

Classe A-041 Scienze e tecnologie informatiche

Capitolo 1

Dal problema al programma

1.1 Soluzione dei problemi: processi euristici e processi algoritmici

Il concetto di metodo informatico è collegato a quello di **processo algoritmico** ed entrambi si agganciano alle tecniche di risoluzione di problemi attraverso sistemi elettronici di calcolo. Un problema è percepito come una situazione nuova, che non si sa con certezza come affrontare. Il concetto di “problema” può avere svariate accezioni in quanto può applicarsi sia a strumenti di valutazione nell’ambito di discipline specifiche, come nel caso di “problemi” di matematica o fisica, sia a metodologie di sviluppo per l’apprendimento integrato del sapere scientifico. In generale, però, alla soluzione di tali situazioni problematiche non si perviene attraverso la deduzione o l’induzione, ma attraverso l’ideazione di una soluzione originale che non deriva dall’applicazione di principi astratti né dalla sola esperienza passata. In linea di principio, sono solo due le tipologie principali di ragionamento utilizzate nella risoluzione di problemi: algoritmi ed euristiche.

- Un algoritmo è un procedimento di calcolo che si basa sull’applicazione di un numero finito di regole che determinano in modo meccanico tutti i singoli passi del procedimento stesso. Nel Medioevo con il termine “algoritmo” si indicava ogni procedimento mediante il quale si eseguivano le operazioni tra i numeri naturali (ad esempio addizione, sottrazione, moltiplicazione e divisione) attraverso la loro rappresentazione decimale con le cifre arabe. In seguito il termine è stato esteso a indicare ogni procedimento che consente di risolvere un qualsiasi problema, relativo anche a enti non numerici, in modo meccanico, mediante l’applicazione di un sistema esplicito di regole effettive (un esempio concreto di algoritmo può essere la ricetta di preparazione di una pizza, dato che descrive a partire dagli ingredienti le operazioni che devono essere eseguite passo a passo).

Per avere un’idea dei campi di maggior impiego di strategie basate sull’impiego di algoritmi, basti pensare che l’individuazione di algoritmi ha accompagnato la storia di tutti i settori della matematica, in quanto la dimostrazione di esistenza delle soluzioni per un qualunque problema è sempre stata accompagnata dalla ricerca di regole per “calcolare effettivamente” tale soluzione. Ma la ricerca di algoritmi ha assunto particolare rilevanza con l’affermarsi dell’informatica, in quanto i computer sono essenzialmente esecutori di algoritmi. Di conseguenza gli algoritmi – ideali per il funzionamento di una macchina con enormi capacità a livello di gestione di sistemi

di memoria come il computer – spesso risultano inattuabili per gli esseri umani, in quanto richiederebbero di prendere in considerazione un numero talmente elevato di possibilità da risultare, quand’anche attuabili, del tutto antieconomici dal punto di vista del tempo e dell’impegno cognitivo.

- Contrariamente a quello algoritmico, il procedimento euristico è basato sull’intuizione e sullo stato temporaneo delle cose con l’obiettivo di generare una nuova conoscenza. In informatica l’euristica è una regola pratica dettata dall’esperienza passata (un esempio pratico potrebbe essere quello del riavvio di una macchina quando è difettosa).

Per risolvere un problema mediante un algoritmo, infatti, si utilizzano conoscenze specifiche che vanno al di là della definizione del problema stesso. Quelli euristici sono procedimenti logici dominati dall’incertezza e quindi legati al probabile e al possibile; i procedimenti algoritmici sono governati da logiche “certe”. **L’algoritmo è tipicamente sequenziale (step by step), l’euristica, invece, è fondamentalmente reticolare.** La procedura euristica è più rapida di quella algoritmica, ma il raggiungimento della soluzione corretta non è assicurato in quanto tale procedura si fonda su strategie di soluzione dei problemi basate sull’analisi di un numero limitato di alternative, selezionate in quanto ritenute le più promettenti, così da ridurre il tempo di ricerca rispetto all’esame completo e sistematico di tutte le possibili risposte. **Un esempio di euristica è l’analisi mezzi-fini, consistente nella progressiva riduzione della distanza tra la condizione di partenza e l’obiettivo da raggiungere.** Tale riduzione si opera scegliendo, tra le alternative che si presentano in ciascun passaggio del processo di soluzione del problema, quella che avvicina maggiormente alla meta finale. **È come se, percorrendo una strada, a ogni incrocio si scegliesse la via che conduce più vicino al posto in cui si vuole giungere.** Le euristiche sono di conseguenza una procedura molto semplificata rispetto agli algoritmi, non indicando con precisione ogni azione che il soggetto impegnato nella soluzione del problema deve o dovrebbe intraprendere, e non garantiscono, inoltre, il raggiungimento della soluzione del problema di partenza. In compenso sono particolarmente flessibili ed “economiche”, perciò vengono spesso utilizzate in maniera spontanea nella vita di tutti i giorni, anche con buoni risultati. Esse vengono utilizzate sia per la risoluzione di problemi di cui conosciamo dati certi su cui lavorare, sia per stimare situazioni dove è lasciato più ampio spazio alla possibilità, e si tratta quindi di stimare la probabilità di eventi che ci sono presentati, per procedere poi alla soluzione o a una valutazione della situazione stessa. Nella risoluzione di problemi applicare un’euristica vuol dire innanzitutto suddividere il problema in più fasi. La prima fase è realizzare l’esistenza di un problema (se non mi rendo conto di dover “fare” qualcosa, difficilmente cercherò di attuare una qualsiasi strategia solutoria), e comprenderne la natura (che informazioni possiedo? di quali informazioni ho bisogno?). Sulla base della rappresentazione del problema che ci si costruisce a partire da questa prima panoramica si passerà a cercare di impostare un piano per la soluzione del problema stesso (quali strategie impiegare,

come impiegarle, che obiettivi intermedi proporsi, anticipare i possibili ostacoli o difficoltà e ipotizzare possibili strade alternative per evitarli ecc.); si tratterà poi di mettere in atto il piano ideato e infine di valutarne gli esiti. Tra tutte le fasi elencate la più importante per la buona riuscita del processo solutorio è la prima: fondamentale è infatti il modo in cui ci si rappresentano il problema, gli obiettivi da raggiungere e le risorse a disposizione.

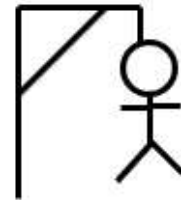
NB: Riassunto. Quindi:

- gli **algoritmi** sono delle procedure che se applicate ricorsivamente consentono di risolvere correttamente il problema.
- le **euristiche** sono strategie generali di soluzione e hanno come obiettivo soltanto la semplificazione dei problemi.

Esempio: Gioco dell'impiccato

Algoritmo: inserire negli spazi vuoti tutte le possibili lettere
non è funzionale perché la soluzione non viene trovata in tempo

Euristica: formulare ipotesi sulla parola tenendo conto delle
sequenze di lettere possibili



1.2 Tecniche di rappresentazione degli algoritmi: Flow-Chart e pseudocodice

Gli algoritmi possono essere rappresentati nei seguenti due modi:


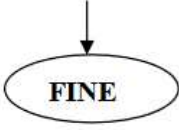
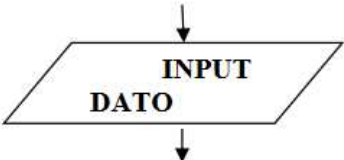
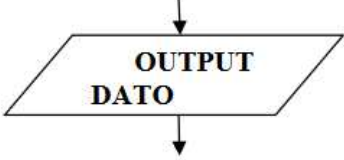
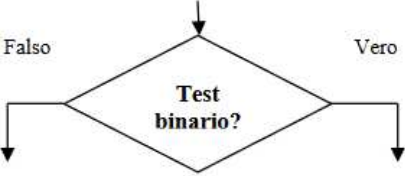



- **Diagrammi di flusso** o **flow-chart**, che hanno il vantaggio di evidenziare visivamente il flusso di esecuzione dell'algoritmo. Permettono la descrizione delle istruzioni con frasi rigorose e non ambigue.
- **Pseudocodifica**, vicino al linguaggio naturale, utilizza un insieme di parole chiave (parole che descrivono il linguaggio) che sono un sotto insieme del nostro vocabolario. Uno pseudolinguaggio non è direttamente eseguibile da un compilatore (programma che permette di trasformare il codice informatico in eseguibile), ma è facilmente comprensibile all'uomo, pur rimanendo un linguaggio formale.

Flow Chart

I diagrammi a blocchi (detti anche diagrammi di flusso, flow chart in inglese) sono un linguaggio di modellazione grafico per rappresentare gli algoritmi. I flow chart vengono creati attraverso l'uso di specifici simboli concatenati fra loro attraverso segmenti orientati (freccie di connessione). Una combinazione di blocchi elementari descrive un algoritmo se:

- Viene usato un numero finito di blocchi.
- Lo schema inizia con un blocco iniziale e termina con un blocco finale.
- Ogni blocco soddisfa delle condizioni di validità.

I blocchi elementari sono i seguenti:

Blocco Iniziale	Blocco Finale
	
Blocco INPUT	Blocco OUTUP
	
Blocco di controllo	Frecce di connessione
	
Blocco Operazione	Insieme di blocchi
	


BLOCCO INIZIALE

Il blocco Iniziale viene posto all'inizio dell'algoritmo ed è unico per ogni dato algoritmo. Il blocco iniziale indica il punto in cui deve iniziare l'esecuzione

BLOCCO ELEMENTARE	PSEUDOCODICE
	INIZIO oppure START Con Algobuid è già sottinteso con: PROG nomeProgramma

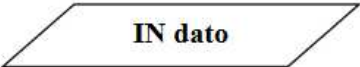
BLOCCO FINALE

Il blocco di fine è analogo al precedente, ma segnala la fine dell'algoritmo. Viene messo dunque per indicare il termine dell'esecuzione. A differenza del blocco iniziale, ci possono essere più blocchi finali per un singolo algoritmo, cioè un algoritmo può terminare, a seconda dei casi, in punti diversi.

BLOCCO ELEMENTARE	PSEUDOCODICE
	FINE oppure END

BLOCCO DI INPUT

Il blocco INPUT rappresenta l'operazione di INPUT che consente di acquisire un dato dalla tastiera (o da altra periferica d'ingresso) e salvarlo in una specificata variabile.

BLOCCO ELEMENTARE	PSEUDOCODICE
	IN dato oppure LEGGI dato


Per esempio:



Significa che il computer si mette in attesa che l'utente digiti il dato. Il dato digitato viene memorizzato nella variabile X di un certo tipo.

BLOCCODI OUTPUT

Il blocco di OUTPUT si riferisce all'operazione di OUTPUT che consente la stampa a video (o su altra periferica di uscita) del dato contenuto in una variabile o di una stringa di caratteri (messaggio).

BLOCCO ELEMENTARE	PSEUDOCODICE	
	VARIABILE	MESSAGGIO
	OUT dato	OUT "messaggio"
	oppure	oppure
	SCRIVI dato	SCRIVI "messaggio"
	Oppure	Oppure
	COMUNICA dato	COMUNICA "messaggio"

L'istruzione OUT permette la stampa a video di un messaggio (indicato fra i doppi apici) o la stampa a video del contenuto di una variabile.

Esempio 1:



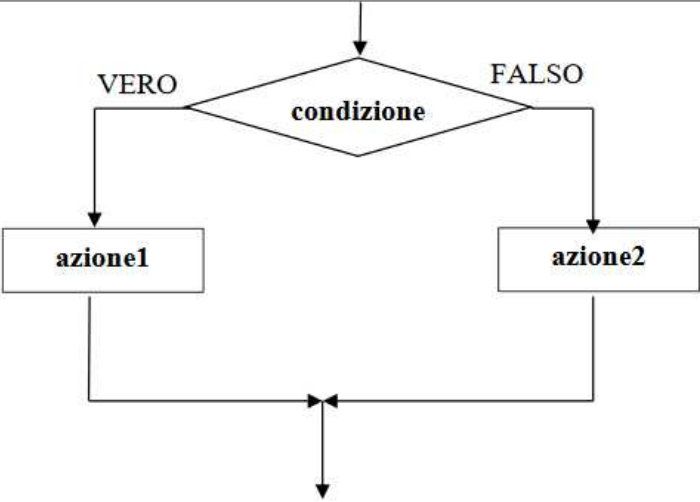
BLOCCO OPERAZIONE/AZIONE

Il blocco operazione o di elaborazione indica l'esecuzione di una qualsiasi operazione all'interno dell'algoritmo. Il simbolismo da utilizzare specifica che occorre utilizzare l'operatore di assegnamento, \leftarrow oppure $=$. Alla destra dell'operatore si scrive l'espressione matematica numerica che dovrà essere eseguita dal calcolatore. Il calcolatore valuta tale espressione e ottiene un risultato numerico che viene assegnato alla variabile indicata alla sinistra dell'operatore

BLOCCO ELEMENTARE	PSEUDOCODICE
<div>variabile \leftarrow espressione</div> <div>oppure</div> <div>variabile = espressione</div>	<div>variabile \leftarrow espressione</div> <div>oppure</div> <div>variabile = espressione</div>

BLOCCO DI SELEZIONE

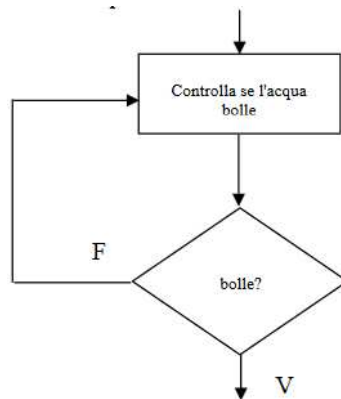
Il blocco di selezione binaria o controllo permette la selezione fra due possibili situazioni. Viene valutata l'espressione logica(condizione) contenuta nel rombo. Se la condizione è vera, allora, la procedura proseguirà eseguendo le azioni del ramo vero, altrimenti verranno eseguite le azioni del ramo falso.

BLOCCO ELEMENTARE	PSEUDOCODICE
	<div>SE condizione ALLORA</div> <div>azione1</div> <div>ALTRIMENTI</div> <div>azione2</div> <div>oppure</div> <div>IF condizione THEN</div> <div>azione1</div> <div>ELSE</div> <div>azione2</div>

CICLO DI ATTESA

Ci sono degli algoritmi che non possono essere rappresentati con i precedenti diagrammi di flusso; un esempio di questi, il processo (algoritmo) di "cottura della pasta", composto dai seguenti passi: prendi la pentola, riempi di acqua, mettila sul fuoco finché l'acqua

bolle, ecc., prevede un "**ciclo d'attesa**", nel senso che si deve rimanere in attesa controllando periodicamente se l'acqua sta bollendo finché l'acqua bolle. Lo schema sarà:



Il "ciclo d'attesa" consiste quindi nella ripetizione (iterazione) di un'azione (il controllo dello stato dell' acqua) per un numero di volte che, in questo caso, non è noto a priori.

Pseudocodifica

Un altro formalismo di codifica, che risulta essere più legato al linguaggio naturale scritto rispetto ai diagrammi di flusso, è la pseudocodifica. In tale rappresentazione si utilizza un linguaggio speciale che descrive le istruzioni con frasi rigorose anziché con i simboli grafici; si usano parole chiave, scritte in maiuscolo ed operatori (\leftarrow , $+$, $*$, $-$, $/$); si usa inoltre l'indentazione, una tecnica che prevede il rientro dei gruppi di istruzioni riferite a cicli o a strutture di scelta.

Scopo della pseudocodifica è di portare il risolutore a esprimere le proprie istruzioni in una forma naturale, utilizzando frasi ed espressioni elementari della lingua scelta. Non esiste un pseudolinguaggio standard e proprio per questo ogni pseudolinguaggio ha un proprio lessico, una propria sintassi e una propria semantica.

1.3 Proprietà degli algoritmi: La programmazione strutturata, complessità

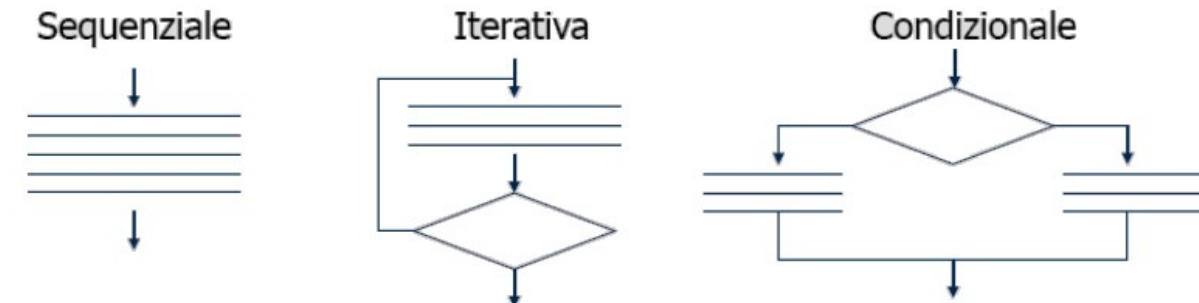
La programmazione strutturata rappresenta una tappa fondamentale dell'evoluzione della così detta **programmazione mainstream**, ovvero di quella sequenza di paradigmi che, nel corso degli anni, l'uno succedendo all'altro, hanno dominato il mondo dell'industria del software.

Tali metodi programmazione si basa sull'utilizzo di tre costrutti sintattici fondamentali, noti come strutture di controllo:

- sequenza;
- selezione;
- iterazione.

Utilizzando opportunamente i tre costrutti sintattici fondamentali possiamo scrivere qualsiasi algoritmo. A tal proposito due matematici italiani, Corrado Bohm and Giuseppe Jacopini nel 1966 enunciarono nel seguente Teorema:

“un algoritmo scritto secondo le regole della programmazione assalti, per quanto complesso, può essere sempre trasformato in un algoritmo a esso equivalente che utilizzi esclusivamente tre costrutti sintattici fondamentali: sequenza selezione e iterazione”



Lo **schema di sequenza** è una sequenza di passi eseguiti uno alla volta, nessun passo è ripetuto e l'ordine di esecuzione dei passi è lo stesso in cui sono scritti.

Lo **schema di selezione** è usato quando si deve effettuare una scelta tra due passi alternativi.

Lo **schema di iterazione** permette la ripetizione di certi passi un numero arbitrario o fisso di volte.

Complessità

Dato un problema, esistono diversi algoritmi che permettono di risolverlo. I fattori che possono influenzare la scelta dell'algoritmo migliore sono principalmente due:

- tempo di elaborazione;
- quantità di memoria.

Il modello che permette di caratterizzare l'efficienza di esecuzione è definito come **complessità computazionale** e l'efficienza in termini di memoria impiegata come **complessità spaziale**.

La valutazione della complessità computazionale deve prendere in considerazione alcuni requisiti:

- 1) **l'indipendenza dal sistema di elaborazione sul quale si intende eseguire il programma**: dato un programma, l'efficienza dell'esecuzione dipende dal compilatore impiegato che effettua una traduzione in codice macchina che può essere più o meno efficiente, dalla potenza della cpu, ecc
- 2) **Dipendenza dei dati in ingresso**. Il tempo di esecuzione dell'algoritmo varia al variare dei dati di ingresso che l'algoritmo elabora per produrne degli altri come possibile soluzione del problema
- 3) **Dipendenza dalla dimensione dei dati**. La complessità computazionale di un algoritmo dipende sempre dal numero dei dati, cioè i dati di input, che devono essere elaborati dall'algoritmo tale numero viene definito dimensione dei dati in ingresso
- 4) **Dipendenza dai valori dei dati**. spesso la complessità computazionale dell'algoritmo non dipende solo dalla dimensione dei dati di ingresso ma anche dei valori assunti da tali dati. Per poter tenere conto dell'influenza dei valori assunti dai dati in ingresso,

solitamente si considerano due possibili scenari: un tempo di calcolo massimo e uno medio.

- 5) **Ha molto interesse la valutazione del tempo massimo**, ossia del caso peggiore che è anche il parametro più difficile da misurare

La seguente tabella riporta degli esempi di classi di complessità di un algoritmo:

Complessità	Terminologia
$O(1)$	Complessità costante
$O(\log n)$	Complessità logaritmica
$O(n)$	Complessità lineare
$O(n \log n)$	Complessità $n \log n$
$O(n^a)$	Complessità polinomiale
$O(a^n)$, con $a > 1$	Complessità esponenziale
$O(n!)$	Complessità fattoriale

Dove la notazione O-grande è utilizzata per descrivere il comportamento asintotico delle funzioni, in questo caso la funzione $f(n)$ indica il tempo di esecuzione.

1.4) Algoritmi notevoli: Ordinamento, ricerca e fusione

Ricerca

In generale un algoritmo di ricerca si pone come obiettivo quello di trovare, all'interno di un gruppo, un elemento avente determinate caratteristiche.

Principalmente esistono due tipi di algoritmi:

- Algoritmo di ricerca sequenziale
- Algoritmo di ricerca binaria.

Il metodo di **ricerca sequenziale** non richiede che i dati siano ordinati: consiste in una serie di confronti tra il valore da ricercare e tutti gli elementi del vettore. I confronti possono terminare nel momento in cui si trova l'elemento cercato, o possono anche continuare sino alla fine del vettore. Qui di seguito si riporta l'algoritmo di codifica sequenziale:

```
int ricercaSequenziale (int Vet[], int x, int n) {
    int i;
    for (i = 0; i < n; i++)
    {
        if (Vet[i] == x)
            return i;
    }
    return -1;
}
```

Il metodo della **ricerca binaria** o dicotomica richiede che gli elementi del vettore siano ordinati. il procedimento è il seguente: Innanzitutto bisogna identificare l'elemento che occupa la posizione centrale del vettore e confrontare questo elemento x con quello cercato.

Si possono verificare tre casi:

- l'elemento X è più piccolo dell' elemento centrale e quindi si scartano tutti gli elementi che occupano la metà destra del vettore.
- l'elemento X è più grande dell' elemento centrale quindi si scartano tutti elementi che occupano alla metà sinistra del vettore.
- l'elemento X è uguale l'elemento centrale, quindi X appartiene all'insieme e si trova in posizione centrale.

Per i casi 1 e 2 il discorso va ripetuto fino a quando si verifica il caso 3. Qui di seguito vediamo un esempio di ricerca binaria scritto in C++.

```
int Binary_Search(int[] array, int elemento)
{
    int start = 0, end = array.Length - 1, centro = 0;
    while (start <= end)
    {
        centro = (start + end) / 2;
        if (elemento < array[centro])
        {
            end = centro - 1;
        }
        else
        {
            if (elemento > array[centro])
                start = centro + 1;
            else
                return centro; // Caso: elemento==array[centro]
        }
    }
    return -1;
}
```

Ordinamento

L'ordinamento il processo che permette di ottenere, a partire da un insieme di dati omogenei, un insieme ordinato, secondo un ordine crescente o decrescente. Principalmente esistono 3 tipi di algoritmi di ordinamento:

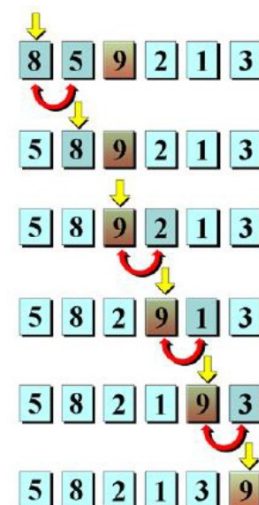
- Bubble Sort
- Selection Sort
- Insertion Sort

Bubble Sort

Il metodo di ordinamento Bubble Sort o "a bolle" consiste nel confronto degli elementi a due a due, e cioè primo e secondo, secondo e terzo, ..., penultimo e ultimo.

Questa ripetizione di scambi fra elementi adiacenti permette di far salire verso l'alto, proprio come Bolle di sapone, gli elementi più grandi.

Siccome questo algoritmo si basa in modo piuttosto marcato sugli scambi, non risulta molto efficiente quando si tratta di ordinare insieme con una grossa percentuale di elementi disordinati.



Riportiamo qui a fianco un esempio di algoritmo bubble sort e il relativo codice.

```
int main()
{
    int a[N], i, j, temp;

    cout<<"Inserisci gli elementi:\n";
    for(i=0; i<N; i++)
        cin>>a[i];

    //ordino gli elementi
    for(j=0; j<N-1; j++)
        for(i=0; i<N-1; i++)
            if(a[i]>a[i+1])
            {
                temp=a[i];
                a[i]=a[i+1];
                a[i+1]=temp;
            }
}
```

Selection Sort

L'algoritmo di selection sort è un algoritmo di ordinamento che consiste nel selezionare volta per volta un elemento, quello minore o quello maggiore, all'interno dell'array per poi posizionarlo nella sequenza ordinata.

Array iniziale:

20 – 13 – 40 – 7 – 1 – 37

Primo passaggio: trovo il minimo che è 1, quindi lo sposto in prima posizione:

1 – 13 – 40 – 7 – 20 – 37

1 – 7 – 40 – 13 – 20 – 37

1 – 7 – 13 – 40 – 20 – 37

1 – 7 – 13 – 20 – 40 – 37

1 – 7 – 13 – 20 – 37 – 40

L'Algoritmo di codifica invece è il seguente:

```
Int main ()
{
    int i, j, min, temp;
    int a[N];

    cout<<"Inserisci gli elementi:\n";
    for (i=0; i<N; i++)
        cin>>a[i];

    for (i=0; i<N-1; i++)
    {
        min=i;
        for (j=i+1; j<N; j++)
            if (a[j]<a[min])
                min= j;
    }
}
```

```
        temp=a[min];  
        a[min]=a[i];  
        a[i]=temp;  
    }
```

Insertion Sort

Per ottenere un vettore ordinato basta costruirlo ordinato, inserendo gli elementi al posto giusto fin dall'inizio. Idealmente il metodo costruisce un nuovo Array contenente gli stessi elementi del primo ma ordinato.

In pratica non è necessario costruire un secondo vettore in quanto le stesse operazioni possono essere svolte direttamente sul vettore originale

Facciamo subito un esempio per capire la logica di funzionamento.

Partiamo dunque da un array con questi elementi:

6 2 4 9 1 8

Innanzitutto confrontiamo 6 con 2

– dato che 6 è maggiore di 2 si sposta il 6 e si procede all’inserimento del 2.

Primo passaggio: 2 6 4 9 1 8

Adesso si confronta 6 con 4

– dato che 6 è maggiore di 4 si dovrebbero scambiare ma prima dobbiamo confrontare il 4 anche con il 2. Dato che 2 non è maggiore di 4 si può scambiare la posizione del 6 con quella del 4.

Secondo passaggio: 2 4 6 9 1 8

Dopo viene confrontato 6 con 9,

– dato che 6 non è maggiore di 9 non si fa nulla, si incrementa solo l’indice.

Terzo passaggio: 2 4 6 9 1 8

Poi si confronta 9 con 1

– dato che 9 è maggiore di 1 si continua a confrontare 1 con gli altri elementi. Quindi, in questo caso essendo tutti maggiori, si sposteranno verso destra lasciando libera la prima posizione dove poter inserire l’elemento.

Quarto passaggio: 1 2 4 6 9 8

Infine si confronta il 9 con il 8,

– dato che è superiore si procede a confrontare il numero 8 con il 4, ma essendo superiore scambiamo il 9 con il numero 8.

Quinto passaggio: 1 2 4 6 8 9

Riportiamo qui di seguito l'algoritmo di insertion sort:

```
for(i=1; i<N; i++)
{
    temp=vet[i];
    j=i-1;
    while(j>=0 && vet[j]>temp)
    {
        vet[j+1]=vet[j];
        j--;
    }
    vet[j+1]=temp;
}
```

Fusione

L'algoritmo di fusione utilizza un algoritmo di bilanciamento che scandisce due Array con due indici diversi e trasferisci nel terzo del array elemento minore. Il merge sort è un algoritmo di ordinamento basato su confronti che utilizza un processo di risoluzione ricorsivo, sfruttando la tecnica del **Divide et Impera**, che consiste nella suddivisione del problema in sotto-problemi della stessa natura di dimensione via via più piccola.

Concettualmente, l'algoritmo funziona nel seguente modo:

1. Se la sequenza da ordinare ha lunghezza 0 oppure 1, è già ordinata. Altrimenti:
2. La sequenza viene divisa (*divide*) in due metà (se la sequenza contiene un numero dispari di elementi, viene divisa in due sottosequenze di cui la prima ha un elemento in più della seconda)
3. Ognuna di queste sottosequenze viene ordinata, applicando [ricorsivamente](#) l'algoritmo (*impera*)
4. Le due sottosequenze ordinate vengono fuse (*combina*). Per fare questo, si estrae ripetutamente il minimo delle due sottosequenze e lo si pone nella sequenza in uscita, che risulterà ordinata

Supponendo di dover ordinare la sequenza [10 3 15 2 1 4 9 0], l'algoritmo procede ricorsivamente dividendola in metà successive, fino ad arrivare agli elementi

[10] [3] [15] [2] [1] [4] [9] [0]

A questo punto si fondono (merge) in maniera ordinata gli elementi, riunendoli in coppie:

[3 10] [2 15] [1 4] [0 9]

Al passo successivo, si fondono le coppie di array di due elementi:

[2 3 10 15] [0 1 4 9]

Infine, fondendo le due sequenze di quattro elementi, si ottiene la sequenza ordinata:

[0 1 2 3 4 9 10 15]

L'esecuzione ricorsiva all'interno del calcolatore non avviene nell'ordine descritto sopra. Tuttavia, si è formulato l'esempio in questo modo per renderlo più comprensibile.

1.4) Linguaggi formali: Sintassi e semantica

E' noto che nei linguaggi naturali, sia che siano verbali, scritti o orali:

- a ogni simbolo non corrisponde solo un significato
- un simbolo, o una parola, trasmette svariati significati a seconda del contesto

I calcolatori devono essere “guidati” per mezzo di istruzioni appartenenti ad un linguaggio specifico e limitato, a loro comprensibile. Ecco che quindi non ci possono essere ambiguità in tale linguaggio. Da questo bisogno, nasce una tipologia di linguaggio detto “**linguaggio formale**”.

I linguaggi formali sono quelli che l'uomo utilizza per la comunicazione simbolica, ossia quella particolare comunicazione che utilizza simboli astratti applicati situazioni concrete.

In questi linguaggi:

- a ogni simbolo corrisponde uno è un solo significato.
- un simbolo racchiude lo stesso significato in qualsiasi contesto.
- una volta definito il codice non può essere modificato dagli elementi che lo utilizzano.

Questi linguaggi, quindi, non presentano né sinonimi, né eccezioni, né ambiguità e sono controllati da regole prefissate. precisiamo che l'aggettivo formare deve essere inteso come sinonimo di “**rigorosamente definito**”.

Il rigore dei linguaggi formali è molto importante informatica: per il linguaggio di programmazione, Infatti, le regole linguistiche devono essere assolutamente rigide, in quanto un computer non ha alcuna capacità di adattarsi a frasi che comunicano seguendo modalità non previste.

Un linguaggio di programmazione e quindi un linguaggio formale, dotato di un lessico, una sintassi e una semantica ben definite, utilizzabile per il controllo del comportamento di una macchina formale

La **sintassi** è un insieme di regole che definisce la struttura di una frase formalmente corretta e ci aiuta a stabilire se una frase appartiene o meno al linguaggio.

La **semantica**, invece, definisce il significato delle parole delle frasi. Una delle principali caratteristiche dei linguaggi naturali è la presenza di sinonimi ambiguità E talvolta anche di eccezioni.

Consideriamo ad esempio la seguente frase:

“La matita mangia il lasagna al forno”

A livello di sintassi, la frase risulta errata, in quanto l'articolo “il” è errato. Perciò dovremmo scrivere “La matita mangia la lasagna al forno”. Tuttavia, dopo la correzione, la frase, pur diventata sintatticamente corretta, rimane semanticamente errata, ossia priva di significato.

1.5) Intelligenza artificiale: problem solving, ragionamento, rappresentazione della conoscenza, apprendimento automatico.

L'**intelligenza artificiale** è una disciplina appartenente all'informatica che studia i fondamenti teorici, le metodologie e le tecniche che consentono la progettazione di sistemi hardware e sistemi di programmi software capaci di fornire all'elaboratore elettronico prestazioni che, a un osservatore comune, sembrerebbero essere di pertinenza esclusiva dell'intelligenza umana.

Una primaria distinzione in seno alla ricerca nel campo dell'intelligenza artificiale è quella di intelligenza artificiale debole e intelligenza artificiale forte a seconda che vengano riprodotte solo alcune o tutte le funzionalità della mente umana.

Problem solving e ragionamento

Inizialmente i ricercatori si concentrarono sullo sviluppo di algoritmi che imitassero fedelmente i ragionamenti impiegati dagli esseri umani per risolvere giochi o realizzare deduzioni logiche in modo da poterli integrare all'interno dei sistemi intelligenti. Tali algoritmi solitamente si basano su una rappresentazione simbolica dello stato del mondo e cercano sequenze di azioni che raggiungano uno stato desiderato. Evoluzioni di questi algoritmi vennero realizzati tenendo in considerazione aspetti più complessi come l'incertezza o l'incompletezza delle informazioni, includendo concetti provenienti dalla probabilità, dalla statistica e dall'economia.

Per difficoltà legate alla complessità intrinseca dei problemi in esame, gli algoritmi per la loro risoluzione possono a volte richiedere enormi risorse computazionali. L'ottimizzazione degli algoritmi ricopre una priorità assoluta all'interno della ricerca in questo ambito.

Rappresentazione della conoscenza

La rappresentazione della conoscenza è una branca dell'intelligenza artificiale che studia il modo in cui avviene il ragionamento umano, e si preoccupa di definire dei simbolismi o dei linguaggi che permettano di formalizzare la conoscenza al fine di renderla comprensibile alle macchine, per potervi fare dei ragionamenti automatici (inferendo le informazioni presenti) ed estrarre così nuova conoscenza.

Quindi, un punto chiave della rappresentazione della conoscenza, è la definizione di linguaggi che siano sufficientemente espressivi da permettere di descrivere il dominio di interesse, ma non troppo ricchi di espressività, in quanto richiederebbero troppe risorse e/o troppo tempo per applicarvi i meccanismi inferenziali.

In linea generale, i linguaggi di rappresentazione della conoscenza forniscono sia una serie di costrutti per definire la sintassi del dominio di interesse (le regole sulle quali costruire delle asserzioni accettabili), sia una serie di operatori (quantificatori, operatori modali, etc.) che permettano di dare un significato, un valore di verità alle asserzioni rispetto al modello di riferimento.

Attraverso il linguaggio scelto si andranno ad effettuare una serie di asserzioni sul mondo, che andranno insieme a costituire una base di conoscenza (KB, Knowledge Base). È inoltre importante che il linguaggio scelto per fare le asserzioni sia anche in grado di operare sulla KB per estrarre nuova conoscenza e per aggiungerne di nuova.

Esistono principalmente due metodologie per rappresentare la conoscenza:

- **I linguaggi formali**
- **Gli alberi di decisione.**

NB: un albero di decisione è un grafo di decisioni e delle loro possibili conseguenze, (incluso i relativi costi, risorse e rischi) utilizzato per creare un 'piano di azioni' (plan) mirato ad uno scopo (goal). Un albero di decisione è costruito al fine di supportare l'azione decisionale (decision making)

Apprendimento automatico

L'apprendimento automatico è la disciplina che studia algoritmi capaci di migliorare automaticamente le proprie performance attraverso l'esperienza. È stato un ambito di ricerca cruciale all'interno dell'intelligenza artificiale sin dalla sua nascita.

L'apprendimento automatico è particolarmente importante per lo sviluppo di sistemi intelligenti principalmente per tre motivi:

- Gli sviluppatori di un sistema intelligente difficilmente possono prevedere tutte le possibili situazioni in cui il sistema stesso si può trovare a operare, eccetto per contesti estremamente semplici.
- Gli sviluppatori di un sistema intelligente difficilmente possono prevedere tutti i possibili cambiamenti dell'ambiente nel tempo.
- Un'ampia categoria di problemi può essere risolta più efficacemente ricorrendo a soluzioni che coinvolgono l'apprendimento automatico. Questa categoria di problemi include, ad esempio, il gioco degli scacchi e il riconoscimento degli oggetti.

Capitolo 2

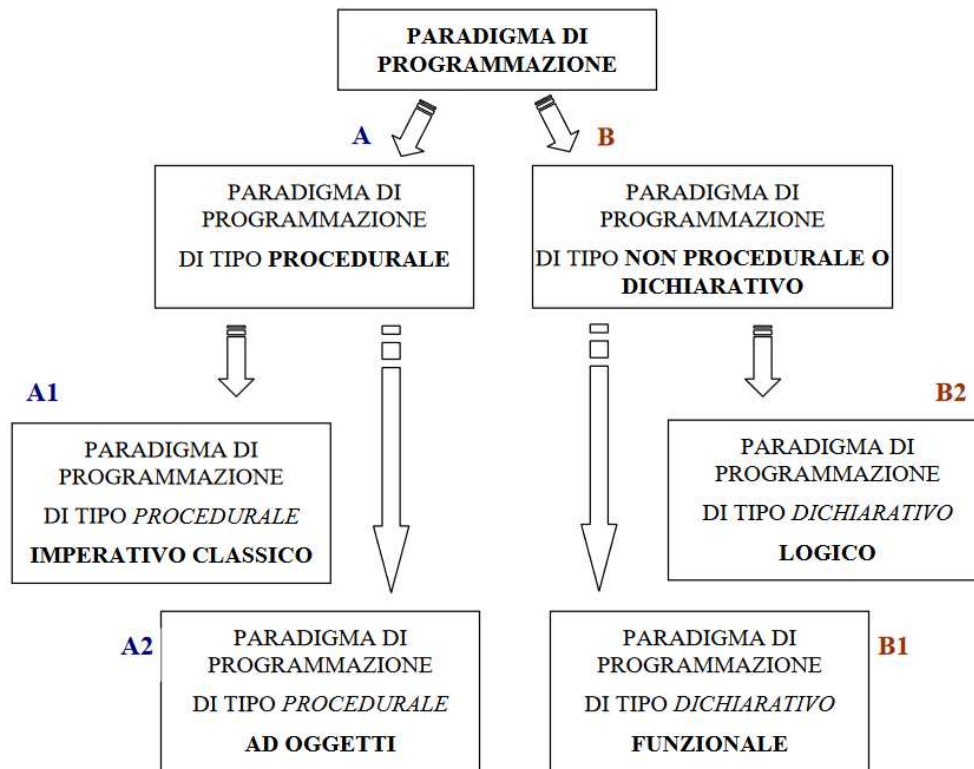
Programmazione e linguaggi

2.1) Linguaggi e tecniche di programmazione secondo i diversi paradigmi

Un computer consente di usare uno più linguaggi di programmazione, ossia quei linguaggi che vengono utilizzati dall'utente per scrivere i programmi che risolvono i problemi specifici alle sue applicazioni. I linguaggi di programmazione in base alle loro caratteristiche fondamentali, possono essere così classificati in:

- **visuali**
- **linguaggi imperativi**
- **funzionali**
- **dichiarativi o logici**
- **orientati agli oggetti**

Schematizzando, i paradigmi di programmazione sono così suddivisibili:



Programmazione Visuale

La programmazione a blocchi, o programmazione visuale, è il modo più semplice e immediato per avvicinarsi al mondo del coding. È il primo passo per imparare a programmare partendo da zero. Il **coding** consente di apprendere le basi della programmazione informatica, aiuta a sviluppare la logica, stimola la creatività ed educa al pensiero computazionale, a ragionare su problemi concreti e sul modo migliore per risolverli.

La programmazione a blocchi è un metodo di programmazione visuale, non devi conoscere un codice di programmazione. Ti basta manipolare degli oggetti, spostare degli elementi, sullo schermo del tuo pc o dei tuoi tablet e il gioco è fatto.

Consente di creare giochi, animazioni, storie interattive, sequenze musicali ma può anche essere usata per **programmare un robot**, per fargli compiere determinati movimenti e comportamenti, per aggiungere delle nuove abilità a un **androide**.

Con la **programmazione a blocchi** puoi fare tutto questo senza scrivere una sola riga di codice informatico. Basta spostare e ordinare in sequenza una serie di blocchi o oggetti grafici su un monitor, come se fossero i pezzi di un puzzle. A ogni mattoncino corrisponde un comando, un'istruzione che non ha bisogno di essere digitata ma solo "incastrata" al blocco precedente.

Scratch, ad esempio è un ambiente di programmazione a blocchi per il coding e la robotica educativa che consente di realizzare giochi e storie interattive, programmare robot in modo intuitive che viene appreso velocemente dai ragazzi fin dalla scuola primaria. Assemblando i

blocchetti sullo schermo lo studente può far muovere un personaggio virtuale a suo piacimento, può farlo cantare, ballare, personalizzarne l'aspetto oppure può creare immagini che ruotano e si animano al ritmo di musica.

Programmazione Imperativa

In informatica, la **programmazione imperativa** è un paradigma di programmazione secondo cui un programma viene inteso come un insieme di istruzioni (dette anche direttive o comandi), ciascuna delle quali può essere pensata come un "ordine" che viene impartito alla macchina virtuale del linguaggio di programmazione utilizzato. Da un punto di vista sintattico, i costrutti di un linguaggio imperativo sono spesso identificati da verbi all'imperativo, per esempio:

- read i
- print i
- goto 1

I linguaggi di programmazione imperativi sono molto concreti e lavorano vicino al sistema. Da un lato, quindi, il codice è di facile comprensione, dall'altro sono necessarie **molte righe di codice sorgente** per descrivere quello che, nei linguaggi di programmazione dichiarativi, si può ottenere con una frazione di questi comandi.

I linguaggi di programmazione imperativi più conosciuti:

- Fortran
- Java
- Pascal
- ALGOL
- C
- C#
- C++
- Assembler
- BASIC
- COBOL
- Python
- Ruby

I linguaggi di programmazione imperativi differiscono dai linguaggi dichiarativi per un aspetto fondamentale. Riassumendo in una frase: la programmazione imperativa si concentra sul **COME**, la programmazione dichiarativa sul **COSA**.

