

Corso Web MVC

Introduzione

Emanuele Galli

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

La calcolatrice si interfaccia con problemi limitati ad un ambito specifico.

Hardware, Software, Firmware

è un programma che entra in gioco all'inizio, controlla le funzionalità minime di base e poi cede il controllo al sistema operativo (se c'è doppio sistema ti fa decidere quale). poi rimane sempre in attività, nella ram.
boot o bootstrap: nome generico dell'operazione di avviamento.

- **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** programmi difficilmente modificabili
 - Programma integrato in componenti elettroniche del computer (ROM, EEPROM)
 - UEFI / BIOS: avvio del computer
 - Avvio e interfaccia tra componenti e computer

read only memory: dentro posso mettere dati e programmi, ma una volta scritti non possono più essere toccati. il senso? dentro ci metto il firmware

memorie riprogrammabili: utile perché anche nel firmware ci sono i bug

Sistema Operativo

deve essere comprensibile tanto per gli umani quanto per gli applicativi perché entrambi si interfacciano con la cpu tramite il 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

attraverso una virtual machine (che fa finta di essere un computer) posso accedere ad un altro sistema operativo che viene visto come un programma qualunque
Problema: il programma che voglio eseguire con l'altro SO è linguaggio macchina scritto per essere eseguito ad es su linux, ma la macchina non è linux, è ad es windows, quindi ci sono una serie di traduzioni di linguaggio macchina che chiaramente rallentano l'esecuzione (non massima performance)

Il file system è un albero (c:/ -> c: in questo caso è il nome del disco ed è la radice, root, dell'albero, i cui nodi rappresentano una cartelletta).
In un file system ci sono folder e file
il folder (o directory) può contenere file e altri folder.
il file è l'unità minima di memorizzazione su una memoria di massa. Il file ha un nome (file name).

Stringa: serie di caratteri. Il tipo è definito, non possono che essere caratteri.

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

GIGO=garbage in, garbage out. se mi dai schifo in entrata, sarà schifo anche l'uscita

AND, = sono operatori
l'inizializzazione, il loop sono istruzioni

serie di passi molto chiari che permette di risolvere un problema partendo dalle condizioni di partenza. ricetta rigorizzata

Algoritmo

istruzione è unità minima
dell'algoritmo. è un comando che
viene dato alla cpu

- 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 KISS=keep it simple, stupid!
solo due valori (0=false, 1=true)



- La macchina di Alan Turing ~1930

è possibile avere un programma automatico che mi dica se una ipotesi di teorema è vera o falsa? Turing dimostra di no.

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

- Ingegneria

la macchina di Turing in forma ingegneristica

- La macchina di John von Neumann ~1940

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

1 byte (8 bit) è l'unità minima indirizzabile nella ram.
se non facciamo errori le nostre variabili non hanno misura più piccola di 1 byte, 8 bit.
un'altra unità fondamentale è una WORD, che mi descrive proprio le caratteristiche della macchina -> 64bit, 1 word (8byte).
ha senso dare nel linguaggio di programmazione al tipo "int" la dimensione di 1 word, al "long" (intero lungo) la dimensione di 2 word.

RAM = random access memory
memoria su cui caricare i programmi che vengono eseguiti.
più c'è di ram più possono essere caricati programmi pesanti.
NB ram e cpu non sono correlati.
la ram è un sistema di memorizzazione volatile. tiene i dati finché c'è corrente, per questo è così veloce. ogni volta che si spegne il pc, sparisce.
la memoria cache è una memoria più veloce (10 volte la ram), si mette fra cpu e ram. è più piccola (di 4 volte).

il clock determina la velocità della cpu
sinonimo di cpu -> CORE
i registri sono le locazioni dove effettivamente viene messo il dato per fare l'operazione; fanno parte della cpu.
la cpu ordina al bus di sistema di andare a recuperare i dati dalla ram per inserirli nei registri. i dati potrebbero però essere già nella memoria cache, quindi il bus le prende (più velocemente) da lì.
il bus di sistema è proprio come un bus che va a prendere i dati nel computer e li trasporta.

Regole:
AND ha la priorità (come
fosse moltiplicazione)

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

unico linguaggio che la macchina conosce; ognuna ha il suo.
il cuore della macchina è la CPU (= central process unit, processore)

- 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

concetto logico con cui si dice ad un programma che quelli sono dati da lavorare

raccolte di 64bit

- 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)

`var c=a` potrebbe voler dire più cose e linguaggi diversi gestiscono questa cosa in maniera diversa

- Supporto a tipi di variabili da linguaggi di:

64bit= si intende la potenza dei registri della cpu, nonché la dimensione del bus. di solito sono uguali.

