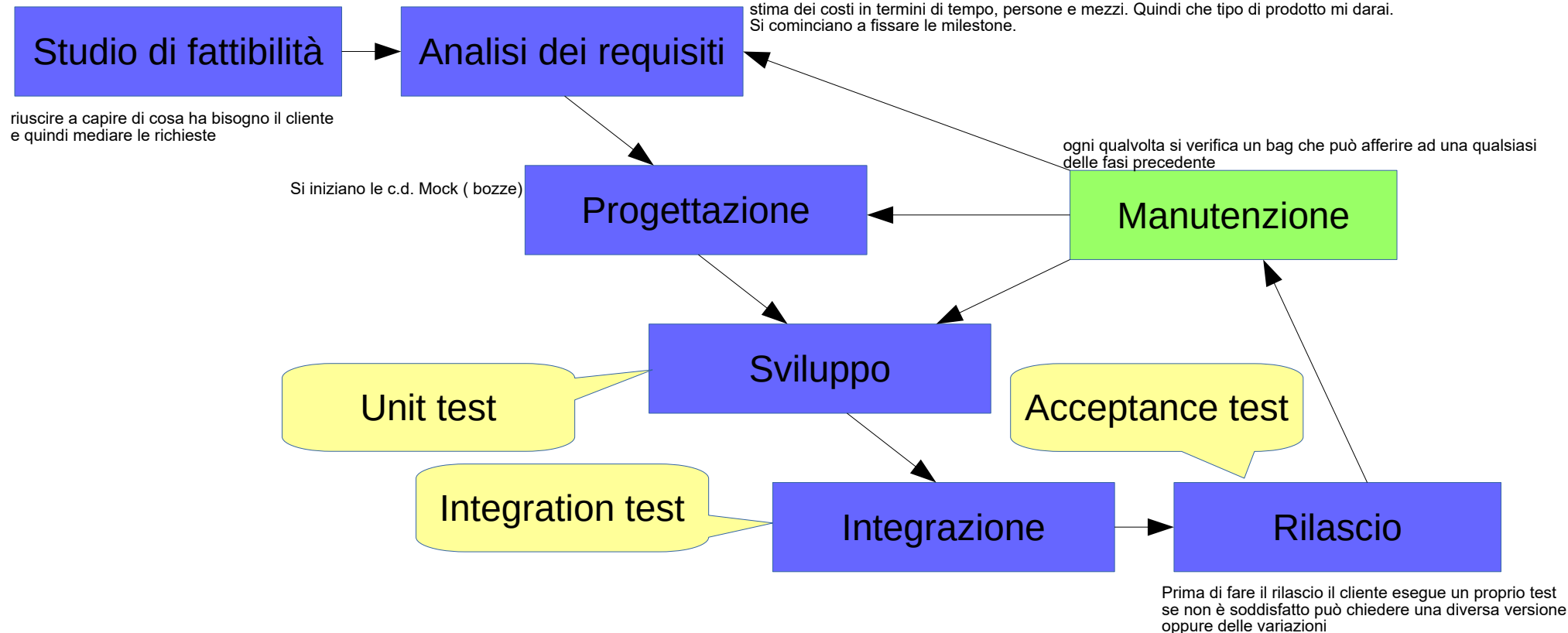
# Ingegneria del software

- Approccio sistematico alla creazione del software
  - Struttura, documentazione, milestones, comunicazione e interazione tra partecipanti
- Analisi dei requisiti non è per il singolo blocco ma per l'intero progetto.
  - Formalizzazione dell'idea di partenza, analisi costi e usabilità del prodotto atteso
- Progettazione definire la struttura del codice come MVC
  - 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**   
cioè che corrisponde alle specifiche richieste
- Manutenzione
  - Modifica dei requisiti esistenti, bug fixing o aggiornamenti