Cosa vuol dire questo? I linguaggi di programmazione imperativi sono scritti come una guida passo passo (come) per il computer. Descrivono esplicitamente quali passaggi devono essere

eseguiti e in quale ordine per raggiungere la soluzione desiderata. Nella programmazione dichiarativa, invece, viene descritto direttamente il risultato finale desiderato (cosa). Facendo un parallelo esplicativo con la cucina, i linguaggi imperativi danno ricette, i linguaggi dichiarativi danno foto del piatto finito.

Di seguito invece si riporta una tabella che illustra i vantaggi e gli svantaggi della programmazione imperativa

Vantaggi	Svantaggi
Facile leggibilità	Il codice diventa rapidamente molto esteso e quindi confuso
Relativamente facile da imparare	Il rischio di errori durante l'elaborazione è maggiore
Modello di pensiero di facile comprensione per i principianti (strada verso la soluzione)	La manutenzione blocca lo sviluppo dell'applicazione poiché la programmazione avviene in prossimità del sistema
Possono essere prese in considerazione le caratteristiche di applicazioni speciali	L'ottimizzazione e l'espansione sono più difficili

Programmazione orientata agli oggetti e relativo linguaggio di modellazione UML

Si considerano linguaggi orientati agli oggetti quei linguaggi di programmazione che hanno come caratteristica principale la definizione di oggetti e strutture dati.

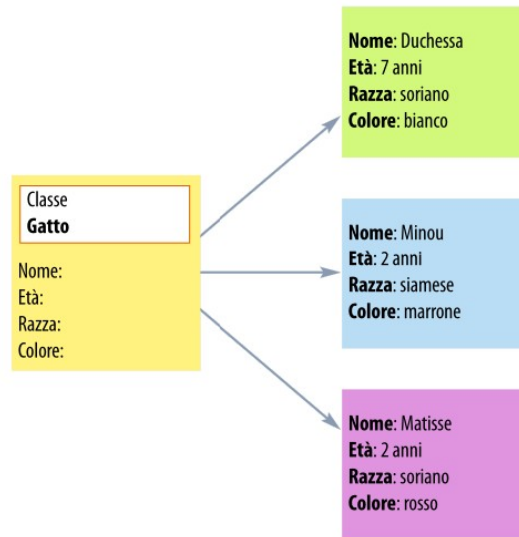
Sono linguaggi basati sul concetto di “oggetto” che è una metafora o rappresentazione del mondo reale. Ad ogni oggetto vengono associate informazioni e funzionalità e gli oggetti interagiscono gli uni con gli altri.

Un oggetto appartiene ad una classe di cui eredita le operazioni ed i predicati, in questo caso l'oggetto viene detto **istanza della classe**.

Tra classe e oggetto vi è la stessa relazione che sussiste fra un tipo e una variabile.

Sottolineiamo inoltre anche due altri aspetti fondamentali di questa relazione:

- le classi definiscono dei prototipi per gli oggetti: sono una specie di "stampino" che può essere utilizzato per realizzare tanti oggetti dello stesso tipo
- all'interno delle classi si incapsulano i dati e le funzioni, dove le funzioni sono strettamente correlate con i dati che manipolano.



Ogni oggetto, anche se appartiene alla stessa classe, è generalmente differente dagli altri, cioè ha proprietà (attributi) che lo caratterizzano, mentre condivide con gli altri suoi simili il comportamento. Formuliamo adesso una prima definizione di oggetto:

Un oggetto è un'entità astratta dotata di specifici attributi e in grado di cooperare con altri oggetti svolgendo specifiche azioni.

Un oggetto è quindi la rappresentazione di una qualsiasi cosa che ha uno stato e un comportamento.

Esempio:

1 Oggetto Gatto

L'oggetto **Gatto**:

- ha caratteristiche specifiche: peso, colore del pelo, età
- compie azioni, come: correre, dormire e bere il latte

2 Oggetto Automobile

L'oggetto **Automobile**:

- ha caratteristiche specifiche: marca, modello, peso, colore, posti ecc.
- esegue azioni, come: avviamento, arresto, accelerazione, cambio marcia, frenata, suono del clacson, avvio tergicristalli ecc.

Le classi

La **classe** costituisce la base della **OOP**: essa descrive la **natura** di un **oggetto** e ne definisce il suo **comportamento**.

Una **classe** è un “modello” per un insieme di oggetti “analoghi”, caratterizzati:

- dalla stessa **rappresentazione interna**;
- dalle stesse **operazioni** con lo stesso **funzionamento**, che permettono di descrivere un insieme di oggetti che hanno gli stessi **attributi (variabili)** ed eseguono le stesse **operazioni (funzioni o metodi)**.



TDA

In un linguaggio di programmazione tradizionale il concetto di **TD, tipo di dato**, è quello di insieme dei valori che può assumere un dato (una variabile): il **tipo di dato astratto (TDA** oppure **ADT, Abstract Data Type**) estende questa definizione includendo anche l'insieme di tutte e sole le operazioni possibili su dati di quel tipo.

La **classe** si può pertanto definire come lo **schema di base** dal quale è possibile creare i **singoli oggetti**.

La **descrizione dei metodi** e delle **variabili** è contenuta **all'interno della classe** e non negli oggetti, che sono singole istanze della classe stessa.

Tutti gli **oggetti istanza** di una stessa classe:

- condividono la **stessa struttura interna** e lo stesso **funzionamento**;
- hanno ciascuno la **propria identità** e il proprio **stato**.

Nella progettazione di una **classe** è necessario:

- definire (e nascondere) la rappresentazione dei dati all'utente;
- definire tutte le operazioni applicabili agli **oggetti** come elementi di un tipo di dato astratto;
- garantire che l'utente possa manipolare gli **oggetti** solo tramite operazioni del **tipo di dato astratto (TDA)** a cui gli **oggetti** appartengono (protezione).

All'interno della **classe** è possibile specificare il tipo di **visibilità** di dati (e, come vedremo, dei **metodi**) e codice tramite i seguenti **modificatori di accesso**:

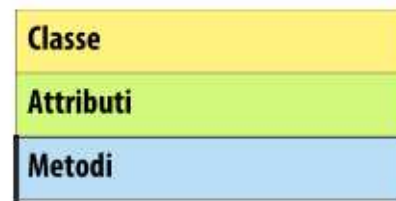
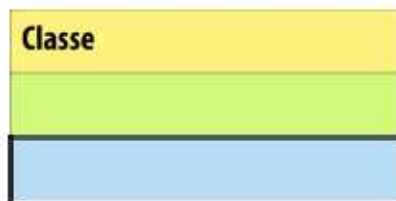
- **public (pubblica)**: permette l'utilizzo di variabili e metodi dall'esterno;
- **private (privata)**: nasconde metodi e codice che risulteranno non utilizzabili;
- **protected (protetta)**: dati e codice nascosti a esterni ma visibili da chi eredita.

Rappresentazione in UML

Nell'ambito dell'ingegneria del software è stato definito un apposito linguaggio per la descrizione grafica dei progetti, il linguaggio UML, nel quale viene utilizzato un simbolismo universalmente riconosciuto da tutti gli sviluppatori a oggetti, per rappresentare le classi e gli oggetti.

Per le classi sono previsti due livelli di rappresentazione:

- **sintetica**, che utilizza un rettangolo con tre sezioni dove viene indicato solo il nome della classe;
- **completa**, che utilizza un rettangolo con indicati anche gli attributi e i metodi.



Anche per gli **oggetti** la notazione è simile, ma il rettangolo ha solo due sezioni:

- la **rappresentazione sintetica** utilizza un rettangolo con solo il nome;
- la **rappresentazione completa** utilizza un rettangolo con indicati anche i valori degli **attributi**.

Oggetto: classe

Oggetto: classe
Attributi: valore

RAPPRESENTAZIONE SINTETICA



Classe



Oggetto



Oggetto

Gatto

Rappresentazione della classe

Pippo: Gatto

Malachia: Gatto

Rappresentazione degli oggetti

RAPPRESENTAZIONE COMPLETA



Classe



Oggetto



Oggetto

Gatto
pelo razza colore nome età cosaFa
mangia dorme corre faFusa gioca

Pippo: Gatto
pelo : "corto" razza : "soriano" colore : "rosso" nome : "Pippo" età : "10-02-17" cosaFa : "ci guarda"

Malachia: Gatto
pelo : "lungo" razza : "di strada" colore : "bianconero" nome : "Malachia" età : "12-03-14" cosaFa : "sonnecchia"

La rappresentazione **UML** della visibilità viene effettuata semplicemente scrivendo i caratteri + e - davanti ai metodi e agli attributi:

- sta per **private**;
- + sta per **public**.

Gatto	
-nome	: String
...	
-cosaFa	: String
+daiNome (String)	: bool
+leggiNome ()	: String
+leggiStato ()	: String
+mangia ()	: boolean
+dorme ()	: boolean
+corre ()	: boolean
+faFusa ()	: boolean
+gioca ()	: boolean
...	

Completiamo quindi la **classe Gatto** aggiungendo la visibilità agli attributi. Per semplicità, ci limitiamo a considerare due attributi, **nome** e **cosaFa**: aggiungiamo quindi due metodi necessari per leggere e scrivere in essi.

Avendo infatti reso **privati** gli **attributi**, l'unico modo per poterli leggere e scrivere in un'applicazione è quello di utilizzare **metodi pubblici**.

AREA DIGITALE



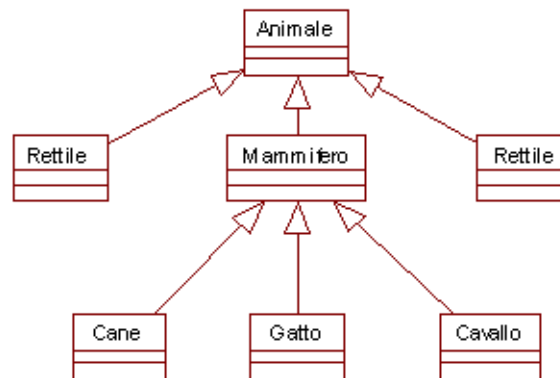
Naming
delle classi

UML: Ereditarietà

Se si conosce qualcosa riguardo ad una categoria di cose, automaticamente si conosce qualcosa che è possibile trasferire ad altre categorie che, in qualche modo, discendono dalla categoria che conosciamo. Chiariamo meglio questo concetto.

Ad esempio, se si conosce qualcosa su un animale generico (ad esempio si sa che un animale mangia, dorme, é nato, si sposta da un luogo ad un altro, ecc.), potremo dire che tutte le sottocategorie di animali (rettili, anfibi, mammiferi, ecc.) ereditano le stesse caratteristiche. Tale meccanismo, in Analisi Object Oriented, viene definito come:

Ereditarietà



Una classe figlia (o sottoclasse) può ereditare gli attributi e le operazioni da un'altra classe (che viene definita classe padre o super classe) che sarà sempre più generica della classe figlia. In UML l'ereditarietà viene rappresentata con una linea che connette la classe padre alla classe discendente e dalla parte della classe padre si inserisce un triangolo (una freccia)

Riassumendo si può dire che una classe può ereditare attributi ed operazioni da un'altra classe. La classe che eredita è figlia della classe (padre) da cui prende gli attributi e le operazioni. Le classi astratte sono utilizzate soltanto come classi base per l'ereditarietà e non forniscono alcun oggetto implementabile.

Programmazione funzionale

I linguaggi funzionali pongono l'accento sulle funzioni. Non significa che non ci siano i dati: però le funzioni sono in qualche modo più importanti.

Un programma scritto in un linguaggio funzionale è costituito da un insieme di definizioni di funzioni. Come nell'usare calcolo matematico, per ottenere il valore di una funzione è necessario specificare il valore dei suoi argomenti, e la coppia *< funzione, lista di argomenti >* è un'espressione che prende il nome di **applicazione**.

La valutazione di un'applicazione comporta, quindi, la valutazione delle espressioni che fungono da argomenti.

Il calcolo del valore di una funzione può anche richiedere la valutazione di un'applicazione della funzione stessa: in tal caso sia una definizione ricorsiva della funzione.

Caratteristiche dei linguaggi funzionali:

La programmazione funzionale offre un alto grado di astrazione perché si basa sul concetto matematico e sul principio di funzionamento. Se usata adeguatamente, questo tipo di programmazione permette di ottenere un **codice molto preciso**. Grazie al maggior numero possibile di unità piccole, ripetutamente utilizzabili e molto specializzate, per l'appunto le funzioni, viene fornito un programma per la risoluzione di un compito sostanzialmente più complesso.

Ci sono quindi numerose ragioni pratiche per cui la programmazione funzionale e i relativi linguaggi di programmazione occupano ancora oggi una posizione speciale nell'ambito dell'informatica, specialmente quando si tratta di compiti matematici complessi e **algoritmi**. Le aree di applicazione molto speciali fanno sì che i linguaggi di programmazione funzionale siano piuttosto di nicchia.

Il **LISP** è il linguaggio funzionale più diffuso, vanta numerosi versioni o dialetti.

La tabella sottostante ci elenca quali sono i vantaggi e gli svantaggi di una programmazione funzionale.

Vantaggi	Svantaggi
Programmi stateless	I dati (es. variabili) non possono essere modificati
Adatto per la parallelizzazione	Recupero di grandi quantità di dati non efficiente
Codice facilmente testabile	Non consigliato per le connessioni a database e server
Codice facilmente verificabile, possono essere verificate anche le funzioni stateless	Non adatto a molte ricorsioni dello stesso stack
Combinabile con la programmazione imperativa, orientata agli oggetti	La programmazione ricorsiva può portare a gravi errori
Codice più preciso e più breve	Non adatto a tutti i compiti

Linguaggi basati sulla logica

Un sistema di programmazione basato sulla logica è, in genere, costituito da un interprete incaricato di memorizzare fatti e regole asseriti dal programmatore, e dedurre mediante l'applicazione di un principio di deduzione, altre verità a partire da quelle specificate

Il programmatore giunge alla soluzione del problema esprimendo una serie di assiomi, formalizzati e comunicati al sistema, il complesso dei fatti che devono essere veri all'interno della realtà esaminata. Il risultato scaturisce dal conseguente processo di calcolo.

L'esecuzione di un programma si ottiene ponendo un quesito al sistema: la risposta consiste nella verifica di verità o di falsità dell'affermazione fatta dal quesito sulla base delle conoscenze specificate all'interno del sistema.

Esempi: linguaggio **PROLOG**, solo esempio di linguaggio di programmazione correntemente in uso.

La caratteristica fondamentale di un linguaggio di programmazione basato sulla logica sta nella richiedere, per la soluzione di un problema, la specifica di cosa deve essere calcolato, piuttosto che le modalità con cui il risultato deve essere ottenuto.

Per questa ragione, i linguaggi di programmazione basati sulla logica sono anche detti linguaggi dichiarativi. Linguaggi non basati sulla logica, quindi quelli imperativi e funzionali sono detti linguaggi procedurali.

2.1) Metodologia di costruzione dei programmi

Ora illustreremo i principi fondamentali per lo sviluppo di un software; essi si riferiscono sia al processo di produzione che al prodotto finale. Questi principi da soli non sono sufficienti a guidare lo sviluppo di un software, poiché ***essi descrivono proprietà desiderabili del processo e del prodotto in termini generali***. La loro applicazione avviene mediante l'utilizzo di metodi e tecniche: i **metodi** sono delle linee guida generali che definiscono un approccio sistematico, rigoroso e disciplinato, mentre per **tecniche** sono invece un approccio rigoroso e sistematico, sintomo di norme più meccaniche rispetto ai metodi. Solitamente metodi e tecniche sono uniti nel formare le **metodologie**, che si occupano di promuovere un approccio alla soluzione di un problema individuando metodi e tecniche relative. Infine a questi 3 elementi vengono forniti degli strumenti per la loro applicazione.

I principi dell'ingegneria del software sono:

- **rigore e formalità**
- **separation of concerns**
- **modularità**
- **astrazione**
- **anticipazione dei cambiamenti**

Rigore e formalità: sviluppare SW è un'attività creativa e come tale, porta all'imprecisione e all'essere inaccurati. Questa ispirazione creativa porta ad una scarsa strutturazione del progetto. Per evitare ciò, soltanto con un approccio rigoroso si possono realizzare prodotti affidabili, controllarne il costo ed essere fiduciosi del corretto funzionamento, avendo strutturato le attività con precisione. Il rigore può essere visto come uno strumento della creatività. Una definizione precisa di rigore non esiste, anche perché vi sono diversi livelli. Di questi, il più alto viene chiamato formalità; esso è un requisito più forte del rigore e consiste in aspetti matematici, che portano all'univocità ed alla non ambiguità. La formalità implica il rigore, ma non è vero il contrario. Si può essere rigorosi anche in situazioni informali

Separation of concerns: la separation of concerns consiste nel suddividere un problema in differenti aspetti, in modo che essi possono essere analizzati separatamente. Questo meccanismo può essere applicato diversi livelli:

- divisione della responsabilità all'interno del team
- separazione delle varie fasi di sviluppo nel tempo.
- Separazione nel considerare le qualità
- Separazione delle viste: un esempio può essere il progetto di una casa (si ha una vista del sistema idraulico, una di quello elettrico ecc); nel caso del software è utile ad esempio concentrarsi separatamente sul flusso dei dati e sul flusso di controllo
- Separazione dei problemi di realizzazione e progettazione da quelli di implementazione;

- Separazione in parti: dividere un problema in diversi problemi minori (**modularizzazione**)

Modularità

Quando siamo di fronte ad un sistema complesso possiamo dividerlo in parti più piccole chiamate moduli. In questo modo considereremo un sistema detto modulare. La modularità offre 4 tipi di benefici:

- La capacità di scomporre un sistema complesso in parti più semplici (approccio top-down)
- La capacità di comporre un sistema complesso partendo dai moduli esistenti (bottom-up)
- La capacità di capire un sistema valutando le sue parti
- La capacità di modificare un sistema modificando soltanto una parte delle componenti.

In pratica non facciamo altro che sfruttare il motto divide et impera, con il quale scomponiamo un problema con un approccio top-down in sottoproblemi, procedendo in modo iterativo finché il problema non è elementare.

Per poter comporre, scomporre, comprendere e modificare il SW in maniera modulare, i moduli devono avere alta coesione e basso accoppiamento. Un modulo ha un'alta coesione se tutti i suoi elementi (istruzioni, procedure, dichiarazioni) sono organizzati per un motivo logico, e non in maniera casuale, così da poter cooperare per raggiungere un obiettivo comune ossia realizzare la funzione richiesta per il modulo. Mentre la coesione è una proprietà interna al modulo, l'accoppiamento indica la relazione del modulo con altri moduli. È una sorta di misura di interdipendenza di due moduli. Si richiede un basso livello di accoppiamento tra i moduli, così da permettere il riutilizzo di ciascun modulo separatamente.

Astrazione

Effettuare un'astrazione significa identificare gli aspetti fondamentali di un fenomeno ignorandone quelli meno importanti, come i dettagli secondari. Esso è uno strumento molto importante per capire ed analizzare i problemi complessi.

Anticipazione dei cambiamenti

Il SW viene modificato in continuazione, sia per eliminare gli errori scoperti dopo il rilascio dell'applicazione sia per supportare l'evoluzione dell'applicazione, dovuta a nuovi requisiti che insorgono o modifica dei vecchi. Per questo la manutenibilità è una caratteristica fondamentale per la qualità del software. Devono essere i progettisti capaci di saper pianificare e anticipare con estrema cura i futuri cambiamenti, intervenendo nel progetto per risolvere questo vuol dire che l'anticipazione del cambiamento dovrebbe essere alla base della modularizzazione. Questo aspetto è fondamentale nei software, poiché essi vengono rilasciati quando i requisiti non sono del tutto stabiliti. Per questo grazie alle reazioni degli utenti, si potranno effettuare le modifiche.

2.1.1) Modularità: funzioni e procedure

La modularità è l'organizzazione in parti (per moduli) di un modello o di un sistema, in modo che esso risulti più semplice da comprendere e manipolare.

– Gran parte dei sistemi complessi sono modulari

Un modulo di un sistema software è un componente che:

- Realizza una astrazione
- È dotato di netta separazione tra:

- **Interfaccia**
- **Corpo**

L'interfaccia specifica il cosa (ovvero l'astrazione realizzata dal modulo)

Il corpo descrive come è realizzata l'astrazione

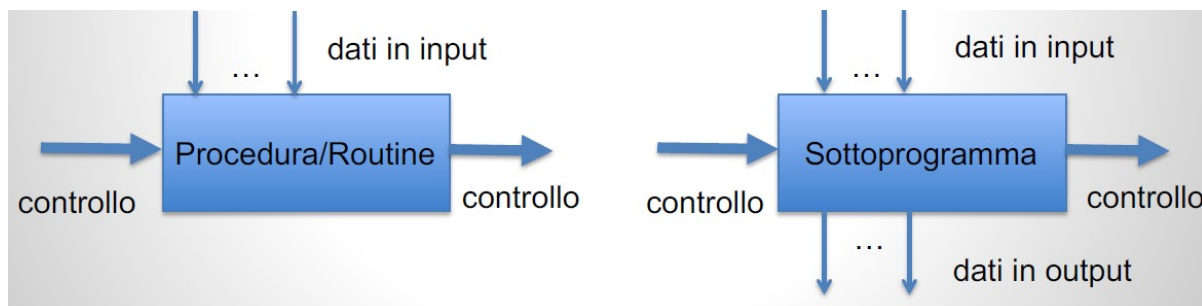


Per gestire la complessità dei programmi è utile individuare al loro interno dei moduli funzionali, detti sottoprogrammi:

- 1) Ai moduli è passato il flusso di controllo in esecuzione: essi hanno un solo punto in ingresso e di uscita (secondo i principi della programmazione strutturata).
- 2) Ai sottoprogrammi si assegnano determinate responsabilità, che includono la risoluzione di un sotto-problema:
 - I sottoprogrammi hanno un nome e possono essere attivati durante l'esecuzione del programma zero, uno o più volte.

Un sottoprogramma scambia informazioni con l'esterno, in ingresso e/o in uscita.

- Se il sottoprogramma non scambia informazioni in uscita, si chiama anche **procedura o routine**.
- Se il sottoprogramma fornisce informazioni in uscita, allora si chiama anche **funzione**.



La definizione di un sottoprogramma si compone di titolo e il corpo.

- Il titolo comprende:
 - il nome del sottoprogramma
 - i parametri di ingresso con il loro tipo

- i dati output con il suo tipo
- Il corpo è tipicamente un blocco sequenziale che comprende una sequenza di dichiarazioni e di istruzioni.

C++

```
returnType functionName(paramType param_1, ..., paramType param_n)
{
    <function body>
}
```

Si definiscono parametri formali i parametri dichiarati nell'interfaccia del sottoprogramma:

- essi sono definiti (gli si assegna un valore) all'atto dell'invocazione dal chiamante
- Quando il sottoprogramma viene invocato ai parametri formali si assegnano i valori dei parametri effettivi (ovvero quelli usati per attivare il sottoprogramma).
- **I parametri effettivi e quelli formali devono essere dello stesso tipo e devono essere specificati nello stesso ordine!**

2.2) Programmazione ad interfaccia grafica (ambienti RAD)

Quasi tutti gli ambienti di sviluppo grafico per programmazione di applicazioni desktop tradizionali sono ormai basati su una comune impostazione che negli anni si è rivelata pratica e capace effettivamente di aumentare la capacità di programmazione del programmatore. Un elemento fondamentale per migliorare lo sviluppo è stato quello di integrare le varie fasi dello stesso in un unico tool che le rendesse sincroniche e facili da richiamare, così come possibile passare velocemente dall'una all'altra. In questo senso si cominciò tanti anni fa a parlare di IDE di sviluppo, piuttosto che di semplici compilatori e tool di sviluppo. Una IDE era qualcosa che integrava in modo armonico i vari tool permettendone un uso più veloce ed ottimizzato, e di richiamare ed utilizzare le varie fasi tipiche dello sviluppo in modo assai più veloce.-

Il principe fra questi tool è senz'altro Visual Basic della stessa Microsoft, che già agli inizi degli anni '90 propone un procedimento di programmazione dell' interfaccia grafica molto intuitivo e veloce, dove altrimenti il programmatore avrebbe dovuto scrivere numerose linee di codice, da modificare e ritoccare magari spesso. Si parla comunemente da quel momento di programmazione "visuale", per indicare un tipo di procedimento che si avvale di tool appositi, di solito interni alla IDE di sviluppo, per disegnare (nel vero senso del termine) una interfaccia grafica che poi sarebbe stata collegata al codice sottostante di programma. Questo paradigma di sviluppo poi accettato universalmente da tutte le IDE (da Netbeans a Visual Studio a molte altre), ha potenziato e velocizzato molto la parte di disegno grafico dei programmi.

Lo sviluppo di interfacce grafiche, in informatica, prende il nome di **RAD**.

Il rapid application development (indicato anche con l'acronimo RAD, letteralmente sviluppo rapido delle applicazioni) è una metodologia di sviluppo del software introdotta inizialmente da James Martin negli anni ottanta. Questa metodologia coinvolge modelli di sviluppo iterativi e la costruzione di prototipi. Solitamente questo approccio allo sviluppo comporta compromessi tra usabilità, funzionalità e velocità d'esecuzione.

Vantaggi e svantaggi

Pro:

- Maggiore velocità di sviluppo attraverso metodi includenti la rapid prototyping, la virtualizzazione delle procedure correlate al sistema, l'utilizzo di CASE tool, e altre tecniche.
- Ridotte funzionalità per l'utente finale (derivante da una più mirata progettazione), da cui una ridotta complessità.
- Maggiore enfasi sulla semplicità e l'usabilità del design delle GUI (Graphic User Interface).

Contro:

- Ridotta Scalability (a capacità di un sistema di aumentare o diminuire di scala in funzione delle necessità e disponibilità), e ridotte funzionalità quando un'applicazione sviluppata tramite RAD inizia come prototipo ed evolve in una applicazione completa.
- Ridotte funzionalità si presentano a causa del time boxing (gestione del tempo) quando queste sono accelerate verso la nuova versione allo scopo di ultimare in tempi brevi la release del software.

2.3) Fondamenti di programmazione: la programmazione di microcontroller

Un microcontrollore (altrimenti detto microcontroller o MCU , MicroController Unit) è un single chip computer , ovvero un microcalcolatore integrato su un singolo chip. Come suggerisce il nome, il microcontrollore è utilizzato principalmente per realizzare sistemi di controllo digitale e, in particolare, nei dispositivi cosiddetti **embedded**. Si tratta di sistemi elettronici di elaborazione a

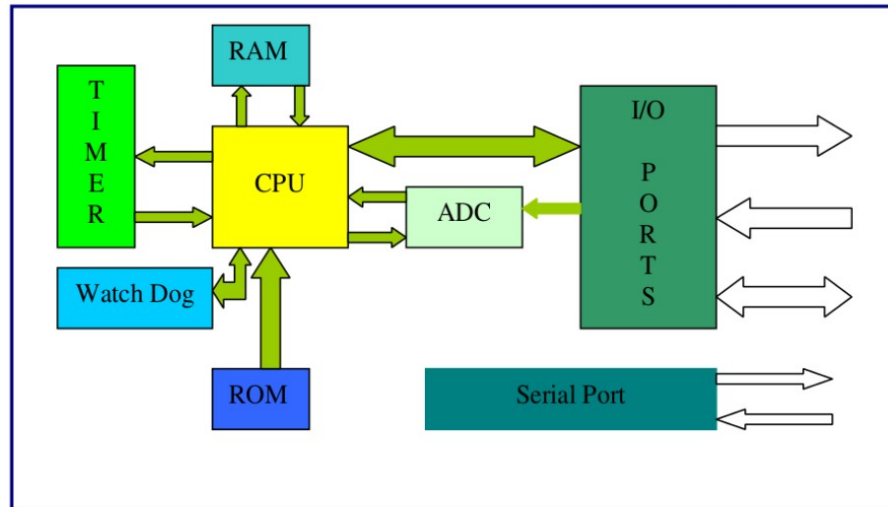
microprocessore progettati appositamente per una determinata applicazione (special purpose) ovvero non riprogrammabili dall'utente per altri scopi.

Alcuni esempi di sistemi embedded in cui vengono usati i microcontrollori sono

- controllo di hard disk e di stampanti;
- calcolatrici;
- telefoni cellulari;
- fotocamere;
- autoveicoli;
- sistemi di allarme;

- sistemi di controllo di macchinari industriali;
- elettrodomestici etc etc.

Il microcontrollore si differenzia rispetto al microprocessore in quanto al proprio interno contiene normalmente anche una certa quantità di memoria RAM e di EPROM e vari dispositivi periferici integrati, come timer, convertitori AD etc. Si tratta dunque di un vero e proprio computer completo di tutto ciò che occorre per il suo funzionamento. La figura seguente mostra uno schema della struttura interna di un MCU:



I microcontrollori hanno una potenza piuttosto limitata e sono utilizzati in applicazioni specifiche, spesso per eseguire sempre lo stesso identico compito. Per fare un semplice esempio, un microcontrollore utilizzato per controllare le funzioni di un forno a microonde potrebbe semplicemente controllare il livello di temperatura all'interno del forno e il tempo di accensione. Dunque i microcontrollori sono notevolmente più semplici ed economici di un normale PC.

Ma come si programma un microcontrollore?

Un microcontrollore non sa cosa fare da solo. Il tuo compito è dirgli cosa vuoi che faccia.

Quindi, devi:

- scrivere il codice del programma sul tuo computer
- compilare il codice con un compilatore per il microcontrollore che stai utilizzando (Ciò significa convertire il codice da codice leggibile dall'uomo in codice leggibile dalla macchina)
- caricare la versione compilata del tuo programma sul tuo microcontrollore

Arduino è un esempio di microcontrollore.

I microcontrollori sono tipicamente programmati in linguaggi di alto livello come C ++ o Java.

Uno degli strumenti essenziali necessari per programmare un microcontrollore è un ambiente di

sviluppo integrato (IDE). Questo software è solitamente sviluppato dai creatori del microcontrollore e contiene strumenti utili per aiutarti a programmare

2.4) Modelli di cicli di vita del software: tecniche di documentazione e manutenzione

Il processo di produzione del software è tutto quell'insieme di attività che vengono svolte per progettare, costruire, mantenere ed evolvere un prodotto software.

Un modello di ciclo di vita del software è una caratterizzazione descrittiva di come un sistema software viene o dovrebbe essere sviluppato.

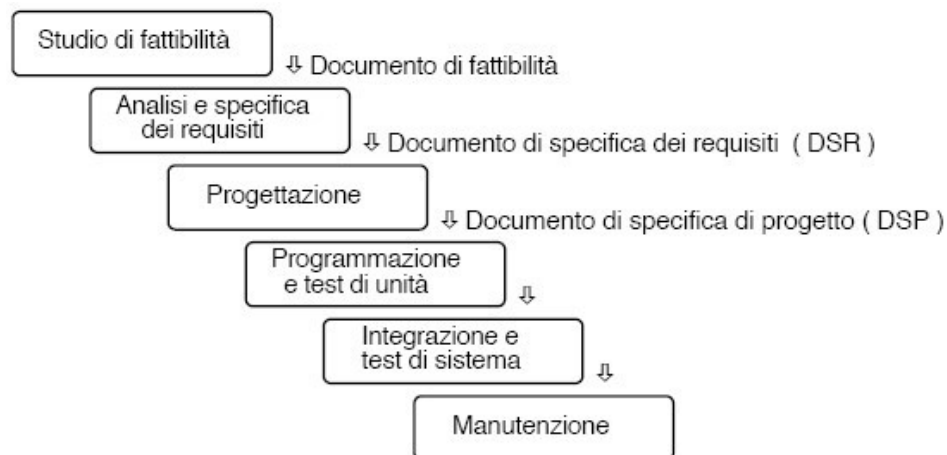
Il modello a cascata

Lo schema più tradizionale per rappresentare il ciclo di vita del software è il modello a cascata che si diffuse tra gli anni 50 e 70.

Il modello a cascata introduce due concetti fondamentali:

- lo sviluppo del software deve essere un processo disciplinato
- l'implementazione deve essere svolta successivamente ad una fase in cui si sono ben compresi gli obiettivi

Le attività tipiche del processo di produzione del software vengono svolte consecutivamente una dopo l'altra. Si ha quindi una progressione lineare tra di esse ma senza retroazioni: al termine di ogni attività sia come output un semilavorato che diverrà poi l'input dell'attività successiva.

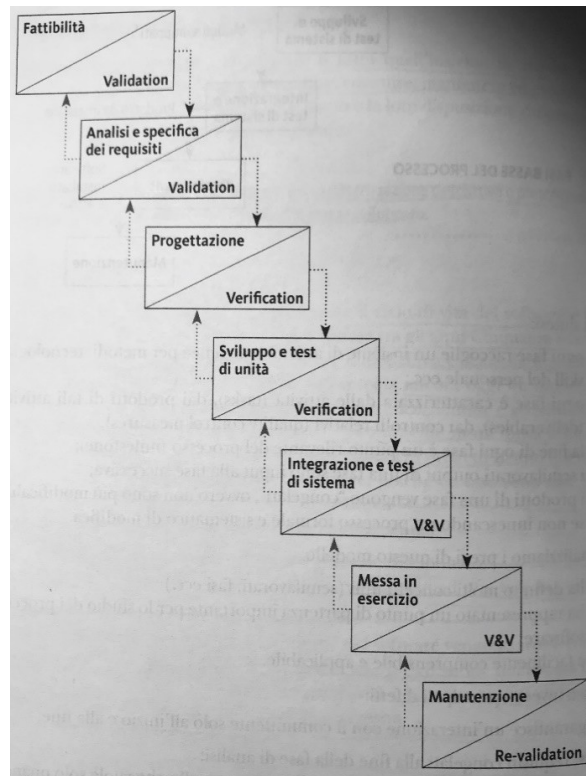


NB: i prodotti di una fase vengono “congelati”, ovvero non sono più modificabili
Principali difetti:

- requisiti congelati alla fine della fase di analisi
- requisiti utente spesso imprecisi: l'utente sa quello che vuole solo quando lo vede
- il nuovo sistema software diventa installabile solo quando è totalmente finito. Ne l'utente ne il Management possono giudicare il grado di adesione del sistema alle proprie aspettative

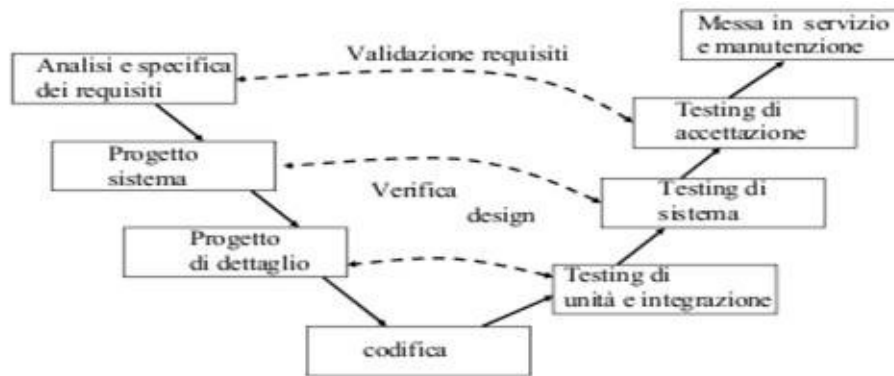
Modello V&V (Verification and Validation)

Questo modello è identico per strutturazione metodologia seguita al modello a cascata, ma in questa variante vengono applicati i ricicli, ovvero, al completamento di ogni fase viene fatta una verifica ed è possibile tornare alla fase precedente nel caso la stessa non esaurisca le aspettative.



Gli elementi chiave quindi sono quindi la verifica, ossia stabilire la verità della corrispondenza tra un prodotto software e la sua specifica, e la convalida, ossia stabilire l'appropriatezza di un prodotto software rispetto alla sua missione operativa

Il modello a V



Osserviamo lo schema precedente:

- le attività a sinistra sono collegate a quelle di destra attorno alla codifica
- se si trova un errore in una fase a destra si riesegue il pezzo della V collegato
- è esplicito che si può iterare migliorando requisiti progetto e codice

Come si evince dalla figura il modello a “V” deriva sempre dal modello a cascata ma presenta la caratteristica saliente di essere un metodo ben strutturato in cui ogni fase è implementabile dalla documentazione dettagliata della fase precedente.

2.5) Metodologie di sviluppo “Agile”

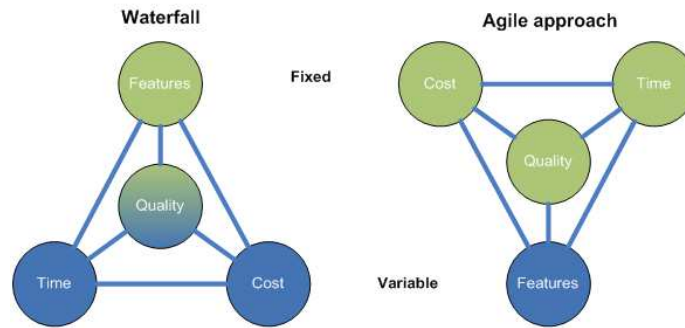
Nell'ingegneria del software, l'espressione metodologia agile (o sviluppo agile del software, in inglese agile software development, abbreviato in ASD) si riferisce a un insieme di metodi di sviluppo del software emersi a partire dai primi anni 2000 e fondati su un insieme di principi comuni, direttamente o indirettamente derivati dai principi del "Manifesto per lo sviluppo agile del software". I metodi agili si contrappongono al modello a cascata (waterfall model) e altri modelli di sviluppo tradizionali, proponendo un approccio meno strutturato e focalizzato sull'obiettivo di consegnare al cliente, in tempi brevi e frequentemente, software funzionante e di qualità.

Fra le pratiche promosse dai metodi agili ci sono la formazione di team di sviluppo piccoli, poli-funzionali e auto-organizzati, lo sviluppo iterativo e incrementale, la pianificazione adattiva, e il coinvolgimento diretto e continuo del cliente nel processo di sviluppo.

Obiettivo:

L'obiettivo è la piena soddisfazione del cliente e non solo l'adempimento di un contratto. Il corretto uso di queste metodologie, inoltre, può consentire di abbattere i costi e i tempi di sviluppo del software, aumentandone la qualità.

La metodologia Agile è fortemente focalizzata sul prodotto da realizzare piuttosto che sul processo per realizzarla e considera le variabili “Qualità”, “Tempo” e “Costi” non negoziabili spostando la flessibilità verso le Funzionalità del prodotto.



Essa è esplosa proprio in concomitanza con la crisi successiva al boom di Internet prendendo spunto dai metodi applicati in piccole software house.

Metodologie:

In senso lato il termine "agile" indica tutte quelle metodologie di sviluppo leggere e flessibili, che rompono con la precedente tradizione di ingegneria del software (modello a cascata, modello a spirale, etc.) basata su una raccolta delle specifiche e su una strutturazione sequenziale dello sviluppo software. Le metodologie agili consentono invece di rivedere di continuo le specifiche adeguandole durante l'avanzamento dello sviluppo del software, mediante un framework iterativo e incrementale, e un forte scambio di informazioni e di pareri tra gli sviluppatori e con il committente.

I 12 sono i principi chiave che guidano la **gestione del progetto** secondo la **metodologia Agile**:

1. La **soddisfazione del cliente** è sempre la massima priorità e viene raggiunta attraverso una consegna rapida e precisa;
2. L'evoluzione, sotto tutti gli aspetti connessi con il progetto, viene adottata in qualsiasi **fase del processo**;
3. Un prodotto o servizio viene consegnato con una frequenza più alta;
4. Gli stakeholder e gli sviluppatori collaborano strettamente ogni giorno;
5. Tutti gli stakeholder e i membri del team devono rimanere motivati per **ottenere risultati di progetto ottimali**. I team dispongono di tutti gli strumenti e il supporto necessari per poter **raggiungere gli obiettivi del progetto**;
6. Le riunioni faccia a faccia sono considerate la **forma di comunicazione più efficiente ed efficace per il successo del progetto**;
7. Un prodotto finale funzionante è la **misura finale del successo**;
8. Lo sviluppo sostenibile si realizza attraverso **processi agili** in cui team di sviluppo e stakeholder sono in grado di mantenere un ritmo costante e continuo;
9. L'agilità è migliorata attraverso una continua attenzione all'eccellenza tecnica e alla corretta progettazione;
10. La semplicità è un elemento essenziale, in ogni **fase del progetto**;
11. I team auto-organizzati hanno maggiori probabilità di sviluppare le migliori idee e progetti e di **soddisfare i requisiti** prefissati;

12. I team effettuano cambi di comportamento per **migliorare l'efficacia e l'efficienza del lavoro**.

La **metodologia Agile** è stata originariamente sviluppata per l'industria del software.

Il suo compito era di **ottimizzare e migliorare il processo di sviluppo** nel tentativo di identificare e correggere rapidamente problemi e difetti.

Questa metodologia permette di fornire un **prodotto migliore**, in modo più rapido, attraverso sessioni / sprint brevi e interattive.

Nell'era della trasformazione digitale, con molte aziende che migrano verso un luogo di lavoro digitale, la metodologia Agile si adatta perfettamente alle organizzazioni che cercano di trasformare il **modo in cui si gestiscono i progetti** e il modo in cui operano nel loro complesso.

In termini di **benefici per l'azienda**, il posto di lavoro digitale e l'Agile forniscono:

- Maggiore flessibilità;
- Maggiore produttività;
- Maggiore trasparenza;
- Prodotti di qualità superiore;
- Diminuzione del **rischio di mancati obiettivi**;
- Maggiore coinvolgimento e soddisfazione delle parti interessate.

Nel campo della **gestione del progetto**, la **metodologia Agile** fornisce ai team, agli sponsor, ai project manager ed ai clienti molti vantaggi specifici, tra cui:

- Implementazione più rapida di soluzioni;
- **Riduzione degli sprechi** grazie alla riduzione al minimo delle risorse;
- Maggiore flessibilità e adattabilità al cambiamento;
- **Maggiore successo** grazie a sforzi più mirati;
- Tempi di consegna più rapidi;
- Rilevamento più veloce di problemi e difetti;
- **Processi di sviluppo ottimizzati**;
- Una struttura più leggera;
- **Controllo ottimale del progetto**;
- Maggiore attenzione a specifiche esigenze del cliente;
- Maggiore frequenza di collaborazione e feedback.

