

# 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 : aree di divisione del pc.

- **Hardware** si tratta delle componenti del pc.
  - Componenti elettroniche usate nel computer
  - Disco fisso, mouse, ...
- **Software**
  - **Programma** es: Windows , il programmatore se ne occupa e il sistema operativo lo esegue e si chiamerà "Processo"(che quindi significa "programma in esecuzione").
    - Algoritmo scritto usando un linguaggio di programmazione
    - Codice utilizzabile dall'hardware
  - **Processo**
    - Programma in esecuzione
  - Word processor, editor, browser, ...
- **Firmware** Tiene insieme il Software e l'Hardware, è una posizione intermedia. (è un codice che non è facilmente modificabile al contrario del Software.
  - **Programma integrato in componenti elettroniche del computer (ROM, EEPROM)** il secondo è modificabile quelli nella Rom no, sono solo leggibili i programmi.
    - **UEFI / BIOS: avvio del computer** il Bios si avvia prima del sistema operativo
    - **Avvio e interfaccia tra componenti e computer**

# Sistema Operativo

Fa parte del Software, ed è un insieme di programmi base che facilità l'utilizzo del computer.

è il primo Software che entra in esecuzione.

- Insieme di programmi di base

- Rende disponibile le risorse del computer : Questo è il suo scopo. Alle "risorse" del pc accedono gli utenti e il pc.

- All'utente finale mediante interfacce

- CLI (Command Line Interface) / GUI (Graphic User Interface)

Accensione che avviene in due modi: CLI è una accensione diversa dove vengono usati i comandi attraverso CMD (che viene scritto sulla barra di start). I comandi variano in base al sistema operativo (due famiglie: Windows e Sistema Unix formato da Linux ed Apple). I comandi tramite CMD vengono scritti a mano.

- Agli applicativi

- Facilità d'uso vs efficienza : queste sono le caratteristiche del sistema operativo. Caratteristiche che devono soddisfare gli utenti e gli applicativi.

- Gestione delle risorse:

- Sono presentate per mezzo di astrazioni per gestire le risorse esse vengono astratte dal sistema operativo organizzate in codice binario e rende l'accesso semplice.


- File System

è un esempio di astrazione. Il file System cos'è? è un insieme di cartelle e file. La cartella è un contenitore di file e altre cartelle (sinonimi: Folder e directory). Il file cos'è? è l'unità minima di memorizzazione della memoria di massa, di particolare ha un proprio nome così da poterlo identificare. Il file system è come se fosse un "Albero" ed è basato su C:/ che serve per accedere ai file del disco fisso.

- Ne controlla e coordina l'uso da parte dei programmi altra cosa che fa il sistema operativo.

- 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? output sta ad indicare il risultato da ottenere, gli input è la spiegazione di ciò che si vuole ottenere.
  - Vanno eliminate le possibili ambiguità Questa è una cosa importante per evitare il nascere di problemi.
- Trovare un **algoritmo** che lo risolva  L'Algoritmo tramite il linguaggio di programmazione si trasforma in un programma.
- Implementare correttamente la soluzione con un linguaggio di programmazione esempio Java. Una volta scritto il programma si esegue con l'input giusto così da avere l'output corretto.
- Eseguire il programma con l'input corretto, in modo da ottenere l'output corretto l'input è importante da capire per ottenere l'output corretto.

L'Algoritmo risolve il problema, è la ricetta che permette la risoluzione del problema. Successivamente ci sarà la traduzione dall'algoritmo al programma, perché l'algoritmo è scritto in maniera comprensibile dall'umano ma il programma per essere eseguito deve essere comprensibile alla macchina. Una volta tradotto in linguaggio macchina il programma verrà eseguito. Esso si determina una volta che input e output sono stati definiti correttamente.

# Algoritmo

Deve essere chiaro, definito e finito in tempo finito.