1 bit è l'unità minima di memorizzazione.

il bit determina l'esponente della base (2, perché si lavora nel sistema binario).

ES. con 64bit si possono mettere 2^{64} -1 valori (perché si deve contare anche lo 0)

- “basso livello” → legati all'architettura della macchina
- “alto livello” → tipi complessi
- script → runtime

si mette la semantica nei dati.

ES. `int a=3`, ti dico che quella è una variabile numero intero

ES. `char b = "x"`, quella variabile è un carattere. ma come faccio a rappresentare x che non è un numero? si fa una tabella di conversione. so che quella variabile è un carattere quindi vado a vedere qual valore a che carattere corrisponde (in Java vedremo la UTF-16 cioè quanti bit vengono utilizzati per visualizzare il carattere)

e per i numeri negativi? per questo è facile. si prende il bit più a sinistra e si mette 1 quando il valore è negativo. quindi se ho 64 bit ne uso per il numero solo 63 perché uno è per il segno positivo o negativo.

quindi se lavoro in 8 bit, 7 li uso per il valore, 1 per il segno. i valori saranno sempre 256, ma non 0...255, ma -128...127

i tipi di dati non sono associati, viene fatto in itinere. non c'è un gran controllo del programmatore sulla variabile

lista indicizzata!

Array

faccio una variabile sola e dico che è una "collezione", una lista; quindi la variabile contiene non solo un valore, ma più valori.
ES: `var temperatures= [12.1, 18.7...]`
tipicamente si prende un blocco in memoria di tot valori, tot celle ognuna del valore di 64 bit (var as, il plurale per indicare che contiene più valori)

- 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)
se voglio accedere ad un elemento, scrivo il nome della variabile e metto fra [...] l'indice che mi interessa.
ES. `temperatures [i]`
ES. `print(as[2])`
problema: devo sapere la posizione del dato.
- Gli elementi sono allocati in un blocco contiguo di memoria, il che permette accesso immediato via indice ai suoi elementi

Linguaggi di alto livello

Tutti i linguaggi turing completi hanno la stessa valenza, nel senso che un problema risolvibile con un linguaggio è risolvibile anche con un altro. Sono però come degli strumenti, alcuni vanno meglio per una cosa, altri per un'altra.

- 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 il compiler fa questo lavoro, cioè prendere il codice comprensibile agli umani (codice sorgente, source code) e trasformarlo in una serie di 0 e 1 comprensibile ad una certa macchina (es. Windows 64bit)
 - **interpretati**: il codice viene eseguito da appositi programmi
C'è un interprete, non c'è un compilatore.
ES. javascript viene seguito dal browser, quindi può correre in un sistema Linux, come Windows

Linguaggi Turing completi:
- variabili
- operazioni condizionali (if - else)
- istruzioni iterative (while/for) -> loop
- blocchi di istruzione
- gestire un input/dare un output

Istruzioni

- Operazioni **sequenziali** più operazioni vengono eseguite in maniera sequenziale, nell'ordine in cui vengono scritte
 - 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
```

praticamente tutti i linguaggi di programmazione hanno l'if, ossia si esegue una o l'altra operazione a seconda che la condizione sia vera o falsa
 - Si valuta una condizione, il risultato determina quale operazione seguente verrà eseguita
- Operazioni **iterative**