Capitolo 3

Architettura dei sistemi di elaborazione

3.1) Codifica digitale delle informazioni

Perché persone o macchine possano utilizzare un'informazione hanno bisogno che essa sia appropriatamente “rappresentata”.

Se non fosse esistita la scrittura non avremmo un resoconto oggettivo degli avvenimenti dell'uomo dalla sua nascita fino ad oggi.

Scrivere, leggere ed elaborare informazioni implica che chi lo fa abbia preliminarmente concordato un codice, ossia una serie di regole e convenzioni da seguire.

Un'informazione, per essere correttamente elaborata, deve essere codificata in una rappresentazione comprensibile all'elaboratore stesso. Quindi si può dire che ***la codifica è l'insieme di convenzioni e di regole da adottare per trasformare un'informazione in una sua rappresentazione e viceversa.***

La stessa informazione può essere codificata in modi diversi (rappresentazioni diverse) a seconda del contesto: le rappresentazioni araba o romana dei numeri ne sono un esempio (i simboli “1” e “I” costituiscono due codifiche diverse della stessa informazione numerica).

I dati appartengono a diverse categorie e possono simboleggiare oggetti concreti o reali (per esempio un monitor) oppure oggetti astratti o concetti (per esempio il concetto di amore). Per gli oggetti concreti si può realizzare una rappresentazione grafica tale da consentire il riconoscimento a qualsiasi individuo, mentre per i concetti astratti si potrebbero avere delle ambiguità.

Il problema ora è quello di rappresentare questa informazione all'interno di un computer.

Rappresentazione digitale o binaria

All'interno dell'elaboratore ogni informazione è codificata usando la rappresentazione binaria o digitale, utilizzando cioè un alfabeto di due soli simboli: 0 e 1. Ma perché?

Scelta della rappresentazione:

- vincolata al tipo di operazione che devo fare su questa informazione
- Le ragioni della scelta di una rappresentazione binaria sono prevalentemente di carattere tecnologico, ossia dipende da come funzionano i dispositivi che costituiscono il computer
- questi dispositivi sono bistabili: assumono sempre uno stato fra due stati fisici
- questi stati possono essere naturalmente tradotti nei simboli numerici del sistema binario 1 e 0

L'entità minima di informazione codificabile attraverso la rappresentazione binaria è il bit

Bit: da **Binary Digit** o cifra binaria – può assumere valori: 0 o 1

Per poter rappresentare un numero maggiore di informazioni si usano sequenze di bit. Ad esempio utilizzando 2 bit si possono codificare quattro informazioni diverse:

00 01 10 11

Il processo che fa corrispondere a una informazione una configurazione di bit prende il nome di codifica binaria dell'informazione.

L'associazione informazione/configurazione di bit è convenzionale: l'importante è che tutti quelli che devono condividere l'informazione usino la stessa convenzione

- Con 1 bit codifico 2 informazioni: 2^1
- Con 2 bit codifico 4 informazioni: 2^2
- Con 3 bit codifico 8 informazioni: 2^3 ...
- Con N bit codifico 2^N informazioni

Byte: unità di misura della capacità di memorizzare informazione.

Si utilizzano i multipli dei byte

–KiloKB210~ un migliaio (1024)

–MegaMB220~ un milione (1KB*1024)

–GigaGB230~ un miliardo (1MB*1024)

–TeraTB240~ mille miliardi (1GB*1024)

Quanta memoria occupa un file: si misura in byte. La capacità di memorizzazione di un dispositivo hardware si misura in byte (capacità di RAM, hard-disk)

Codifiche numeriche

Le codifiche numeriche sono codici usati esclusivamente per la rappresentazione delle informazioni numeriche. Un esempio di codifica è il codice BCD (binary code decimal). Ogni numero decimale è rappresentato con un **nibble** (un gruppo di 4 bit), in quanto la cifra 9, poiché la più grande, ha bisogno di 4 bit per essere codificata. Con questa codifica, pertanto, ogni cifra decimale è rappresentata dal binario puro corrispondente:

Decimale	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Con 4 bit le configurazioni possibili sono 16 ma per rappresentare le cifre che vanno da 0 a 9 ne vengono usate solo 10; ne restano quindi 6 inutilizzate, il codice quindi è ridondante.

Avremo dunque: $146 = 0001 \mid 0100 \mid 0110$, $352 = 0011 \mid 0101 \mid 0001$. Vediamo come poter effettuare somme in BCD:

- Si convertono i due numeri da sommare;
- Si prova ad effettuare una semplice somma binaria;

Ci si accorgerà che ciò in alcuni casi funziona, in altri no.

ESEMPIO

Cominciamo con la seguente somma in decimale: $236 + 151 = 387$

$236 = 0010|0011|0100$

$151 = 0001|0101|0001$

Si incolonnano e sommano le rappresentazioni binarie. Il tutto funziona correttamente:

111			
0010	0011	0100	236
0001	0101	0001	151
<hr/>			
0011	1000	0101	
3	8	7	

Codici a controllo di errore

Una problematica che va affrontata quando si tratta di codificare informazione che deve essere trasmessa a distanza in modo affidabile, è la gestione degli errori. Per quanto la tecnologia usata sia affidabile, può sempre succedere che per disturbi, interferenze elettro-magnetiche o malfunzionamenti di qualsiasi genere, l'informazione venga alterata. In altri termini qualcosa che è partito come 0 è arrivato come 1 o viceversa.

Esistono delle tecniche delle quali qui daremo un breve cenno che permettono almeno di scoprire se ci sono stati errori ed in alcuni casi correggerli.

Esempio

Si vuole trasmettere un messaggio numerico codificato con il codice BCD. Si parte dall'ipotesi che la probabilità che avvenga un errore sia sufficientemente bassa da ritenere che al massimo ce ne possa essere uno ogni quattro bit. Supponiamo di trasmettere la cifra BCD 7 ed aggiungiamo un bit in coda alla sua codifica secondo la seguente regola:

facciamo in modo che il numero di 1 presenti tra i cinque bit che si trasmettono sia dispari.

Cioè $7 = 0111$ allora si aggiunge uno 0 in modo che nella cinquina 01110 si abbiano tre 1, cioè un numero dispari. In ricezione supponiamo sia giunto il messaggio 01010.

È cambiato a causa di un disturbo sulla linea elettrica il valore di un bit. Chi riceve (si tenga presente l'ipotesi di un solo possibile errore nei cinque bit), si accorge dell'errore perché adesso il numero di 1 presenti nella cinquina non è più dispari. Riflettete sul fatto che con un errore o un 1 diventa zero o uno zero diventa 1, in ogni caso il numero totale di 1 varia di un unità passando da pari a dispari e quindi non torna più il conteggio. Il ricevente chiederà la ritrasmissione del dato errato.

Questa tecnica si chiama tecnica del bit di **parità dispari**. Se avessimo fatto in modo che il numero di 1 debba essere pari avremmo parlato di **parità pari**. E' indifferente che si scelga la parità pari piuttosto che quella dispari, la scelta deve però essere fissata nel nostro sistema una volta per tutte ed ovviamente nota sia al trasmettitore sia al ricevitore.

Vediamo un esempio di parità pari. Trasmetto 3 in BCD a parità pari. La codifica del 3 in quel codice è 0110 dunque anche in questo caso devo aggiungere uno 0 e trasmettere 01100. I codici a parità pari o dispari si chiamano codici rivelatori di errore, in quanto permettono solo di scoprire (rivelare) la presenza di un errore. La presenza di bit di parità genera uno spreco di memoria e di velocità di trasmissione, per cui può valere la pena cercare di capire ogni quanti bit mi convenga inserirne uno di parità. Ad esempio se si scopre che è bassissima la probabilità di 1 errore su 8 bit, invece di inserire un bit per ogni cifra del codice, se ne inserisce 1 ogni coppia. Ciò non potrà che essere frutto di un'analisi statistica delle caratteristiche del mezzo trasmissivo.

I codici alfanumerici (codice ASCII)

Oltre ai numeri i calcolatori elaborano ovviamente anche i testi. Vediamo come. Per la codifica dei testi ha uso molto diffuso il cosiddetto codice **ASCII**. La sigla sta per *American Standard Code for Information Interchange*. Esso è definito da uno standard internazionale noto come standard ISO 646. L'**ISO** è l'*International Standard Organization*, un ente internazionale preposto alla definizione di norme omogenee in campo tecnico valide in tutto il mondo.

Tabella ASCII a 7 bit																									
NUL	0	NL	10	DC4	20	RS	30	(40	2	50	<	60	F	70	P	80	Z	90	d	100	n	110	x	120
SOH	1	VT	11	NAK	21	US	31)	41	3	51	=	61	G	71	Q	81	[91	e	101	o	111	y	121
STX	2	NP	12	SYN	22	SP	32	*	42	4	52	>	62	H	72	R	82	\	92	f	102	p	112	z	122
ETX	3	CR	13	ETB	23	!	33	+	43	5	53	?	63	I	73	S	83]	93	g	103	q	113	{	123
EOT	4	SO	14	CAN	24	"	34	,	44	6	54	@	64	J	74	T	84	^	94	h	104	r	114		124
ENQ	5	SI	15	EM	25	#	35	-	45	7	55	A	65	K	75	U	85	_	95	i	105	s	115	}	125
ACK	6	DLE	16	SUB	26	\$	36	.	46	8	56	B	66	L	76	V	86	`	96	j	106	t	116	~	126
BEL	7	DC1	17	ESC	27	%	37	/	47	9	57	C	67	M	77	W	87	a	97	k	107	u	117	DEL	127
BS	8	DC2	18	FS	28	&	38	0	48	:	58	D	68	N	78	X	88	b	98	l	108	v	118		
HT	9	DC3	19	GS	29	*	39	!	49	;	59	E	69	O	79	Y	89	c	99	m	109	w	119		

Una successiva standardizzazione ha definito versioni diverse per i cosiddetti codici **ASCII estesi ad 8 bit** (ISO 8859). In ognuno dei vari codici le prime 127 configurazioni coincidono con quelle del codice a sette bit, mentre le restanti variano a seconda dei vari gruppi di alfabeti disponibili nel mondo. Non tutte le lingue vi sono comprese, in particolare le lingue basate su ideogrammi (cinese, giapponese e coreano). Notiamo che i primi 32 caratteri ASCII non hanno corrispondente visuale a schermo o in stampa ma sono caratteri di controllo usati per le telecomunicazioni o per rappresentare alcuni particolari simboli. Ad esempio i caratteri di tabulazione (HT), l'invio (CR) il salto pagina (NP), lo spazio bianco (SP)

Nel 1993 è stato definito un nuovo standard che promette di sostituire nel tempo l'ASCII esteso. Si tratta dell'**UNICODE**. È una codifica a 16 bit, quindi con 2^{16} , cioè più di 64.000 possibilità. Le prime 256 combinazioni corrispondono all'ASCII esteso dell'Europa Occidentale e dell'America. Con il resto vengono codificati gli altri simboli necessari agli altri ASCII, simboli matematici, più di ventimila ideogrammi ed altro ancora.

3.2) Sistemi digitali e programmabili: i microprocessori, programmazione a livello macchina e con linguaggi orientati alla macchina.

I sistemi digitali e sono tutti quegli apparati al cui interno le grandezze fisiche impiegate come segnali sono vincolate ad assumere solo due valori discreti.

In particolare i **sistemi binari** sono quei sistemi in cui i segnali sono limitati a due valori di regime.

Un sistema digitale è un circuito costituito da componenti elementari e dai collegamenti che li intercorrono.

I microprocessori

Un microprocessore è un dispositivo complesso che integra funzionalità sia hardware che software:

- Dal punto di vista hardware è dotato di circuiti organizzati per le funzionalità interne e per l'interfacciamento con dispositivi esterni;
- Dal punto di vista software è dotato di capacità di calcolo di base logiche e aritmetiche, ma soprattutto può leggere e eseguire programmi.

Un programma è una sequenza di istruzioni che comandano al microprocessore l'esecuzione di operazioni elementari.

Il microprocessore è basilamente dotato di una "intelligenza", CPU, e una unità logica-aritmetica, ALU, per le operazioni. Si appoggia a unità esterne, come per esempio memorie di massa, tastiera, video, per scambiare le informazioni da elaborare.

Possiamo dire più precisamente che il microprocessore è l'elemento che determina le caratteristiche e le prestazioni del computer: tutta la capacità di calcolo e di elaborazione sono condensati nei pochi centimetri quadrati di un chip.

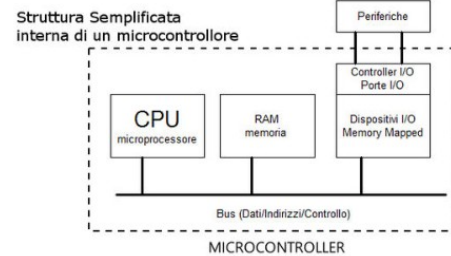
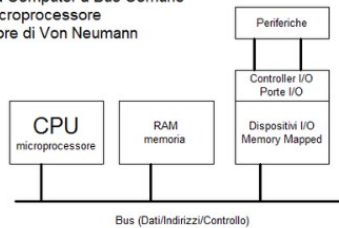
Il microprocessore risiede in un alloggiamento chiamato socket, dove vengono inseriti i piedini di input/output del processore stesso. Sopra di esso viene montata solitamente una ventolina di raffreddamento ed eventualmente un dissipatore di calore.

I microprocessori, indipendentemente dalle prestazioni, sono catalogati per famiglie: appartenere a una determinata famiglia vuol dire dividerne l'architettura, cioè la struttura logica interna in rapporto con il bus dei dati, come vedremo in seguito.

Distinguiamo il termine microprocessore da microcontrollore:

- Un **microprocessore** è un circuito integrato contenente esclusivamente la **CPU (central processing unit)** del sistema, ed ha bisogno di altri dispositivi esterni per funzionare, quali **RAM, ROM (o Flash Disk)** e, se si pensi ad un computer, anche una scheda grafica;
- Un **microcontrollore** al suo interno contiene non solo una CPU ma anche :una RAM, una ROM, dei timers, dei convertitori ADC e DAC etc.... Un microcontroller è equiparabile per funzionalità a un piccolo computer , questa integrazione dei componenti permette di contenere i costi a pochi euro per dispositivo semplificando la costruzione della scheda elettronica.

Struttura Semplice Computer a Bus Comune
Sistema digitale a microprocessore
Architettura Elaboratore di Von Neumann



I componenti principali del sistema a microprocessore, che è possibile trovare anche all'interno del microcontrollore, sono:

- **CPU:** La CPU o μP è il Microprocessore , un circuito elettronico in grado di eseguire le operazioni matematiche e logiche di un programma software caricato in RAM , inoltre tramite segnali elettrici digitali che viaggiano sul BUS sovrintende al funzionamento dei dispositivi elettronici collegati al BUS. Il dispositivo è molto veloce , infatti riesce ad eseguire anche miliardi di operazioni al secondo ; inoltre supporta caratteristiche quali multitasking, interrupt, timer etc. Ad esempio la CPU nello schema elettronico in figura è collegato a un multiplexer in cui alcuni piedini detti "selettori" sono gestiti dalla cpu e permettono di selezionare uno dei sensori collegato ai piedini d'ingresso del multiplexer .
- **MEMORIA FLASH:** La memoria flash corrisponde a quell'unità di memoria che nei vecchi computer si chiamava ROM. Sostanzialmente la flash è in grado di conservare le informazioni anche in assenza di alimentazione (come le istruzioni di avvio) con la possibilità di sovrascrivere i dati. La memoria FLASH contiene anche il software del sistema, che permette al microprocessore di svolgere le operazioni nel modo corretto.
- **RAM:** La RAM (Random Access Memory) è un altro tipo di memoria ad accesso casuale più veloce della memoria flash. È in grado di memorizzare i dati solo nel momento in cui si sta lavorando su di essi (e solo nel caso il sistema sia alimentato), in assenza di alimentazione elettrica i dati vengono eliminati (memoria volatile). La RAM opera più velocemente rispetto alla memoria FLASH, poiché è una memoria ad accesso casuale: le informazioni non devono essere ricercate in sezioni, in reparti precisi della memoria.
- **START:** Lo start è un elemento fondamentale del sistema e decide in quale istante temporale iniziare a fare la misura del segnale.
- **PORTE I/O:** Sono le porte Input e Output. Ad esse vengono collegati i dispositivi di Input (ad esempio sensori, o una banale tastiera) e di output. Essi permettono rispettivamente

la ricezione di dati e la loro trasmissione. Le porte I/O sono dette "*memory mapped*" quando vengono gestiti come fossero celle di memoria RAM o di scrittura.

- **IL BUS:** Non meno importante degli altri componenti elencati sopra, il BUS è un canale di comunicazione che trasferisce dati e segnali tra le componenti del sistema a μP o del μC . Un solo bus può connettere simultaneamente più dispositivi. A sua volta un Bus è costituito da molteplici linee:
 - o **BUS DATI:** è responsabile del trasferimento di dati;
 - o **BUS INDIRIZZI:** comunica la posizione dei dati trasmessi o scritti nella RAM o nelle periferiche I/O.

Programmazione a livello macchina

Un linguaggio di programmazione a basso livello in informatica, indica un linguaggio di programmazione che coincide con il linguaggio macchina o che differisce poco dal linguaggio macchina, fornendo poca o nessuna astrazione dai dettagli del funzionamento fisico del calcolatore.

L'espressione si contrappone a "linguaggio di programmazione ad alto livello". Si può dire che i linguaggi di programmazione di basso livello sono orientati "alla macchina" (ovvero il loro scopo è di essere direttamente eseguibili dal processore, o di poter essere tradotti facilmente in programmi eseguibili dal processore), mentre i linguaggi ad alto livello sono orientati "al programmatore" (il loro scopo è quello di essere facilmente utilizzabili dai programmatori umani).

Intesa in senso assoluto, l'espressione "linguaggio di programmazione a basso livello" si riferisce normalmente al linguaggio macchina o all'assembly, che differisce dal linguaggio macchina solo nella sua rappresentazione testuale. In senso relativo, si può dire che un linguaggio è "a più basso livello" di un altro, intendendo che fornisce meccanismi di astrazione meno potenti. In questo senso si può dire per esempio che il linguaggio C è (relativamente) a basso livello, pur essendo caratterizzato da una significativa astrazione rispetto al linguaggio macchina.

Linguaggi orientati alla macchina

Il linguaggio macchina (o codice macchina) è il linguaggio in cui sono scritti i programmi eseguibili per computer: può venire classificato come linguaggio di programmazione, sebbene quest'ultima espressione sia più spesso riservata per indicare i linguaggi di alto livello con cui si scrivono programmi non direttamente eseguibili, ma che richiedono una traduzione in linguaggio macchina, per es. per mezzo di un compilatore.

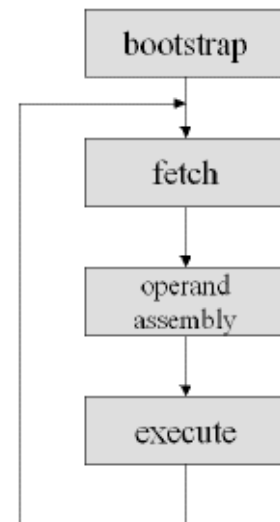
Il linguaggio macchina è basato su un alfabeto detto binario in quanto comprende due soli simboli, generalmente indicati con 0 e 1: un simbolo di questo alfabeto viene detto bit. Il processore o CPU è quella componente hardware di un computer che è in grado di eseguire i

programmi scritti in linguaggio macchina. In altre parole in linguaggio macchina sono definite l'insieme di istruzioni fondamentali che un processore è in grado di compiere (instruction set) in cui i codici di programmi da eseguire devono essere tradotti. In particolare i linguaggi a più basso livello si ottengono come semplice codifica (tabella di associazione) a partire dal linguaggio macchina in un crescendo di astrazione. Il grafico qui a fianco mostra le operazioni principali che esegue un microprocessore: questo ciclo prende il nome di fetch-execute.

L'espressione ciclo di fetch-execute si riferisce alla dinamica generale di funzionamento logico dei processori dei computer. In termini generali, un processore esegue iterativamente tre operazioni: preleva (fetch) un'istruzione dalla memoria primaria, in seguito avviene la decodifica (decode) con cui interpreta l'istruzione, infine la esegue (execute) combinandola coi dati relativi all'istruzione stessa. In questo modo il processore esegue sequenzialmente istruzioni che danno vita a thread e processi, sotto la supervisione del sistema operativo attraverso lo scheduler. (È un componente di un sistema operativo che, dato un insieme di richieste di accesso ad una risorsa (tipicamente l'accesso al processore da parte di un processo da eseguire), stabilisce un ordinamento temporale per l'esecuzione di tali richieste, privilegiando quelle che rispettano determinati parametri secondo una certa politica di scheduling.

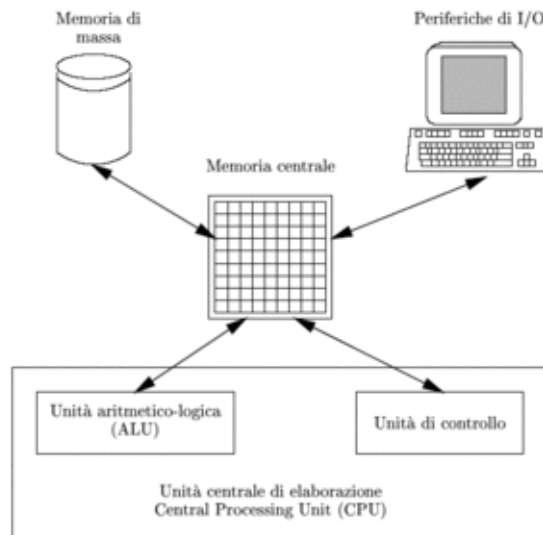
L'estrema velocità di elaborazione dei processori moderni rispetto alla velocità di accesso intrinseca alla memoria rende quest'ultima un collo di bottiglia in fase di progettazione dell'intero sistema di elaborazione

3.3) Componenti di un sistema di elaborazione: motherboard, unità centrale, unità periferiche, memorie e loro caratteristiche e gerarchia.



L'architettura della maggior parte dei moderni elaboratori è basata sul modello della macchina di Von Neumann; secondo tale architettura un sistema di calcolo è costituito da 4 elementi base:

1. **L'unità di elaborazione**, formata dai dispositivi elettronici che consentono di acquisire, interpretare ed eseguire le singole istruzioni;
2. **La memoria**, che contiene le istruzioni e i dati necessari per eseguire un programma;
3. **Le periferiche**, che consentono lo scambio di informazioni tra l'elaboratore e l'esterno;
4. **Il bus di sistema**, che funge da collegamento tra i vari elementi funzionali



Motherboard

Una scheda madre (detta anche piastra madre, in lingua inglese motherboard ("scheda madre")), è un tipo di scheda elettronica principale, raccoglie in sé tutta la circuiteria elettronica e i collegamenti di interfaccia tra i vari componenti interni principali di un personal computer come memoria e le altre schede elettroniche montate o alloggiare sopra, comprendendo anche i bus di espansione e le interfacce verso le periferiche esterne.

I componenti di una scheda madre possono variare a seconda di che tipo di computer si sta considerando: nel seguito di questa descrizione faremo riferimento a una generica scheda madre per personal computer.

- **Chipset:** l'insieme di [chip](#) che si occupano di smistare e dirigere il traffico di informazioni passante attraverso il [Bus di sistema](#), fra [CPU](#), [RAM](#) e [controller delle periferiche di input/output](#) (come [Floppy disk](#), [Hard disk](#) ecc.).
- **Socket** (CPU), è una parte fondamentale del computer che accoglie la [CPU](#). Nelle schede embedded (o in quelle vecchie e molto economiche) è assente, e il processore è saldato direttamente sulla scheda. Lo zoccolo (socket) può essere di tipo [PGA](#) o [LGA](#). Nel caso di processori di tipo PGA, i pin di interconnessione risiedono sulla parte inferiore della CPU. Se il socket è di tipo LGA (ovvero *Land Grid Array*) i pin risiedono sul socket ed è necessaria una piastra di caricamento per tenere in posizione la CPU dato che, a differenza delle CPU PGA, non è tenuta in posizione dai piedini che vanno ad incastrarsi nel socket. La soluzione LGA è adottata da diverso tempo da Intel con molti dei suoi processori Pentium IV, Core, entrambi interfacciati con 775 pin. A differenza di Intel, la rivale AMD ha adottato più tardi soluzioni LGA con l'avvento dei processori Athlon FX serie 7x interfacciati con 1207 pin alla scheda madre. Ora la soluzione LGA è divenuta di fatto standard.
- **ROM** (*Read Only Memory*), è la piccola memoria presente su tutti i personal computer, che in alcuni casi può essere riprogrammata, (può essere PROM, EEPROM, flash o altro) contenente l'[Uefi](#) della scheda madre. L'Uefi è un tipo di [firmware](#) dalle funzionalità

molto limitate. Le sue funzioni sono essenzialmente tre: eseguire il controllo dell'hardware all'accensione (il POST, *Power On Self Test*), eseguire alcune istruzioni basilari per il controllo dell'hardware stesso e caricare il [sistema operativo](#).

- **RAM** (*Random Access Memory*), può essere di diversi tipi, esistono vari tipi di RAM diffusi dalle industrie sin dai primi anni ottanta. Attualmente le schede madri in commercio adottano slot DDRAM (DDR, DDR2, DDR3, DDR4 nelle diverse evoluzioni delle prestazioni), derivate delle precedenti SDRAM, che a loro volta erano derivate dalle SIMM e SIPP presenti sulle macchine che montavano processori compatibili con l'[80386](#).
- **CMOS**, piccola memoria RAM, in cui sono memorizzate le impostazioni dell'[UEFI](#). Il CMOS è un semiconduttore che richiede pochissima energia per funzionare. Quando il computer viene spento, per mantenere memorizzate le impostazioni dell'UEFI, utilizza una piccola batteria al litio.
- **Il bus di espansione.** Si tratta di un collegamento dati generico punto-multipunto, progettato per permettere di collegare alla scheda madre delle altre schede *di espansione* alloggiare su connettori (*slot*), che ne estendono le capacità. Attualmente il tipo di bus più diffuso è il bus [PCI](#), destinato nel tempo a lasciare strada alla sua estensione [PCI Express](#), arrivata alla versione PCI e 4.0, molto più veloce. In linea di principio ad un bus può essere collegato hardware di ogni tipo: schede video aggiuntive, schede audio professionali, schede acquisizione dati, unità di calcolo specializzate, coprocessori: nella pratica si ricorre ad una scheda di espansione su slot interno solo per hardware che ha bisogno di una collaborazione estremamente stretta con la CPU o con la memoria RAM; per le espansioni hardware meno critiche si sfruttano le connessioni "lente" (USB, seriali ecc.).

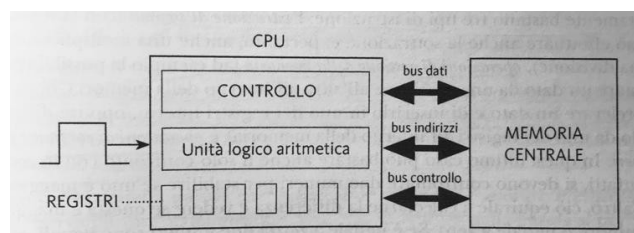
Unità centrale (processore)

Il processore rappresenta la mente del sistema informatico, la parte cioè che controlla e gestisce il flusso dei programmi ed esegue le singole istruzioni.

A livello fisico la CPU è un circuito integrato della Ddimensione di pochi centimetri quadrati. compiti del processore sono:

- lo spostamento dei dati
- le operazioni di tipo aritmetico
- controllo del flusso delle istruzioni

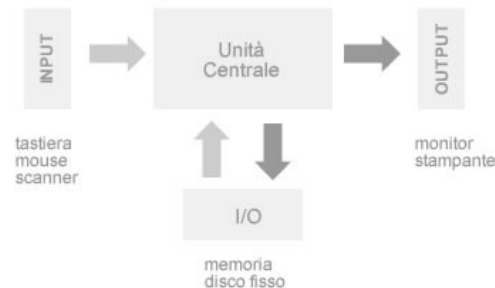
Il processore è un dispositivo sincrono, In quanto ogni cambiamento al suo interno avviene soltanto ogni volta che riceve un impulso di clock. All'interno di questo si trovano **un'unità di controllo, un unità logico aritmetica**, è un insieme di **registri** che servono alla CTU per contenere al suo interno i dati da cui dipendono i risultati delle operazioni che esegue.



come rappresentato in figura, la c p u è collegata con la memoria centrale tramite un bus di dati, un bus di indirizzi e un bus di controllo a seconda della tipologia di segnali che vengono trasmessi.

Unità Periferiche

Le periferiche informatiche sono dispositivi hardware del computer che permettono all'utente di interagire con il computer. L'unità periferica è un'interfaccia tra la scheda madre del computer e l'utente. Si chiama periferica poichè il dispositivo è generalmente un componente esterno della scheda madre che contiene la CPU. La periferica del computer è un dispositivo elettronico che viene collegato al computer per ricevere o inviare i dati, rispettivamente periferica di input e periferica di output. Nell'architettura hardware del computer le periferiche gestiscono il flusso di dati in entrata (input) e in uscita (output) del computer. Tutte le periferiche sono collegate all'unità centrale dell'elaboratore elettronico, a cui spetta il compito di controllo.



Le periferiche di un computer sono classificate in:

- **Periferiche di input.** Le periferiche di input consentono all'utente di inserire dati o di richiedere l'esecuzione di un comando o di un programma informatico, di selezionare una funzione, ecc. Sono unità periferiche di entrata. Il flusso di dati si muove dall'utente verso il computer. Alcuni esempi di periferiche di input sono la tastiera, il mouse, lo scanner, ecc.
- **Periferiche di output.** Le periferiche di output permettono all'utente di osservare il risultato dell'elaborazione dati. Sono unità periferiche di uscita. Il flusso di dati si muove dal computer verso l'utente. Alcuni esempi di periferiche di output sono il monitor (video), la stampante, ecc.
- **Periferiche di input/output.** Le periferiche di input/output (I/O) possono svolgere sia operazioni di lettura dei dati (input) che operazioni di scrittura dei dati (output). Sono periferiche informatiche utilizzate prevalentemente per le operazioni di archiviazione e di memorizzazione dei dati. Alcuni esempi di periferiche I/O sono la memoria di massa, il disco fisso, il floppy disk, la pen drive, ecc.

In base alla collocazione dell'unità periferica nell'architettura hardware del computer, è possibile suddividere le periferiche anche in periferiche interne e periferiche esterne.

- **Periferiche interne.** Le periferiche interne sono collocate all'interno del cabinet (case) del computer. Si tratta prevalentemente di schede informatiche, installate sulla scheda madre o ollegate a quest'ultima tramite bus, che consentono di aggiungere delle funzionalità al computer. Un esempio di periferica interna è la scheda video, la scheda di rete, la scheda audio, la memoria di massa, ecc.
- **Periferiche esterne.** Le periferiche esterne sono situate all'esterno del cabinet (case) del computer. Sono collegate all'unità centrale del computer tramite cavo o via wireless. Un esempio di periferica esterna è la stampante, lo scanner, ecc.

Memorie: caratteristiche e gerarchia

La memoria, in informatica, è un elemento di un computer o di un suo sottosistema deputato alla persistenza dei dati e/o delle istruzioni dei programmi, la cui implementazione fisica dà vita ai vari supporti di memorizzazione esistenti.

Una memoria può essere considerata astrattamente come una sequenza finita di celle in cui ogni cella contiene una sequenza finita di bit, normalmente gestiti a gruppi di otto detti byte. Pertanto lo spazio fisico della memoria può essere immaginato come una sequenza di posizioni, ognuna contenente un byte. Ogni posizione è individuata da un preciso indirizzo di memoria, normalmente espresso tramite un numero intero positivo. Ad oggi tra le tecnologie che implementano questo modello astratto, le più diffuse sono la memoria elettronica, la memoria magnetica e la memoria ottica.

Nell'architettura dei calcolatori, si distinguono due tipi di memoria: la memoria primaria, che lavora a più diretto contatto con il processore, costituita fondamentalmente da memoria RAM, memoria ROM, memoria Cache, e la memoria secondaria di cui maggiori rappresentanti sono gli hard disk, ma anche supporti rimovibili come dischi floppy, CD, DVD, nastri magnetici, memorie flash di ogni tipo ed altro ancora.

Memoria primaria o centrale

Collegata alla scheda madre tramite connettori chiamati socket ed alla CPU tramite il BUS di sistema, la memoria primaria, chiamata anche memoria centrale o memoria principale, contiene dati ed istruzioni prelevati dalla memoria di massa in attesa che questi siano a loro volta prelevati ed elaborati dal microprocessore, lavorando dunque in maniera strettamente accoppiata con esso ed essendo dunque assimilabile ad una memoria di transito o appoggio.

Molto spesso si tratta di memoria RAM e memoria cache e nelle moderne architetture dei processori è spesso incorporata nella scheda CPU o direttamente nel chip del processore stesso. È una parte importante del computer in quanto dalle sue dimensioni in termini di capacità di immagazzinamento dipende dunque la quantità massima di dati che possono essere prelevati e dunque elaborati dal processore in condizioni di monotasking e multitasking ed è quindi

considerata a tutti gli effetti un parametro prestazionale del computer stesso. Qualora la memoria primaria venga esaurita molti sistemi di elaborazione moderni sono in grado di implementare il cosiddetto meccanismo della memoria virtuale come estensione provvisoria della memoria primaria.

Bisogna distinguere tra vari tipi di memorie primarie, a seconda della funzione svolta e delle loro caratteristiche peculiari. Di seguito vengono elencate quelle più importanti.

- [RAM](#), l'acronimo per "random access memory", ovvero "memoria ad accesso casuale", è la memoria in cui vengono caricati i dati che devono essere utilizzati dal calcolatore per elaborare. La RAM può essere volatile (si cancella spontaneamente ed ha bisogno di essere aggiornata), statica o tamponata (mantiene l'alimentazione anche a macchina spenta). Il processore identifica le celle della RAM tramite indirizzi preassegnati che ne specificano la posizione: la memoria si presenta, quindi, come un enorme vettore (stringa ordinata di elementi detti byte, ciascuno individuabile con un indirizzo). Il termine "random" evidenzia che non ci sono differenze ad accedere alle varie celle della memoria. Le caratteristiche della RAM vengono ereditate anche da tutte le altre memorie ad accesso casuale (individuabili facilmente dal fatto che contengono RAM alla fine). Sostanzialmente le memorie RAM si suddividono in [DRAM](#) (dinamiche), [SRAM](#) statiche e che vengono utilizzate per la [memoria cache](#).
- [Cache](#) RAM, una memoria associativa integrata nel processore, che ha la caratteristica di essere molto veloce; dato l'elevato costo, viene utilizzata esclusivamente per contenere i dati e le istruzioni utilizzati più di frequente (in modo da migliorare notevolmente le prestazioni del processore).
- [ROM](#), l'acronimo per "read only memory", ovvero "memoria in sola lettura (o solamente leggibile)", è una memoria permanente (cioè ha un contenuto fisso che non può essere cancellato ed inoltre non è volatile), presente sulla scheda madre, che contiene le istruzioni che la CPU deve caricare per consentire l'avvio del sistema e le routine di base che prendono il nome di [BIOS](#) (Basic I/O System).

La memoria secondaria, chiamata anche memoria ausiliaria o memoria di massa, è un'unità che si aggiunge alla memoria primaria (o centrale) dell'elaboratore per accrescerne le capacità di memorizzazione. Consiste in una classe di [dispositivi](#) che non sono posti a diretto contatto con il [processore](#). Di conseguenza i dati in essi contenuti non vengono persi una volta spento il processore stesso.

Memoria secondaria

Qui sotto sono elencate le varie categorie di memorie secondarie:

- I [dischi magnetici](#), composti da uno o più dischi (i [disk pack](#)) ricoperti di materiale ferromagnetico, vengono "letti e scritti" (cioè su questi dischi vengono salvati e recuperati i dati) mediante un braccio mobile dotato della "testina di lettura/scrittura". I dati vengono trasferiti ai dischi magnetici tramite un buffer nella memoria centrale ed occupano successive posizioni lungo le tracce, sotto forma di differenti stati di magnetizzazione. I settori dei dischi vengono letti e

scritti interamente utilizzando il numero della superficie, della traccia e del settore. Il tempo di accesso ai dischi magnetici è superiore rispetto a quello della memoria centrale, ma i costi, a parità di quantità di informazione memorizzata, sono decisamente più bassi. Di questa categoria fanno parte ad esempio gli [hard disk](#) e i [floppy disk](#) (realizzati con materiale plastico flessibile).

- I [dischi ottici](#), composti da materiale riflettente ricoperto da una sostanza protettiva, dove l'informazione viene registrata realizzando modificazioni della superficie riflettente e viene letta mediante un raggio [laser](#) che riscontra le irregolarità della superficie riflettente. I dischi ottici sono senza dubbio i supporti di memoria secondaria più diffusi: ne esistono di vari tipi, alcuni riscrivibili (cioè una volta scritti possono essere riscritti nuovamente) e non riscrivibili (una volta immagazzinati dei dati sul disco, questo non è più riscrivibile con altri dati). Fanno parte di questa categoria i [CD](#), i [CD-ROM](#) e i [DVD](#).
- I [nastri magnetici](#), composti da fettucce di nastri magnetizzabili e gestiti dalle unità a nastro (che dispongono della testina lettura/scrittura), servono per svolgere funzioni di [backup](#) e [log](#) (registrazione delle operazioni effettuate in un certo tempo). I nastri magnetici consentono solo un accesso sequenziale ai dati (cioè è necessario leggere tutti i dati precedenti prima di accedere ad un certo dato). Tra i vari tipi di nastri magnetici, le prestazioni migliori sono ottenute dai [nastri per la memorizzazione di dati digitali](#).
- La [Memoria Flash](#), memoria elettronica non volatile di tipo [EEPROM](#). Si presenta come [schede di memoria](#) dall'ingombro ridotto.

3.3) Elaboratori monoprocesso: tipologie e caratteristiche funzionali

Con il termine monoprocesso si intende un sistema equipaggiato con 1 solo processore. Tale termine viene utilizzato per distinguere questo tipo di sistemi da quelli biprocessore e multiprocessore in cui sono presenti, come facilmente si può intuire, 2 o più processori che operano in parallelo. Normalmente i sistemi dotati più di 1 processore vengono utilizzati negli ambiti in cui è richiesta una grande potenza di elaborazione, ed essi sono generalmente delle workstation o dei server.

In una macchina monoprocesso e multiprocesso un solo processo alla volta può essere in esecuzione sulla CPU.

Gli altri processi sono rispettivamente

- in attesa (di un evento, quale il completamento di un'operazione di I/O)
- pronti (ad essere eseguiti, aspettano che la CPU si liberi e sia loro assegnata)

Ciascun processo può trovarsi in uno dei seguenti stati :

- **esecuzione:** sta effettivamente usando il processore per la sua esecuzione
- **pronto:** è potenzialmente in condizione di usare il processore che è però occupato da un altro processo in esecuzione
- **attesa:** aspetta la terminazione di un evento esterno

Per quanto concerne il tipo di istruzioni che è in grado di svolgere un microprocessore, si possono avere due tipi di architetture: CISC e RISC.

L'architettura CISC è caratteristica dei microprocessori che sono in grado di eseguire operazioni molto complesse, per attuare le quali sono necessari molti cicli di clock

Nel caso dell'architettura RISC, invece, si ha la situazione opposta: si dispone di un numero molto basso di operazioni molto elementari e attraverso la combinazione di queste operazioni è possibile consentire tutte le funzionalità della CPU. In questo modo, avendo a disposizione un numero molto elevato di registri, si ottiene un' elevatissima velocità di elaborazione.

Un'altra caratteristica fondamentale della CPU è la frequenza di clock alla quale può operare: maggiore è il numero di impulsi di clock che la CPU riceve in un secondo, maggiore è il numero delle operazioni che la CPU è in grado di svolgere

3.4) Architetture parallele. Sistemi multiprocessori superscalari, sistemi a memoria condivisa, sistemi a memoria distribuita. sistemi a matrice.

Architetture parallele

L'elaborazione parallela, al contrario di quella tradizionale (sequenziale di tipo Von Neumann), non si riferisce ad una unica soluzione architetturale ma obbliga a definire una classificazione dei diversi sistemi paralleli. Una classificazione di macchine parallele è stata proposta da Flynn [Hwa94], dove i concetti discriminanti sono il *flusso di istruzioni (instruction stream)* e il *flusso dei dati (data stream)*. Con il flusso di istruzioni si intende la sequenza di istruzioni eseguita da una *unità di elaborazione (processore)*; con il flusso di dati si intende invece la sequenza di operandi da essa manipolati. Controllare un flusso di istruzioni e un flusso di dati significa determinare un ordine con cui le istruzioni del programma verranno eseguite da un processore e su quali dati opereranno. Per esempio, nel caso di una macchina sequenziale vi è un unico flusso di istruzioni ed un unico flusso di dati possibile. In base alla tassonomia di Flynn una macchina sequenziale viene detta macchina SISD (*Single Instruction stream Single Data stream*).

L'individuazione dei concetti fondamentali di flusso d'istruzioni e flusso di dati conduce alla determinazione di tre classi di macchine parallele, indipendentemente dalla realizzabilità e utilità di queste:

- 1) Macchine il cui flusso di istruzioni viene applicato a più flussi di dati; queste vengono dette macchine parallele di tipo SIMD (*Single Instruction stream Multiple Data stream*).
- 2) Macchine di tipo MISD (*Multiple Instruction stream Single Data stream*), in cui ad uno stesso flusso di dati vengono applicati più flussi di istruzioni.