- Sequenza di istruzioni che garantisce di dare il risultato di un certo problema un esempio è la ricetta per spiegare come si cucina il pollo.
  - Ordinata, esecuzione sequenziale (con ripetizioni)
  - Operazioni ben definite ed effettivamente eseguibili
  - Completabile in tempo finito Tempo che fa riferimento al tempo dell'esecuzione.
- Definito in linguaggio umano ma artificiale
  - Non può contenere ambiguità
  - Deve essere traducibile in un linguaggio comprensibile dalla macchina Questa è la parte finale, una volta che l'algoritmo viene creato deve essere tradotto in linguaggio macchina per poter eseguire il programma.

L'algoritmo è ciò che sarà codice e i programmi si scrivono in questo modo perché l'informatica è formata da una parte matematica e una parte ingegneristica.

# Le basi dell'informatica

Serve per capire come sono fatte le macchine!  
così da comprendere il linguaggio della macchina.

- Matematica

- L'algebra di George Boole ~1850

- Notazione binaria

Ha semplificato l'algebra riducendola a due valori: 0 e 1. (false-true). KISS: acronimo che significa "Mantieni il problema facile".  
Inventa la base matematica che si usa per scrivere il codice informatica.



- La macchina di Alan Turing ~1930

Che cerca di risolvere il problema della decisione, "può esistere un problema dove il programma risponde?" Questa fu la sua intuizione.  
Turing elabora una risposta e nella sua teoria inventa l'informatica moderna con la c.d "Macchina di Turing".

- 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

Macchina espressa in maniera ingegneristica, tenendo conto della memoria, degli input, degli output. Considerato anche lui il padre dell'informatica moderna. Turing e Neumann hanno posto le basi dell'informatica come la conosciamo oggi.



# Algebra Booleana

: Stabilisce come sono fatti i dati in informatica.

L'Algebra di Boole

- Due valori

sono i valori su dove si lavora, due valori: 0 e 1.

- false (0)
- true (1)

- Tre operazioni fondamentali

Quelle più comunemente usate, ma ne esistono altre.

- AND (congiunzione) Lavora su due variabili sulla tabella della verità.
- OR (disgiunzione inclusiva)
- NOT (negazione) è la più semplice come operazione.

Tabella della verità, con due variabili: True e false.

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 base di ogni macchina ed è l'unico linguaggio che conosce. Ogni macchina ha il suo linguaggio macchina. CPU è il cuore della macchina, ovvero il processore. a seconda del CPU vi è un linguaggio macchina diverso. Per questo si scrive il codice per il sistema operativo in maniera tale da avere un linguaggio quasi universale.
  - È il linguaggio naturale di un dato computer
  - Ogni hardware può averne uno suo specifico possono cambiare dei dettagli anche significativi, in generale quando cambiano le versioni cambia il linguaggio.
  - Istruzioni e dati sono espressi con sequenze di 0 e 1 Algebra binaria che viene usata.
  - Estremamente difficili per l'uso umano
- **Linguaggi Assembly** Ovvero un linguaggio comprensibile per l'essere umano. Attraverso i comandi. Sono una traduzione umana del linguaggio macchina, ma che rimane incomprensibile la macchina per questo ci sarà bisogno del programma Assembly che tradurrà il linguaggio assembly in linguaggio macchina. Nel linguaggio Assembly vengono usate le variabili.
  - Si usano abbreviazioni in inglese per le istruzioni esempio ADD che significa SOMMA.
  - Più comprensibile agli umani, incomprensibile alle macchine
  - Appositi programmi (assembler) li convertono in linguaggio macchina