```
for(each el in >) el= el*2
```

si può chiedere di eseguire un'operazione per più valori, o in un certo intervallo, ciclo. arrivato in fondo si ferma
 - Richiede di ripetere un blocco di operazioni finché non si verifica una certa condizione – se ciò non accade: loop infinito

while -> looppa finché la condizione è vera
for -> looppa per tutta la durata dell'array

Flow chart vs Pseudo codice

viene rappresentato il programma come una specie di diagramma

codice che assomiglia al codice che si dovrà fare

- 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 sopra un'altra funzione e che possibilmente le si avvicina il più possibile

- “O grande”, limite superiore della funzione asintotica

- Costante $O(1)$
- Logaritmica $O(\log n)$ poco peggio della costante, perché cresce, ma in maniera estremamente lenta
- Lineare $O(n)$ Più è lungo l'array più mi posso aspettare che il costo aumenti.
è lineare perché cerca elementi uno per uno, quindi continua a ciclare finché non lo trova.
- Linearitmico $O(n \log n)$ un po' peggio di quella logaritmica.
accade quando c'è un sorting
- Quadratica $O(n^2)$ – Polinomiale $O(n^c)$ cresce in maniera esponenziale (curva della parabola).
- Esponenziale $O(c^n)$
- Fattoriale $O(n!)$

- Tempo e spazio come costo mi interessa quanto tempo ci mette e in quanto spazio, ma soprattutto il tempo

- Caso migliore, peggiore, medio


Alberi