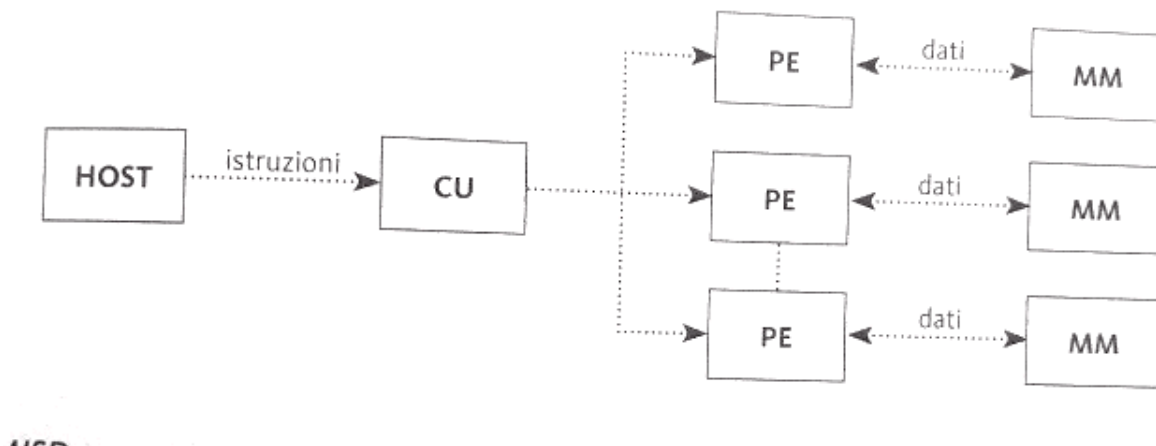
3) Macchine di tipo MIMD (*Multiple Instruction stream Multiple Data stream*), in cui più flussi di istruzioni vengono applicati a più flussi di dati.

Sebbene esistono dei problemi per cui la classe MISD potrebbe essere estremamente utile è evidente che si tratti, tuttavia, di un modello generalmente non realistico per il calcolo parallelo.

Invece le classi SIMD e MIMD si sono rilevate, ancor prima che un mezzo di classificazione, uno strumento concettuale ampiamente utilizzato per analizzare o proporre nuove architetture parallele. Dunque in tali classi ricade la quasi totalità di macchine parallele realmente costruite.

Parallelismo SIMD

La programmazione di macchine SIMD viene anche detta programmazione parallela sincrona. Infatti tutti i processori sono vincolati ad eseguire la stessa operazione su dati diversi. In questo modo i flussi paralleli che vengono generati automaticamente da un'istruzione su dati differenti si ricongiungono, al completamento dell'istruzione stessa, in virtù di meccanismi che non sono controllabili dal programmatore.

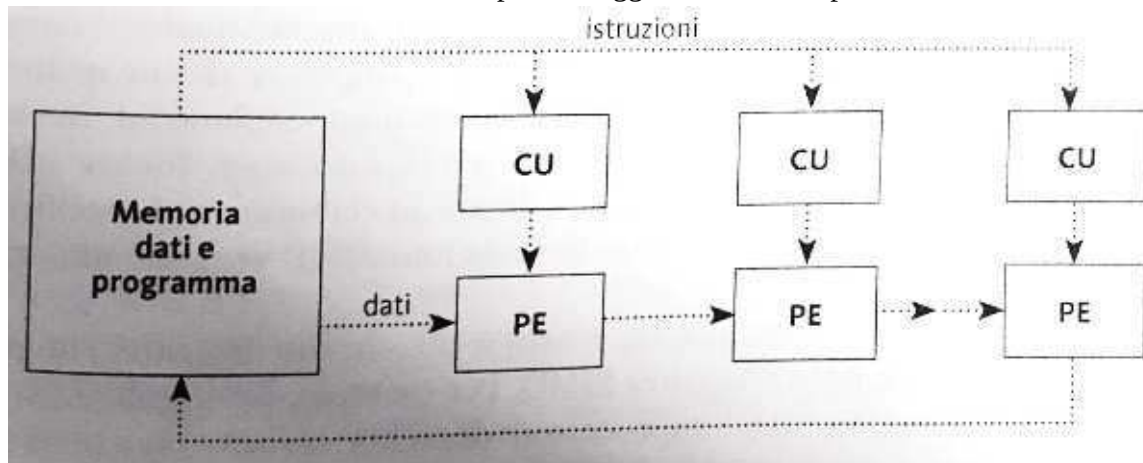


Un primo tipo di macchine SIMD è quello costituito dai processori vettoriali (*vector processor*). In questi una singola istruzione vettoriale controlla la manipolazione simultanea (anche se in diverse fasi funzionali) di più dati. Più precisamente un *vector processor* è un coprocessore progettato per eseguire calcoli vettoriali e cioè di applicare un'istruzione vettoriale che può anche essere un'operazione di tipo aritmetico o logico, ad una stringa di dati formata da un unico vettore. In questo modo di operare si ha la differenza principale con l'elaborazione scalare che come noto coinvolge solamente uno o due operandi.

Parallelismo MISD

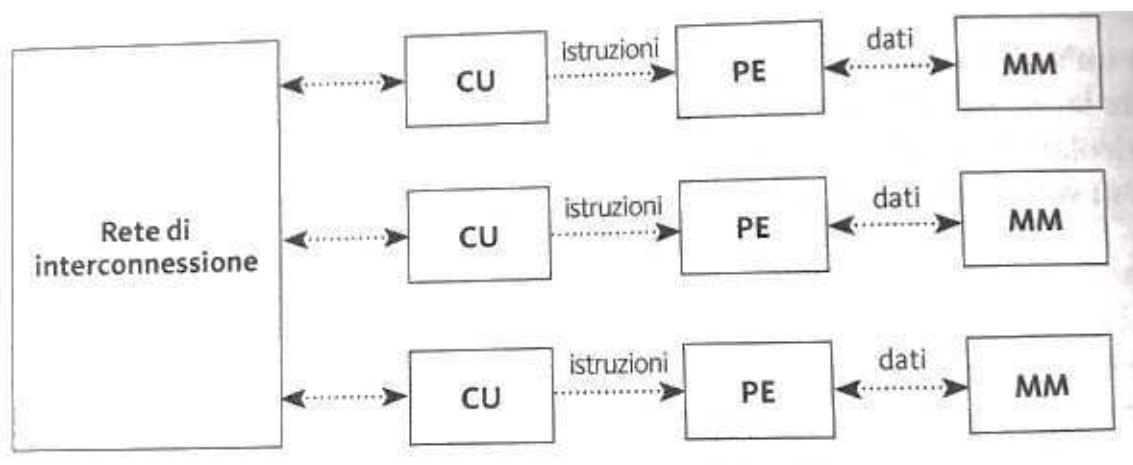
In questo tipo di architettura (figura 1.1) un 'unico flusso di dati viene utilizzato da tutte le unità funzionali simultaneamente, ciascuno in accordo all'istruzione che riceve dalla sua unità di controllo. Quindi il parallelismo viene raggiunto permettendo ai processori di fare cose diverse

nello stesso tempo sullo stesso dato. Questa classe di calcolatori si presta naturalmente a quei calcoli che richiedono che uno stesso input sia soggetto a diverse operazioni



1.3. Parallelismo MIMD

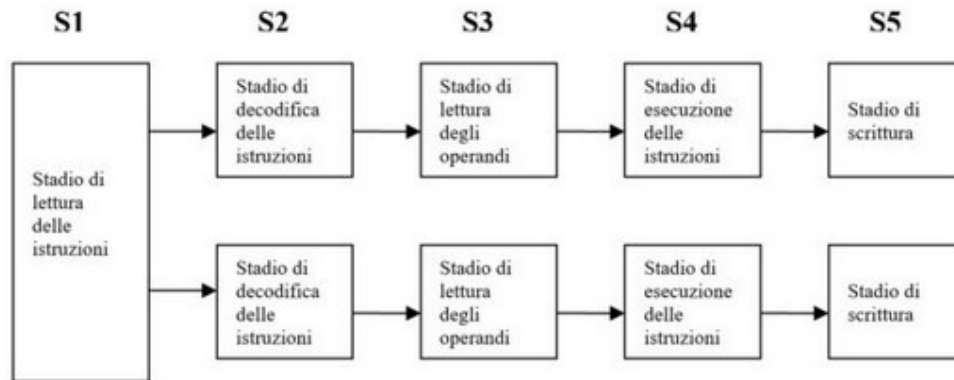
Nel caso di parallelismo MIMD (Multiple Instruction stream Multiple Data stream) il programmatore ha la possibilità di controllare più flussi di istruzioni e più flussi di dati. Un flusso di istruzioni eseguite da un processore reale viene detto *processo* oppure *task*. A causa di questa maggiore generalità il parallelismo delle macchine MIMD è molto più generale rispetto a quello delle architetture SIMD. Infatti, un problema che non è caratterizzato da una struttura regolare, ma che tuttavia presenta un potenziale parallelismo, si presta ad una elaborazione di tipo MIMD. Un tipico esempio è fornito dai problemi combinatori che possono essere risolti generando e analizzando una struttura dati nota come grafo dello spazio degli stati, dove ogni processore ha il compito di generare e analizzare un sottografo al fine cercare una soluzione del problema.



Processore Superscalare

Per processore superscalare si intende, in generale, un processore in grado di realizzare più di un'istruzione per ciclo di clock. Per ottenere questo obiettivo nel processore sono presenti diverse unità funzionali dello stesso tipo, con dispositivi aggiuntivi per assegnare le istruzioni alle varie unità. Per esempio, sono generalmente presenti numerose attività per il calcolo interno (ALU). Le unità di controllo identificano quali istruzioni possono essere svolte in parallelo e le trasmettono alle relative unità

Una caratteristica dei processori superscalari è l'inserimento di più processori indipendenti (core) in un singolo processore. Questi processori sono forniti di pipeline totalmente separate e quindi sono in grado di eseguire programmi differenti, cosa non possibile nelle CPU classiche.



Esiste anche un altro approccio per realizzare processori superscalari. È possibile implementare una pipeline singola ma con unità funzionali multiple.

Questo permette di realizzare più thread in parallelo senza dover riprodurre tutte le unità funzionali di un processore, e quindi risparmiando molti transistor rispetto a una soluzione pura.

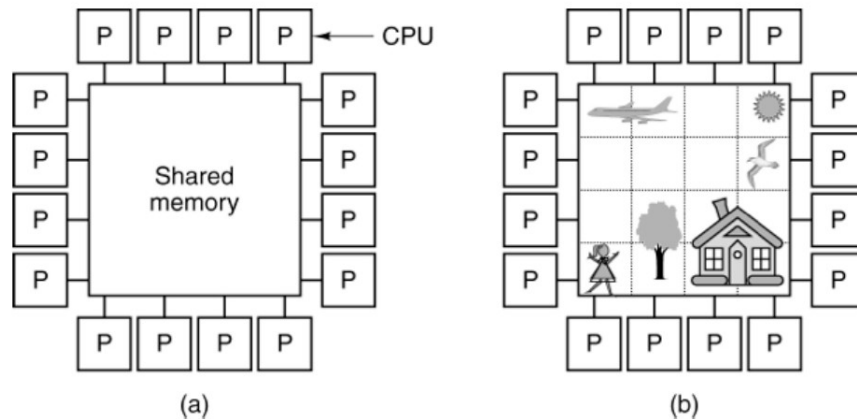
Sistemi a memoria condivisa

Utilizzando p processori sarebbe teoricamente possibile raggiungere uno *speedup* uguale a p . Anche nel caso in cui esista effettivamente la possibilità di decomporre il calcolo in almeno p attività distinte, tale soluzione ideale è comunque irraggiungibile a causa di conflitti per l'utilizzo di risorse comuni (la rete di interconnessione, i singoli percorsi all'interno di tale struttura, i moduli di memoria e le locazioni in ciascun modulo) e la presenza in ogni algoritmo di porzioni non parallelizzabili, la necessità di introdurre punti di sincronizzazione e il tempo che ogni processore spreca nell'inviare o nell'attendere un dato.

E' evidente che nel caso in cui la memoria sia realizzata come blocco unico si hanno dei conflitti se più di un processore necessita di leggere o scrivere un dato in memoria. Quindi una soluzione di questo tipo porta notevoli aggravii sul tempo complessivo per portare a termine una computazione nell'ipotesi reale che un'operazione di lettura o di scrittura richieda un tempo non nullo. Per risolvere questi problemi la memoria viene suddivisa in un certo numero di moduli; tuttavia tale suddivisione presenta alcuni svantaggi da tenere in dovuta considerazione. Uno di

questi riguarda la complessità, è dunque, il costo della struttura di interconnessione che cresce all'aumentare della suddivisione della memoria.

Il fatto che in sistemi con memoria condivisa la rete di interconnessione sia bidirezionale vuol dire che nel collegamento logico che unisce il processore P_i ed il modulo di memoria M_j i dati possono viaggiare nei due sensi. Anche se suddivisa in moduli, da un punto di vista logico e pertanto dell'utente tale suddivisione della memoria rimane del tutto trasparente. La ripartizione serve per permettere a più processori di accedere contemporaneamente a locazioni di memoria che risiedono in moduli differenti e quindi di raggiungere un certo grado di parallelismo nell'accesso in memoria. Un modulo di memoria può essere acceduto di volta in volta da un solo processore. Se più processori vogliono leggere o scrivere in locazioni, anche differenti, contenute nello stesso modulo, è necessario che gli accessi siano serializzati, cioè venga stabilito un ordinamento in base al quale le richieste vengono soddisfatte una dopo l'altra e non parallelamente. Per minimizzare i conflitti nell'uso di un singolo modulo i dati vengono opportunamente distribuiti fra i vari moduli.



Nel corso di una computazione parallela i dati elaborati da un processore possono essere ulteriormente elaborati da un altro processore. Se esiste una memoria comune questa può anche essere usata come mezzo di comunicazione: un processore deposita un messaggio in una locazione di memoria ben precisa e successivamente un altro processore lo preleva. Questa comunicazione indiretta necessita di una sincronizzazione a livello dei dati, il che significa garantire che il processo destinatario del messaggio non legga il contenuto dell'area destinata a contenere il messaggio stesso prima che esso sia stato interamente scritto. Di qui la necessità di primitive di sincronizzazione concorrenti come quelle menzionate precedentemente.

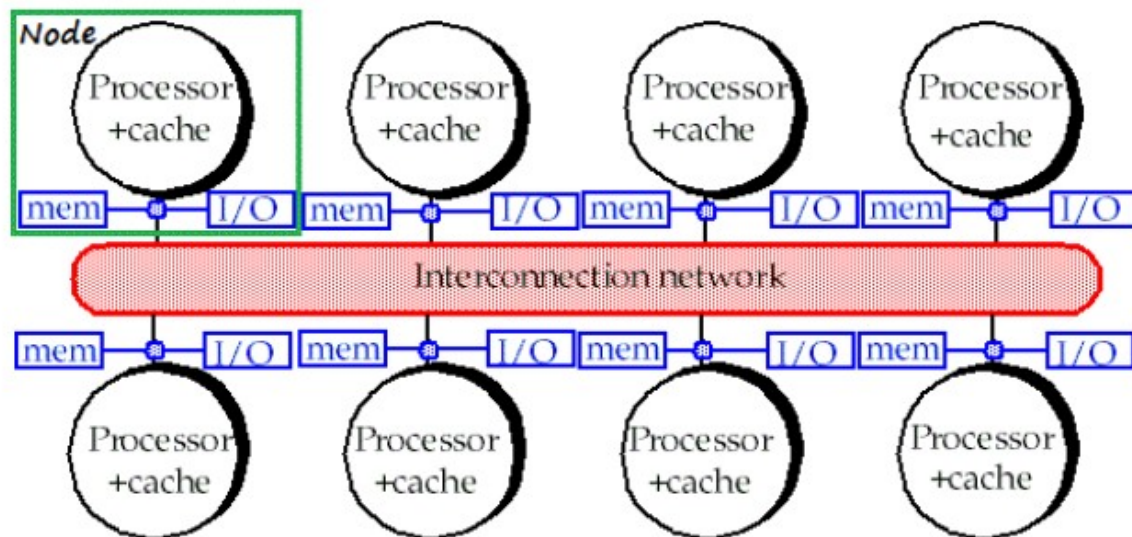
Caratteristiche dei sistemi multiprocessore a memoria condivisa:

- La memoria logica è la stessa per tutti i processori; ad esempio, tutti i processori associati alla stessa struttura dati lavoreranno con gli stessi indirizzi logici, in quanto globali, accedendo così alle stesse locazioni di memoria.
- La sincronizzazione è ottenuta leggendo i compiti dei vari processori e concedendo a turno la memoria condivisa; infatti i processori possono solo accedervi uno alla volta.

- Una locazione di memoria condivisa non deve essere modificata da un compito mentre un altro compito concorrente vi accede.
- La condivisione dei dati tra i vari compiti è veloce; infatti il tempo necessario alle attività per comunicare tra loro è il tempo che una di esse impiega per leggere una singola locazione (ciò dipende dalla velocità di accesso alla memoria).
- La scalabilità è limitata dal numero di vie d'accesso alla memoria; questo limite si presenta soprattutto quando ci sono più compiti che connessioni alla memoria. In queste situazioni si avranno dei processori in stato d'attesa e quindi tempi di latenza maggiori.
- Il programmatore è responsabile della gestione della sincronizzazione, inserendo opportuni controlli, semafori, lock ecc nel programma che gestisce le risorse.

Sistemi a memoria distribuita

In un sistema a memoria distribuita la memoria è associata ai singoli processori e un processore è solamente in grado di indirizzare la propria memoria. Alcuni autori fanno riferimento a questo tipo di sistema come “multicomputer” [10], riflettendo il fatto che i blocchi del sistema sono a loro volta piccoli sistemi completi di processore e memoria, come si può vedere in figura:



Questa organizzazione presenta diversi vantaggi. In primo luogo, non vi sono conflitti a livello di bus o switch. Ogni processore può utilizzare l'intera larghezza di banda della propria memoria locale, senza subire interferenze da parte di altri processori. In secondo luogo, la mancanza di un bus comune significa che non c'è limite intrinseco al numero di processori. In terzo luogo, non ci sono problemi di coerenza della cache. Ogni processore è responsabile dei propri dati, e non deve preoccuparsi di aggiornare eventuali copie. Il principale svantaggio nel disegno a memoria distribuita è che la comunicazione interprocessore è più difficile da implementare. Se un processore richiedesse dei dati presenti nella memoria di un altro processore, i due processori dovrebbero necessariamente scambiarsi dei messaggi tramite il Message Passing.

Ciò introduce due fonti di rallentamento: per costruire e inviare un messaggio da un processore all'altro ci vuole tempo, e inoltre un qualsiasi processore deve essere interrotto al fine di gestire i messaggi ricevuti da altri processori. Un programma, creato per funzionare su una macchina a memoria distribuita, deve essere organizzato come un insieme di attività indipendenti che comunicano tra loro tramite messaggi.

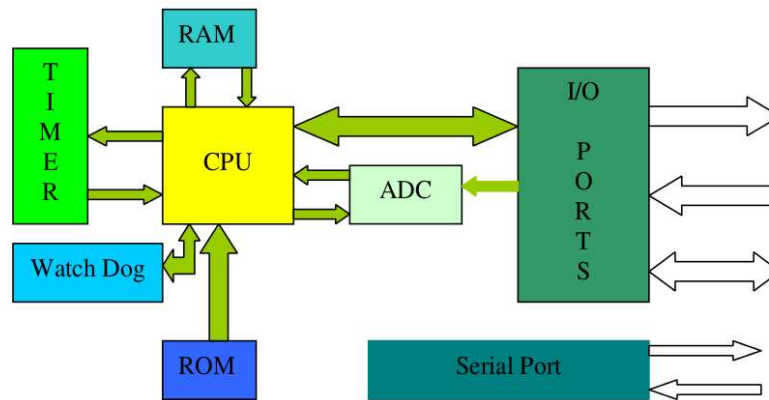
Caratteristiche dei sistemi multiprocessore a memoria distribuita:

- La memoria è fisicamente distribuita tra i vari processori; ogni memoria locale è accessibile direttamente solo dal suo processore.
- La sincronizzazione è ottenuta mediante lo spostamento di dati (anche se è solo il messaggio stesso) tra i processori (comunicazione).
- La suddivisione dei dati nelle memorie locali incide molto sulle prestazioni della macchina: è fondamentale fare una suddivisione accurata in modo da ridurre al minimo le comunicazioni tra le CPU. Inoltre, il processore che coordina queste operazioni di decomposizione e composizione deve comunicare efficacemente con i processori che operano sulle singole parti delle strutture dati.
- Il Message Passing consiste nel far comunicare le CPU tra di loro tramite scambi di pacchetti dati. I messaggi trasmessi sono unità discrete di informazione; nel senso che hanno una identità ben definita, perciò deve essere sempre possibile poterli distinguere gli uni dagli altri.

3.5) Architettura dei microcontrollori e loro programmazione

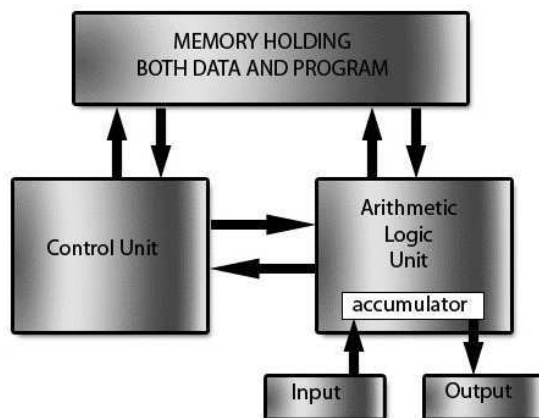
Un microcontrollore (microcontroller o MCU, MicroController Unit) è un single-chip computer, ovvero un microcalcolatore integrato su un singolo chip. Come suggerisce il nome, il microcontrollore è utilizzato principalmente per realizzare sistemi di controllo digitale e, in particolare, nei dispositivi cosiddetti embedded. Si tratta di sistemi elettronici di elaborazione a microprocessore progettati appositamente per una determinata applicazione (special purpose) ovvero non riprogrammabili dall'utente per altri scopi.

Il microcontrollore si differenzia rispetto al microprocessore in quanto al proprio interno contiene normalmente anche una certa quantità di memoria RAM e di EPROM e vari dispositivi periferici integrati, come timer, convertitori AD etc. Si tratta dunque di un vero e proprio computer completo di tutto ciò che occorre per il suo funzionamento. La figura seguente mostra uno schema della struttura interna di un MCU:

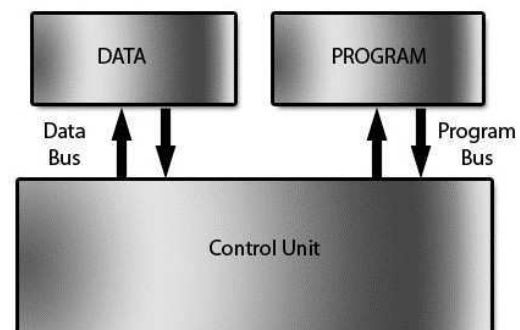


A differenza dei calcolatori classici, i microcontrollori adottano generalmente un'architettura interna detta Harvard e caratterizzata dalla separazione fra bus dati e bus istruzioni e, conseguentemente, fra la memoria che contiene i dati e quella contenente le istruzioni. La figura seguente mostra un confronto fra l'architettura di Von Neumann (comunemente utilizzate nei microprocessori classici) e quella Harvard. Come si può osservare in questo secondo caso esistono due bus separati che collegano l'unità di controllo alla memoria: uno è dedicato ai dati e l'altro alle istruzioni del programma. Invece nella struttura classica di Von Neumann, dati e istruzioni condividono la stessa memoria e lo stesso bus.

The Von Neumann or Stored Program architecture



The Harvard architecture



La struttura generale di un programma per un microcontrollore può essere schematizzata come segue:

1. **Inizializzazione:** in questa prima sezione del software, vengono impostate le configurazio-ni degli SFR, in base al funzionamento specifico richiesto per le periferiche integrate. Ad esempio, nel caso si debba eseguire operazioni periodiche con temporizzazioni predefinite, sarà necessario predisporre una periferica di conteggio (es. il Timer0 nell'8051) in modo che funzioni in modalità "timer" e che sia possibile gestire la condizione di roll-over in modo che essa avvenga con le tempistiche necessarie.
2. **Ciclo principale:** in questa sezione, viene programmato il compito vero e proprio del microcontrollore, vale a dire il controllo del sistema fisico ad esso collegato (tramite

sensori/attuatori). Tipicamente, le istruzioni che costituiscono il programma principale devono essere eseguite ciclicamente per una ripetizione virtualmente infinita, in modo da mantenere sempre attivo il controllo in tempo reale del sistema

3. **Terminazione:** in questa sezione, tipicamente eseguita solo in caso di guasto o in fase di spegnimento del sistema di controllo, si eseguono eventuali operazioni necessarie alla “messa in sicurezza” del sistema controllato (es. spegnimento degli azionamenti elettrici ed attesa dei transistori di scarica di condensatori, ecc.).

Capitolo 4

Sistemi operativi e software applicativo

4.1) Sistemi operativi: tipologie, architettura e funzioni

Il software di un sistema informatico viene suddiviso in due categorie:

- Software di base, dedicato alla gestione delle funzioni elementari dell'elaboratore;
- Software applicativo dedicato alla realizzazione di particolari esigenze dell'utente e che riesce ad agire sull'elaboratore solo con il tramite del software di base.

Il componente principale del software di base è il sistema operativo, il programma incaricato di gestire le varie risorse fisiche dell'elaboratore svolgendo compiti differenti a seconda della complessità del sistema posto sotto il suo controllo.

- Il sistema operativo in pratica gestisce le risorse fisiche del sistema di elaborazione: processore, memoria centrale, memoria di massa, dispositivi di input/output;
- Consente la gestione delle informazioni
- Fornisce l'interfaccia uomo macchina.

I sistemi operativi più diffusi sono i Windows di Microsoft (che hanno soppiantato l'MS-DOS), il MacOS di Macintosh, Linux e Unix utilizzati soprattutto per i server o in ambito di ricerca.

Esiste una sostanziale differenza tra le vari "marchi":

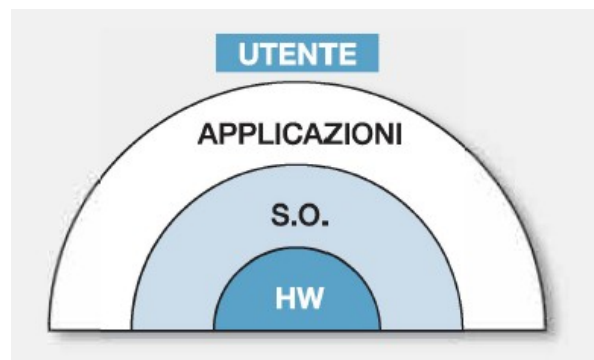
- Linux, Unix e, un tempo, MS-DOS di Microsoft sono sistemi operativi a linea di comando: sullo schermo non compare alcuna grafica e tutti i comandi vanno digitati con la tastiera. Ovviamente sono sistemi complessi e dotati di scarsa usabilità. Ecco perché sono state create delle interfacce grafiche, che si appoggiano ai sistemi. Esistono interfacce user-friendly per Unix e Linux e, tutte le versioni di Windows precedenti al 98, erano solo degli applicativi supportati da MS-DOS.
- I sistemi Unix e Linux sono molto simili tra di loro. Si differenziano solamente dal fatto che Linux gira sui PC, mentre Unix ha bisogno di macchine più potenti. Entrambi degli open source, ovvero non sono il prodotto di un'azienda ma vengono implementati da una comunità di ricercatori. Per questo motivo sono completamente gratuiti.
- Windows 98 (e i successivi, dal 2000 all'XP, al Windows Vista) e MacOS di Macintosh (vai al sito) sono invece sistemi operativi ad interfaccia grafica. Tutte le operazioni si svolgono con il mouse, tramite una visualizzazione a finestre. Si tratta di sistemi che, a differenza di quelli a linea di comando, sono molto più facili da usare, anche da un utente inesperto.

Architettura di un SO

Per semplificarne la progettazione e lo sviluppo un sistema operativo è di solito organizzato internamente in moduli corrispondenti alle diverse risorse che esso deve gestire:

- il gestore del processore e dei programmi in esecuzione;
- il gestore della memoria per il codice e per i dati dei programmi in esecuzione;
- il gestore dei file memorizzati su disco o su altri supporti;
- i gestori dei dispositivi di input e di output;
- il gestore della comunicazione di rete.

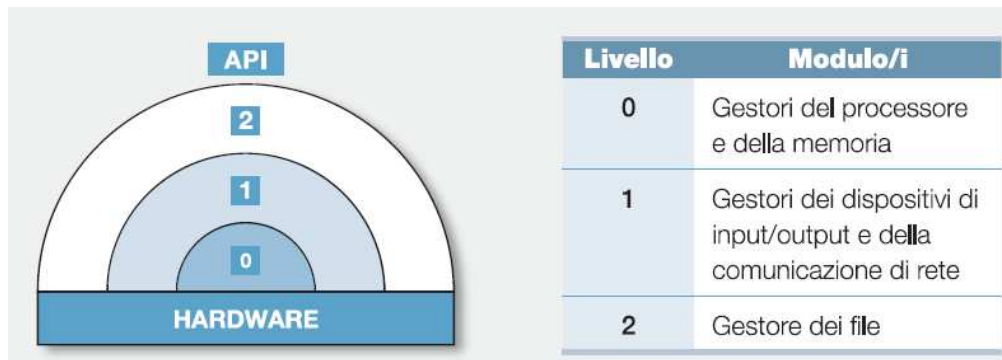
Dal punto di vista dell'utente del computer il sistema operativo (S.O.) costituisce un'estensione dell'hardware (HW) che incapsula consentendo l'esecuzione dei programmi applicativi con cui interagisce



In quest'ottica l'interfaccia grafica di interazione con l'utente è considerata come un'applicazione e non come un componente del sistema operativo.

Pur essendo un elemento software, la particolarità del sistema operativo di costituire un'estensione dell'hardware del computer lo rende un programma con privilegi speciali: il processore esegue il codice del sistema operativo in una modalità protetta, distinta dalla modalità utente in cui sono eseguiti i normali programmi applicativi; il kernel o nucleo di un sistema operativo è costituito dai moduli eseguiti in modalità protetta.

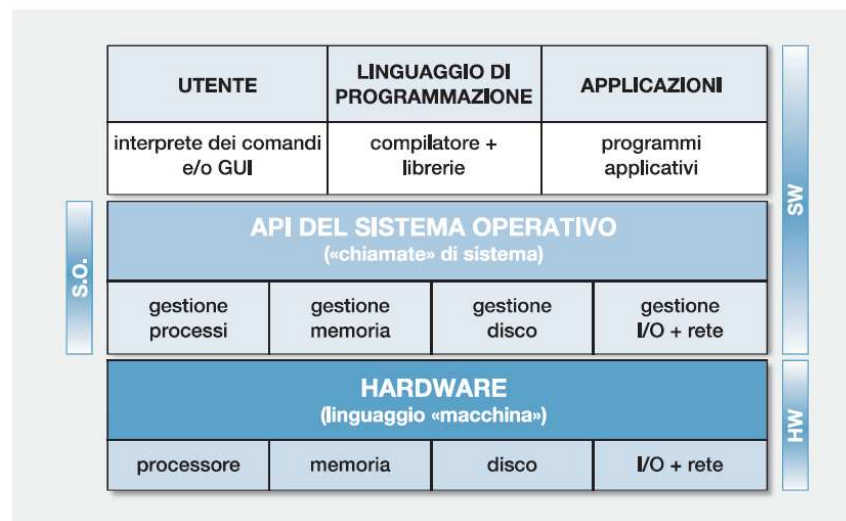
OSSERVAZIONE I sistemi operativi per i quali tutti i moduli sono eseguiti in modalità protetta, come un unico programma, sono definiti **monolitici**. Invece i sistemi operativi in cui tutti i moduli, esclusi i gestori del processore e della memoria, vengono eseguiti in modalità utente sono denominati **micro-kernel**. La maggior parte dei sistemi operativi più diffusi sono monolitici, o ibridi; in quest'ultimo caso solo alcuni moduli sono eseguiti in modalità utente come programmi esterni.



Anche nei sistemi operativi monolitici il kernel ha un'organizzazione interna gerarchica distribuita su più livelli.

OSSERVAZIONE L'invocazione di una system-call da parte di un programma non può consistere in una semplice chiamata di funzione; infatti essa genera una vera e propria interruzione dell'esecuzione del programma e trasferisce al sistema operativo la richiesta dell'operazione desiderata. L'esecuzione del programma riprende solo dopo che il sistema operativo ha completato l'operazione richiesta.

Lo schema di FIGURA 7 sintetizza l'architettura di un sistema operativo moderno e il contesto hardware e software in cui opera.



Funzionalità di un SO

Le funzionalità principali di un sistema operativo sono:

- Gestione dei processi
- Gestione della Memoria Principale
- Gestione della Memoria Secondaria
- Gestione dell'I/O

- Gestione dei file
- Sistemi di protezione
- Networking (Sistemi Distribuiti)
- Sistema di interpretazione dei comandi

Gestione dei processi

Un processo è un programma in esecuzione che necessita di certe risorse, tra cui tempo di CPU, memoria e dispositivi di I/O, per assolvere il suo compito. Il sistema operativo è responsabile delle seguenti attività, relative alla gestione dei processi:

- creazione e cancellazione dei processi;
- sospensione e riesumazione dei processi;
- fornire meccanismi per sincronizzazione dei processi;
- comunicazione tra processi;
- evitare, prevenire e risolvere le situazioni di stallo (deadlock).

Gestione della Memoria Principale

Il sistema operativo è responsabile delle seguenti attività relative alla gestione della memoria principale:

- Tener traccia di quali parti della memoria sono correntemente utilizzate, e da chi.
- Decidere quale processo caricare in memoria, quando dello spazio si rende disponibile.
- Allocare e deallocare spazio in memoria, su richiesta.

Gestione della Memoria Secondaria

Il sistema operativo è responsabile delle seguenti attività relative alla gestione della memoria secondaria:

- Gestione dello spazio libero
- Allocazione dello spazio
- Schedulazione dei dischi

Gestione dell'I/O

Uno tra gli scopi di un sistema operativo è nascondere all'utente le caratteristiche degli specifici dispositivi di I/O. Per nascondere queste caratteristiche viene realizzato un sottosistema di I/O che è composto da:

- un componente di gestione delle regioni di memoria riservate ai trasferimenti di I/O;
- una interfaccia per i driver dei dispositivi;

- i driver per ogni specifico dispositivo hardware;

Gestione dei file

Un file è una collezione di informazioni correlate, definite dal suo creatore.

Comunemente, i file rappresentano programmi e dati. I file sono generalmente organizzati in direttori (directory), che ne facilitano l'uso. Il sistema operativo è responsabile delle seguenti attività connesse alla gestione dei file:

- Creazione e cancellazione dei file
- Creazione e cancellazione delle directory
- Supporto di funzioni per la manipolazione di file e directory
- Allocazione dei file nella memoria secondaria
- Salvataggio dei dati (backup) su supporti non volatili

Sistemi di protezione

Per protezione si intende un meccanismo per controllare l'accesso di programmi, processi e utenti sia al sistema, sia alle risorse degli utenti. Il meccanismo di protezione deve:

- distinguere tra uso autorizzato e non autorizzato.
- fornire un modo per specificare i controlli da imporre
- forzare gli utenti e i processi a sottostare ai controlli richiesti

Networking (Sistemi Distribuiti)

Un sistema distribuito è una collezione di processori che non condividono memoria o clock, ogni processore ha una memoria propria ed è connesso agli altri processori attraverso una rete di comunicazione. Il sistema operativo deve gestire l'accesso alla rete come una forma di accesso ai file, dove i particolari riguardanti l'interconnessione sono contenuti nel driver del dispositivo di interfaccia della rete stessa.

Sistema di interpretazione dei comandi

Uno dei programmi più importanti di un sistema operativo è l'interprete dei comandi. Si tratta di un'interfaccia tra l'utente e il sistema operativo. Spesso viene chiamata shell. Molti sistemi operativi si differenziano proprio per l'interprete dei comandi. Esistono interpreti amichevoli che rendono il sistema più adatto a certi utenti: ad esempio sistemi basati su mouse, finestre e menu come Windows. In altri di questi interpreti invece i comandi si immettono con la tastiera e appaiono su uno schermo. Il tasto invio segnala la fine di un comando, indicando che il comando è pronto per essere eseguito. Gli interpreti dei comandi dell' MS-DOS e dello UNIX funzionano così.

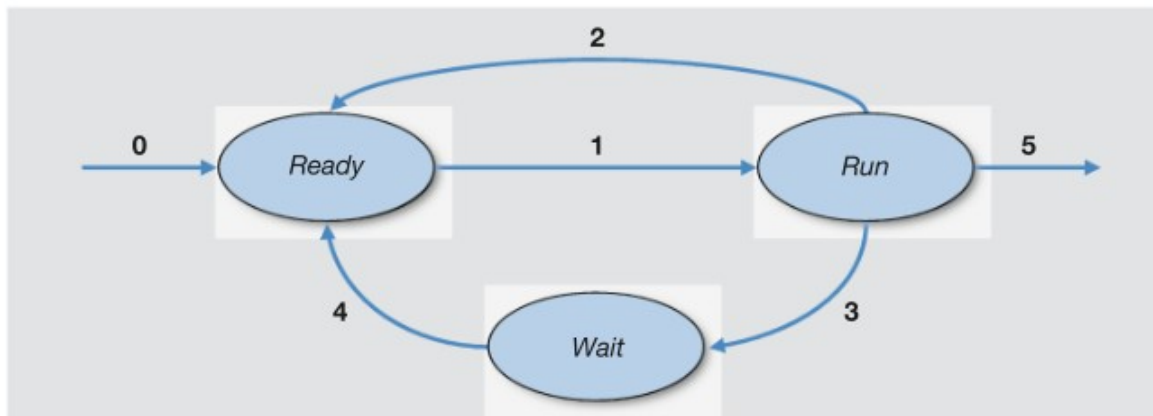
4.2) Gestione delle risorse fisiche e dei programmi da parte del SO

Gestore dei processi

Il gestore dei processi del sistema operativo associa ogni processo attivo a uno dei seguenti 3 stati:

Stato	Descrizione
<i>Ready</i>	Il processo non è attualmente in esecuzione, ma è pronto per essere eseguito in quanto l'unica risorsa mancante è la CPU
<i>Wait</i> ²	Il processo ha invocato un servizio del sistema operativo (per esempio ha iniziato un'operazione di lettura/scrittura da/su un file o è in attesa di una segnalazione da parte di un altro processo, cioè ha richiesto una risorsa non ancora disponibile: non può essere eseguito fino al termine della richiesta)
<i>Run</i>	Il processo è attualmente in esecuzione, cioè dispone di tutte le risorse necessarie al suo avanzamento

Le transizioni da uno stato all'altro avvengono secondo il diagramma di questa figura:



- Il processo viene creato nello stato Ready.
- Il sistema operativo seleziona il processo per l'esecuzione: passa nello stato Run.
- Il sistema operativo interrompe l'esecuzione del processo che ritorna nello stato Ready.
- Nel corso dell'esecuzione il processo richiede un servizio del sistema operativo: viene posto nello stato Wait.
- Il sistema operativo completa la richiesta di un processo che può così riprendere l'esecuzione: dallo stato Wait passa nuovamente allo stato Ready.
- Il processo termina l'esecuzione e il sistema operativo lo distrugge

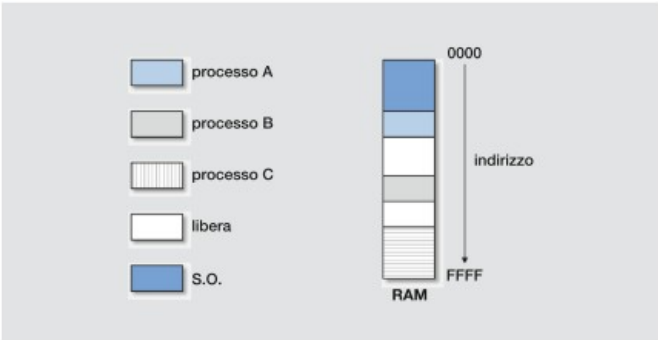
Gestione della memoria:

Tutti i processi in esecuzione necessitano della memoria del computer per:

- le istruzioni che costituiscono il codice dei programmi da cui sono originati;

- i dati su cui operano. Di conseguenza la gestione efficiente di questa risorsa è uno dei compiti fondamentali di ogni sistema operativo.

Un metodo semplicistico per assegnare la memoria principale ai vari processi in esecuzione contemporanea consiste nell’allocare a ogni processo un settore di memoria compreso tra un indirizzo iniziale e uno finale.



Per quanto semplice, questo schema di allocazione della memoria principale ha un grave difetto: quando un processo termina, viene liberato il settore di memoria allocato per la sua esecuzione, ma la memoria libera di cui il sistema operativo dispone per l’allocazione a un nuovo processo rimane «frammentata» in diversi settori non contigui. Non è da escludere il caso che vi sia complessivamente memoria libera sufficiente per l’esecuzione di un nuovo processo, ma che non esista nemmeno un settore sufficientemente grande da contenerle e quindi per eseguirlo effettivamente. Il problema della frammentazione viene risolto dai processori e dai sistemi operativi moderni impiegando la tecnica della paginazione della memoria: la memoria principale viene «vista» dal processore come un array di settori aventi tutti la stessa dimensione predefinita (FIGURA 3). Il modulo di gestione della memoria del sistema operativo assegna a ogni processo in esecuzione un numero di pagine sufficiente per contenere il codice e i dati, ma esse non necessariamente sono contigue (FIGURA 4). In questo caso il gestore della memoria del sistema operativo mantiene una tabella delle pagine come la seguente:

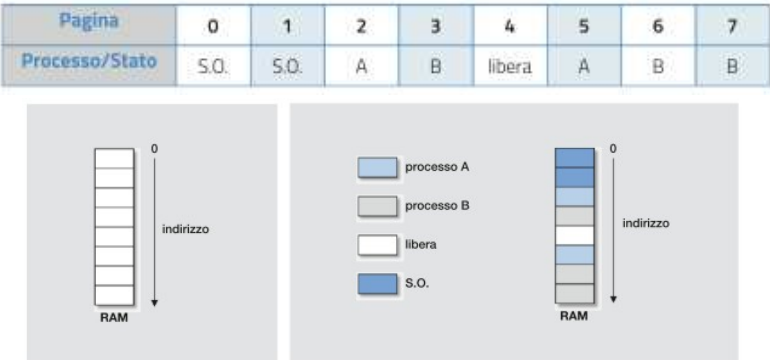


FIGURA 3

FIGURA 4

OSSERVAZIONE Il problema della frammentazione è completamente risolto dalla tecnica della paginazione: al termine di un processo il gestore della memoria rende nuovamente disponibili tutte le pagine in precedenza utilizzate dal processo stesso; al momento della creazione di un

nuovo processo il gestore della memoria dispone per l'assegnazione di tutte le pagine di memoria libere, cioè di tutto lo spazio di memoria effettivamente disponibile.