CPU: è un acronimo che vuol dire "Unità centrale di calcolo", è la parte del pc che fa i calcoli. Operazioni che svolge la CPU, per fare un'operazione ha bisogno di dati che si trovano in memoria. La CPU riceve i comandi dall'utente e lui li fa. Che operazioni fa? prende i dati dalla memoria (dalla RAM, memoria di accesso casuale). Sinonimo di CPU è CORE e lo si trova ad esempio usato nella vendita del cellulare che hanno più CPU. Serve anche per capire quanta RAM si ha e quindi quanta memoria ci sta, più RAM più velocità. CPU che accede alla RAM, quest'ultima quindi non è esterna, la CACHE è in posizione intermedia tra CPU e RAM. La CACHE è sempre una memoria però più vicina alla CPU e quindi CPU la legge prima della RAM. RAM e CACHE la differenza sta nella velocità la CACHE è più veloce della RAM. Poi, abbiamo i REGISTRI all'interno della CPU, il dato una volta trovato lo si porta nei REGISTRI. La CACHE è più veloce e costosa e quindi le dimensioni sono più piccole rispetto alla RAM. Il BUS prende i valori e li porta nella CACHE o nei REGISTRI della CPU.

RAM: ci sono tutti i dati e programmi che si vuole eseguire.

CPU: legge le istruzioni da eseguire e dati su cui deve lavorare.

Di conseguenza, la CPU interagisce con la RAM ma i dati non sono nella RAM, è un sistema volatile nel senso che tiene i dati finché c'è corrente. Se si vogliono dei dati più permanenti si mettono sulla memoria di massa (chiavetta USB). Per eseguire dalla memoria di massa il sistema operativo lo mette nella RAM e poi passa nella CPU. (La cache è nascosta, il sistema operativo la pesca ma in teoria è nascosta).

# Variabile

VARIABILE: è un modo per comunicare alla CPU che determinati dati ci interessano. E' un concetto logico per esprimere che si vuole lavorare su determinati dati. Tipicamente la variabile ha un nome ( var a = 3). In base a come si definiscono si definiscono i linguaggi di programmazione. Tabella di conversione: l'hanno i linguaggi di programmazione e serve per convertire un numero a un carattere e viceversa. esempio 65 che corrisponde alla lettera A questo perché la semantica è associata, questo avviene nei linguaggi ad "alto livello". (Tabelle UTF-8 in HTML, mentre in Java si usa la tabella UTF-16).

- **Locazione di memoria associata a un nome, contiene un valore** es var a = 3.

La costante la si usa quando non si vogliono cambiare dei valori. Si può anche avere var c = a che vuol dire: che "c" e "a" sono diverse o che si riferiscono alla stessa cella di memoria.

- **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** : Non ci sono variabili con associazioni, ma solo l'architettura. Hanno però il vantaggio di essere aderenti alle macchine. Sono linguaggi simili a quello della macchina. E vengono usati di rado perché sono complessi. Ci sono solo 0 e 1.
- **“alto livello” → tipi complessi** : Significa che nei dati c'è la semantica e quindi i dati vengono definiti e vengono usate le variabili primitive per la definizione dei dati, per affermare se si tratta di un numero intero (int), di un carattere (char) o di un valore booleano (True or false).
- **script → runtime** : I tipi di dati non sono associati quando si compila ma vengono associati in fase di esecuzione. esempio con Java script.

Variabili sono contenute nella Ram e quando servono alla CPU vengono posizionate nei Registri. Quest'ultimi sono contenuti nella CPU e servono per ordinare i dati. Poi la CPU prende un'istruzione per volta e la esegue.

Quando si vuole lavorare su un insieme di variabili collegate l'una con l'altra si usa l'Array. Quindi invece di compilare una serie di variabili si crea un'Array con tutti i valori che servono. Quindi tutti i valori sono all'interno dell'Array e questi elementi si salvano in memoria in particolare nelle celle di memoria, create personalizzate a quanto dovrebbe occupare l'Array.

Concetto base: vettore matematico monodimensionale.

Si tratta di una variabile che ha "n" celle, al contrario della variabile che ha una sola cella. L'Array quindi contiene può valori per questo è una variabile particolare e in memoria viene memorizzata tutta la cella, per accedere ai valori si usa l'indice e quindi si esplicita la posizione del valore scelto da stampare.