# Unit Test

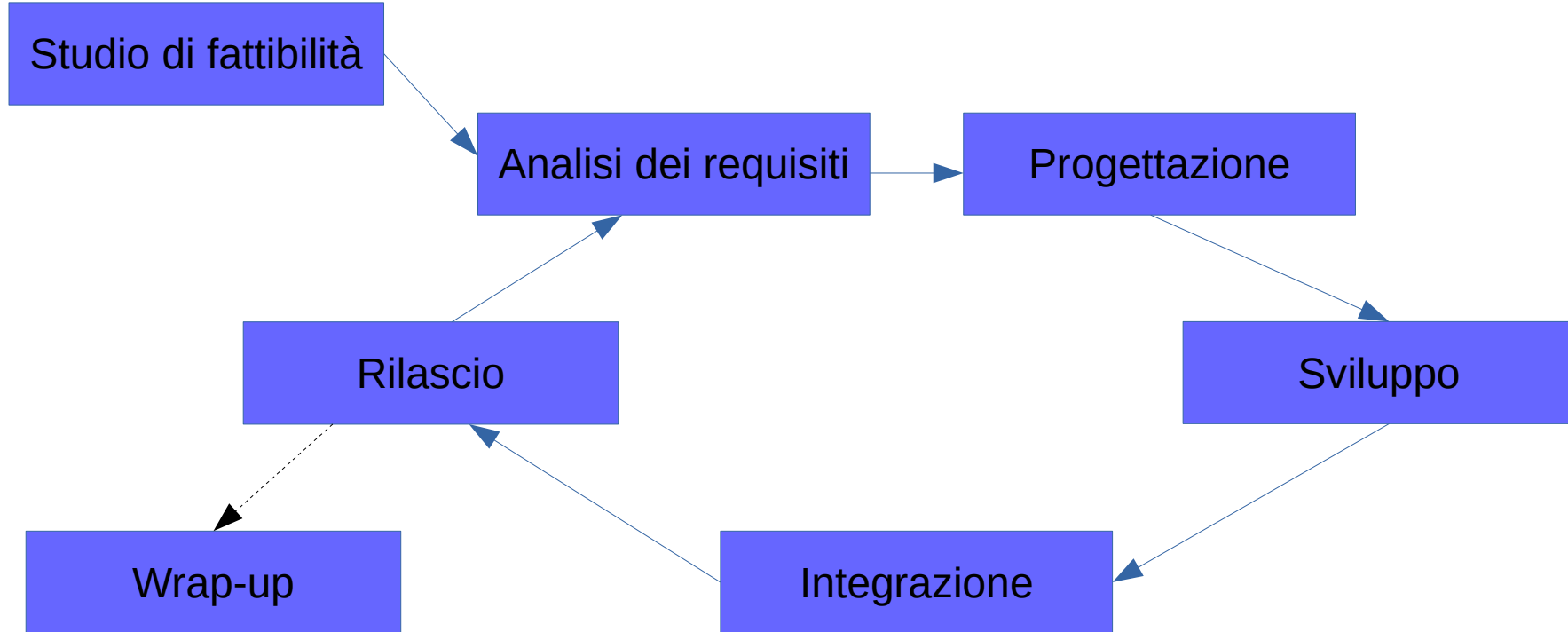
- Verificano la correttezza di una singola “unità” di codice
  - Mostrano che i requisiti sono rispettati
- Verifica
  - Casi base (positivi e negativi)
  - Casi limite 
- Ci si aspetta che siano
  - Ripetibili: non ci devono essere variazioni nei risultati
  - Semplici: facile comprensione ed esecuzione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

# Modello a cascata (waterfall)



# Modello agile

Tipico modello da start up. Si ha una visibilità molto più limitata perchè è impossibile dare una stima precisa dei costi e ovviamente questo alle aziende clienti non piace. Tutti comunicano con tutti e possono interagire con loro.



si continuano ad inserire nuove funzionalità

# Software Developer

- **Front End Developer** scrivo codice che correrà sulla macchina del cliente
  - Pagine web, interazione con l'utente
    - HTML, CSS, JavaScript
    - User Experience (UX)
- **Back End Developer** una volta si parlava di client server, scrivo un codice che correrà sulla mia macchina
  - Logica applicativa
    - Java, C/C++, Python, JavaScript, SQL, ...
    - JavaEE, Spring, Node, DBMS, ...
- **Full Stack Developer** combinazione dei primi 2
  - Sintesi delle due figure precedenti