La gestione dei dispositivi di input/output (I/O)

L'aggiunta di un nuovo dispositivo hardware al computer è normalmente seguita dall'installazione del driver software, cioè dall'integrazione del sistema operativo con un componente specifico – il cui codice viene normalmente eseguito in modalità protetta – per la gestione del dispositivo stesso.

L'installazione di un driver software integra il codice del sistema operativo con un insieme di funzioni di gestione delle interruzioni che il controllore dell'hardware è in grado di generare e con un insieme di funzioni che – tramite l'API del sistema operativo stesso – i programmi applicativi possono invocare (FIGURA 3). La frequenza e la velocità con cui un dispositivo hardware invia i dati di

input o riceve i dati di output sono molto variabili: non è pensabile che un programma applicativo sincronizzi l'invocazione delle funzioni del sistema operativo che gestiscono le operazioni di input/output con il flusso di dati del dispositivo hardware.

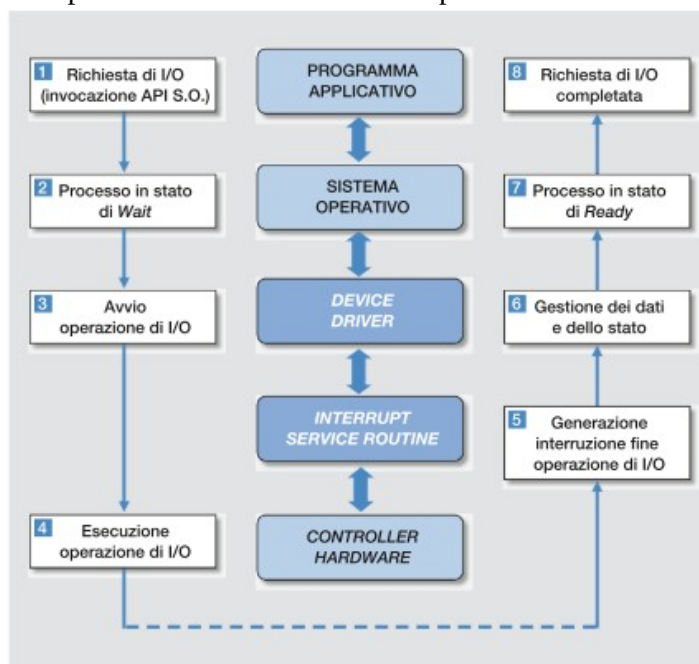
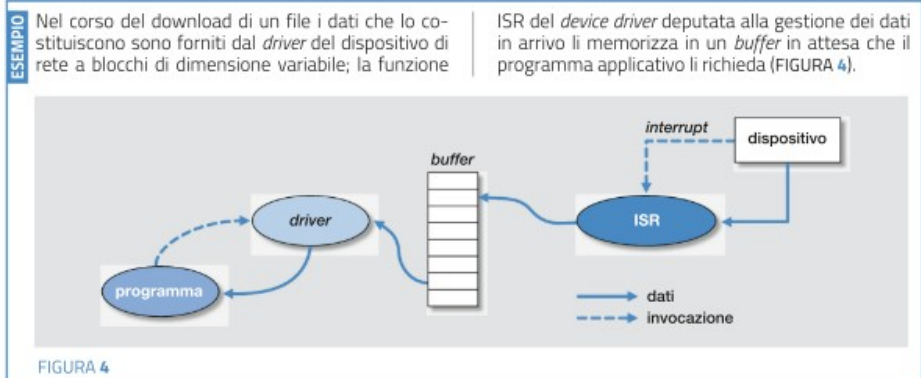


FIGURA 3



Gestione del file-system

2 Organizzazione del *file-system* nell'unità di memorizzazione

Nel caso in cui l'unità di memoria persistente è costituita da un disco magnetico, questo è mantenuto in rapida rotazione intorno al proprio asse ed è possibile posizionare una testina per la lettura/scrittura dei dati (FIGURA 2).

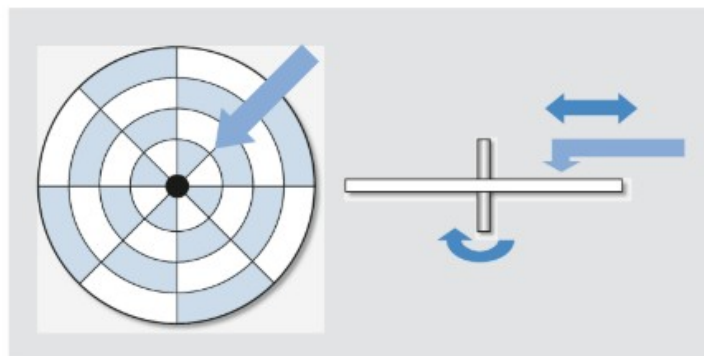


FIGURA 2

OSSERVAZIONE La grande capacità di memorizzazione dei dischi attuali è data dal fatto che sono costituiti da più «superfici» impilate sullo stesso asse, tra cui scorre una testina di lettura/scrittura multipla.

I movimenti meccanici del disco (rotazione continua intorno al proprio asse e traslazione della testina per il suo posizionamento) lo rendono l'unico elemento non completamente elettronico dei computer moderni.

Il posizionamento della testina individua sopra alla superficie del disco una successione di **tracce** circolari concentriche, mentre la rotazione del disco sotto di essa individua una sequenza di **settori** circolari per ciascuna di esse.

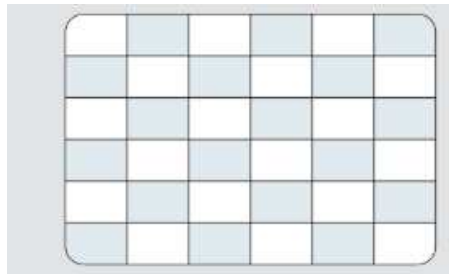
La lettura/scrittura dei dati dal e sul disco avviene per singoli **settori di traccia** individuati dalle rispettive coordinate nella geometria del disco: traccia e settore (nel caso di dischi multipli è necessario includere nella localizzazione anche la superficie).

L'integrità dei dati memorizzati su disco

La correttezza delle operazioni di lettura/scrittura dei dati dalla e sulla unità di memoria persistente è ovviamente della massima importanza. Nonostante l'elevata affidabilità dei dispositivi attuali, insieme ai dati sono sempre memorizzati anche dei codici di controllo calcolati a partire dai dati stessi. In fase di lettura dei dati i codici di controllo vengono nuovamente calcolati e confrontati con quelli memorizzati in fase di scrittura: in caso di difformità si ha un errore che può in questo modo essere rilevato. Volendo garantire un'alta immunità agli errori, è possibile utilizzare due unità di memoria persistente identiche per memorizzare in «parallelo» (*mirroring*) gli stessi dati: in caso di errore in uno dei due dispositivi è possibile utilizzare i dati memorizzati nell'altro.

OSSERVAZIONE Per motivi tecnologici un settore di traccia ha la tipica capacità di 512 byte, ma i gestori del *file-system* dei sistemi operativi considerano più settori consecutivi della stessa traccia come un unico **blocco di dati (cluster)** da leggere e scrivere sempre nella sua interezza. Questa modalità di lettura/scrittura dei dati è finalizzata all'ottimizzazione delle prestazioni di un dispositivo per sua caratteristica estremamente lento: una volta posizionata la testina sulla traccia specificata, si deve infatti attendere che il settore richiesto si trovi sotto di essa; la lettura/scrittura di più settori consecutivi non richiede nuovi spostamenti della testina, o attese della rotazione del disco.

Nel caso invece in cui l'unità di memoria persistente è costituita da un dispositivo SSD internamente organizzato in pagine di dimensione fissa, tipicamente di 4 Kilobyte, vedi figura.



La pagina di un dispositivo SSD è identificata da uno o più valori numerici e rappresenta l'insieme minimo di dati che possono essere letti o scritti in un'unica operazione da parte del computer.

La scrittura deve avvenire in una pagina cancellata (non è infatti possibile sovrascrivere una pagina già scritta) e la cancellazione deve essere effettuata per un intero «blocco» di pagine, formato, per esempio, da 128 pagine. Inoltre il numero di volte che una pagina può essere cancellata è limitato e, per garantire una lunga vita operativa di un dispositivo SSD, la cancellazione delle pagine deve essere effettuata nel tempo in modo uniforme su tutti i blocchi. Dato che la cancellazione di un blocco è un'operazione lenta, le versioni più recenti dei sistemi operativi prevedono una specifica azione – denominata *trim* –con cui il gestore del *file-system* indica al driver dell'unità SSD quali blocchi di pagine può cancellare anticipandone in questo modo la disponibilità per il futuro riuso.

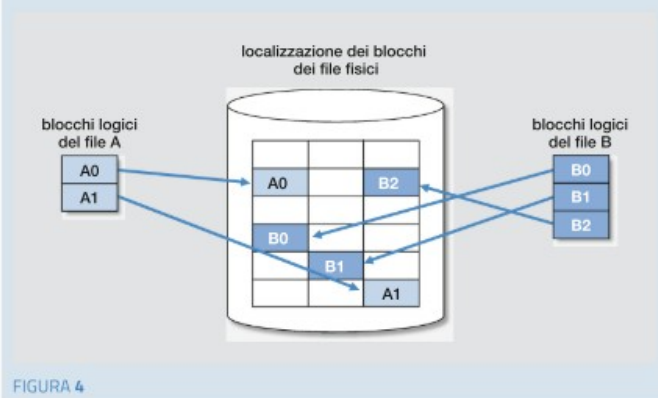
OSSERVAZIONE Per quanto tecnologicamente molto diversa da un dispositivo HDD, un'unità SSD espone al sistema operativo del computer un'interfaccia operativa simile, costituita da un numero elevato di elementi (settori o pagine) leggibili e scrivibili, ciascuno dei quali indirizzabile mediante uno o più indici numerici.

Il gestore del *file-system* di un sistema operativo deve normalmente organizzare centinaia di migliaia di file con dimensioni estremamente variabili (si passa dalle poche centinaia di byte di un file contenente un'icona ai diversi miliardi di byte di un file contenente un video). Oltretutto, nonostante la tipica caratteristica di memorizzazione persistente che il *file-system* realizza, la variazione dei dati è – nel normale uso del computer – estremamente alta: i file sono infatti

continuamente creati ed eliminati e il loro contenuto viene frequentemente aggiornato. L'idea ingenua di memorizzare sequenzialmente in settori consecutivi di tracce contigue del disco, o in pagine consecutive e contigue di un'unità SSD, ogni singolo file non è mai stata presa in considerazione dai progettisti di file-system perché presenta due gravi problemi:

- una volta che lo spazio destinato alla memorizzazione di un file è incastrato tra gli spazi allocati ad altri file, diventa impossibile aumentarne le dimensioni, se non copiandolo preventivamente in una zona libera sufficientemente grande;
- la continua cancellazione dei file causa il problema della frammentazione dello spazio libero: in breve tutto lo spazio disponibile per la memorizzazione di nuovi file sarebbe distribuito in brevissime sequenze di blocchi o pagine contigui in cui non risulterebbe possibile memorizzare file se non di dimensioni piccolissime.

Per questo motivo – in analogia a quanto già visto per la gestione della memoria principale – i singoli file costituiti logicamente da una sequenza di byte sono memorizzati nell'unità di memoria persistente come una **sequenza non contigua di blocchi/pagine**, ciascuno individuato dalla sua coordinata costituita dai numeri di traccia e di settore nel caso di dispositivi HDD, o di pagina nel caso di unità SSD (FIGURA 4).



La memorizzazione in blocchi non contigui dei file risolve il problema della frammentazione dello spazio libero, ma l'elevato numero di blocchi di un'unità di memoria persistente attuale rende non praticabile la memorizzazione di tabelle di allocazione dei blocchi logici dei singoli file nei corrispondenti blocchi fisici del disco, o nelle corrispondenti pagine di un'unità SSD. Una soluzione possibile ed effettivamente implementata in passato da

alcuni sistemi operativi è quella di collegare i blocchi che compongono un file in una catena dove in ogni blocco sono inseriti i numeri di traccia e di settore o il numero di pagina del blocco successivo o, in alternativa, l'indicazione che si tratta dell'ultimo blocco del file: in questo modo, nella tabella che il gestore del file-system deve mantenere per localizzare i singoli file contenuti in una directory, è sufficiente riportare i numeri relativi al primo blocco del file. Questa soluzione risolve il problema della variazione della dimensione di un file perché i blocchi possono essere collocati in modo non contiguo, ma non viene mai adottata.

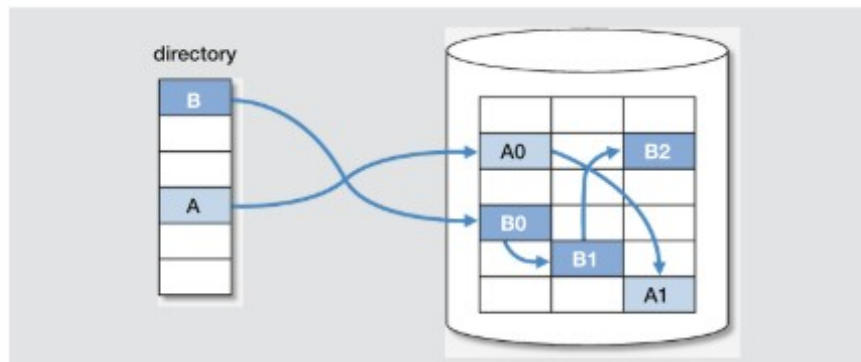


FIGURA 5

La soluzione dei blocchi concatenati ha infatti un grave difetto: dovendo accedere agli ultimi dati di un file – per esempio per estenderne il contenuto – è necessario scorrere sequenzialmente tutti i blocchi precedenti; nel caso di file di dimensioni notevoli questo metodo è altamente inefficiente.

La tecnica adottata dalla maggior parte dei *file-system* è denominata **indicizzazione ad albero** ed è un intelligente compromesso tra l'impraticabilità delle tabelle di indicizzazione di tutti i blocchi e l'inefficienza della concatenazione dei blocchi stessi.

Nella tabella che rappresenta la directory, per ogni file sono mantenuti i numeri (di traccia/settore o di pagina) relativi a un numero limitato di blocchi; se il file è costituito da pochi blocchi di dati, questi sono gli «indici» dei blocchi corrispondenti (FIGURA 6).

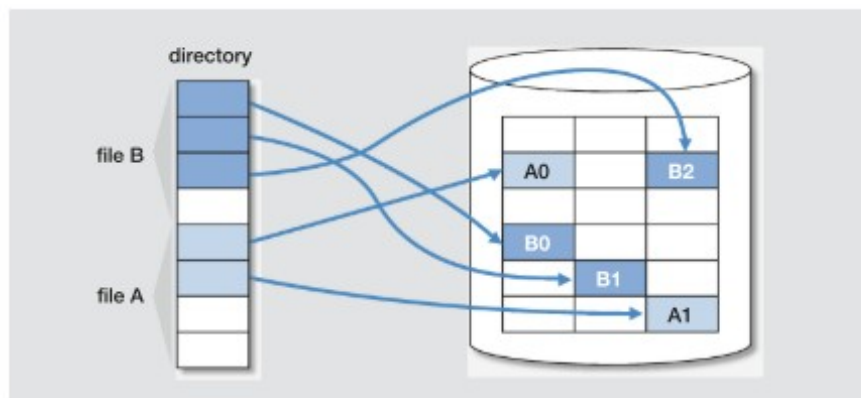


FIGURA 6

Altrimenti, se il file è costituito da più blocchi, gli ultimi blocchi conterranno le coordinate di traccia/settore o il numero di pagina che costituiscono un indice per ulteriori blocchi di dati (FIGURA 7).

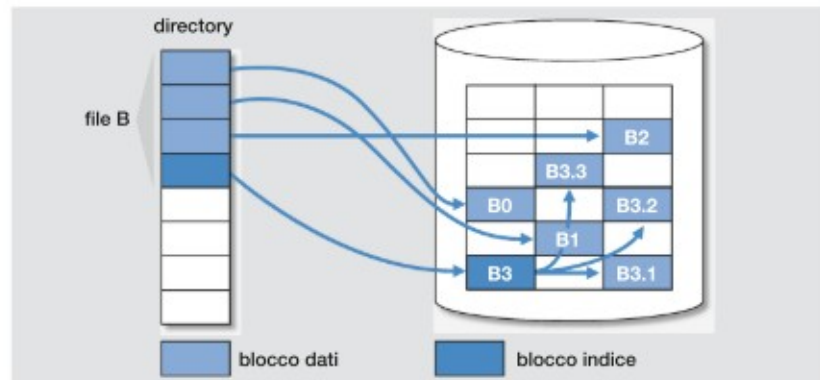


FIGURA 7

All'aumentare del numero di blocchi di dati che costituiscono il file, anche i blocchi di secondo livello (cioè i blocchi che sono riferiti all'interno di altri blocchi e non direttamente alla directory) individuano, anziché direttamente blocchi di dati, blocchi contenenti riferimenti ad altri blocchi, realizzando una struttura ad albero come quella di FIGURA 8.

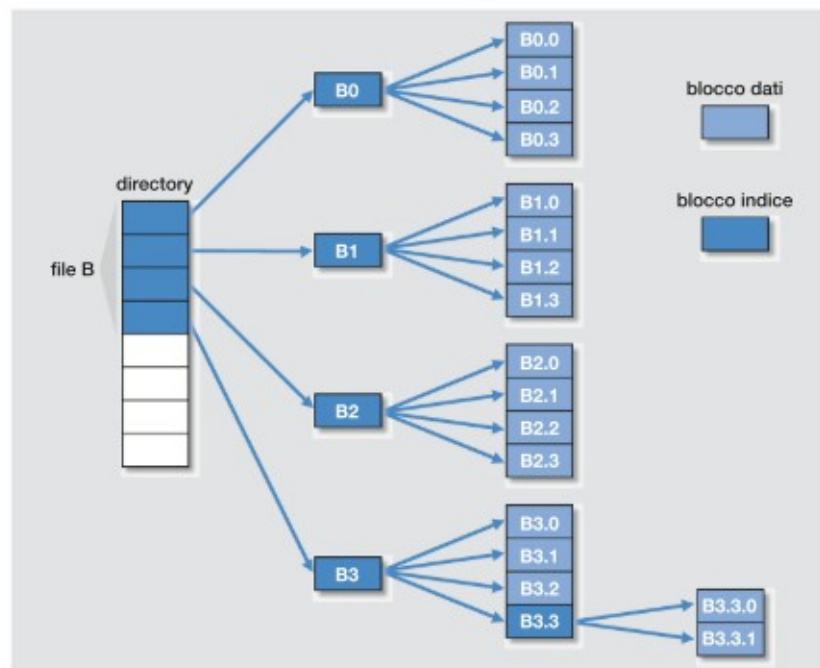


FIGURA 8

L'adozione dello schema di memorizzazione dei file con indicizzazione ad albero dei blocchi ha come conseguenza l'accesso relativamente rapido ai dati contenuti in file di piccola dimensione – in questo caso, infatti, i blocchi sono accessibili direttamente dalla directory, o al massimo utilizzando un blocco indice intermedio – e una minore velocità di reperimento dei dati contenuti in file di grande dimensione, per i quali diviene necessario leggere diversi blocchi indice in cascata prima di ottenere i numeri di traccia/settore o il numero di pagina del cluster richiesto. La limitazione del numero di

livelli di indicizzazione consente comunque di mantenere un'elevata efficienza rispetto alla tecnica della concatenazione dei blocchi.

I sistemi operativi moderni limitano l'uso della concatenazione dei blocchi unicamente alla gestione dello spazio libero: i blocchi non occupati sono concatenati in una lista da dove vengono rimossi quando allocati per l'estensione di file già esistenti, o per la creazione di nuovi file (FIGURA 9).

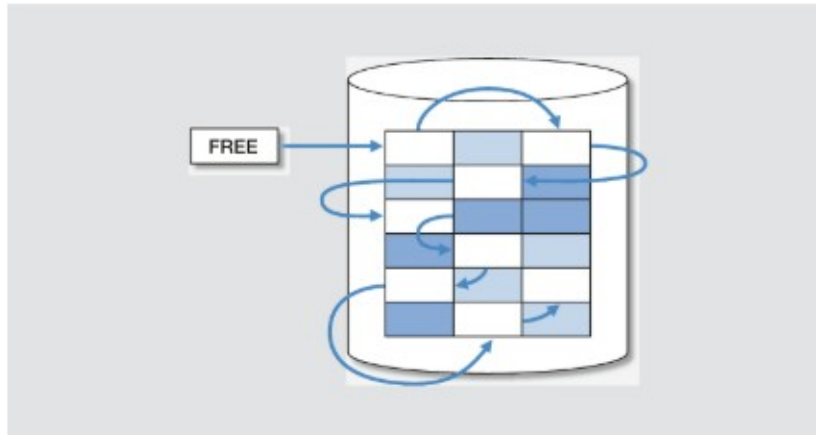


FIGURA 9

Un modo alternativo di gestire lo spazio libero è quello di mantenere una *bitmap* che associa in forma binaria a ogni *cluster* del disco o a una pagina di un'unità SSD il suo stato: libero o occupato.

Questa sequenza di bit viene memorizzata in una posizione fissa.

4.3) Parallelismo e concorrenza

La concorrenza e il parallelismo sono termini correlati ma non uguali, e spesso errati come i termini simili. La differenza cruciale tra concorrenza e parallelismo è che la **concorrenza** riguarda il trattare molte cose contemporaneamente (dà l'illusione della simultaneità) o gestire eventi concomitanti che nascondono essenzialmente latenza. Al contrario, il **parallelismo** consiste nel fare un sacco di cose allo stesso tempo per aumentare la velocità.

I processi di esecuzione paralleli devono essere concomitanti, a meno che non vengano gestiti nello stesso istante, ma i processi in esecuzione simultanea non potrebbero mai essere paralleli perché questi non vengono elaborati nello stesso istante.

Definizione di concorrenza

Atto di eseguire più calcoli nello stesso intervallo di tempo.

- Task multipli (due o più) vengono eseguiti negli stessi intervalli di tempo, senza ordine specifico particolare
- L'esecuzione dei task sembra contemporanea, ma non lo è
- L'effetto è dovuto al time-slicing della CPU, ovvero allo scheduler (context switching) che dedica l'unità di calcolo ai vari task per unità di tempo infinitesimi

Definizione di parallelismo

Atto di eseguire più calcoli simultaneamente:

- Task multipli (o parti diverse dello stesso task) vengono eseguiti (eseguite) contemporaneamente in sistemi multi-processori o multi-core
- Viene permesso da strutture hardware apposite (multi-CPU o multi-core)
- Si ottiene trasformando un flusso di esecuzione sequenziale in uno parallelo

➤ Concorrenza e parallelismo non identificano la stessa cosa

- Concorrenza = espletare azioni nello stesso intervallo di tempo dando l'illusione della simultaneità
- Parallelismo = espletare azioni nello stesso istante
- I termini spesso si usano in maniera intercambiabile

In informatica la concorrenza è una caratteristica dei sistemi di elaborazione nei quali può verificarsi che un insieme di processi o sottoprocessi (thread) computazionali sia in esecuzione nello stesso istante. Tale sistema viene appunto chiamato sistema a concorrenza o sistema concorrente. L'esecuzione contemporanea può condurre a interazione tra processi quando viene coinvolta una risorsa condivisa. Un'importante classe di sistemi informatici nei quali gli aspetti di concorrenza sono fondamentali è quella dei sistemi operativi.

La concorrenza può portare ad una serie di problematiche legate all'utilizzo di una stessa [risorsa](#) condivisa da parte di più processi. Un determinato susseguirsi di eventi può causare condizioni critiche. La [programmazione](#) concorrente sfrutta alcuni principi per fronteggiare e risolvere questo tipo di problematiche.

- **Corse critiche ([Race conditions](#))**

In alcuni sistemi può accadere che i processi in esecuzione condividano una risorsa comune di qualsiasi natura (sia essa un'area di memoria condivisa o una periferica). In particolare se si verifica che il risultato finale dell'esecuzione di più processi dipende dall'ordine in cui essi vengono eseguiti, questa è una *condizione di corsa critica* (race condition). Il risultato dell'esecuzione nel caso di corse critiche è assolutamente imprevedibile.

Il problema delle "corse critiche" può essere evitato impedendo che più di un processo per volta acceda a risorse condivise. Con la **mutua esclusione** si evita che più processi che contendono una risorsa riescano ad accedervi contemporaneamente.

- **Stallo ([Deadlock](#))**

Quando ad un processo viene garantito l'accesso esclusivo (ad esempio tramite una mutua esclusione) ad una risorsa, possono crearsi situazioni di stallo. Formalmente un insieme di processi è in stallo (*deadlock*) quando ogni processo dell'insieme attende un evento che può avvenire soltanto tramite un altro processo dell'insieme. Essendo tutti i processi in attesa, nessuno potrà mai creare l'evento di sblocco protraendo l'attesa all'infinito. Le tecniche per individuare situazioni di stallo prevedono l'analisi di grafi delle risorse allocate oppure mediante la creazione di cosiddette "matrici di allocazione". La risoluzione degli stalli può essere affrontata in vari modi. Concettualmente si possono suddividere in:

- **Risoluzione mediante prerilascio:** viene scelto un processo che detiene una risorsa dall'insieme dei processi in stallo e viene tolto l'accesso esclusivo (prerilascio o preemption) ad una risorsa condivisa (causa di stallo). L'operazione è talvolta difficile, talvolta impossibile e dipende dal tipo di risorsa che il processo stava bloccando.
- **Risoluzione mediante punto di controllo:** vengono creati dei registri (checkpoint) che descrivono lo stato di utilizzo delle risorse condivise. Quando viene rilevato uno stallo si effettua un "ritorno" (*rollback to checkpoint*) alle condizioni precedenti. Anche questa tecnica è di difficile o addirittura impossibile realizzazione poiché il ritorno potrebbe causare perdita o inconsistenza di dati.
- **Risoluzione mediante eliminazione:** viene scelto un processo e viene terminato. Questa tecnica è anch'essa molto complessa e richiede di fare stime e assunzioni sul tipo di processo da eliminare. Inoltre non è garantita l'uscita dalla condizione di stallo per cui potrebbe essere necessario terminare altri processi, situazione che complica ulteriormente la problematica.

È anche possibile effettuare delle stime sulle risorse che verranno impegnate da un processo. Grazie a tali stime, esistono sistemi in cui invece di risolvere le situazioni di stallo risulta più semplice evitarle a priori.

Tutte le tecniche prevedono la costruzione di matrici che tengono traccia dell'utilizzo delle risorse (matrici di traiettoria di risorse) o si basano su algoritmi noti come l'[algoritmo del banchiere](#).

- **[Starvation](#)**

Letteralmente inedia, è un problema in stretta relazione con lo stallo. Quando le politiche di assegnazione delle risorse condivise favoriscono un processo rispetto ad un altro, il processo a cui vengono assegnate in minor misura soffre di *starvation*. Esso è infatti bloccato e non può proseguire sebbene non si trovi in una condizione di stallo. Nei sistemi in cui si accede a risorse condivise è sempre necessario stabilire una politica per le priorità e l'ordine con cui esse vengono ripartite. Sebbene queste politiche possano risultare quanto più eque, esse possono portare a condizioni di starvation.

4.4) Programmi di elaborazione dei linguaggi: interpreti e compilatori

Il compilatore e l'interprete sono entrambi programmi in grado di tradurre le istruzioni di un programma scritto in un linguaggio di programmazione ad alto livello detto [programma sorgente](#) (o [codice sorgente](#)), in istruzioni comprensibili al computer ossia in istruzioni di un programma scritto in linguaggio macchina detto programma oggetto (sequenze di '0' ed '1' comprensibili per l'elaboratore).

Che cos'è un interprete?

Un **interprete**, in [informatica](#) e nella [programmazione](#), è un [programma](#) in grado di [eseguire](#) altri programmi a partire direttamente dal relativo [codice sorgente](#) scritto in un [linguaggio di alto livello](#), senza la previa [compilazione](#) dello stesso ([codice oggetto](#)), eseguendo cioè le [istruzioni](#) nel linguaggio usato traducendole di volta in volta in istruzioni in [linguaggio macchina](#) del [processore](#).

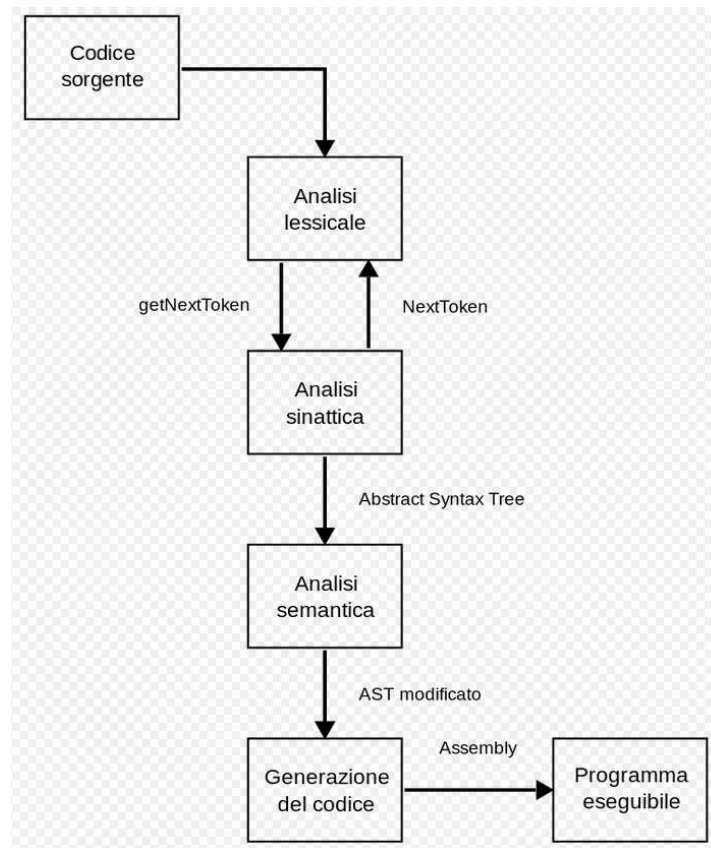
Un interprete è un **programma per computer** che elabora il codice sorgente di un progetto software durante il suo runtime, cioè mentre è in esecuzione, e funge da **interfaccia** tra quel progetto e il processore. Un interprete procede sempre riga per riga di codice, in modo che le singole istruzioni individuali siano lette, analizzate e preparate una dopo l'altra per il processore. Questo principio si applica anche alle istruzioni ricorrenti, che vengono eseguite ogni volta che è il loro turno. Gli interpreti utilizzano le proprie librerie interne per elaborare il codice del software: una volta che una **riga di codice sorgente è stata convertita nelle corrispondenti istruzioni leggibili dalla macchina**, viene inoltrata direttamente al processore.

Il processo di conversione non è completato fino a quando tutto il codice non è stato interpretato ed è interrotto in corso d'opera solo se si verifica un errore durante l'elaborazione, circostanza che semplifica notevolmente la gestione degli errori, poiché la linea di codice problematica viene identificata non appena si verifica l'errore.

N.B. I linguaggi di programmazione più noti che utilizzano un interprete per convertire il codice sorgente in linguaggio macchina sono **BASIC, Perl, Python, Ruby e PHP**. Spesso questi linguaggi vengono definiti "**linguaggi interpretati**".

Che cos'è un compilatore?

Un **compilatore** è un [programma informatico](#) che traduce una serie di [istruzioni](#) scritte in un determinato [linguaggio di programmazione](#) ([codice sorgente](#)) in istruzioni di un altro linguaggio ([codice oggetto](#)): il processo di traduzione si chiama compilazione mentre l'attività inversa - ovvero passare dal codice oggetto al codice sorgente - è chiamata [decompilazione](#) ed è effettuata per mezzo di un [decompilatore](#).



Un compilatore è un **programma per computer** che traduce l'intero codice sorgente di un progetto software nel linguaggio macchina prima che sia eseguito. Solo allora il progetto è eseguito dal processore, che possiede tutte le istruzioni necessarie in linguaggio macchina. Vale a dire che il processore dispone di tutti i componenti necessari per eseguire il rispettivo software, elaborare input e generare output. In molti casi, tuttavia, durante il processo di compilazione avviene una fase intermedia cruciale: prima della traduzione finale nel linguaggio macchina, la maggior parte dei compilatori convertono il codice sorgente in un **codice intermedio** (chiamato anche “codice oggetto”), che è spesso adatto a diverse piattaforme e può essere utilizzato anche da un interprete.

Durante la generazione del codice, i compilatori determinano **quali istruzioni** sono trasmesse al processore e in **quale ordine**. Se le istruzioni non sono connesse tra loro, il processore può elaborare le istruzioni in parallelo.

N.B.

I **linguaggi compilati** più noti sono, tra gli altri, **C**, **C++** e **Pascal**.

Le differenze tra compilatore e interprete mostrano anche i **punti di forza e di debolezza** della scelta di traduzione del codice del programma: i programmi con interpreti possono essere eseguiti immediatamente e sono quindi avviati molto più velocemente. Inoltre, lo sviluppo è

molto più facile che con un'applicazione compilata, perché il **processo di debug** (cioè la correzione degli errori) avviene linea per linea. Nel caso di un software con compilatore, prima di poter gestire gli errori o avviare l'applicazione, il codice deve essere prima completato. Una volta che il programma è in esecuzione, tuttavia, i servizi del compilatore non sono più necessari, mentre un interprete continua a utilizzare la **potenza di calcolo**.

	Vantaggi	Svantaggi
Interprete	Processo di sviluppo semplice (specialmente nel debug)	Processo di traduzione inefficace e bassa velocità di esecuzione
Compilatore	Trasmette al processore il linguaggio macchina completo pronto all'uso ed eseguibile	Eventuali modifiche al codice richiedono una nuova traduzione (gestione degli errori, estensione del software, ecc.)

3.5) Software di utilità, software applicativo e software gestionali

Il software di base comprende anche i software di utilità che estendono le funzionalità del sistema operativo per attività di uso comune nel lavoro con il computer. Esempi di software di utilità sono i programmi per la compressione dei file, i programmi per il backup e i programmi antivirus.

Compressione dei file

Comprimere un file significa ridurre le dimensioni, conservando il contenuto e le caratteristiche originali, in modo da occupare meno spazio sui supporti di memorizzazione, oppure per diminuire i tempi di trasmissione dei file sulle linee di rete di Internet o degli allegati ai messaggi di posta elettronica. Per comprimere i file di una cartella o di un insieme di cartelle si devono utilizzare programmi di compressione, quali WinZip, WinRAR e 7Zip. I file compressi si chiamano spesso anche file zippati. I formati di compressione sono zip, rar oppure 7z

Backup

Il termine backup indica le copie di sicurezza dei file. Il backup è importante perché si può verificare la perdita di file a causa della relativa eliminazione o sostituzione accidentale, dell'attacco di virus, di un errore hardware o software oppure di un errore dell'intero disco rigido. L'operazione di backup deve essere un'attività svolta con regolarità, anche ogni giorno, soprattutto quando riguarda lavori e dati importanti o che comunque richiederebbero tempi lunghi per il recupero. L'operazione a rovescio, cioè il recupero dei dati o documenti dai supporti contenenti le copie di sicurezza, si chiama ripristino, in inglese restore. Le copie di backup vanno effettuate su supporti diversi da quelli contenenti i file originali, perciò si usano unità di memoria di massa rimovibili, quali chiavi USB, CD, DVD o nastri magnetici.

Antivirus

Un programma antivirus è uno strumento software che viene installato nella memoria del computer al momento dell'accensione in modo da controllare il contenuto dei file e delle cartelle sul disco durante il lavoro dell'utente (in background, cioè in sottofondo), impedendo l'ingresso e l'attivazione dei virus. Inoltre il programma consente di fare la scansione di un disco o di un'unità esterna per controllare se contiene virus. Poiché ogni giorno vengono prodotti nuovi virus di diverso tipo, è necessario mantenere aggiornata la tabella delle definizioni dei virus, scaricando dal sito del produttore del programma antivirus i file di aggiornamento e installando poi le nuove definizioni sul proprio computer.

I software applicativi (o applicazioni) sono quelli realizzati dal programmatore utilizzando le prestazioni del sistema operativo e i linguaggi di programmazione. Tra essi si distinguono:

- i programmi di applicazione del computer a problemi di tipo gestionale (contabilità aziendale, fatturazione, magazzino, stipendi); questi programmi vengono di solito forniti all'utente finale da aziende specializzate nella produzione di programmi, dette software house;
- gli strumenti software (tools), detti anche pacchetti o prodotti office, perché vengono forniti già pronti per l'uso all'utente finale e sono orientati alla produttività nella gestione dei più comuni lavori di ufficio:

1. **Elaborazione di testi** (word processing) per scrivere lettere e rapporti usando il computer.
2. **Gestori di archivi elettronici** (database) per creare e gestire con il computer gli schedari presenti nell'ufficio, quali rubriche personali o di gruppo, i listini prezzi, gli indirizzari
3. **Fogli elettronici** (spreadsheet) per usare tabelle di calcolo nelle quali inserire dati e formule di ricalcolo, con eventuali rappresentazioni attraverso grafici statistici.
4. **L'agenda personale per pianificare il tempo** e gli impegni; le diverse agende personali possono essere integrate nell'agenda di gruppo, per trovare le date possibili per una riunione oppure per fissare la prenotazione di una risorsa comune.
5. **La gestione del progetto** (project management) per definire il piano per raggiungere l'obiettivo finale del progetto e la definizione di tutte le risorse necessarie per completare le singole attività.
6. **Il programma per la posta elettronica** (e-mail) con l'invio di messaggi attraverso la rete aziendale o la rete Internet.
7. **Il programma** browser per consultare le pagine Web dei siti Internet.

Capitolo 5

Reti di elaboratori e reti di comunicazione

4.1) Fondamenti di comunicazioni: segnali, canali e metodi di trasmissione:

Con il termine **telecomunicazione** si intende l'invio a distanza di informazioni di varia natura (audio, testi, dati, video, dati rilevati da sensori e oggetti «intelligenti», ecc.) da sorgenti a destinatari.

In termini generali le informazioni sono contenute in messaggi che le sorgenti inviano ai destinatari. Una sorgente emette i propri messaggi utilizzando una certa forma di energia, dando così origine a un segnale informativo. Un segnale è genericamente definibile come una grandezza fisica che, con le sue variazioni, trasporta informazioni.

La natura della grandezza fisica può essere diversa a seconda del tipo di sorgente, per esempio possiamo avere:

- **segnali acustici** quando si utilizza un suono udibile per trasferire informazioni, come un *segnale vocale*, detto anche *fonia*, o un *segnale musicale*, indicato genericamente come *audio*; un segnale acustico si propaga in un mezzo fisico tramite variazioni di pressione, per questo motivo i segnali acustici non si propagano nel vuoto;
- **segnali luminosi** o **ottici** quando si fa variare una luminosità per trasferire informazioni, per esempio immagini sullo schermo di un televisore oppure emissioni e interruzioni di luce associate a un opportuno codice, che ne determina il contenuto informativo, come avviene negli avvisatori ottici;
- **segnali elettrici**, quando si utilizza una variazione di tensione o di corrente per trasferire informazioni; per esempio nel corso di una conversazione telefonica il telefono invia un segnale elettrico sulla linea che lo collega a una centrale telefonica;
- **informazioni codificate**, quando per esempio si inviano testi a distanza o si trasmettono dati tra computer.

Se il segnale deve essere trasferito a distanza è necessario impiegare una forma di energia che sia facilmente generabile, manipolabile e inviabile sugli appositi canali trasmissivi che collegano la sorgente alla destinazione. Tale forma di energia è quella *elettrica ed elettromagnetica*, comprendendo in essa sia quella associata alle onde elettromagnetiche utilizzate nel campo delle comunicazioni radio sia quella ottica.

In un sistema di telecomunicazione si possono avere:

- **segnali elettrici** in tensione e corrente, per la comunicazione attraverso una linea di trasmissione metallica;
- **segnali radio** *costituiti da onde elettromagnetiche*, generate e captate da apposite *antenne*, per la comunicazione senza fili (wireless) attraverso lo spazio;
- **segnali ottici**, considerabili come onde elettromagnetiche ad altissima frequenza in grado di essere guidate da mezzi trasmissivi non metallici denominati *fibre ottiche*.

Sistema di telecomunicazione

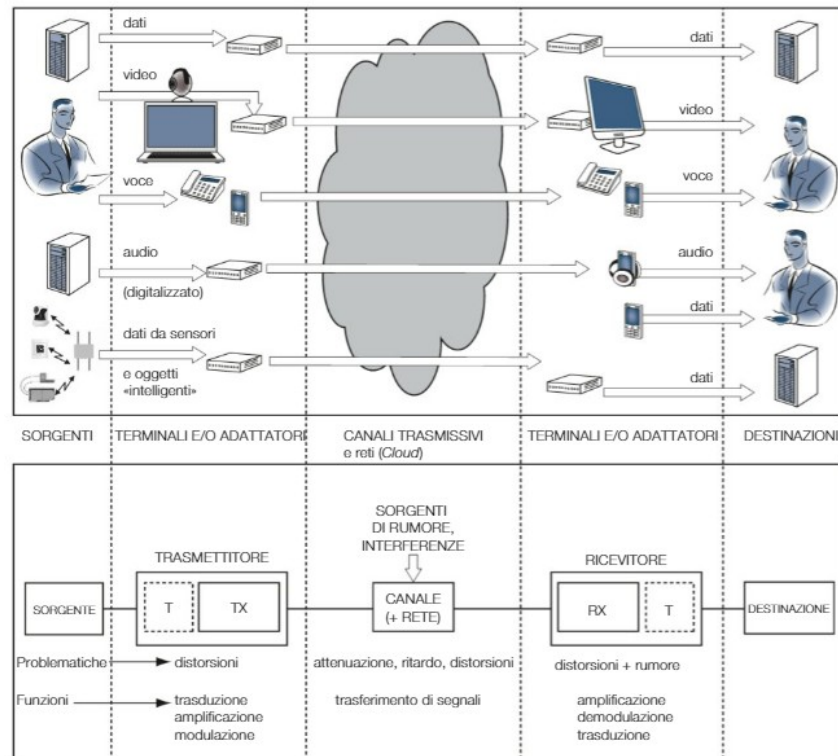
Un sistema di telecomunicazione può essere definito come l'insieme degli apparati e dei mezzi trasmissivi che consentono di trasferire a distanza le informazioni emesse da delle sorgenti verso delle destinazioni remote. Normalmente la comunicazione è bidirezionale per cui accanto a una sorgente vi è anche una destinazione di informazioni. Un sistema di telecomunicazione che interconnette una sorgente con una destinazione attraverso un singolo canale viene definito punto-punto. I sistemi di telecomunicazione che interconnettono molteplici sorgenti e destinazioni, poste in luoghi diversi, comprendono una rete di telecomunicazione.