Sono collezioni di valori dello stesso tipo anche se Java Script fa eccezione, tutti valori dello stesso tipo ad esempio tutti numeri interi. Il vantaggio di essere tutti dello stesso tipo è che facilmente si accede alla cella che contiene un determinato valore senza scorrere tutti gli elementi.

# Array

La Stringa è come se fosse un Array ma solo di caratteri. Contiene una collezione di caratteri, una sequenza di caratteri. E' un concetto simile ma non è del tutto un Array. L'array potrebbe essere di numeri e caratteri. Ci sono dei linguaggi dove l'array non contiene oggetti dello stesso tipo. (elementi di qualunque tipo: numeri, stringhe, caratteri)

- 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

L'INDICE: Serve per accedere ad un certo elemento.

I linguaggi di alto livello: è un linguaggio comprensivo, sono fatti con termini inglese e notazioni matematiche perché l'informatica moderna deriva dai paesi anglofoni.

Si usano perché sono comprensibili agli umani, e ne sono stati introdotti tanti e se sono di Turing completi saranno equivalenti e quindi risolveranno gli stessi problemi. Ne sono stati introdotti tanti per risolvere le varie esigenze ad esempio linguaggio troppo macchinoso. Problemi che in base agli anni cambiano. Tutti i linguaggi ad alto livello sono scritti allo stesso modo e questo è l'elemento in comune e quindi troveremo la matematica e termini in inglese perché i padri fondatori sono inglesi.

# Linguaggi di alto livello

sono linguaggi simili al modo di ragionamento umano.

- Molto più comprensibili degli assembly assembly è un linguaggio di basso livello perché è una traduzione del linguaggio macchina.
- Termini inglesi e notazioni matematiche
- Possono essere espressi in forma I modi per i linguaggi di programmazione ad alto livello:
  - **imperativa**: si indica cosa deve fare la macchina sequenza di ordini. Ed è la logica prevalente dove alla CPU si danno ordini. es Java  
Java che comunque sta diventando dichiarativo.
  - **dichiarativa**: si indica quale risultato si vuole ottenere esempio HTML (che non è un linguaggio di programmazione); Anche con Java si cerca di farlo diventare dichiarativo perché i compilatori sono più capaci dei programmatori.
- A seconda di come avviene l'esecuzione si parla di linguaggi per renderlo comprensibile alla macchina e questo lavoro lo fa il compilatore che prende il codice comprensibile all'umano e lo rende comprensibile a una specifica macchina esempio per Windows. Il compilatore è una macchina!!
  - **compilati**: conversione del codice in linguaggio macchina, ottenendo un programma eseguibile File scritto in un linguaggio di programmazione che poi il compilatore lo trasforma in un file comprensibile alla macchina(win64), ma non a tutte le macchine(file comprensibile per Windows e uno comprensibile per Linux ad esempio. L'eseguibile è il linguaggio macchina(CPU + sistema operativo). Viene scritto il codice sorgente quando si usa il linguaggio ad alto livello, è il codice che scrive l'umano. Compilatore che prende il codice sorgente e lo trasforma in linguaggio macchina. Una volta fatto questo il programma sarà eseguibile.
  - **interpretati**: il codice viene eseguito da appositi programmi Qui si avrà sempre bisogno dell'interprete. es Java Script  
La logica: scritto il codice lo si dà all'utente, quest'ultimo poi lo carica su un programma chiamato l'"interprete" che ha l'incarico di eseguire il programma. Il browser fa da interprete e quindi non si deve compilare ma il codice deve essere leggibile.  
Questo è comprensibile alla concorrenza per questo meno usato. / utente che scrive il codice sorgente ma non lo compila, non c'è il compilatore ma si dà l'eseguibile all'utente.

Qui non si ha sempre bisogno del compilatore.