Alberi binari: un ramo può avere solo uno o due figli. Alberi della famiglia BST (binary sorted tree): albero che permette di gestire le sequenze ordinate (es. agenda telefonica un nodo: i nodi alla sx sono più piccoli, i nodi alla dx più grandi) Ul livello 0 c'è un nodo, al livello 1, 2 e così via. Se vengono inseriti valori “strani” ci sono dei programmi per riarrangiare l'albero in modo che la ricerca di un valore costi sempre in maniera logaritmica

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

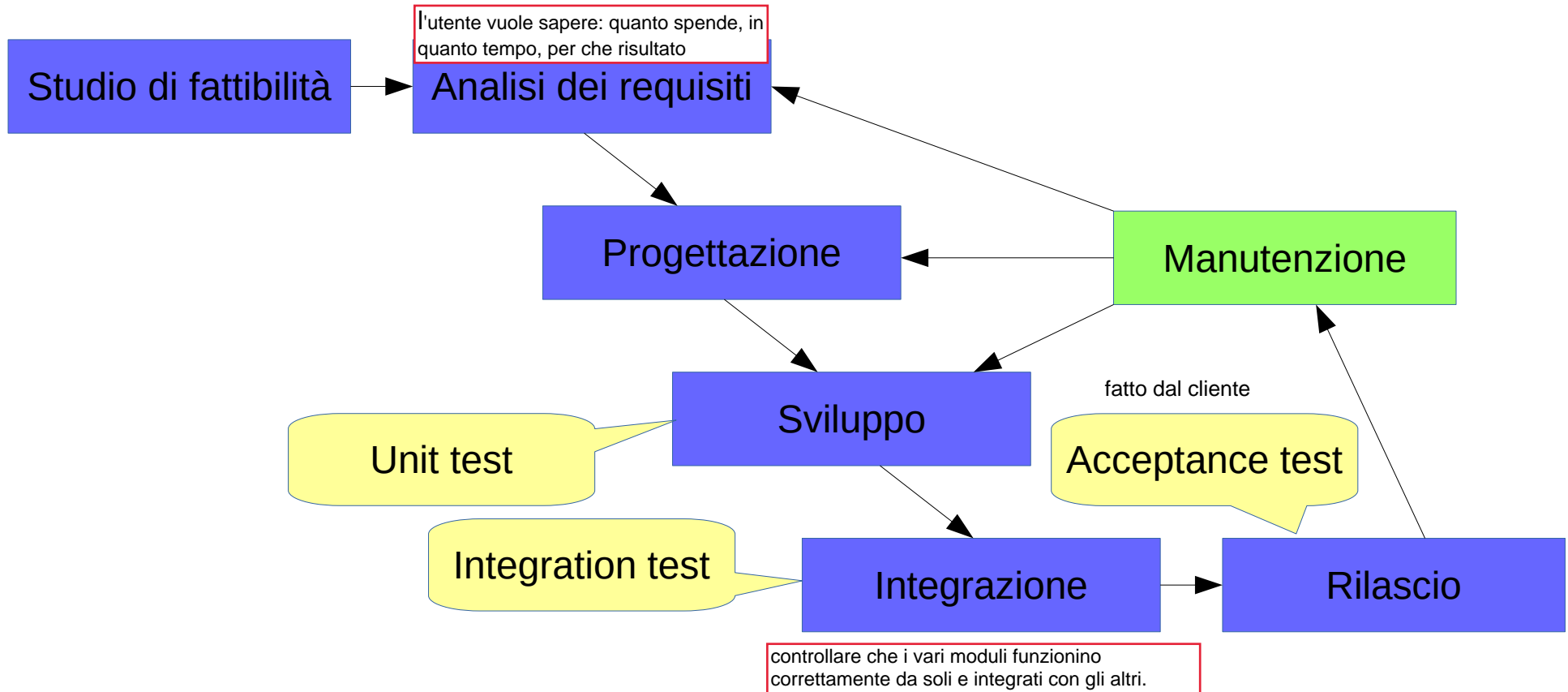
- Approccio sistematico alla creazione del software
 - Struttura, documentazione, milestones, comunicazione e interazione tra partecipanti
- Analisi dei requisiti capire davvero cosa si vuole ottenere, quale vuole essere il risultato finale
 - Formalizzazione dell'idea di partenza, analisi costi e usabilità del prodotto atteso
- Progettazione MVC: model view controller
 - 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

Test Driven Development: prima di scrivere la funzione dovrei cominciare a scrivere i test. Cosa può andare storto?

Unit Test

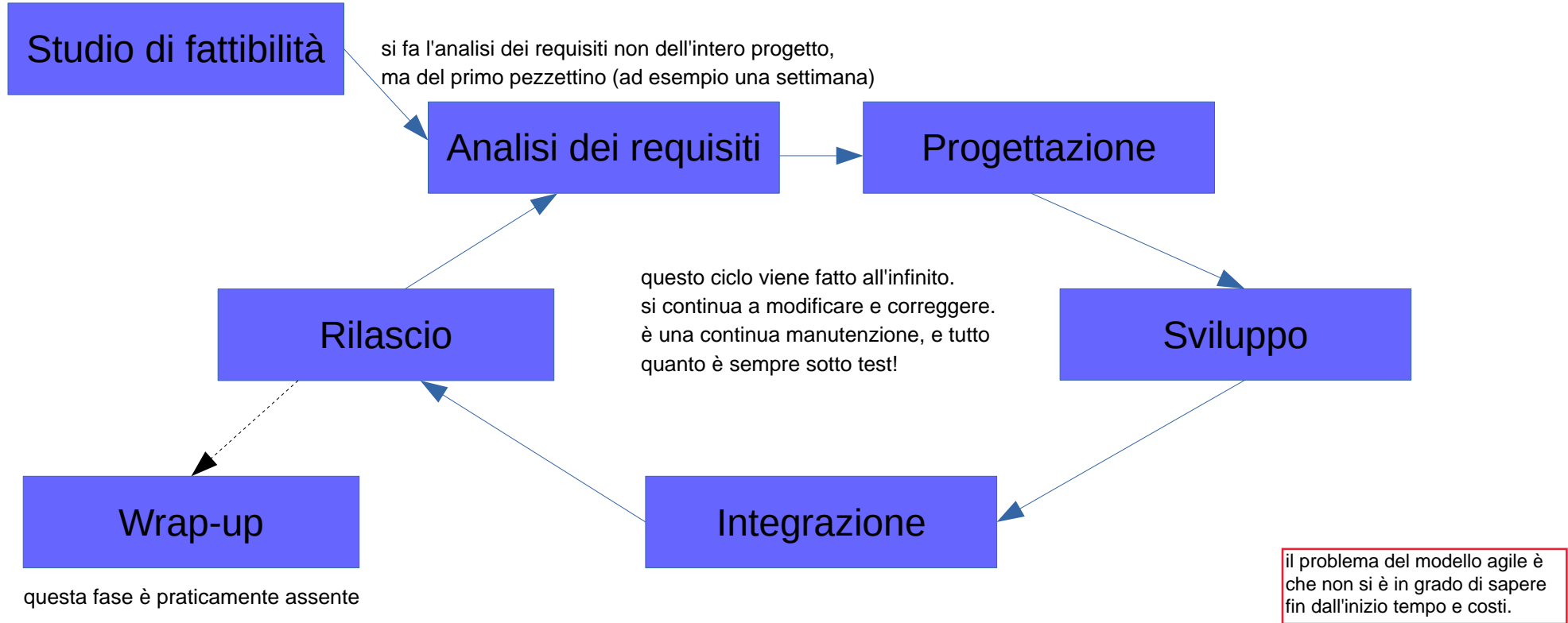
- Verificano la correttezza di una singola “unità” di codice
 - Mostrano che i requisiti sono rispettati
- Verifica
 - Casi base (positivi e negativi)
 - Casi limite sono i casi in cui di solito si presentano gli errori
- Ci si aspetta che siano
 - Ripetibili: non ci devono essere variazioni nei risultati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

Modello a cascata (waterfall)



N.B. tutti comunicano con tutti. è il classico modello da start-up

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