Per contestualizzare meglio l'esame dei principi e delle tecniche utilizzati nei sistemi di telecomunicazione, è utile schematizzare in termini generali il problema dell'invio a distanza di messaggi informativi tra sorgenti e destinazioni, considerando una sola direzione di un processo di scambio di informazioni, che però nella maggior parte dei casi è bidirezionale.

A livello elementare un sistema di telecomunicazione punto-punto può essere suddiviso in tre blocchi, ciascuno dei quali svolge delle funzioni specifiche, come quelle indicate in figura.

Trasmittitore: riceve i messaggi informativi dalla sorgente e li imprime su un segnale elettrico avente le caratteristiche adatte alla trasmissione sul canale trasmissivo, o canale di comunicazione, utilizzato; può comprendere un trasduttore nel caso in cui la sorgente trasferisca i messaggi informativi con una forma di energia diversa da quella elettrica, come nel

caso di un terminale di tipo telefonico, e/o un adattatore nel caso in cui il terminale non fornisca un segnale adatto a viaggiare sul canale a disposizione dell'utente; per esempio quando si utilizza un computer come



terminale per recuperare informazioni su Internet, per la trasmissione in linea è necessario impiegare un adattatore denominato modem;

- **canale trasmissivo o canale di comunicazione:** comprende i mezzi trasmissivi utilizzati per collegare i luoghi in cui sono posti la sorgente e la destinazione, nonché gli apparati che consentono al segnale emesso dal trasmettitore di giungere al ricevitore con qualità accettabile, limitando quindi il degrado che l'informazione subisce nel transitare attraverso il canale stesso;
- **ricevitore (RX):** preleva il segnale dal canale trasmissivo e lo pone nella forma richiesta dalla destinazione, cercando di limitare il degrado del segnale informativo scambiato.

In una comunicazione bidirezionale l'insieme di un trasmettitore e di un ricevitore viene denominato **ricetrasmittitore** o **transceiver** (TRX).

Se vi è un numero a priori indefinito di sorgenti e di destinazioni, poste in luoghi diversi, allora essi sono posti in comunicazione attraverso una o più **reti di telecomunicazione** interconnesse. In questo caso si attraversano più canali trasmissivi, interconnessi attraverso un'opportuna forma di *commutazione* (PARAGRAFO 2) per andare da una sorgente a una destinazione, FIGURA 2.

L'insieme delle reti interconnesse è spesso rappresentato come una nuvola (**cloud**) per evidenziare il fatto che agli utenti non interessano quali tecnologie di rete sono impiegate, ma quali **servizi** vengono loro offerti e con che prestazioni.

Canali di trasmissione

Un canale di telecomunicazione, nell'ambito delle telecomunicazioni, è in generale una via di comunicazione o propagazione di un segnale, assumendo poi una varietà di significati più o meno specifici in base al contesto, comunque complementari tra loro. Si tratta di un elemento logico e fisico indispensabile all'interno di un sistema di telecomunicazioni cioè nell'apparato deputato alla trasmissione a distanza dell'informazione tra utenti oppure all'interno di apparati elettronici dando vita ai cosiddetti bus di collegamento dati tra sottosistemi elettronici.

Dal punto di vista del trasporto dell'informazione un canale può essere:

- [simplex](#), consente una comunicazione monodirezionale (ad es. [radiodiffusione](#) e [telediffusione](#)).
- [half-duplex](#), consente una comunicazione bidirezionale, ma solo un utente alla volta (ad es. [walkie-talkie](#)).
- [full-duplex](#), consente una comunicazione bidirezionale contemporanea tra due utenti (ad es. [telefono](#)).

Metodi di trasmissione

Una qualsiasi trasmissione può viaggiare a distanza su diversi [mezzi trasmissivi](#) che rappresentano a livello fisico il [canale](#) di comunicazione. Si distinguono trasmissioni:

- [cablate](#) le quali si suddividono a loro volta in [trasmissioni elettriche](#) e [trasmissioni ottiche](#).
- [wireless](#) le quali si suddividono a loro volta in *ottiche* e [radiocomunicazioni](#) che a sua volta si suddividono in comunicazioni terrestri e [comunicazioni satellitari](#)

Mezzi di trasmissione

Un **mezzo trasmissivo** è il supporto fisico tramite il quale un segnale si propaga da un sistema trasmittente a uno ricevente.

La forma di energia più adatta alla trasmissione a distanza dei segnali è quella *elettromagnetica*. In funzione della frequenza dei segnali da trasmettere, della distanza del collegamento, delle interferenze, della tecnologia disponibile, si possono impiegare mezzi trasmissivi diversi.

Una prima distinzione fra essi può essere effettuata distinguendo fra i seguenti.

- *Portanti fisici*

Sono i mezzi trasmissivi costruiti dall'uomo per l'impiego in sistemi di trasmissione cablati (detti *wireline*). Il segnale si propaga al loro interno ed essi lo guidano dal sistema trasmittente fino a quello ricevente; per questo motivo sono anche detti *mezzi trasmissivi con onde guidate*.

I *vantaggi* dei portanti fisici sono: caratteristiche e parametri (attenuazione, banda, ecc.) stabili e determinabili con buona precisione; possibilità di ridurre o eliminare (fibre ottiche) i disturbi elettromagnetici; capacità trasmissiva elevata o elevatissima (fibre ottiche).

Gli *svantaggi* sono: collegamento tra punti fissi; costi che possono essere elevati per via della posa dei cavi; tempi di attivazione relativamente lunghi se i cavi non sono già posati; non sempre è possibile cablare.

- *Portante radio*

La ricetrasmissione delle informazioni avviene via radio, irradiando e captando onde elettromagnetiche tramite antenne trasmittenti e riceventi. È impiegato nei sistemi di trasmissione che non fanno uso di cavi, detti *wireless*. Il portante radio (o portante hertziano) costituisce un *mezzo trasmissivo con onde irradiate*. Il segnale irradiato deve essere di frequenza elevata e quindi sono necessari opportuni metodi di modulazione.

Vantaggi: collegamento in mobilità; velocità di installazione e messa in funzione del sistema di trasmissione; costi complessivi relativamente contenuti; distanze coperte senza ripetitori maggiori dei cavi in rame.

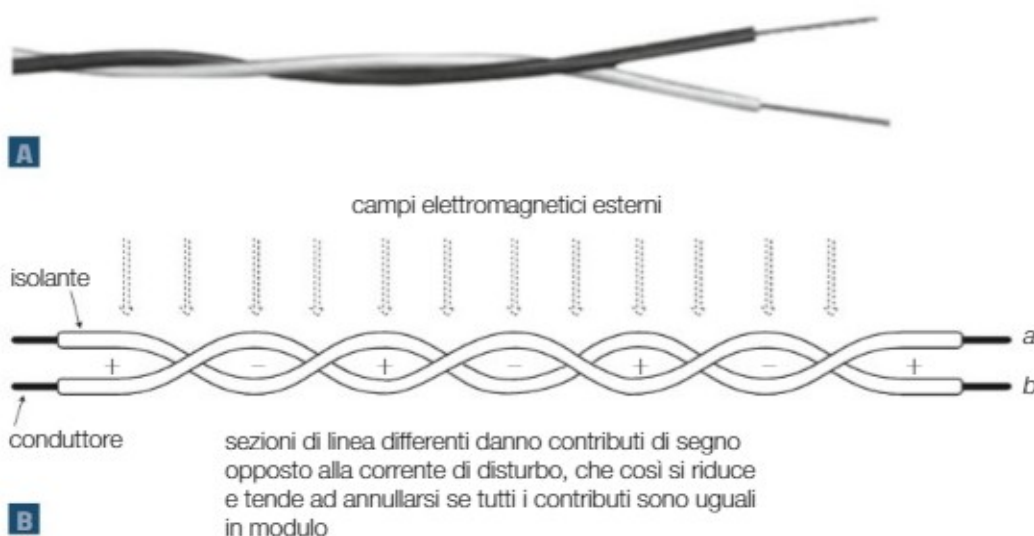
Svantaggi: caratteristiche del mezzo trasmissivo variabili e note solo statisticamente; disturbi e interferenze; tecniche trasmissive sofisticate; può richiedere la visibilità ottica tra le antenne TX ed RX; le frequenze sono risorse limitate e vanno rigidamente assegnate; esposizione ai campi e.m. delle persone che si trovano vicino ad antenne trasmittenti.

Doppino intrecciato

Una **coppia simmetrica intrecciata**, o *Twisted Pair* (TP), è una linea a due fili (denominati filo *a* e filo *b*), costituita da una coppia di conduttori ricoperti da un isolante (polietilene, ecc.) e intrecciati («twistati» dall'inglese *twisted*) (FIGURA 1B). Vi è poi un rivestimento esterno in materiale plastico.

Nell'ambito delle reti telefoniche prende il nome di **doppino telefonico** (FIGURA 1A) una singola coppia simmetrica utilizzata per collegare l'impianto telefonico di un utente alla rete telefonica.

Un cavo a coppie simmetriche può comprendere un certo numero di coppie simmetriche.



Se non si adottano accorgimenti particolari una coppia simmetrica può captare e irradiare molta energia elettromagnetica da/verso l'esterno. Infatti un filo di materiale conduttore immerso in un campo elettromagnetico si comporta come un'antenna ricevente, per cui su di esso viene indotta una corrente di disturbo dal campo e.m. esterno. Viceversa, se il filo è attraversato da una corrente di segnale avente frequenza sufficientemente elevata, esso si comporta come un'antenna trasmittente e quindi è in grado di generare un campo elettromagnetico, che può essere fonte di interferenza per altri sistemi.

Cavi Coassiali

I **cavi coassiali** sono costituiti da due conduttori coassiali, separati da un dielettrico, opportunamente rivestiti da una guaina di materiale plastico⁴ (FIGURA 4).

Questi cavi vengono utilizzati solo in alta frequenza, ove risultano schermati.

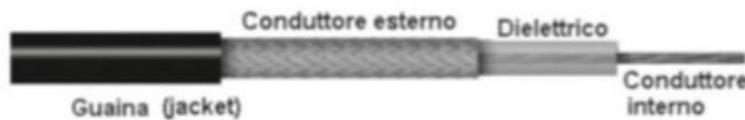


FIGURA 4 Cavo coassiale.

I cavi coassiali sono linee di tipo sbilanciato, in quanto le caratteristiche dei due conduttori, rispetto a terra, sono diverse.

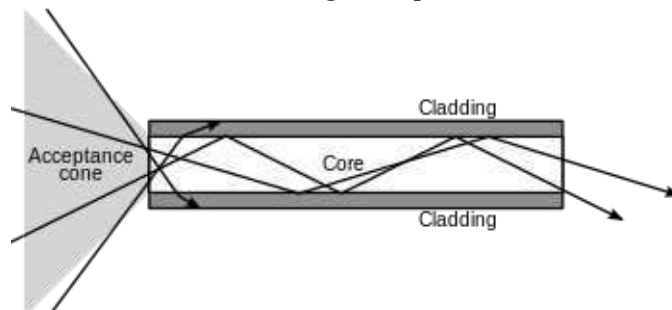
Attualmente l'utilizzo principale dei cavi coassiali è nell'interconnessione di apparati all'interno di un sistema di telecomunicazione, per esempio come cavo di collegamento tra un'antenna e un ricetrasmittitore e come cavi per la strumentazione di laboratorio. Essi sono stati quasi del tutto soppiantati come mezzo trasmissivo nei sistemi di telecomunicazione, dalle coppie simmetriche nei collegamenti a breve distanza e dalle fibre ottiche nei collegamenti a media-lunga distanza.

Nella **TABELLA 2** sono riportate le caratteristiche di alcuni cavi coassiali.

Recentemente sono stati introdotti sul mercato cavi coassiali ad alte prestazioni, in grado di supportare trasmissioni fino a circa 5,8 GHz, come cavi di interconnessione con le antenne in sistemi wireless a larga banda, e fino a 3 GHz in sistemi riceventi satellitari.

Fibre Ottiche

Sono uno dei mezzi più recenti e stanno rivoluzionando il mondo delle telecomunicazioni. Sono fatte di un sottilissimo cilindro centrale in vetro, circondato da uno strato esterno di vetro avente un diverso indice di rifrazione e da una guaina protettiva.



Le fibre ottiche sfruttano il principio della deviazione che un raggio di luce subisce quando attraversa il confine fra due materiali diversi. La deviazione dipende dagli indici di rifrazione dei due materiali. Oltre un certo angolo, il raggio rimane intrappolato all'interno del materiale.

Le fibre ottiche sono di due tipi:

- Multimodali: raggi diversi possono colpire la superficie con diversi angoli, proseguendo quindi con diversi cammini.
- Monomodali: sono così sottili che si comportano come una guida d'onda: la luce avanza in modo rettilineo, senza rimbalzare.

5.2) Il modello ISO/OSI: livelli e primitive di interfaccia

Il modello ISO/OSI è un argomento di fondamentale importanza per chi si affaccia al mondo delle telecomunicazioni, perché fornisce un linguaggio comune per poter sviluppare i protocolli di rete.

Nella comunicazione, un protocollo consiste nella definizione delle regole, affinché due entità possano scambiarsi efficacemente delle informazioni.

L'obiettivo dei protocolli è permettere che un'applicazione nel pc possa trasferire dati con un'altra applicazione attraverso un sistema di interconnessione di reti anche su scala globale.

Proprio per questo è importante il Modello OSI (sviluppato dall'ISO). Tale strumento è fondamentale per capire il ruolo dei protocolli di comunicazione a diversi livelli.

Tali livelli sono numerati dall'1 al 7.

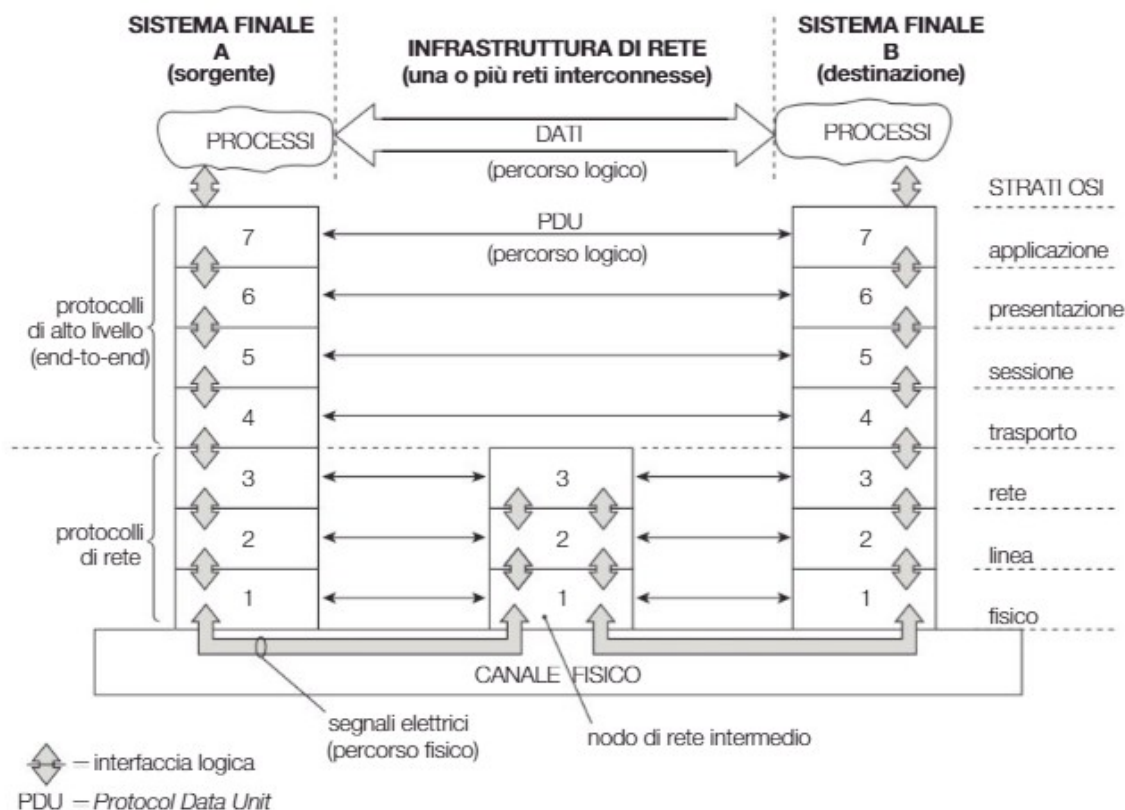
Il Modello di Riferimento OSI definisce 7 strati, suddivisibili in due parti:

- 3 strati inferiori, che affrontano le problematiche della comunicazione attraverso un sistema di reti interconnesse e danno origine ai protocolli di rete. Essi hanno il compito di effettuare il trasferimento dei dati attraverso una o più reti interconnesse e sono implementati sia nei nodi di rete (o sistemi intermedi) sia nei sistemi finali;
- 4 strati superiori, che affrontano le problematiche di alto livello (end-to-end) e danno origine ai protocolli di alto livello, i quali da un punto di vista logico consentono la comunicazione diretta tra le entità di alto livello che risiedono nei sistemi finali.

Nella figura sottostante è mostrato il modello di riferimento OSI, con evidenziati i protocolli di alto livello e quelli di rete. Da un punto di vista logico le entità presenti in uno strato comunicano direttamente tra loro, mediante uno scambio di unità dati, o PDU, definite dal protocollo di quello strato. Il percorso effettivamente compiuto dalle PDU che l'entità di uno strato emette è invece costituito dall'attraversamento di tutti gli strati sottostanti, oltre che dall'invio dei segnali elettrici sul canale fisico.

Si riassumono ora sinteticamente i servizi offerti e le principali funzioni svolte dagli strati OSI.

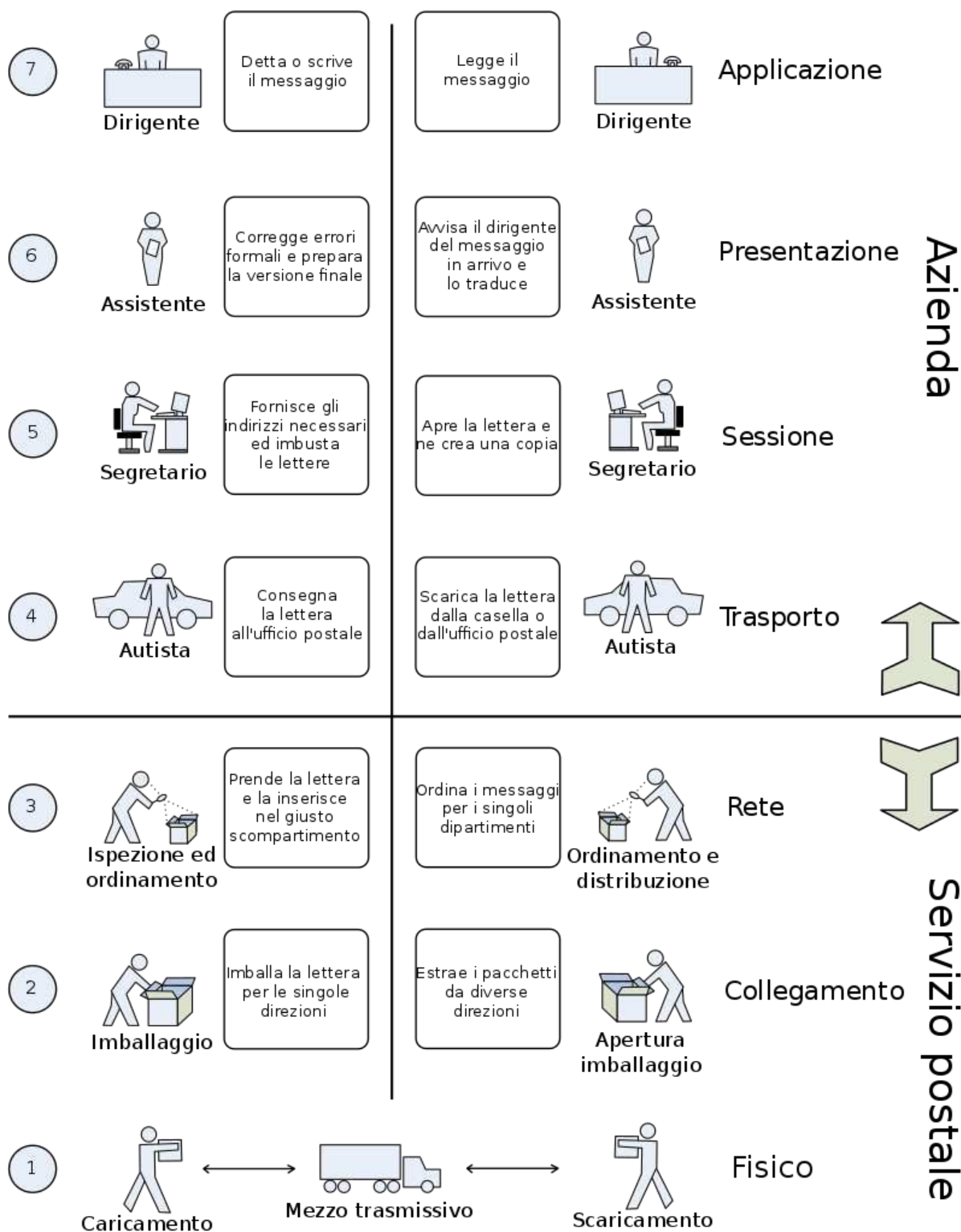
- **Strato fisico (*Physical Layer*) o strato 1:** fornisce un corretto accesso al canale trasmissivo utilizzato; definisce le caratteristiche elettriche,



meccaniche e funzionali dell'interfaccia fisica con cui si accede al canale trasmissivo (livello di segnale, codice di linea, ecc.).

- **Strato di linea (Data Link Layer) o strato 2:** svolge le funzioni che consentono il controllo di uno scambio di dati attraverso un canale di comunicazione fisico (*link*); a questo scopo deve essere adottato un *protocollo di linea* che struttura i bit da trasmettere in (2)-PDU (*Protocol Data Unit dello strato 2*), denominati *frame*, per svolgere funzioni quali: indirizzamento a livello 2 di uno tra più apparati (terminali, schede di rete, ecc.) o entità, rivelazione degli errori (obbligatoria), eventuale correzione degli errori e controllo di flusso (queste funzioni sono opzionali).
- **Strato di rete (Network Layer) o strato 3:** ha la funzione di fornire allo strato superiore un servizio volto a consentire la comunicazione attraverso una rete o un insieme di reti interconnesse. È lo strato che nelle reti a commutazione di pacchetto si occupa del corretto instradamento dei pacchetti attraverso un insieme di reti interconnesse.

- **Strato di trasporto (*Transport Layer*) o strato 4:** fornisce un trasferimento di dati end-to-end efficiente e (opzionalmente) affidabile. Un protocollo dello strato di trasporto può implementare la correzione degli errori e il controllo di flusso end-to-end, per garantire l'affidabilità della comunicazione, oppure può privilegiare la velocità dello scambio di dati, omettendo tali funzioni.
- **Strato di sessione (*Session Layer*) o strato 5:** gestisce e controlla il dialogo tra le entità dello strato superiore, che avviene aprendo e chiudendo *sessioni di comunicazione* (per sessione si intende una connessione logica tra le entità dello strato superiore).
- **Strato di presentazione (*Presentation Layer*) o strato 6:** questo strato può effettuare diverse trasformazioni, quali compressione dati, crittografia e riformattazione, mascherando allo strato superiore eventuali differenze nella rappresentazione e nella sintassi con cui i dati sono trasmessi e permettendo così uno scambio di dati intellegibili.
- **Strato di applicazione (*Application Layer*) o strato 7:** è lo strato che interfaccia i processi utente dei sistemi (cioè il software applicativo), ai quali fornisce tutte le funzioni necessarie per comunicare con i processi residenti in altri sistemi.



Parallelo tra invio di una lettera e modello OSI

Primitive di interfaccia

Le *primitive* sono procedure che permettono di attivare i servizi forniti dal livello inferiore.

In ambito OSI sono previsti quattro tipi di primitive:

- richiesta (*request*)
- indicazione (*indication*)
- risposta (*response*)
- conferma (*confirm*)

Qualunque primitiva OSI rientra in una di queste categorie.

Per descrivere le primitive si utilizzano *diagrammi temporali* simili a quello riportato nella figura 2.10, in cui l'asse dei tempi cresce verso il basso, mentre orizzontalmente corre una coordinata spaziale. La zona centrale rappresenta la rete e tutti i protocolli dal livello (N-1) fino al livello 1 sui due sistemi, mentre a sinistra e a destra sono rappresentate le due (N)entità che vogliono scambiarsi informazioni.

Ad esempio la richiesta di apertura di una connessione viene eseguita nelle fasi seguenti.

- La (N)entità del sistema chiamante A attiva la primitiva (N-1)CONNECTION.request.
- L'(N-1)entità di A, attraverso una serie di fasi intermedie, trasmette l'informazione relativa alla richiesta di apertura di una connessione alla (N-1)entità del sistema ricevente B, che attiverà la primitiva (N-1)CONNECTION.indication.
- La (N)entità del sistema B, se accetta la richiesta di connessione, attiva la primitiva (N-1)-CONNECTION.response.
- La (N-1)entità del sistema B, attraverso una serie di fasi intermedie, trasmette l'informazione relativa alla accettazione della apertura di una connessione alla (N-1)entità del sistema A, che attiva la primitiva (N-1)CONNECTION.confirm che informa l'(N)entità dell'avvenuta connessione.

Da questo momento in poi è aperta una connessione tra le (N)entità del sistema A e del sistema B.

Esistono primitive OSI che permettono di rifiutare la connessione in questa fase iniziale o di chiuderla successivamente e primitive per lo scambio di dati. Le primitive saranno descritte in modo approfondito quando si studieranno i diversi livelli del modello di riferimento OSI, ma il susseguirsi delle primitive è tipicamente analogo a quello appena visto.

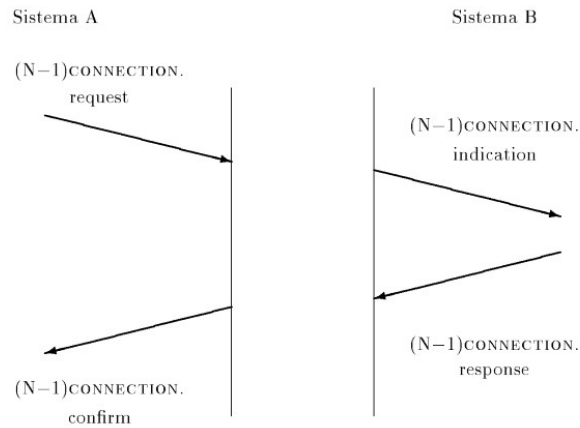


Figura 2.10. Fasi di apertura di connessione

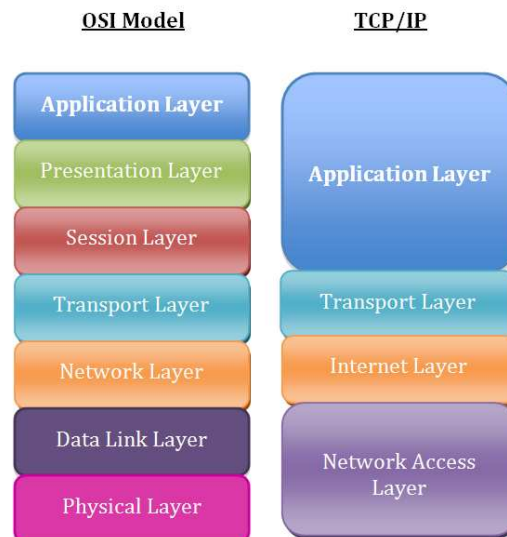
5.3) La suite di protocolli TCP/IPv4

Nel corso dello sviluppo, per integrare via via tipi diversi di reti, si vide la necessità di una nuova architettura, mirata fin dall'inizio a consentire l'interconnessione di molteplici reti.

L'architettura divenne "architettura TCP/IP".

Tale suite può essere descritta per analogia con il **modello OSI**, che descrive i livelli (layer) della pila di protocolli. In una pila di protocolli ogni livello risolve una serie di problemi che riguardano la trasmissione di dati e fornisce un ben definito servizio ai livelli più alti. I livelli più alti sono logicamente più vicini all'utente e funzionano con dati più astratti lasciando ai livelli più bassi il compito di tradurre i dati in forme mediante le quali possono essere fisicamente manipolati e trasmessi infine sul canale di comunicazione.

Per comprendere la struttura della suite TCP/IP, si utilizza una schematizzazione a livelli. Ogni livello esegue una specifica serie di operazioni; ad ogni livello, ci si avvicina sempre più dall'interfaccia utente (quella con cui interagiamo) all'interfaccia di rete. Il messaggio trasmesso è modificato di conseguenza.



A differenza del modello ISO/OSI sono presenti quattro livelli invece di sette, partendo dall'alto:

- **Livello di applicazione**
- **Livello di trasporto**
- **Livello di rete**
- **Livello di accesso alla rete**

Essendo importante la distinzione fra livello di collegamento e livello fisico, è possibile utilizzare un modello ibrido che distingue i due strati e porta il numero di livelli a cinque.

Livello di applicazione

Il primo livello è quello dell'[applicazione](#): esso rappresenta l'[interfaccia](#) con l'utente ed abilita, ad esempio, la consultazione di [pagine web](#), stabilendo e gestendo le [sessioni](#) di lavoro dei processi [client](#) tra il nostro [browser](#) ed il [server](#) web.^[8]

Il protocollo di [trasporto TCP](#) mette in coda i messaggi generati da client e server e li trasmette sotto forma di [pacchetti](#) su una connessione [full-duplex](#); il buon fine della spedizione è attestato da una ricevuta di ritorno o riscontro. Anche questo è un collegamento virtuale tra le due applicazioni, i cui dettagli sono demandati al successivo livello, detto di [trasporto](#).

Livello di trasporto

Quindi, il livello di trasporto offre un servizio al livello delle applicazioni avvalendosi dei servizi del sottostante livello di rete (ed in particolare dell'[Internet Protocol](#)).^[9] Per gestire molteplici processi attivi nel trasferimento dati sul medesimo nodo (o computer), cioè più [sessioni](#) di navigazione attive, il livello di trasporto (TCP o [UDP](#)) utilizza più numeri di [porta](#). TCP nell'invio dei pacchetti usa il meccanismo dello [Sliding Window](#) (o finestra scorrevole). Una serie di pacchetti viene inviata da TCP seguendo delle regole ben precise:

- Ad ogni finestra di pacchetti spedita il trasmettitore fa partire un [timeout](#).
- Il Ricevitore invia per ogni pacchetto ricevuto un [ACK](#) indicando il successivo pacchetto atteso.
- Il trasmettitore considera quindi spediti tutti i pacchetti precedenti.
- Se il timeout scade oppure sono ricevuti 3 ACK duplicati, TCP presume che si sia verificata la perdita di uno o più pacchetti e provvede ad implementare opportune strategie di ritrasmissione dei dati e di [controllo della congestione](#).

Questa è una tecnica molto importante perché fornisce un canale di comunicazione affidabile. Inoltre TCP contiene meccanismi per gestire la congestione ed il controllo di flusso.

Livello di rete

E' il livello che tiene insieme l'intera architettura. Il suo ruolo è permettere ad un host di iniettare pacchetti in una qualunque rete e fare il possibile per farli viaggiare fino alla destinazione. Internet Protocol (IP) è il protocollo di InterNetworking che si occupa di gestire l'indirizzamento

dei nodi e l'instradamento: a ciascun nodo viene infatti assegnato un indirizzo IP che lo identificherà in modo non ambiguo in rete; le funzionalità di instradamento, invece, consentono di selezionare il percorso migliore per veicolare un messaggio verso un dato nodo destinatario, noto che sia il suo indirizzo IP.

Livello di accesso alla rete

Sotto il livello internet, il modello di riferimento TCP/IP specifica solo che ci deve essere un livello di accesso alla rete in grado di spedire i pacchetti IP.[11] Nel modello ISO/OSI questo strato corrisponde ai primi due livelli: il livello di collegamento e il livello fisico.

Al livello di collegamento si decide come fare il trasferimento del messaggio per ogni singolo tratto del percorso: dal computer del web browser al primo router, dal primo router al secondo, dal secondo al terzo e dal terzo al computer del server. Questo è un collegamento virtuale tra due computer (o router) adiacenti. Anche in questo caso le interfacce di comunicazione dei nodi adiacenti saranno individuate per mezzo di un indirizzo univoco, usualmente denominato indirizzo MAC.

Il livello fisico trasmette il messaggio sul canale di comunicazione usualmente sotto forma di segnali elettrici o elettromagnetici, sebbene sia anche possibile utilizzare onde acustiche (come ad esempio nelle reti di sensori sottomarine).

Confronto fra modello ISO/OSI e architettura TCP/IP

Somiglianze:

- Basati entrambi sul concetto di pila di protocolli indipendenti
- Funzionalità simili in entrambi per i vari livelli

Differenze di fondo:

- OSI nasce come modello di riferimento, i protocolli vengono solo successivamente
- TCP/IP nasce coi protocolli, il modello di riferimento viene a posteriori.

I protocolli OSI non sono riusciti ad affermarsi sul mercato per una serie di ragioni:

1. Scelta di tempo non appropriata: la definizione dei protocolli è arrivata troppo tardi, quando cioè quelli TCP/IP si erano già considerevolmente diffusi; le aziende non se la sono sentite di investire risorse nello sviluppo di una ulteriore architettura di rete;
2. L'infelicità di scelte tecnologiche: i sette livelli (e i relativi protocolli) sono stati dettati in realtà dalla architettura SNA dell'IBM, più che da considerazioni di progetto; grande complessità e conseguente difficoltà di implementazione; inutili i livelli session e presentation; la non ottimale attribuzione di funzioni ai vari livelli: alcune funzioni appaiono in molti livelli ad es. controllo errore e flusso in tutti i livelli e altre funzioni mancano del tutto ad es. sicurezza e gestione rete ;

3. l'infelice implementazione: le prime realizzazioni erano lente ed inefficienti, mentre contemporaneamente TCP/IP era molto ben implementato (e per di più gratis!). In effetti i protocolli dell'architettura TCP/IP invece sono stati implementati efficientemente fin dall'inizio, per cui si sono affermati sempre più, e quindi hanno goduto di un crescente supporto che li ha resi ancora migliori.

Ad ogni modo, neanche l'architettura TCP/IP è priva di problemi, infatti :

- l'architettura TCP/IP non ha utilità come modello (non serve ad altro che a descrivere se stessa);
- non c'è una chiara distinzione fra protocolli, servizi e interfacce, il che rende più difficile l'evoluzione dell'architettura;
- alcune scelte di progetto cominciano a pesare oggi (ad es., indirizzi IP a soli 16 bit).

In conclusione: il modello OSI è ottimo come “modello”, mentre i suoi protocolli hanno avuto poco successo; TCP/IP è ottima (per ora e si crede per ancora molti anni con l'introduzione di IPv6) come architettura di rete, ma inutile come modello.

Algoritmo di routing

La funzione principale del livello network è di instradare i pacchetti sulla subnet, tipicamente facendo fare loro molti hop (salti) da un router all'altro.

Un algoritmo di routing è quella parte del software di livello network che decide su quale linea di uscita instradare un pacchetto che è arrivato.

Da un algoritmo di routing desideriamo:

- chiarezza(dove muovere il pacchetto nella giusta direzione)
- semplicità
- robustezza (deve funzionare in caso di cadute di line)
- stabilità
- equità (non deve favorire nessuno)
- ottimalità (deve scegliere la soluzione globalmente migliore).

Gli algoritmi di routing si dividono in due classi principali:

- algoritmi non adattivi: le decisioni di routing sono prese in anticipo, all'avvio della rete, e sono comunicate ai router, che poi si attengono sempre a quelle.
- Algoritmi adattivi: le decisioni di routing sono riformulate molto spesso sulla base del traffico, della topologia della rete.

Protocollo IPv6

Dopo un lungo lavoro, IETF ha approvato il successore di IP versione 4, cioè la versione 6 (**IPv6**, RFC [1883](#), [1884](#), [1885](#), [1886](#), [1887](#)).

I requisiti principali di progetto erano:

- aumentare il numero di indirizzi, ormai quasi esauriti;
- ottenere una maggiore efficienza nei router (tavole più piccole, routing più veloce);
- supportare meglio il traffico real time;
- offrire maggiore sicurezza ai dati riservati.

Le principali differenze rispetto alla versione 4 sono:

- indirizzi di 16 byte, il che significa disporre di 2^{128} indirizzi IP, e cioè $7 \cdot 10^{23}$ indirizzi IP per metro quadro su tutto il nostro pianeta;
- header semplificato: 7 campi contro 13;
- funzioni di autenticazione e privacy, basate su crittografia;
- gestione della qualità di servizio attraverso un campo **flow label**, che consente di istituire delle pseudoconnessioni con caratteristiche negoziate in anticipo.

5.4) Reti locali e reti geografiche

Le reti locali, note col termine LAN (Local Area Network), sono reti private ad alta velocità di piccole estensioni utilizzate per la trasmissione dei dati tra due o più apparati che, generalmente, sono computer localizzati in un'area limitata.

La tecnologia più affermata è la Ethernet nella quale la massima velocità di trasmissione dei dati è di 10Mbit/s, 100Mbit/s nella Fast Ethernet e di 1Gbit/s nella Gigabit Ethernet. Il tasso di errore di trasmissione è assai basso: 10^{-8} - 10^{-9} , ovvero un errore medio di un bit ogni 100 milioni – 1 miliardo di bit trasmessi correttamente.

Si rammenta che l'architettura di una rete locale è costituita da un insieme di nodi collegati tra di loro attraverso i rami.

Il nodo può essere un punto terminale di un ramo, cioè un punto della rete dove risiedono le risorse che si intendono condividere, o un punto di congiunzione in cui confluiscono due o più rami, cioè un apparato di rete come, ad esempio, Hub o Switch.

Il ramo rappresenta il canale di comunicazione che collega due nodi e si avvale di mezzi trasmissivi in cavo o ad onde elettromagnetiche.

I mezzi trasmissivi in cavo utilizzati sono:

- doppino intrecciato; è un mezzo economico; se non è schermato presenta una certa sensibilità ai disturbi elettromagnetici, consente notevoli velocità di trasmissione per distanze fino a qualche centinaio di metri.
- cavo coassiale; per la sua elevata larghezza di banda consente elevate velocità per distanze di diversi chilometri.
- fibra ottica; presenta una totale immunità al rumore elettromagnetico e consente velocità di trasmissione fino a 12Gbit/s avendo una larghezza di banda di oltre 10GHz.
- rete elettrica; si utilizza la normale rete elettrica presente negli edifici per collegare i computer alla rete locale. La distanza massima tra i computer è dell'ordine di 100m. e la velocità di trasmissione è intorno a 10Mbit/s. Sono in atto studi per perfezionare ulteriormente tale tecnica.

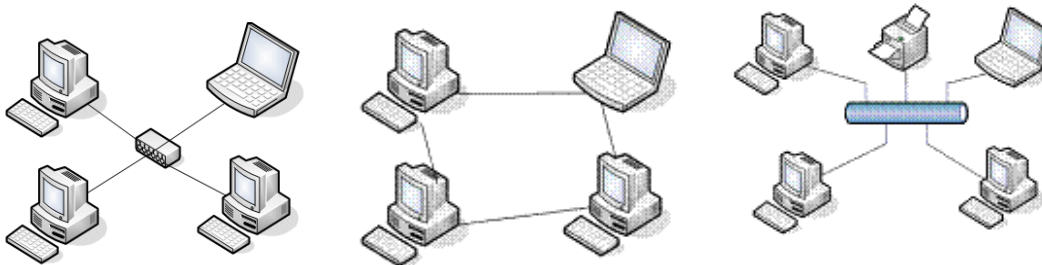
Topologia delle reti locali

Le strutture delle reti sono numerose ma tutte riconducibili a tre tipiche configurazioni fondamentali che sono:

- rete a stella;
- rete ad anello;
- rete a bus.

Per ciascuna di esse è possibile scegliere il mezzo trasmissivo da utilizzare, la tecnica di modulazione, il metodo di accesso alla rete ed il relativo tipo di controllo.

Nella **rete a stella** si individua un nodo centrale a cui sono collegati gli altri nodi attraverso trasmissioni bidirezionali. In fig.1a si mostra una tipica configurazione di rete LAN a stella.



a) Rete a stella

b) Rete ad anello

c) Rete a bus

Questo tipo di collegamento presenta il vantaggio di un basso costo, facilità di espansione ed agevole manutenzione. Il centro stella spesso è un apparato di rete come Hub o Switch. Svantaggio: in caso di avaria del centro stella la rete è paralizzata. Vantaggi: l'apparato di rete rigenera i livelli elettrici del segnale. L'hub riceve i dati da un nodo e li smista su tutti gli altri nodi, intasando così tutte le linee disponibili. Lo switch è più intelligente: "legge" i dati ricevuti, individua il destinatario ed invia i dati solo al nodo del destinatario evitando, così, di intasare tutte le linee come fa l'hub.