# Istruzioni

- Operazioni **sequenziali** : esempio moltiplicare i valori delle variabili. Quindi una serie di operazioni da fare potrebbe essere un esempio di operazione sequenziale. Sono quelle maggiormente usate.
  - Chiedono al computer di eseguire un compito ben definito, poi si passa all'operazione successiva
- Operazioni **condizionali** : se "c" > 0 allora moltiplicato per 2. Esempio di comando di un'operazione condizionale. E troveremo la parola IF
  - Si valuta una condizione, il risultato determina quale operazione seguente verrà eseguita

Ovvero un linguaggio comprensibile per l'essere umano. Attraverso i comandi. Sono una traduzione umana del linguaggio macchina, ma che rimane incomprensibile la macchina per questo ci sarà bisogno del programma Assembly che tradurrà il linguaggio assembly in linguaggio macchina.
- Operazioni **iterative** : ad esempio quando si lavora su un array e si vuole raddoppiare il valore dell'array, l'istruzione sarà quella di fare un ciclo.
  - Richiede di ripetere un blocco di operazioni finché non si verifica una certa condizione – se ciò non accade: loop infinito . I loop: istruzione che viene ciclata all'infinito o fino a quando si arriva al risultato che si vuole.

LINGUAGGIO TURIG COMPLETO deve avere input, output, macchina che deve saper prendere delle decisioni(IF-Else), possibilità di eseguire più volte le stesse istruzioni(loop), contenere istruzioni e variabili.

Variabili: dove metto i dati; Istruzioni: cosa voglio fare con i dati.

# Flow chart vs Pseudo codice

Elementi che usa l'architetto, che farà sapere la sua idea agli sviluppatori.

- Diagrammi a blocchi – flow chart : funziona per blocchi contenenti informazioni dove i rettangoli sono per le operazioni e i rombi per le condizioni. E' un grafico che spiega cosa fare.
  - 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 : si specifica il pensiero scrivendolo.
  - 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

La funzione O GRANDE è una funzione ricca di andamenti(diagramma ad onda del mare) e tratteggiare una retta adiacente per descrivere al meglio l'andamento della funzione. A cosa serve la funzione? Per spiegare quanto costa l'algoritmo(tempo e memoria che occupa) a grandi linee. Stima che non sarà tanto precisa, bisogna però dare una buona approssimazione e per fare questo base utilizzare le funzioni di base (es lineare, algoritmica, etc)

- “O grande”, limite superiore della funzione asintotica
  - Costante  $O(1)$  è parallela all'asse delle "x" che esprime il costo in base ai dati che diamo negli input;
  - Logaritmica  $O(\log n)$  è una curva che parte da - infinito e cresce lentamente all'infinito; Ciò che interessa è la sua crescita. Questa curva viene fuori quando si lavora su un "albero binario", termine matematico che fa riferimento a una struttura che ha una radice con dei nodi che hanno dei figli fino ad arrivare ad una foglia che è un nodo che non ha figli. L'albero binario significa che ogni albero ha 0, 1 o 2 figli. E' una tipologia di dati astratto. Gli alberi che si usano sono quelli della famiglia BST, questi permettono di gestire le strutture ordinate ad esempio l'agenda telefonica in cui la chiave di ordinamento è il nome. ( i dati più piccoli si mettono a sinistra quelli più grandi a destra). Con l'utilizzo di un algoritmo si trova un determinato dato presente nell'albero.
  - Lineare  $O(n)$  è la bisettrice del primo e terzo quadrante. E' una retta che parte dall'origine. esempio: trovare nell'array un determinato valore e si tiene conto del caso peggiore e del caso medio. Tipicamente succede che statisticamente quel determinato valore si troverà in mezzo. Se invece il valore non si trova nell'array saremo presente al caso peggiore(perché si devono visionare tutti gli elementi dell'array). Più è grande l'array più il costo sale.
  - Linearitmico  $O(n \log n)$  Ha un costo contenuto ed è sia Lineare che logaritmica. Fa riferimento al caso migliore.
  - Quadratica  $O(n^2)$  – Polinomiale  $O(n^c)$  è una parabola e l'algoritmo sarà complesso e quindi si dovrà optare di lavorare con pochi dati o si lascia perdere perché si perderebbe troppo tempo
  - Esponenziale  $O(c^n)$  si tratta di un'altra situazione complessa
  - Fattoriale  $O(n!)$  situazione molto complessa!!
- Tempo e spazio Le variabili che interessano, maggiormente il tempo conta. Se l'algoritmo occupa tanto spazio saranno più veloci al contrario saranno più tempo.
- Caso migliore, peggiore, medio ciò che va preso in considerazione per capire la complessità di un determinato algoritmo. e si deve guardare al caso medio al fine di evitare il caso peggiore e far avverare quello migliore. Il caso peggiore si cerca sempre di evitarlo quando si può! e bisogna capire qual è il caso migliore! e il caso migliore è "n log n".