Nella rete ad **anello** ogni nodo risulta connesso ai due nodi adiacenti da rami con collegamento punto-punto unidirezionale. La struttura realizza un percorso chiuso come si mostra in fig.1b.

Ciascun nodo deve essere in grado, attraverso il confronto del proprio indirizzo con quello associato al flusso dei dati, di riconoscere se il messaggio che transita in rete è destinato ad esso. In caso contrario il nodo ritrasmette il messaggio al nodo adiacente.

I vantaggi presentati da questo tipo di rete derivano dal fatto che ogni nodo rigenera elettricamente il segnale ricevuto: ciò consente di realizzare reti di lunghezze più elevate rispetto

alle altre; un altro vantaggio consiste nella semplificata procedura di instradamento in quanto il messaggio ricevuto deve essere inviato solo al nodo adiacente.

Fra gli svantaggi annoveriamo la possibilità di paralisi della rete derivante da un guasto ad un nodo o alla linea. Per ovviare a tale inconveniente si impiegano reti ad anello bidirezionali per cui, in caso di guasto sulla rete o in un nodo, i dati possono transitare dal trasmettitore al ricevitore seguendo l'altro percorso.

La relativa lentezza nella trasmissione dipende dal numero di nodi costituenti la rete: infatti ogni nodo deve leggere e poi trasmettere al nodo successivo le informazioni che viaggiano in linea. Per evitare conflitti dovuti alla richiesta simultanea della rete da parte di due o più nodi si utilizza la tecnica Token-ring per il controllo dell'accesso alla rete.

La tecnica consiste nel far circolare nella rete una particolare stringa binaria, denominata token (gettone). Il nodo in attesa di trasmissione che riceve il token ha il consenso all'utilizzo della rete. Alla fine della trasmissione il nodo cede il token a quello successivo. Se il nodo non deve trasmettere alcun dato, legge il token e lo rispedisce immediatamente a quello adiacente.

Una rete a BUS è costituita da un'unica linea multipunto a cui risultano collegati, tramite cavo coassiale, tutti i nodi della rete come si mostra in fig.1c. Le estremità del bus vanno chiuse con resistenze di terminazione.

È una configurazione concettualmente molto semplice, di facile espansione e caratterizzata da buona affidabilità e flessibilità.

I dati trasmessi da un generico nodo vengono immessi sul BUS e letti da tutti gli altri nodi. Essi saranno acquisiti solo dal nodo destinatario il quale confronta il suo indirizzo con quello associato al messaggio in transito.

Un vantaggio consiste nella immunità a situazioni critiche: se un nodo va in avaria la rete continua a funzionare correttamente con la sola esclusione, ovviamente, del nodo guasto.

Uno svantaggio di questa configurazione è la ridotta lunghezza della rete poiché non è possibile, a differenza della rete ad anello, rigenerare il segnale.

Reti geografiche

Le reti geografiche o Wide Area Network, WAN, si estendono a livello di una nazione, di un continente o dell'intero pianeta.

Per introdurci alle principali aspetti tecnologici legati alle reti, prendiamo consideriamo una rete di comunicazione planetaria di uso comune, la rete telefonica. La rete è disegnata in questo modo: esiste un distretto telefonico che contiene una centrale di smistamento, che comunica con le centrali degli altri distretti. Quando facciamo una telefonata, la chiamata viene fatta passare attraverso una o più centrali, fino a raggiungere il numero chiamato: le centrali costruiscono una connessione diretta fra i due telefoni, che dura per tutto il tempo della telefonata. In altre parole, viene creato un circuito virtuale che unisce i due telefoni all'atto della chiamata: per questo, tale tipologia di comunicazione viene anche detta a "commutazione di circuito", in quanto il canale

viene "costruito" per ogni nuova sessione di comunicazione, collegando singoli tratti di linee dedicate come evidenziato in figura .

E' chiaro che la soluzione adottata per le reti telefoniche non si adatta bene alle comunicazioni tra elaboratori: si utilizza in questo caso, infatti, un diverso tipo di commutazione detta "a pacchetto".

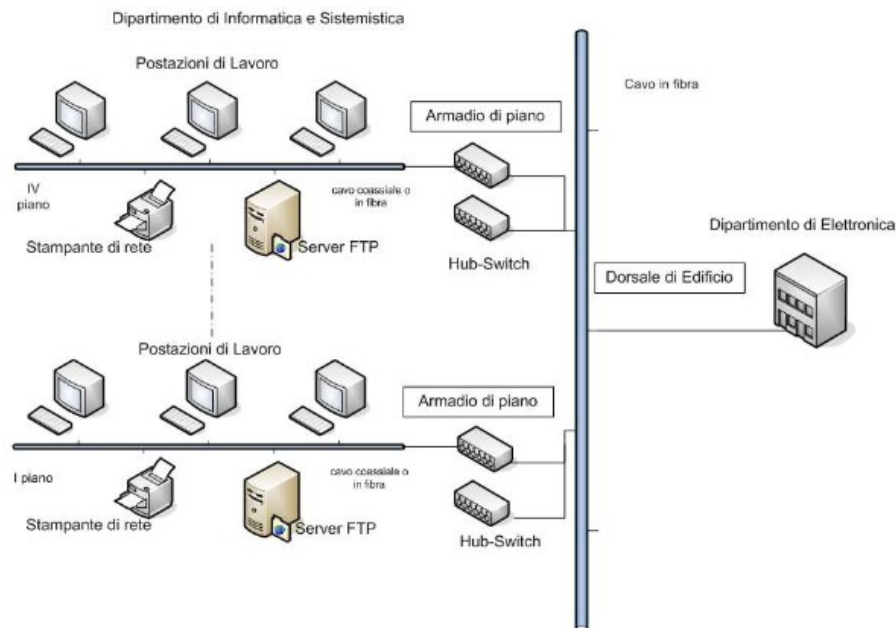


Fig. 7. Componenti di una rete locale

Nella "commutazione a pacchetto" (packet switching), ogni messaggio `e diviso in tanti elementi di dimensione fissa detti pacchetti e opportunamente numerati. Ogni pacchetto contiene altresì l'informazione relativa all'indirizzo del computer destinatario e a quello del mittente e viene trasmesso separatamente, facendo virtualmente una strada diversa dagli altri pacchetti del messaggio per arrivare al destinatario. Si noti, allora, che i pacchetti non arrivano necessariamente nello stesso ordine con cui sono stati inviati: per questo il destinatario deve aspettare la ricezione di tutti i pacchetti per poterli poi ricomporre e ricostruire il messaggio (vedi figura 8). Una rete di comunicazione che `e basata su questo principio viene anche detta punto a punto, o store and forward.

Una WAN, dunque, `e costituita di due componenti distinte:

un **insieme di elaboratori** detti host oppure end system ed una **communication subnet** (o subnet). A sua volta, la subnet consiste di due componenti: linee di trasmissione (dette anche circuiti, canali, trunk) ed elementi di commutazione (switching element), detti anche "sistemi intermedi", oppure "nodi di commutazione" o anche "router".

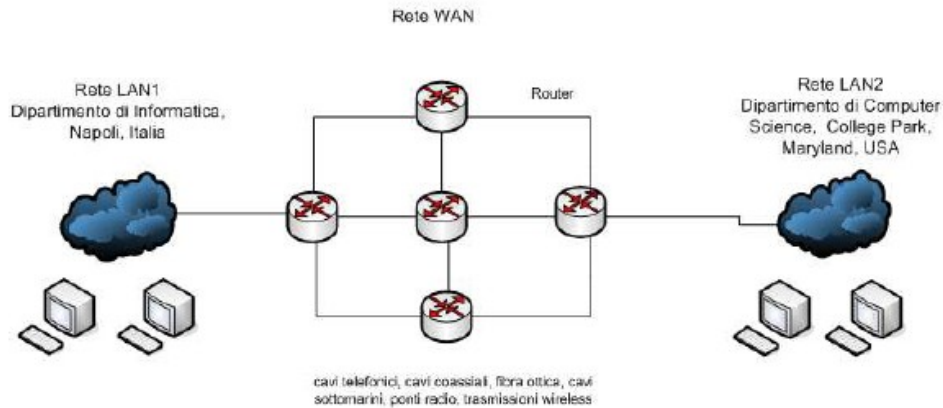


Fig. 9. LAN collegate da una WAN

Sistemi operativi di rete

Il software specializzato chiamato NOS (sistema operativo di rete) fornisce molte funzioni Lan. Come indica il suo nome, un sistema operativo di rete estende la funzione di un sistema operativo (OS) usato per gestire internamente un computer e le sue risorse come dischi e stampanti. Il NOS estende le funzioni di un OS ad un ambiente di rete.

Funzione fornite da un NOS

- **Servizi di stampa:** scaricano le informazioni dalle stazioni sulla rete alle stampanti collegate altrove sulla Lan. E' possibile dirigere le informazioni ad ogni stampante oltre che ad una collegata localmente al computer. E' anche permesso ad una stampante di essere condivisa da molteplici stazioni sulla rete. Di solito i servizi di stampa operano come processi in due fasi. Prima le informazioni sono inviate sulla Lan dal computer client al server di stampa. Qui sono temporaneamente salvate sul disco. Più tardi le informazioni sono inviate alla stampante desiderata. Questo è chiamato Spooling. L'utilizzo dello spooling permette alle stazioni di inviare i dati alle stampanti senza curarsi se la stampante è occupata o se i dati possano mischiarsi con altri di un'altra fonte. Il server di stampa si prende cura del mantenimento dei flussi di stampa separati e dell'invio dei dati quando la stampante è disponibile.
- **Servizi di gestione file e database:** I Servizi di gestione file e database forniscono l'accesso ai file e database memorizzati su altri sistemi sulla Lan. I sistemi dove i file o i database risiedono e che forniscono in tal modo i servizi sono chiamati server di file o di database. Tipicamente i server di file o di database hanno proporzionalmente una grande quantità di memoria disco che può essere condivisa dai sistemi sulla Lan. Questo elimina la necessità per ogni sistema di avere una grande capacità di memoria. Può anche evitare la duplicazione di file e di database. Inoltre per la memorizzazione di file e di database, questi server controllano l'accesso ai dati. Con l'uso di nomi di login e di password, l'amministratore di sistema può limitare l'accesso alle informazioni sul server a solo quelli che lo richiedono.

- **Servizi di accesso remoto:** sistemi operativi di rete forniscono anche dei servizi che permettono ai programmi su un sistema di accedere ai programmi e alle altre risorse di un altro sistema. Questa capacità di accesso remoto permette agli utenti ed alle applicazioni su un sistema di interagire con le applicazioni che risiedono su altri sistemi.
- **Servizi di messaggistica:** I sistemi operativi di rete forniscono spesso servizi di messaggistica di email. Questi servizi permettono agli utenti di sistemi differenti di scambiare messaggi. Gli utenti possono inviare messaggi ad uno o più utenti e possono ricevere messaggi da diverse fonti.
- **Servizi di gestione di rete:** La gestione della rete è un problema importante in una Lan e la maggior parte dei sistemi operativi di rete forniscono il supporto alla gestione di rete. Il controllo statistico delle caratteristiche operative come errori del mezzo trasmissivo, le collisioni e la ritrasmissione sono messe insieme e possono essere segnalate. Inoltre i server di file e database possono anche conservare le statistiche per segnalare gli errori dei dischi, le informazioni di accesso o le violazioni della sicurezza.
- **Servizi di comunicazione:** I Sistemi Operativi di rete possono anche fornire servizi di comunicazione, permettendo la comunicazione al di fuori della Lan. La comunicazione può essere con altre Lan, con sistemi minicomputer o mainframe o attraverso reti Wan. Sono fornite sia la comunicazione da Lan a Lan che da Lan a Wan.
La maggior parte dei sistemi operativi di rete è strutturata sul modello client/server. Uno o più sistemi operano come server degli altri sistemi sulla Lan. I server comprendono i server di stampa, di file, di database e di comunicazione.

I sistemi che richiedono questo tipo di servizi sono chiamati client. I client e i server comunicano attraverso la Lan utilizzando un particolare insieme di protocolli. I protocolli variano al dipendere dal tipo di sistema, NoS, Server e applicazione in uso.

5.5) Normative per il cablaggio strutturato: standard EIA

STANDARD PER CABLAGGIO

Le norme che riguardano il [cablaggio strutturato](#) sono: La norma [EIA/TIA 568A](#) (e 568B dal 2002) è lo standard americano per il cablaggio per telecomunicazioni in edifici commerciali; in questo standard si definisce un generico sistema di cablaggio per le telecomunicazioni che dovrà supportare un ambiente multiprodotto e multifornitore installato in edifici commerciali. Poiché questa norma è stata la prima normativa sul cablaggio strutturato pur essendo americana è stata e continua ad essere utilizzata anche in altri paesi.

La **norma ISO/IEC 1180110**, ultima versione del 2002, e in questo standard si definisce un generico sistema di cablaggio che è indipendente dal tipo di applicazione e compatibile con i componenti di cablaggio di differenti costruttori rispondenti a tale standard.

La **norma EN5017311**, dal 2002 EN50173-1, è lo standard Europeo per un generico cablaggio per telecomunicazioni per uffici, ma applicabile anche nei suoi principi generali agli ambienti industriali e gli edifici residenziali. Questo standard deriva dalla norma ISO/IEC 11801 e ad essa è correlata ma non identica. E' importante sottolineare che, in Italia, si devono inoltre rispettare anche altre normative non comprese nella EN, come: la Sicurezza 626, l'Antincendio, la Privacy, e le Eventuali normative ambientali.

Norma EIA/TIA 568

E' uno standard americano per il cablaggio di edifici commerciali; è stato approvato nel luglio 1991 ed è attualmente quello più applicato e diffuso in tutto il mondo.

Questo standard specifica i requisiti minimi richiesti per il cablaggio di un edificio o un gruppo di edifici facenti parte di uno stesso comprensorio.

I limiti del comprensorio sono i seguenti:

- estensione geografica massimo di 3.000 m;
- superficie massima degli edifici di 1.000.000 m²;
- popolazione massima degli edifici 50.000 persone.

La validità minima di un progetto è di dieci anni e significa che durante questo periodo non deve essere necessario apportare al cablaggio modifiche sostanziali.

Le specifiche dello standard riguardano:

- la topologia;
- gli elementi del cablaggio;
- i mezzi trasmissivi;
- le dorsali;
- il cablaggio orizzontale;
- le norme d'installazione;
- l'identificazione dei cavi;
- la documentazione;
- tipi di connettori e giunzioni.

La topologia

- La topologia del cablaggio è di tipo *stellare gerarchica* e di conseguenza le altre topologie, ad esempio quella a bus e quella ad anello, tipiche di alcuni standard per LAN, devono essere ricondotte ad una topologia stellare.

Elementi del cablaggio

Gli elementi costituenti un sistema di cablaggio sono i seguenti.

MAIN CROSSCONNECT (MC)

Permutatore principale, identifica un locale tecnologico od un armadio di distribuzione, situato

nell'edificio centrale di un comprensorio, da cui vengono distribuiti i cavi di dorsale degli altri edifici. Esso è il primo livello di gerarchia del cablaggio (centro stella di comprensorio).

INTERMEDIATE CROSSCONNECT (IC)

Permutatore intermedio, identifica un locale tecnologico od un armadio di distribuzione di un edificio facente parte di un comprensorio, da cui vengono distribuiti i cavi di dorsale di edificio ai vari piani. Esso è il secondo livello di gerarchia del cablaggio (centro stella di edificio).

TELECOMMUNICATION CLOSET (TC)

Identifica l'armadio di piano da cui vengono distribuiti i cavi che raggiungono l'utente. Esso è il terzo livello di gerarchia del cablaggio (centro stella di piano).

INTERBUILDING BACKBONE (DORSALE DI COMPRESORIO)

E' la dorsale di interconnessione tra l'edificio centro stella di comprensorio e un altro edificio. Essa parte dal Main Crossconnect e termina su un Intermediate Crossconnect.

INTRABUILDING BACKBONE (DORSALE DI EDIFICIO)

E' la dorsale di interconnessione tra il locale tecnologico di edificio e l'armadio di piano.

EQUIPMENT ROOM (ER)

E' un locale tecnologico che può contenere degli apparati passivi, quali pannelli di permutazione, scaricatori di tensione e può ospitare apparati attivi quali il centralino telefonico.

INTERBUILDING ENTRANCE FACILITY (EF)

Identifica un insieme di infrastrutture e di componenti passivi utilizzati per l'ingresso delle dorsali di comprensorio nell'edificio. Nell' EF è richiesto l'utilizzo di protezioni elettriche per i cavi in rame e deve essere particolarmente curato l'aspetto della massa a terra dei vari componenti.

TRANSITION POINT (TP)

E' un punto di transizione del cablaggio orizzontale dove un cavetto rotondo di tipo ritorto viene connesso, tramite un giunto meccanico, ad un cavo piatto che è normalmente pre-installato.

WORK AREA (WA)

Identifica il posto di lavoro dell'utente.

PRIVATE BRANCH EXCHANGE (PBX)

E' il centralino telefonico.

PATCH PANEL

E' il pannello di permutazione per i mezzi trasmissivi che può assumere due forme:

- per cavi in rame, può contenere uno o più blocchi di terminazione;
- per fibre ottiche, può contenere una serie di connettori passanti, chiamati barrel o bussole, che servono a permutare le fibre tra pannelli diversi oppure tra un pannello ed un apparato attivo.

PATCH CORD

E' un cavetto di permutazione per cavi in rame o per fibre ottiche. Quando è per fibre ottiche assume il nome di bretella ottica.

CROSS CONNECT (PERMUTATORE)

E' composto da almeno due blocchi di terminazione, di cui uno per i cavi entranti e uno per i cavi uscenti.

TELECOMMUNICATION OUTLET (TO)

E' la presa utente che può contenere due o più connettori.

ADAPTER

E' un adattatore per il cablaggio e lo standard prevede che sia installato esternamente alla presa utente.

I mezzi trasmissivi

I mezzi trasmissivi ammessi sono i seguenti:

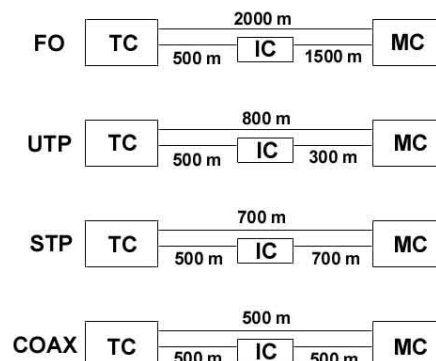
- cavi coassiali da 50 ohm;
- fibre ottiche multimodali 62.5/125 μm ;
- cavi UTP a 4 coppie;
- cavi UTP multicoppia;
- cavi STP a 150 ohm.

Le dorsali

Le dorsali sono gli elementi portanti del cablaggio e possono interconnettere, con topologia stellare gerarchica:

- edifici diversi con l'edificio centro-stella del comprensorio (interbuilding backbone);
- armadi di piano diversi con l'armadio di edificio (intrabuilding backbone).

Le distanze ammesse per le dorsali variano a seconda dei mezzi trasmissivi utilizzati e di ciò che essi interconnettono:

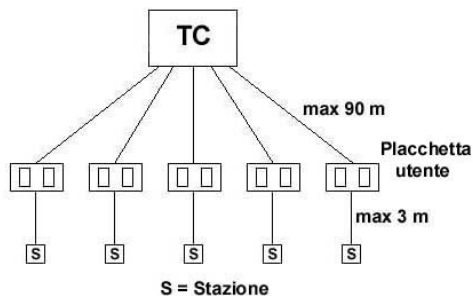


Il cablaggio orizzontale

Interconnette i vari posti di lavoro all'armadio di piano e deve essere progettato per fornire almeno i seguenti servizi:

- trasporto di fonia;
- trasmissione dati in modalità seriale;
- trasporto dati per le reti locali;
- trasporto di segnali per il controllo di dispositivi all'interno dell'edificio.

La topologia è di tipo stellare a partire dall'armadio di piano. Le distanze ammesse per i cavi di distribuzione e i cavetti di permutazione sono indicati in figura.



I cavi ammessi sono i seguenti:

- cavo UTP a 4 coppie con impedenza da 100 ohm;
- cavo STP a 2 coppie con impedenza da 150 ohm;
- cavo coassiale da 50 ohm, tipo Ethernet sottile (thin), intestato alle due estremità con appositi connettori BNC;
- fibra ottica multimodale 62.5/125 μm .

La placchetta o presa a muro, relativa al singolo posto di lavoro, deve almeno contenere due cavi, di cui almeno uno deve essere di tipo UTP a 4 coppie di categoria 3 o superiore. Il cavo UTP va intestato su una presa RJ45. Il secondo cavo può essere uno qualunque dei cavi ammessi per il cablaggio orizzontale sopra elencati, compreso un secondo cavo UTP, che è attualmente la soluzione più adatta.

Le norme d'installazione

Il cablaggio, a seconda dei componenti utilizzati e della qualità dell'installazione, potrà essere considerato di categoria 3, 4 o 5.

Norme per l'installazione dei cavi UTP

- la massima tensione di tiro applicabile sui cavi è di 11.3 Kg. Se si supera tale valore viene compromessa la corretta geometria delle coppie e si ha un degrado delle caratteristiche elettriche.
- il raggio di curvatura minimo ammesso varia a seconda della categoria del cablaggio. Il valore richiesto è di 25.4 mm per i cablaggi di categoria 3, ed otto volte il diametro esterno del cavo (50.8 mm) per i cablaggi di categoria 4 e 5.
- la parte del cavo non ritorta sulla terminazione non deve superare i 25 mm per i cablaggi di categoria 4, e 13 mm per i cablaggi in categoria 5.

Norme per l'installazione sotto-moquette

- il cablaggio non deve essere effettuato in locali umidi o soggetti al rovesciamento di solventi;
- si raccomanda che la pavimentazione sia realizzata con dei moduli quadrati per facilitare l'accesso al cablaggio;
- i cavi di telecomunicazione di tipo under-carpet possono incrociare i cavi di potenza a patto che questi non siano del tipo under-carpet;
- la distanza minima tra i cavi di telecomunicazione e quelli di potenza, quando viaggiano paralleli tra loro, è di 152 mm.

La messa a terra va effettuata sui seguenti tipi di cavi:

- cavi di tipo schermato;
- cavi in fibra ottica ove sia presente una protezione meccanica di tipo metallico.

Identificazione dei cavi

Lo standard specifica che i cavi di dorsale devono avere un numero unico che deve contenere almeno due campi indicanti:

- l'identificativo del cavo;
- il numero di coppie, nel caso di cavo multicoppie, o il numero di fibre nel caso di cavo multifibra.

Ogni posto di lavoro ed il relativo cavo sono identificati con una targhetta, composta normalmente da 8-10 caratteri, che può contenere numeri o lettere alfabetiche. La numerazione deve contenere:

- il riferimento al piano dell'edificio dove è situato il posto di lavoro;
- il riferimento all'armadio di piano a cui il posto di lavoro è stato collegato;
- un campo di tre caratteri che identifica il posto di lavoro stesso.

Normalmente gli armadi di piano vengono identificati con delle lettere alfabetiche.

ESEMPIO

Come si numera il posto di lavoro ed il relativo cavo con la targhetta "PD02109A"

- PD indica "Palazzo Dante"
- 02 indica il piano del posto di lavoro
- 109 indica il posto di lavoro
- A indica l'armadio di piano a cui il posto di lavoro è stato collegato.

Documentazione

Deve comprendere:

- il disegno logico dell'intero comprensorio o del singolo edificio;
- una tabella per indicare le dorsali;
- una tabella di armadio che indichi le connessioni tra l'armadio di piano e i posti di lavoro.

La tabella di documentazione delle dorsali deve contenere:

- gli identificativi di tutti i cavi ed il loro corrispondente numero di coppie o fibre;
- la localizzazione e l'identificativo dei due armadi a cui ogni cavo è attestato.

Ogni armadio di piano deve contenere la documentazione consistente in una tabella delle permutazioni, tramite cui è possibile ricostruire il percorso del cavo che, partendo da una certa posizione del permutatore, raggiunge il posto di lavoro. Vanno inoltre indicate le coppie attive ed il loro utilizzo.

Tipi di connettori e giunzioni

I connettori ammessi sono i seguenti:

- connettore RJ45 per i cavi UTP a 4 coppie;
- connettore ermafrodita per i cavi STP a 2coppie;
- connettore "N" per i cavi coassiali di dorsale;
- connettore "BNC" per i cavi coassiali di distribuzione orizzontale;
- connettore per fibra ottica in grado di sopportare almeno 200 cicli di estrazione/inserzione senza introdurre attenuazioni superiori a 1 dB; normalmente quello utilizzato è di tipo "ST";
- gli splices che servono per giuntare la fibra ottica; l'attenuazione massima ammessa sulla giunzione è di 0.3 dB.

Capitolo 5

Sicurezza dei sistemi informatici e delle reti

6.1) Metodologie e tecnologie per la sicurezza informatica

I dati contenuti nei nostri computer spesso sono importanti per lo studio e il lavoro, oppure possono avere un valore affettivo, come nel caso delle foto o dei video. Ma oltre a essere significativi per noi, questi dati possono interessare anche a persone malintenzionate.

Collegarci a Internet con il nostro computer può esporci ad alcuni rischi, come il «furto di identità»: se un malfattore riesce a carpire dati sensibili, per esempio le nostre password, può derubarci facendo acquisti online con i nostri conti bancari, oppure può compiere a nostro nome azioni illecite con la posta elettronica o sulle reti sociali.

Proteggere i dati personali è perciò un'esigenza prioritaria per chiunque usi computer collegati alla Rete, o anche dispositivi mobili come tablet e smartphone.

In ambito IT garantire la sicurezza di un sistema informativo significa garantirne:

- **riservatezza delle informazioni** (*confidentiality*): solo chi è autorizzato deve poter accedere all'informazione;
- **integrità delle informazioni** (*integrity*): le informazioni non devono essere danneggiate o modificate per caso o con intenzioni malevole;
- **disponibilità delle informazioni** (*availability*): le informazioni devono essere sempre disponibili a chi è autorizzato ad utilizzarle

Occuparsi di sicurezza informatica significa quindi predisporre **politiche, processi, controlli e contromisure informatiche** in grado di **contrastare le minacce** che rischiano di compromettere la riservatezza, l'integrità e la disponibilità delle informazioni

Nell'ambito della sicurezza informatica l'oggetto più prezioso da proteggere è l'**informazione**, il **dato**, o il **servizio di business** erogato con il supporto del sistema informatico (es.: il valore di un computer non è dato solo dal suo prezzo di acquisto, ma soprattutto dall'importanza del dato che gestisce e del servizio che eroga).



Gesture

Con il termine **gesture** si intendono tutti quei movimenti o gesti che possono essere effettuati sullo schermo mediante il tap, o tocco delle dita. Le applicazioni di blocco/sblocco tramite gesture prevedono la conoscenza della sequenza gestuale memorizzata nel dispositivo.

Q PER SAPERNE DI PIÙ

REGOLE PER LA SICUREZZA

1 Proteggere il dispositivo che accede a Internet

Innanzitutto è bene aggiornare costantemente il browser, le versioni più recenti infatti sono protette verso i nuovi tentativi di intrusione. È sempre necessario dotarsi di **firewall**, **antivirus** e **antispam**. Per una maggiore sicurezza impostiamo anche lo **screen saver** dello schermo con password o **gesture** di sblocco.

2 Proteggere le password

Le password devono sempre essere modificate, almeno ogni **tre mesi** e devono essere custodite gelosamente. È buona regola non custodire tutte le password nello stesso luogo, e comunque è bene scriverle su un supporto cartaceo (diario, agenda, quaderno ecc.) e non su supporti digitali che possono essere violati o che si possono deteriorare o smagnetizzare. Inoltre utilizziamo sempre password diverse per ciascun account.

3 Utilizzare reti sicure

Quando navighiamo in Internet accedendo da una rete poco sicura, come per esempio la **Wi-Fi** gratuita di un locale pubblico, evitiamo di digitare **password** o informazioni **personali**, ricordiamo sempre che le informazioni trasmesse nella rete possono essere facilmente intercettate da malintenzionati. Nella rete **Wi-Fi** casalinga dobbiamo provvedere a proteggere i dispositivi con **password crittografata** secondo lo standard **WPA2 (Wi-Fi Protected Standard)**. Tutti i router **Wi-Fi** più recenti possiedono un menu che consente di impostare una password con questo metodo di criptaggio.

4 Proteggere le informazioni personali

Verifichiamo sempre, prima di inserire dati personali nei moduli delle pagine Web, la presenza di indicatori che ne attestino la sicurezza, come per esempio il protocollo **https** e il relativo simbolo del lucchetto chiuso posto accanto, oppure l'uso di **certificati digitali**. Facciamo poi molta attenzione nel fornire informazioni personali, come per esempio il proprio indirizzo, il numero di telefono, o il numero di carta di credito.

5 Evitare le frodi e le truffe

La **frode informatica** consiste sia nel creare anomalie di funzionamento che nell'intervenire su programmi, dati, notizie e collegamenti in maniera illegittima. La **truffa** è invece una frode che viene perpetrata con danni al patrimonio dell'interessato, che è stato raggirato e indotto all'errore. Dobbiamo sempre essere consapevoli che il clic su di un collegamento ipertestuale oppure sull'allegato di un messaggio di posta, possono aprire pagine potenzialmente pericolose, oppure aprire la strada a **malware** in grado di nuocere al dispositivo o addirittura rubare le nostre informazioni personali. Facendo acquisti online, verifichiamo sempre la reputazione digitale del venditore online.

6 Prevenire il furto di identità

Per evitare il **furto di identità** è bene non inviare mai la nostra **password** o i nostri **dati personali** tramite posta elettronica e soprattutto non condividerla con altre persone. Attraverso il **pharming** i ladri d'identità utilizzano siti Web fasulli sostituendosi a organizzazioni legittime: facendo leva sulla fiducia inducono l'utente sprovveduto a divulgare i propri dati personali. Attraverso gli antivirus possiamo bloccare software malvagi chiamati **keystroke logger**, che acquisiscono furtivamente informazioni dell'account mentre vengono digitati sul dispositivo.

7 Utilizzare i social network con prudenza e rispetto

Nei social network limitiamo la visione dei nostri dati personali soltanto agli account dei quali siamo certi delle credenziali, utilizzando gli strumenti presenti nelle opzioni sulla privacy. Prestiamo atten-

zione nel **pubblicare** video, foto o post con informazioni personali: dobbiamo essere consapevoli che non potremo più controllare la diffusione di un contenuto **condiviso**. Inoltre sui social network vengono a volte create **identità false**, data la modalità di custodia delle credenziali di autenticazione degli utenti, dalle quali dobbiamo guardarci.

8 Non rispondere alle provocazioni

Evitiamo di rispondere a messaggi provenienti da email, blog, chat o social network a carattere minaccioso o provocatorio. Piuttosto utilizziamo gli strumenti leciti messi a disposizione dalla rete, **bloccando** o **segnalando** il contatto che ci ha infastidito. Fenomeni di questo tipo prendono il nome di **cyberbullismo** e **cyberstalker**.

9 Segnalare i contenuti illeciti o inappropriati

Segnaliamo i contenuti illeciti o inappropriati che troviamo su Internet e che a nostro parere violano le norme della community, per consentire di esaminarli, di difenderci e per garantire una esperienza di navigazione online migliore per tutti.

10 Attenzione ai file che scarichiamo da Internet o come allegati di posta elettronica

Recentemente sono saliti alla ribalta dell'opinione pubblica alcuni particolari virus chiamati **ransomware**, che bloccano il sistema criptando tutti i dati memorizzati nel computer. Per ottenere la chiave di sblocco è necessario pagare un riscatto, generalmente in **bitcoin**, la moneta di Internet. In caso contrario gli hacker non forniranno la chiave di sblocco e l'unica soluzione sarà quella di formattare il computer. Per difendersi dai ransomware è importante aggiornare periodicamente il proprio antivirus, oltre al firewall, in modo che possa essere in grado di individuare i malware prima che infettino il dispositivo. Può essere utile effettuare periodici backup dei dati in modo da recuperare i dati eventualmente danneggiati dai ransomware.

11 La sicurezza delle informazioni

La **sicurezza** delle informazioni è caratterizzata dagli aspetti che seguono.

- **Confidenzialità**: garantisce che l'informazione sia accessibile solamente a coloro che hanno l'autorizzazione ad accedervi. Si indica con l'acronimo **AAA** (**A**uthentication, **A**uthorization e **A**ccounting) che consente di identificare l'utente prima che sia autorizzato all'accesso alla rete.
- **Integrità**: garantisce l'accuratezza e la completezza dell'informazione e dei metodi di elaborazione.
- **Disponibilità**: garantisce che gli utenti autorizzati possano accedere all'informazione quando vi è necessità.

12 Classificare le informazioni

La **classificazione** delle informazioni permette di valutare il rischio che un utilizzo non corretto può apportare al patrimonio informativo. La classificazione è riferita a tutte le tipologie di informazioni e di dati, oltre ai software che li elaborano. Esistono tre livelli di classificazione che devono essere utilizzati.

LIVELLO 0 (Non classificato)

L'utilizzo di queste informazioni non comporta alcun pregiudizio all'attività, per cui non è prevista alcuna limitazione alla loro circolazione interna ed esterna. Queste informazioni sono solitamente di pubblico dominio (per esempio listini, cataloghi, dati forniti tramite il web di pubblico dominio ecc.).

LIVELLO 1 (Uso interno)

L'utilizzo di queste informazioni avviene unicamente, all'interno dell'azienda, da personale interno: la divulgazione di tali dati deve essere, pertanto, circoscritta all'area interna. La diffusione all'esterno comporta pregiudizi di entità, più o meno notevole, al patrimonio, alle finanze o all'immagine aziendale.

LIVELLO 2 (Riservato)

L'utilizzo improprio di queste informazioni può costituire pregiudizio al patrimonio, alle finanze o all'immagine aziendale. Il loro utilizzo è circoscritto a un numero ridotto di persone, opportunamente identificate, all'interno dell'azienda e il loro trattamento può avvenire con strumenti specifici, ai quali gli utenti individuali sono espressamente sballati. Per questa tipologia di informazione è interdotta la

6.2) Vulnerabilità, minacce e contromisure

Una **vulnerabilità informatica** può essere intesa come una componente (esplicita o implicita) di un [sistema informatico](#), in corrispondenza alla quale le misure di sicurezza sono assenti, ridotte o compromesse, il che rappresenta un punto debole del [sistema](#) e consente a un eventuale aggressore di compromettere il livello di sicurezza dell'intero sistema.

Riferita a persone, individui singoli o gruppi di individui (gruppi o comunità di vulnerabili),^[1] può indicare una debolezza che può consentire ad un [attacco informatico](#) di compromettere un sistema, cioè di ridurre il livello di protezione fornito da tale sistema, fino al caso limite di inficiarne il funzionamento. Particolarmente importanti sono le vulnerabilità dei [sistemi informatici](#) nei confronti di [hacker](#) o [cracker](#).

Le vulnerabilità si possono presentare a qualsiasi livello di un [sistema informatico](#), ne esistono principalmente di due tipi:

1. Vulnerabilità [software](#) ([bug](#) software): Si presenta ovunque ci sia un difetto di progettazione, codifica, [installazione](#) e configurazione del software. Le vulnerabilità software dovute a difetti di progettazione e codifica sono causate principalmente dalla mediazione incompleta dei dati forniti in input direttamente dall'utente e mancanza di controlli di sistema
2. Vulnerabilità dei protocolli: si manifestano quando i [protocolli di comunicazione](#) non contemplano il problema legato alla [sicurezza](#); l'esempio classico di vulnerabilità consiste nel permettere una [connessione](#) in chiaro (non [crittografata](#)) consentendo a possibili malintenzionati di intercettare le [informazioni](#) scambiate ([eavesdropping](#))

Una vulnerabilità può essere causata da:

- Complessità: Sistemi informatici molto grandi e complessi, possono essere causa di falle e involontari punti di accesso.^[5]
- Connettività: Più sono presenti porte, protocolli, privilegi, connessioni fisiche e servizi, e più il tempo in cui queste sono accessibili, più è probabile la presenza di un attacco.^[6]
- Password: L'utente utilizza password deboli, facilmente soggette a [password cracking](#) attraverso il [metodo "forza bruta"](#)^[7], oppure utilizza troppo spesso la stessa password^[5] o la memorizza nel computer.
- Sistemi operativi basilari: Certi sistemi operativi permettono agli utenti e ai programmi un ampio accesso alle risorse, permettendo tutto ciò che non è stato esplicitamente negato. Questa politica, sebbene più funzionale, permette a [virus](#) e [malware](#) di eseguire comandi a [livello amministratore](#).
- Navigazione in Internet: Certi siti possono contenere [Spyware](#) o [Adware](#) che vanno ad infettare il dispositivo, rubando informazioni personali.
- Bug Software: In un programma viene lasciato un [bug](#). Questo potrebbe essere sfruttabile per un possibile attacco all'applicazione.

- Mancanza di controllo degli Input: Un programma può assumere che tutti gli input dell'utente siano sicuri, permettendo la diretta esecuzione di programmi e istruzioni [SQL](#), causando [Buffer overflow](#), [SQL injection](#) o [vulnerabilità di formato della stringa](#).
- Non imparare dagli errori: Quando un programmatore ripete gli stessi errori in nuovi programmi, aumenta la possibilità che questi siano conosciuti e quindi attaccabili. Si veda, ad esempio, le vulnerabilità nei protocolli [IPv4](#), scoperte essere ancora presenti nei nuovi [IPv6](#).

Minacce

La sicurezza informatica è un settore in rapida evoluzione. Gli amministratori e gli esperti di sicurezza informatica inventano e adottano un'ampia gamma di termini e frasi per descrivere i rischi potenziali o gli incidenti indesiderati su computer e rete. Di seguito vengono analizzati alcuni dei termini e il loro significato in base a come vengono utilizzati in questo documento.

Virus/minaccia informatica.

Un virus informatico/una minaccia informatica è un programma, ovvero un codice eseguibile, con la capacità esclusiva di riprodursi. Penetrano in qualsiasi tipo di file eseguibile e si diffondono nel momento in cui i file vengono copiati e inviati da un utente all'altro.

Oltre alla moltiplicazione, alcuni virus/minacce informatiche hanno un altro elemento comune: una routine di danneggiamento che trasporta la carica distruttiva. Benché a volte si limitino a visualizzare messaggi o immagini, le cariche distruttive possono anche danneggiare file, formattare il disco rigido o causare danni di altra natura.

- **Minacce informatiche:** le minacce informatiche sono dei software progettati per infiltrarsi o danneggiare un computer all'insaputa del proprietario.
- **Cavalli di Troia:** un cavallo di Troia è un programma dannoso mascherato da applicazione innocua. Contrariamente ai virus e alle minacce informatiche, i cavalli di Troia non si moltiplicano ma possono essere altrettanto dannosi. Un esempio di cavallo di Troia è un'applicazione che a prima vista serve per eliminare virus/minacce informatiche dai computer ma in realtà li introduce.
- **Worm:** un worm è un programma (o gruppo di programmi) autonomo in grado di diffondere copie funzionanti di se stesso o di suoi segmenti ad altri sistemi di computer. La propagazione avviene in genere mediante le connessioni alla rete o gli allegati e-mail. A differenza dei virus e delle minacce informatiche, non hanno la necessità di attaccarsi a programmi host.
- **Backdoor:** un programma backdoor è un metodo per oltrepassare i normali processi di autenticazione così da permettere l'accesso remoto a un computer e ottenere accesso alle informazioni rimanendo, nel frattempo, nascosto.

- **Rootkit:** un rootkit è un insieme di programmi progettato per danneggiare il controllo legittimo di un sistema operativo da parte dei suoi utenti. Solitamente un rootkit mantiene nascosta la propria installazione e tenta di evitare la rimozione provando a modificare i sistemi di protezione standard.
- **Virus da macro:** i virus da macro sono specifici di un'applicazione. Questo tipo di virus si trova all'interno di file destinati ad applicazioni come Microsoft Word (.doc) e Microsoft Excel (.xls). Possono, pertanto, essere rilevati in file con estensioni comuni ad applicazioni in grado di eseguire macro come .doc, .xls e .ppt. I virus da macro viaggiano tra i file di dati nell'applicazione e possono, se non scoperti, infettare centinaia di file.

I programmi dell'agente presenti sui computer client, detti anche Client/Server Security Agent rilevano la presenza di virus/minacce informatiche durante la scansione antivirus. In presenza di virus/minacce informatiche, Trend Micro consiglia di eseguire una *disinfezione*.

Spyware/grayware

Con il termine grayware si intendono tutti quei programmi che eseguono operazioni inaspettate o non autorizzate. È un termine generico utilizzato per indicare spyware, adware, dialer, programmi ingannevoli, strumenti di accesso remoto e qualsiasi altro tipo di file e programma indesiderato. A seconda del tipo, queste minacce possono contenere codici dannosi in grado o meno di replicarsi.

- **Spyware:** gli spyware sono software che si installano sul computer senza il consenso o la consapevolezza dell'utente per raccogliere o trasmettere informazioni personali.
- **Dialer:** i dialer sono necessari per collegare a Internet i computer che non dispongono di connessione a banda larga. I dialer dannosi sono progettati per collegare i computer a numeri telefonici con tariffe altissime invece che al provider di servizi Internet abituale. I provider di tali dialer dannosi riscuotono le tariffe pagate. I dialer vengono utilizzati anche per trasmettere informazioni personali e scaricare software dannoso.
- **Strumenti per hacker:** uno strumento per hacker è un programma o un gruppo di programmi progettato per l'attività degli hacker.
- **Adware:** per adware s'intende il software finalizzato alla pubblicità e si riferisce a qualsiasi pacchetto software che, dopo la sua installazione o quando l'applicazione viene utilizzata, esegue, visualizza o scarica automaticamente del materiale pubblicitario su un computer.
- **Keylogger:** i keylogger sono dei software che registrano tutte le sequenze di tasti digitate dall'utente. Queste informazioni possono quindi essere recuperate da un hacker che le utilizza per i propri scopi.
- **Bot:** un bot (diminutivo di "robot") è un programma che opera come agente per un utente o per un altro programma e simula un'attività umana. I bot, una volta eseguiti, possono

replicarsi, comprimersi e distribuire copie di se stessi. I bot possono essere utilizzati per coordinare un attacco automatico su computer collegati in rete.