Quando si fa un loop un esempio quotidiano è quello di sistemare in ordine le carte da gioco!



# 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 "n" quadro è il costo del logaritmo.
- $O(n \log n)$ : dimostrato ottimale per algoritmi basati su confronto
- $O(n)$ : casi o uso di tecniche particolari

Con un ragionamento matematico si può arrivare agli algoritmi  $n \log n$ . Matematici che hanno dimostrato che questo equivale al caso migliore.

ci sono dei casi particolari in cui "n log n" può essere sostituito da un altro caso migliore. Ma normalmente non si riesce a scendere da "n log n".

Quindi ci possono essere dei casi logaritmici meno complicati, ma ci vogliono delle tecniche particolari. Di solito il meno complicato dove si arriva è "n log n".

# Ingegneria del software

- Approccio sistematico alla creazione del software
  - Struttura, documentazione, milestones, comunicazione e interazione tra partecipanti
- Analisi dei requisiti che fa riferimento al progetto e quindi si analizza l'idea e si analizzano i costi, l'utilità del prodotto.
  - Formalizzazione dell'idea di partenza, analisi costi e usabilità del prodotto atteso
- Progettazione Si definisce la struttura del codice, e quindi ci si prepara allo sviluppo. Si preparano le specifiche per i programmatori.
  - Struttura complessiva del codice, definizione architetturale
  - Progetto di dettaglio, più vicino alla codifica ma usando pseudo codice o flow chart
- Sviluppo nel momento in cui arrivano le direttive dal progettista si trasformano in codice, si scrive il codice e si testa per dimostrare se sia buono o meno.
  - Scrittura effettiva del codice, e verifica del suo funzionamento via **unit test** 
- Manutenzione il codice una volta prodotto andrà modificato perché ci saranno nuovi requisiti magari spunteranno fuori degli errori e quindi andrà rivisto. E' un lavoro complesso, è più difficile mantenere il codice invece di crearlo. es. l'aggiornamento è un esempio di manutenzione del codice.
  - Modifica dei requisiti esistenti, bug fixing

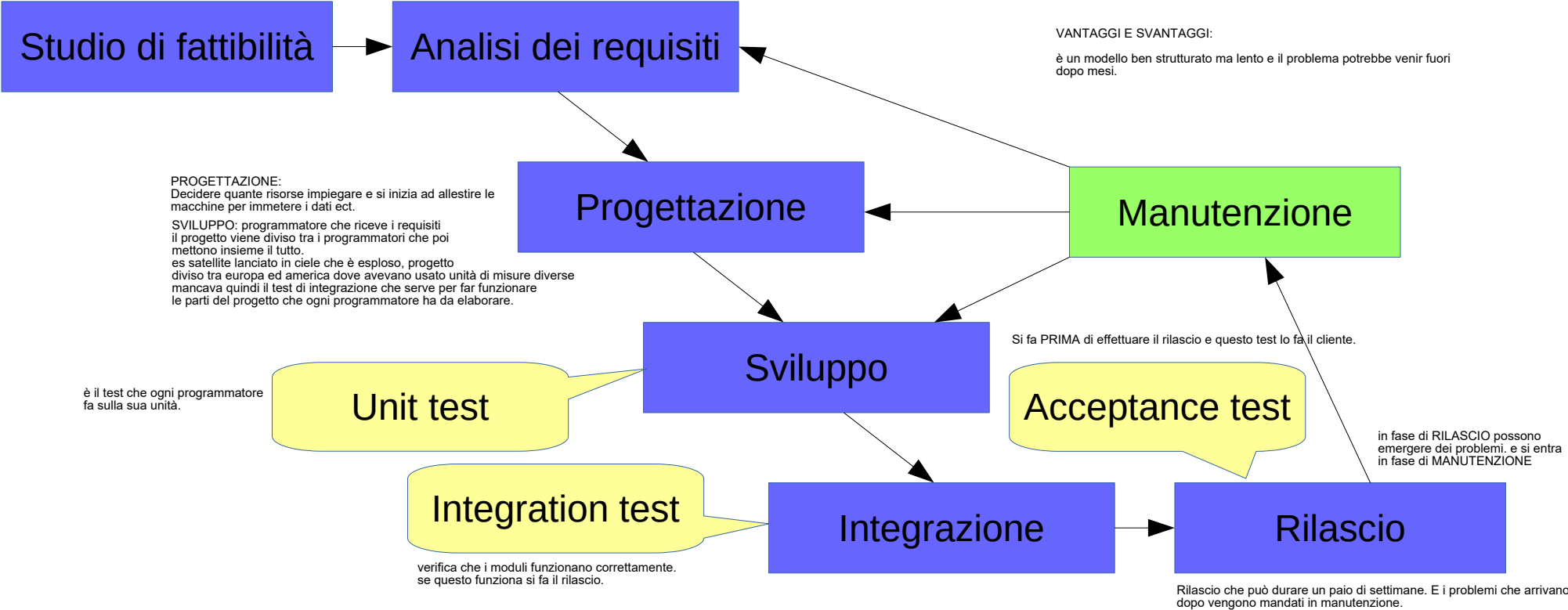
# Unit Test

- Verificano la correttezza di una singola “unità” di codice
  - Mostrano che i requisiti sono rispettati
- Verifica ciò che si verifica durante il test:
  - Casi base (positivi e negativi)
  - Casi limite importanti perché gli errori tipicamente si fanno sui casi limite.
- Ci si aspetta che siano Le caratteristiche del test:
  - Ripetibili: non ci devono essere variazioni nei risultati deve ritornare lo stesso risultato. Requisito difficile da ottenere se l'input non è standard. ad esempio se il test è basato su una funzione che genera numeri casuali. (bisognerebbe lavorare su numeri pseudo casuali in modo tale che possano essere ripetibili).
  - Semplici: facile comprensione ed esecuzione
  - E che offrano una elevata copertura del codice nel caso di codici complicati se possibile tutti i possibili flussi devono essere coperti, questo se presenti tanti -IF.

# Modello a cascata (waterfall)

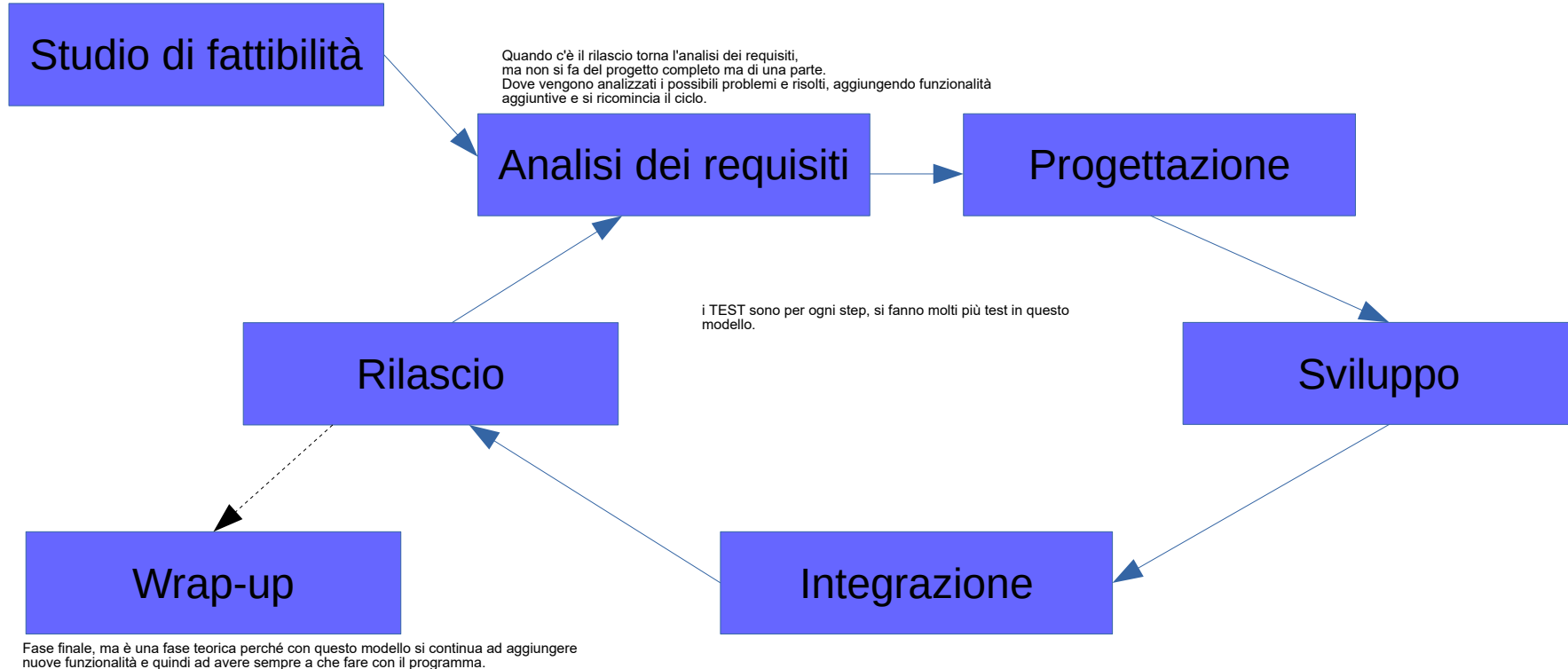
es: nuova compagnia telefonica che vuole un programma per l'emissione delle fatture per i propri clienti. Quindi il cliente dice cosa vuole e il programmatore dice cosa si può fare.

una volta capito cosa vuole il cliente si analizzano i requisiti. In questo step si fa una stima dei costi e quando si consegnerà il prodotto finito. Tipicamente alla fine di questa fase al cliente si danno dei documenti e il cliente decide se accettare o meno. documenti che spiegano bene i costi e l'elaborazione del prodotto in questione. Quando si ha l'ok dal cliente si passa alla progettazione.



# Modello agile

Lo svantaggio è che non si può dare una stima dei costi e questo è un grosso problema e per questo non piace alle aziende che vogliono un prodotto, non un programma da incrementare con il tempo. Dove le scadenze non sono certe.  
E' un modello utilizzato dalle start up.  
il vantaggio è che c'è tanta comunicazione perché il progetto è piccolo e tutti possono interagire.  
è difficile fare previsioni: Svantaggio  
Tanta comunicazione : Vantaggio



Ci sono pro e contro in entrambi i modelli, vengono scelti in base alle esigenze dei clienti.

# 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