Alcune applicazioni vengono classificate da Trend Micro come spyware/grayware non perché siano dannose per il sistema su cui sono installate, ma perché potrebbero esporre il client o la rete ad attacchi da parte di minacce informatiche o hacker.

Hotbar, ad esempio, è un programma che inserisce una barra degli strumenti nei browser Web. Hotbar rileva gli URL visitati dagli utenti e registra le parole o le frasi che questi immettono nei motori di ricerca. Queste informazioni vengono poi utilizzate per visualizzare messaggi pubblicitari prestabiliti, comprese le finestre a comparsa, nei browser in uso. Poiché le informazioni raccolte da Hotbar potrebbero essere inviate a un sito di terzi e utilizzate dalle minacce informatiche o dagli hacker per raccogliere informazioni sugli utenti, Worry-Free Business Security Services impedisce che questa applicazione venga installata ed eseguita per impostazione predefinita.

Se l'utente desidera eseguire Hotbar o qualsiasi altra applicazione classificata da WFBS-SVC come spyware/grayware, dovrà aggiungerla all'elenco di spyware/grayware affidabili.

Impedendo l'esecuzione di applicazioni potenzialmente dannose e fornendo all'utente il controllo totale sull'elenco di spyware/grayware affidabili, WFBS-SVC garantisce che client e server eseguano sui client solo le applicazioni approvate dagli utenti.

I moduli Client/Server Security Agent sono in grado di rilevare grayware. L'azione consigliata da Trend Micro per spyware e grayware è la *disinfezione*.

Minacce e vulnerabilità

I sistemi informatici sono soggetti a minacce che possono sfruttare vulnerabilità (minaccia e vulnerabilità sono concetti contigui ma distinti^[4]): questo potrebbe causare attacchi volti all'accesso ai dati in esso contenuti oppure o a minarne la funzionalità o [disponibilità](#) di servizio. Spesso dal funzionamento o meno del sistema informatico dipende anche la sicurezza dei dati in esso contenuti. Le stesse cause di *out of service* dei sistemi informatici possono anche essere raggruppate in due classi di eventi:

- eventi accidentali;
- eventi indesiderati.

Eventi accidentali

Gli eventi accidentali non riguardano attacchi malevoli, ma fanno riferimento a eventi causati accidentalmente dall'utente stesso, tipo: uso difforme dal consigliato di un qualche sistema, incompatibilità di parti [hardware](#), guasti imprevisti, ecc. Tutti questi eventi compromettono la sicurezza del sistema soprattutto in termini di [disponibilità](#). Per evitare gli eventi accidentali non esistono soluzioni generali; un primo rimedio è il fare regolarmente una copia di [backup](#) del

sistema, comprendente dati e applicazioni, com'è tipico delle procedure di [disaster recovery](#), in modo da poter fronteggiare un danno imprevisto.

Eventi indesiderati

Tra i due eventi sopra citati, quelli indesiderati sono quelli per lo più inaspettati, anche se è prudente aspettarsi di tutto, e sono i cosiddetti attacchi da parte di utenti non autorizzati al trattamento di dati o all'utilizzo di servizi. Alcuni degli eventi indesiderati che si possono subire possono essere:

- attacchi malevoli;
- uso delle proprie autorizzazioni per l'accesso a sistemi da parte di utenti non autorizzati.

Principali cause di perdita di dati

Le cause di probabile perdita di dati nei sistemi informatici possono essere classificate in:

1. [Malware](#),
2. Smarrimento o Furto di documenti, [dispositivi mobili](#) o fissi,
3. Divulgazione non intenzionale,
4. Frodi con [carte di pagamento](#).

Contromisure

Per contromisure intendiamo tutto ciò che concorre, attivamente o passivamente, a minimizzare la probabilità che gli eventi indesiderati accadano, rilevare il fatto che sono accaduti, individuarne e minimizzarne le conseguenze, ripristinare il corretto funzionamento del sistema. L'evoluzione storica delle contromisure è legata a quella delle tecnologie, alla crescente interconnessione fra i sistemi e, soprattutto, al crescente grado di sofisticazione degli attacchi deliberati. Un'elencazione estesa delle contromisure adottate nei sistemi informativi moderni va oltre gli scopi di questa trattazione, ma una classificazione delle principali tipologie di contromisura, è comunque utile per affrontare la tematica in modo organico.

Una prima possibilità è quella di classificare le contromisure in funzione degli eventi indesiderati che vanno a contrastare. Affiancare a ciascun evento indesiderato le contromisure applicabili è in effetti lo schema adottato da molti testi destinati agli amministratori di sistema. Accanto a questo criterio di classificazione molto diffuso, è possibile adottarne altri basati su caratteristiche generali proprie delle contromisure, piuttosto che sulla loro specifica applicabilità. Possiamo in particolare distinguere fra contromisure:

- **PREVENTIVE o CORRETTIVE:** per contromisure preventive intendiamo quelle finalizzate a minimizzare la probabilità che un evento indesiderato accada, mentre per

contromisure correttive indichiamo quelle le tese a riparare i danni causati dagli eventi indesiderati che, a dispetto delle contromisure preventive, sono ugualmente accaduti;

- **INFORMATICHE:** le prime sono basate sulla tecnologia informatica e le seconde riconducibili alla organizzazione che utilizza il sistema informatico ed alle norme e regole di comportamento stabilite per il personale;
- **A LIVELLO LOGICO:** le contromisure operanti a livello fisico proteggono dispositivi (calcolatori, cavi ed apparecchiature di rete, locali, impianti di alimentazione e condizionamento, etc.) da attacchi di tipo fisico quali furto o danneggiamento; quelle operanti a livello logico, come ad esempio i software antivirus, proteggono risorse logiche (basi di dati, registri di configurazione, moduli software, etc.) da attacchi di tipo logico.

Le possibili tecniche di attacco sono molteplici, perciò è necessario usare contemporaneamente diverse tecniche difensive per proteggere un sistema informatico, interponendo barriere fra l'attaccante e l'obiettivo. Il sistema informatico deve essere in grado di impedire l'alterazione diretta o indiretta delle informazioni, sia da parte di utenti non autorizzati, sia a causa di eventi accidentali; inoltre deve impedire l'accesso abusivo ai dati. Inoltre in generale non è buona norma assumere che le contromisure adottate in un sistema siano sufficienti a scongiurare qualsiasi attacco.

Per far fronte a evenienze derivanti da possibili guasti o danni fisici, come sicurezza fisica o passiva molte volte si opera in un contesto di [ridondanza](#) degli apparati (es. [server cluster](#)) ovvero con [sistemi distribuiti](#) all'interno di piani di [disaster prevention](#) che, assicurando la tolleranza ai guasti ([fault tolerance](#)), garantiscano [affidabilità](#) e disponibilità, cioè la [continuità operativa](#) del [sistema informatico](#) e dell'azienda. A volte si preferisce agire anche in maniera preventiva tramite piani di [disaster prevention](#). Tra le contromisure più comuni di tipo logico sulla [rete locale](#) di un sistema e sui suoi sottosistemi troviamo:

- [Sistema di autenticazione](#): potrebbe rivelarsi utile, in particolare nelle aziende, l'utilizzo di software per l'autenticazione sicura con un secondo elemento di autenticazione basato su un insieme di caratteri disposti in uno schema suddiviso in file e colonne conosciute dall'utente che dovrà poi inserirle in una combinazione di valori per dimostrare di essere in possesso dei dati corretti. Altro sistema, più sofisticato, è quello del riconoscimento dell'utente tramite l'utilizzo dell'impronta digitale come forma di autenticazione.
- Gestione utenti e relativi [permessi](#);
- [Mandatory Access Control](#) (MAC), tipologia di controllo di accesso a un [sistema informatico](#).
- [Firewall](#): installato e ben configurato un firewall garantisce un sistema di controllo degli accessi verificando tutto il traffico che lo attraversa. Protegge contro aggressioni provenienti dall'esterno e blocca eventuali programmi presenti sul computer che tentano di accedere a internet senza il controllo dell'utente.
- [Intrusion detection system](#) (IDS): è un dispositivo software e hardware (a volte la combinazione di tutti e due) utilizzato per identificare accessi non autorizzati ai computer. Le intrusioni rilevate possono essere quelle prodotte da cracker esperti, da tool automatici o da utenti inesperti che utilizzano programmi semiautomatici. Gli IDS vengono utilizzati per rilevare tutti gli attacchi alle reti informatiche e ai computer. Un

IDS è composto da quattro componenti. Uno o più sensori utilizzati per ricevere le informazioni dalla rete o dai computer. Una console utilizzata per monitorare lo stato della rete e dei computer e un motore che analizza i dati prelevati dai sensori e provvede a individuare eventuali falle nella sicurezza informatica. Il motore di analisi si appoggia a un database ove sono memorizzate una serie di regole utilizzate per identificare violazioni della sicurezza.

- [Network Intrusion Detection System](#) (NIDS): sono degli strumenti informatici, software o hardware, dediti ad analizzare il traffico di uno o più segmenti di una LAN al fine di individuare anomalie nei flussi o probabili intrusioni informatiche. I più comuni NIDS sono composti da una o più sonde dislocate sulla rete, che comunicano con un server centralizzato, che in genere si appoggia ad un Database. Fra le attività anomale che possono presentarsi e venire rilevate da un NIDS vi sono: accessi non autorizzati, propagazione di software malevolo, acquisizione abusiva di privilegi appartenenti a soggetti autorizzati, intercettazione del traffico (sniffing), negazioni di servizio (DoS).
- [Honeypot](#): un honeypot (letteralmente: "barattolo del miele") è un sistema o componente hardware o software usato come trappola o esca a fini di protezione contro gli attacchi di pirati informatici. Solitamente consiste in un computer o un sito che sembra essere parte della rete e contenere informazioni preziose, ma che in realtà è ben isolato e non ha contenuti sensibili o critici; potrebbe anche essere un file, un record, o un indirizzo IP non utilizzato.
-

6.3) Tecniche crittografiche e loro applicazioni

La crittografia è un meccanismo per agevolare la protezione dei dati. Essa garantisce la riservatezza dei dati "camuffandoli" in modo che solo le persone autorizzate possano accedervi o leggerli.

Nel processo di crittografia i dati originali, o testo normale, in combinazione con un valore nominato "chiave", vengono elaborati tramite una o più formule, procedura che rende illeggibili file originali; i dati originali sono denominati testo crittografato. Per rendere leggibili nuovamente i dati, il destinatario li decrittografa invertendo il processo matematico con l'ausilio della chiave corretta.

NB= una chiave di crittografia più lunga aiuta a rendere più sicuro il testo crittografato rispetto a una chiave più breve; tuttavia la maggiore complessità dell'operazione di crittografia comporta un costo maggiore in termini di tempo del processore rispetto all'utilizzo di una chiave più breve.

Nello studio di questa materia, si sono studiati due tipi di crittografia:

- **Crittografia simmetrica**: nota anche come crittografia a chiave pubblica condivisa.
- **Crittografia asimmetrica**: nota anche come crittografia

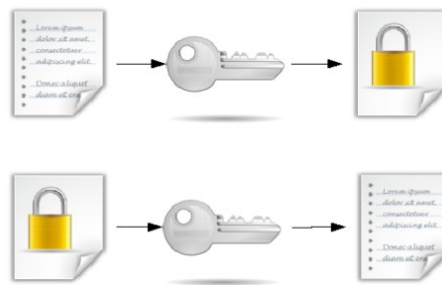
Il processo di **crittografia simmetrica** è caratterizzato dall'utilizzo di una medesima chiave condivisa sia per l'operazione di cifratura che per quella di decifratura. Negli schemi di cifratura

simmetrica, la sicurezza del processo risiede nella **lunghezza della chiave utilizzata** e nella **capacità di mantenerla segreta**. La dimensione della chiave è un parametro fondamentale per valutare la **robustezza** dello schema usato, insieme alla resistenza dell'algoritmo agli attacchi di crittoanalisi. Nei processi di crittografia simmetrica è possibile utilizzare **chiavi lunghe quanto il messaggio**, che rendono il messaggio cifrato estremamente sicuro.

La **criticità principale** della crittografia simmetrica risiede nella necessità di doversi **scambiare le chiavi in maniera sicura**. Nei processi completamente informatizzati questo scambio di solito avviene ricorrendo alla crittografia asimmetrica (ad es. tramite algoritmo di Diffie-Hellmann); altro metodo consiste nell'**incapsulamento** di una chiave in un'altra, ovvero generando una chiave master che viene scambiata attraverso un canale sicuro: le chiavi di cifratura sono quindi cifrate con la chiave master e scambiate.

Tra gli algoritmi di crittografia simmetrica più popolari ricordiamo: DES, Triple-DES (3DES), IDEA, CAST5, BLOWFISH, TWOFISH.

Cifratura / Decifratura Simmetrica



La **crittografia asimmetrica**, anche nota come crittografia a chiave pubblica/privata, è un tipo di crittografia in cui a ciascuna entità è associata una coppia di chiavi:

- Una **chiave pubblica** accessibile a tutti coloro che necessitano di scambiare informazioni con l'entità proprietaria;
- Una **chiave privata**, custodita e tenuta segreta dal legittimo proprietario.

In un processo di crittografia asimmetrica, qualsiasi messaggio crittografato con una chiave privata potrà essere decifrato solo utilizzando la chiave pubblica corrispondente. Il processo svincola quindi l'utente dalla necessità di dover scambiare le chiavi prima di qualunque comunicazione. Nei processi di crittografia a chiave pubblica non è necessario preoccuparsi dello scambio di chiavi: gli algoritmi utilizzati non consentono di risalire alla chiave privata partendo dalla chiave pubblica. I **principali utilizzi** della crittografia asimmetrica sono:

1. **Invio di un messaggio cifrato ad un destinatario:** il mittente cifra il messaggio con la chiave pubblica del destinatario, che sarà quindi l'unico a poter decifrare il messaggio utilizzando la sua chiave privata;

2. **Non ripudiabilità del messaggio:** il proprietario della chiave privata cifra il messaggio con la propria chiave privata, qualunque entità che possa accedere alla chiave pubblica del firmatario potrà decifrarlo con la stessa ed essere quindi certo della sua autenticità.



La crittografia asimmetrica presenta molti ambiti di impiego tra cui la firma digitale, la verifica dell'autenticità di un documento, la verifica dell'integrità del messaggio, l'informativa del messaggio, oppure la protezione dello scambio delle chiavi simmetriche durante la procedura di scambio tra due entità.

6.4) Controllo degli accessi

Il controllo degli accessi identifica gli utenti verificando varie credenziali di accesso, che possono includere nome utente e password, PIN, scansioni biometriche e token di sicurezza. Molti sistemi di controllo degli accessi includono anche l'autenticazione multi-fattore, un procedimento che richiede più metodi di autenticazione per verificare l'identità di un utente.

Una volta che l'utente è stato autenticato, il controllo degli accessi autorizza il livello di accesso appropriato e determinate azioni consentite associati alle credenziali dell'utente e all'indirizzo IP.

Se un computer può eseguire più programmi contemporaneamente, allora può ricevere comandi da più utenti contemporaneamente. Proprio per questo un sistema operativo multiutente pone problematiche particolari dovute alla presenza di più utenti. Un sistema operativo deve tener conto anche dei seguenti aspetti:

- L'univoca identificazione ed autenticazione degli utenti
- La protezione delle informazioni relative a ciascun utente nei confronti degli altri
- La garanzia di accesso alle risorse esclusivamente agli utenti abilitati
- Il controllo diversificato dell'accesso alle risorse del sistema per ciascun utente
- La tracciabilità di qualsiasi evento di modifica delle informazioni trattate e l'individuazione del suo autore.

Esaminiamo ora che cosa si intende per controllo degli accessi.

Alla base dello sviluppo di un sistema di riconoscimento per il controllo degli accessi vi è la definizione di regole secondo cui consentire o negare l'accesso alle risorse; successivamente si definisce un meccanismo in base al quale si fa in modo che le regole vengano rispettate. Quindi prima si definisce la politica di sicurezza, poi si definiscono i meccanismi software e hardware che implementano tale politica scelta. Un passo importante è la scelta del tipo di riconoscimento che si vuole adoperare per consentire o meno l'accesso alla risorsa.

Politiche di controllo degli accessi

Le politiche di controllo degli accessi possono essere divise in tre classi principali: discrezionali, obbligatorie e basate sui ruoli.

Politiche discrezionali

Le politiche discrezionali (**Discretionary Access Control, DAC**) costruiscono il controllo degli accessi sull'identità del richiedente e su gruppi di regole che stabiliscono chi può e chi non può eseguire determinate azioni su determinate risorse. Il termine "discrezionali" denota appunto la possibilità che gli utenti trasferiscano i propri diritti ad altre entità a loro discrezione. Non è dunque necessario l'intervento dell'amministratore di sistema nel processo di conferimento e revoca dei permessi. Ad ogni modo l'amministratore è colui che possiede tutti gli oggetti del sistema e può violare a piacimento le diverse restrizioni d'accesso.

Le politiche obbligatorie

Le politiche obbligatorie (Mandatory Access Control, MAD) gestiscono il controllo degli accessi in conformità ad un insieme di regole definite ed impartite da sistema o autorità centrale.

Il controllo di accesso si basa sulla specifica a priori di una serie di attributi di sicurezza rispettivamente per soggetti e oggetti: quando un soggetto tenta l'accesso a un oggetto, il kernel del sistema operativo avvia una regola di autorizzazione che esamina gli attributi di sicurezza di soggetti e oggetti e decide se l'accesso può aver luogo oppure essere negato.

Le regole di accesso che vengono generalmente impiegate sono le seguenti:

- No read up: l'utente non può leggere informazioni con un livello d'accesso superiore al suo
- No write down: l'utente non può scrivere informazioni con un livello di accesso inferiore al suo.

Le politiche basate sui ruoli

Nella sicurezza informatica, il **role-based access control** (in italiano: controllo degli accessi basato sui ruoli; in sigla RBAC) è una tecnica di accesso a sistemi ristretto per utenti autorizzati. È una più recente alternativa al mandatory access control ("controllo degli accessi vincolato", in sigla MAC) e al discretionary access control ("controllo degli accessi discrezionale", in sigla DAC).

Si tratta di un meccanismo di accesso definito basandosi sui concetti di ruolo e privilegio. I componenti del RBAC, come i permessi dei ruoli, il ruolo utente e le relazioni ruolo-ruolo, fanno in modo di semplificare l'assegnamento dei ruoli agli utenti.

La motivazione dietro a questo tipo di politica è la necessità di specificare ed imporre politiche di controllo degli accessi strettamente legate alla struttura dell'organizzazione aziendale. Ai fini del controllo degli accessi piuttosto che conoscere l'identità di un soggetto è invece molto più importante conoscere i ruoli e le funzioni che tale soggetto svolge all'interno di un'organizzazione..

La politica RBAC supporta i principi di minimo privilegio e di separazione delle responsabilità.

In un modello RBAC si possono individuare i seguenti componenti fondamentali:

- Un utente è un soggetto facente parte di un'organizzazione.
- Un ruolo è un insieme di operazioni che un utente può svolgere all'interno di un'organizzazione.
- Un'operazione è una procedura alla quale viene associato un certo set di dati, che viene definito con il termine oggetto.

Va notato il fatto che le operazioni sono allocate ai diversi ruoli dell'amministratore di sistema, che si preoccupa anche di assegnare i vari utenti ai diversi ruoli presenti all'interno dell'organizzazione.

6.5) Principali aspetti normativi

Iniziamo, intanto, a definire le principali attività cyber criminali indicando per ciascuna di esse il relativo riferimento normativo.

Hacking

- **Accesso abusivo ad un sistema informatico o telematico** (*Articolo 615 ter del codice penale*). Questo reato richiede che una persona ottenga l'accesso ad un sistema informativo protetto contro il consenso esplicito o implicito della persona avente diritto di escludere terzi dall'ottenimento di tale accesso. **La pena è la reclusione fino a 3 anni.**
- **Frode digitale** (*Articolo 640 ter del codice penale*). Questo reato si configura quando chi – consapevolmente e con l'intento di frodare – manomette uno o più dispositivi digitali, in violazione di legge, utilizzando informazioni, dati o software al fine di ottenere un guadagno economico o danneggiare qualcun altro. **La pena è la reclusione da sei mesi a tre anni.**
- **Falsa identità** (*Articolo 494 del codice penale*). L'articolo in questione è applicabile alle identità reale, nonché alle identità digitali; il reato in questione è perpetrato quando qualcuno falsamente e volontariamente si sostituisce a qualcun altro. **La pena è la reclusione fino ad un anno.**
- **Detenzione e diffusione abusiva di password** (*Articolo 635 bis del codice penale*). Si verifica quando qualcuno intenzionalmente danneggia, distrugge, cancella o disabilita qualsiasi tipo di informazione digitale, dati o software di proprietà di qualcun altro. **La pena è la reclusione da sei mesi a tre anni.**

Attacchi DOS

- **Danneggiamento di informazioni, dati o software** (*Articolo 635 bis del codice penale*). Il reato si configura quando qualcuno intenzionalmente danneggia, distrugge, cancella o disabilita qualsiasi tipo di informazione digitale, dati o software di proprietà di qualcun altro. **La pena è la reclusione da sei mesi a tre anni.**

Phishing

- **Frode digitale** (*Articolo 640 del codice penale*): come già descritto sopra.
- **Falsa identità** (*Articolo 494 del codice penale*): come già descritto sopra.

Compromissione di sistemi IT con malware (ransomware, spyware, worm, trojan e virus)

- **Accesso abusivo ad un sistema informatico o telematico** (*Articolo 615 ter del codice penale*): come già descritto sopra.
- **Danneggiamento di informazioni, dati o software** (*Articolo 635-bis del codice penale*): come già descritto sopra.

Possesso o uso di hardware, software o altri strumenti utilizzati per commettere il crimine informatico (ad esempio, strumenti di hacking)

- **Detenzione e diffusione abusiva di password ai sistemi digitali** (*Articolo 615 quarto del codice penale*): come già descritto sopra.

Qualsiasi altra attività che incide negativamente o minaccia la sicurezza, la riservatezza, l'integrità e la disponibilità di qualsiasi sistema IT, infrastrutture, reti di comunicazione, dispositivi o dati

- **Intercettazioni illegali e distruzione delle comunicazioni** (*Articolo 616 del codice penale*). Il reato è commesso quando una persona apre, ruba o distrugge la corrispondenza, comprese le e-mail, non indirizzate a lui o lei. **La pena è la reclusione fino a un anno.**
- **Intercettazioni illegali, distorsione, falsificazione e distruzione delle comunicazioni** (*Dall'articolo 617 bis a 617 sexies del codice penale*). Questi diversi reati, puniti da diversi articoli del codice penale, sono commessi quando una persona apre, ruba o distrugge la corrispondenza altrui, comprese le e-mail, anche con software, malware o qualsiasi tipo di strumento digitale avente uno di questi scopi. **La pena è della reclusione da sei mesi/un anno a quattro anni.**
- **Divulgazione illecita di e-mail** (*Articolo 618 del codice penale*). Tale ipotesi di reato si configura nel caso in cui un soggetto intenzionalmente divulghi, o cerchi di divulgare, a qualsiasi altro soggetto, il contenuto di qualsiasi comunicazione via cavo, verbale o elettronica, sapendo o avendo motivo di sapere che l'informazione è stata ottenuta mediante intercettazione via cavo, verbale o elettronica in violazione della presente disposizione. **La pena è la reclusione fino a sei mesi.**

Leggi di sicurezza informatica: cosa cambia con il GDPR

Il Regolamento generale per la protezione dei dati è entrato in vigore il 25 maggio 2018 e il suo arrivo ha cambiato, tra le altre cose, anche il modo in cui su Internet vengono raccolte e gestite le informazioni private degli utenti.

Con questa riforma, i cittadini dell'UE hanno finalmente un maggiore controllo sulla sicurezza dei loro dati personali e i siti web saranno tenuti a seguire rigorosi adeguamenti di conformità per garantire la protezione di questi dati.

Che cosa significa questo per il futuro della sicurezza informatica e in che modo l'economia digitale ne risentirà nel suo complesso? Anche se il [sito Web](#) non ha un'affiliazione diretta con il Regno Unito o con l'UE, è comunque importante essere consapevoli di cosa comporta la conformità al GDPR e creare in anticipo una strategia di elaborazione dei dati.

Capitolo 6

Relational Database Management System

6.1) Progettazione concettuale, logica e fisica di una base di dati

Una **base di dati** o **database** può essere considerata come una raccolta di dati logicamente correlati, utilizzata per modellare una realtà. I dati sono memorizzati su un supporto di memoria di massa e sono progettati per essere fruiti in maniera ottimizzata da differenti applicazioni e utenti diversi.

Requisiti fondamentali di un database:

- **Sicuro**, progettato in modo da impedire che venga danneggiato da eventi accidentali o interventi non autorizzati.
- **Integro**, ossia deve essere garantito che le operazioni effettuate da utenti autorizzati non provochino una perdita di consistenza dei dati.
- **Consistente**, ossia i dati devono essere significativi ed effettivamente utilizzabili.
- **Condivisibile**, cioè applicazioni e utenti devono poter accedere, ai dati comuni.
- **Persistente**, deve avere un tempo di vita (TTL-Time to live) che non è limitato a quello delle singole esecuzioni dei programmi che lo utilizzano.
- **Scalabile**, cioè deve mantenere intatte le proprie performance all'aumentare della quantità di dati.

IL MODELLO DEI DATI

Un modello di dati è un insieme di concetti e costrutti usati per organizzare i dati descrivendone struttura, associazioni e vincoli.

Esistono vari modelli:

1) **Modelli concettuali**: cercano di descrivere i concetti del mondo reale. Essi sono usati nella fase preliminare di progettazione e il più noto è il modello E/R.

2) **Modelli logici**: consentono una specifica rappresentazione dei dati attraverso tabelle, grafi, alberi, oggetti, e descrivono ciò che l'utente finale può vedere. Tra questi modelli abbiamo:

- **Modello gerarchico**: piuttosto obsoleto, in cui i dati sono connessi secondo una struttura ad albero, con una forte dipendenza dei programmi alle strutture, causa di ridondanze ed inconsistenze.

- **Modello reticolare:** la struttura è ad anello con accesso ai dati più semplice, anche se tale struttura è abbastanza complessa e la sua modifica comporta la cancellazione dell'intero database.
- **Modello ad oggetti:** molto meno moderno, è un'evoluzione del modello relazionale e viene usato per applicazioni multimediali.

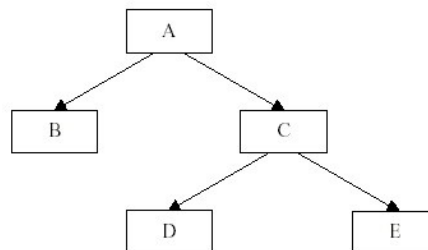
Modello gerarchico

Il modello gerarchico consente di rappresentare informazioni usando la relazione tra segmenti "padre" e segmenti "figli": ogni padre può avere molti figli, ma ogni figlio può avere un solo padre. In questo caso si parla di relazioni 1:N, chiamate anche relazioni uno-a-molti.

I dati sono organizzati in **record** (ogni record è un insieme di **campi**, cioè di informazioni elementari) connessi tra loro secondo una struttura **ad albero**. I record che realizzano l'albero sono chiamati **nodi**: ogni nodo può avere uno o più nodi figli mentre ha un solo nodo padre (eccetto il nodo **radice** che non ha padre).

I nodi possono essere di tre tipi:

- nodo **radice**: dall'inglese **root**, è il nodo in testa all'albero. Nella figura a destra è il nodo A;
- nodo **foglia**: i nodi dell'albero che non hanno figli. Nella figura sono i nodi: B, D, E;
- nodo **interno**: sono i nodi rimanenti. Nella nostra figura il nodo interno è soltanto C.



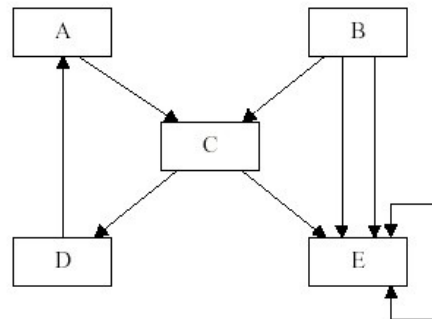
Modello reticolare

Anche nel modello reticolare i dati sono organizzati in **record** e **puntatori**. Un record è un insieme di informazioni (anche dette **campi**) e sono connessi tra loro secondo una struttura **a grafo**: i record del grafo sono chiamati **nodi** e i collegamenti tra un nodo e l'altro sono chiamati **archi**. Un nodo può puntare ad uno o più figli (come nel modello gerarchico) ma anche a uno o più padri.

Il collegamento tra i nodi dell'albero è realizzato tramite un tipo di dato chiamato "puntatore" e a differenza del modello gerarchico la navigazione può avvenire in più direzioni.

Il grafo è detto **orientato** se gli archi di collegamento hanno un verso di percorrenza, e in questo caso sugli archi sono presenti delle frecce; è invece detto **non orientato** se i collegamenti non hanno frecce e pertanto la navigazione può avvenire in ambedue le direzioni. L'esempio in figura rappresenta un grafo orientato.

Inoltre, il grafo è detto **pesato** se sugli archi di collegamento c'è un **peso** (un valore numerico); è invece detto **non pesato** se i collegamenti non hanno pesi e pertanto ogni collegamento ha la stessa valenza di un altro. L'esempio in figura rappresenta un grafo orientato non pesato.



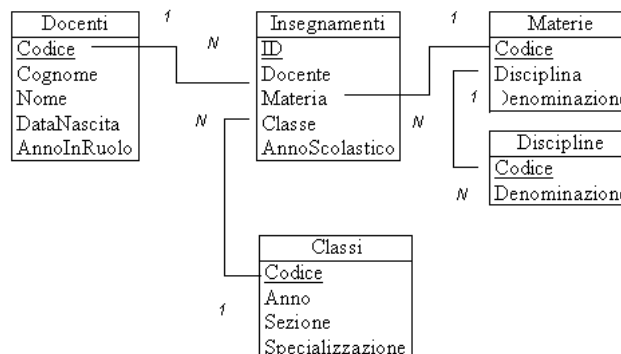
Modello relazionale

L'**entità** implementa "il tipo di dato", esempio lo STUDENTE, organizzandolo in un insieme di attributi (nome, cognome, matricola, data di nascita, ecc.). Le **relazioni**, invece, sono i collegamenti che possiamo avere tra questi diversi tipi di dati. Una relazione tra due tipologie di dati diverse (cioè tra due entità) e sono implementate per mezzo della presenza di valori comuni.

Il collegamento tra una entità e l'altra è realizzato tramite la presenza di campi contenenti valori uguali. Il campo della prima entità che fa da collegamento con la seconda è chiamato **chiave esterna** (o **foreign key**) la quale "referenzia" un altro campo che è la **chiave primaria** della seconda entità.

Questo modello si basa molto sulla teoria degli insiemi e sulla logica del primo ordine ed è strutturato intorno al concetto matematico di relazione. Per il suo trattamento ci si avvale di strumenti quali il *calcolo relazionale* e l'*algebra relazionale*.

Nel modello Relazionale non si fa riferimento più al *file* bensì si parla di **tabelle** e **colonne**. Facendo riferimento alla figura a destra, alla fine avremo una tabella DOCENTI che ha cinque colonne: Codice, Cognome, Nome, DataNascita e AnnoInRuolo.



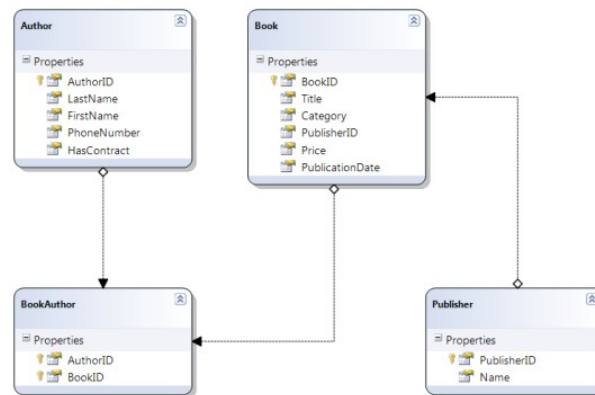
Modello a oggetti

L'OODB (Object Oriented DataBase) è un modello più recente di database che nasce dall'esigenza di gestire informazioni multimediali: immagini, audio, video, documenti e risorse Internet.

dati del database sono definiti **classi** nella cui struttura interna sono specificati sia i dati, o **proprietà**, che le modalità di accesso: i **metodi**. Un oggetto è una **istanza** della classe.

Esempio: possiamo definire la classe IMPIEGATO e l'istanza può essere "Rossi Paolo". Rossi è un oggetto di tipo impiegato, proprio come prescrive il paradigma della OOP: Programmazione Object Oriented.

In questi database il collegamento tra un oggetto e l'altro si realizza dichiarando l'istanza del secondo oggetto nella struttura interna del primo oggetto; ossia come un suo campo.



PROGETTAZIONE DI UN DATABASE E SUE FASI

Le fasi di progettazione di un database sono tre:

- 1) **PROGETTAZIONE CONCETTUALE**: ha lo scopo di trasformare la rappresentazione astratta in uno schema logico (tabelle).
- 2) **PROGETTAZIONE FISICA**: ha lo scopo di implementare lo schema logico definendone gli aspetti fisici di memorizzazione in memoria di massa.

IL DBMS

Il DBMS è un insieme di strumenti software che, sulla base delle specifiche dell'utente, genera lo schema per aggiornare i dati.

Le funzioni che un DBMS deve assolvere sono:

- 1) Gestione del database, con inserimento, cancellazione, aggiornamento e interrogazione dello stesso.
- 2) Persistenza e consistenza, cioè la conservazione del contenuto del database in caso di guasti o malfunzionamenti.
- 3) Privatezza e sicurezza dei dati attraverso meccanismi di autorizzazione.
- 4) Integrità dei dati mediante il controllo sui vincoli imposti per un dato database.

LA PROGETTAZIONE CONCETTUALE: IL MODELLO E/R

La progettazione concettuale consiste nel riorganizzare tutti gli elementi a disposizione per rappresentare la realtà di interesse. Il documento ufficiale è lo SCHEMA CONCETTUALE, che serve poi per la successiva fase logica. Il più diffuso modello concettuale è quello E/R o Entità/Associazioni, ed è un modello grafico per la descrizione dei dati e delle loro relazioni. Per la costruzione di uno schema E/R, si parte dal concetto che la realtà da rappresentare sia composta da ENTITÀ, ognuna delle quali è caratterizzata da proprietà dette ATTRIBUTI. Le varie entità sono connesse tra loro attraverso ASSOCIAZIONI.

LE ENTITÀ

Le entità sono ciò che esiste nella realtà che si vuole modellare come ad esempio il libro “I Promessi Sposi” dalla biblioteca che si sta esaminando. Gli attributi sono i fatti che si intende rappresentare e descrivono le entità, come ad esempio il numero di pagine di un libro. Ogni singolo esemplare appartenente ad una certa entità si dice ISTANZA. Ad esempio “I Promessi Sposi” è un’istanza dell’entità Libri.

GLI ATTRIBUTI

Possiamo dire che gli attributi semplici per l’entità Persona, sono ad esempio: Nome, Cognome, Età, Sesso. In sostanza ogni attributo è specificato da:

- 1) Nome
- 2) Formato, cioè il tipo di valori che assume
- 3) Dimensione, cioè la quantità massima di caratteri o cifre
- 4) Valore, il cui insieme determina il DOMINIO dell’attributo

Nel caso dell’entità Persona, abbiamo che il dominio dell’attributo Età va da 1 fino a 100, mentre il dominio del Sesso è {uomo, donna}.

Un attributo è OBBLIGATORIO se il suo valore è non nullo (Nome, Cognome), mentre è FACOLTATIVO se può essere vuoto (Titolo di studio) e coincide con il valore NULL.

Inoltre abbiamo gli attributi COMPOSTI che risultano dall’aggregazione di più attributi semplici, come ad esempio l’attributo DataDiNascita, composto da giorno, mese ed anno.

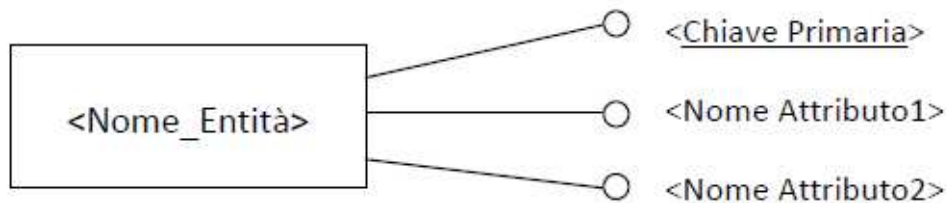
Esistono anche gli attributi MULTIPLI, come ad esempio Hobby.

L’esigenza di creare un oggetto come entità piuttosto che come attributo o viceversa, dipende dal contesto e dall’uso che ne vogliamo fare. Se di un certo concetto vogliamo descrivere un insieme di proprietà, si crea un’entità.

ATTRIBUTI CHIAVE

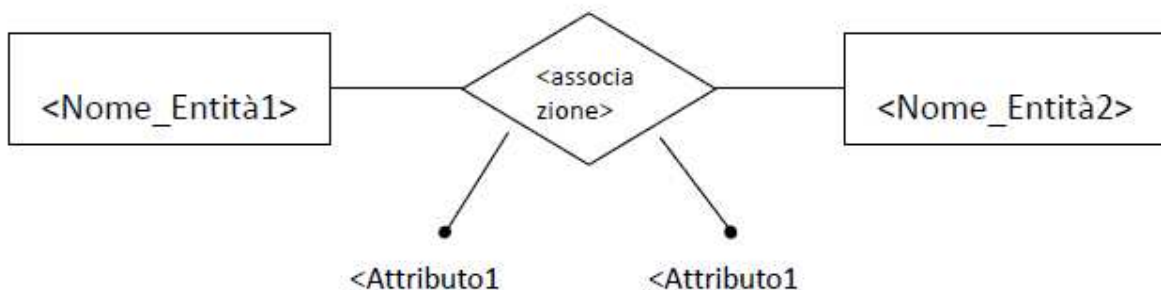
Si definisce CHIAVE CANDIDATA o SUPERCHIAVE una chiave che consente di distinguere un’istanza di entità dall’altra in modo univoco. Ad esempio per l’entità Persona, possiamo dire che l’attributo CodiceFiscale, diverso per ogni individuo, è una CHIAVE PRIMARIA, con minor numero di attributi.

RAPPRESENTAZIONE GRAFICA DI ENTITÀ ED ATTRIBUTI



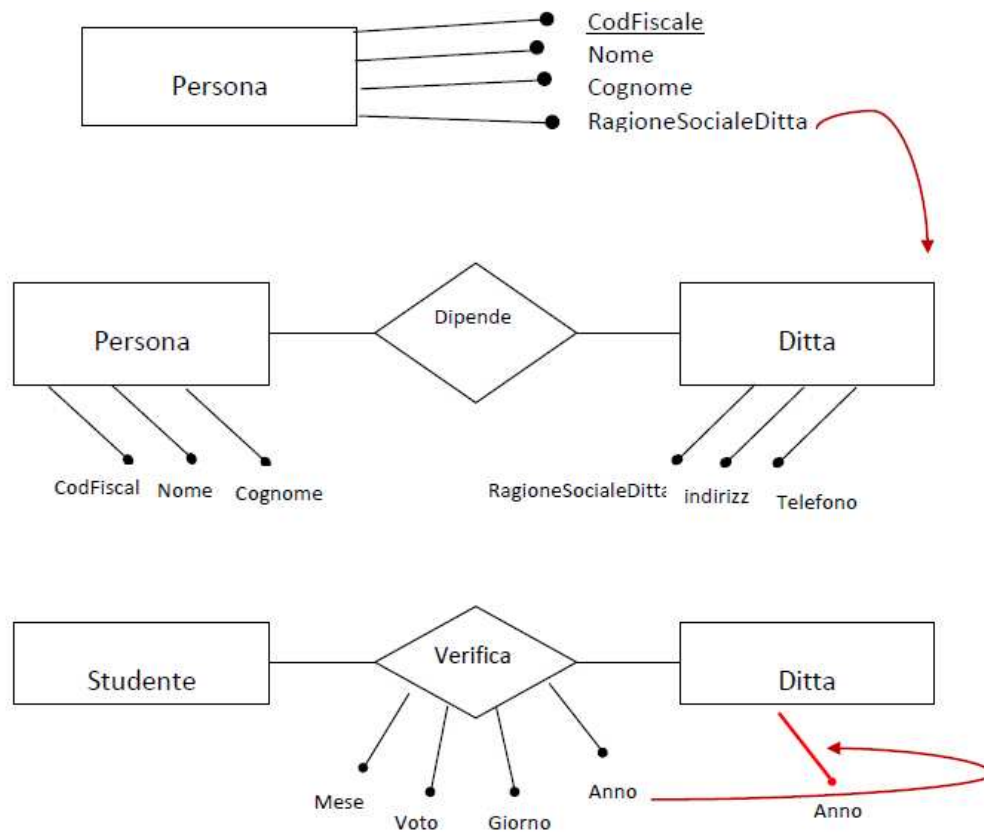
LE ASSOCIAZIONI

L'associazione è un legame logico tra due o più entità rilevanti nella realtà considerata. L'ISTANZA di un'associazione è una combinazione di istanze delle entità che prendono parte all'associazione.



Ogni associazione tra due entità ha due versi. Così, ad esempio, tra l'entità Persona e l'entità Automobile esiste l'associazione "Possiede", mentre nel verso contrario l'associazione è "è posseduta".

Di solito si hanno associazioni binarie, come quella vista poco fa, ma esistono anche associazioni multiple, che collegano più di due entità e che spesso possono essere scomposte in binarie. Possono anche esistere attributi che da un'associazione passano ad un'entità, e questo avviene di solito se un attributo ha un valore fisso e può essere identificato anche nell'entità. Un attributo può diventare un'entità quando ad esso possono attribuirsi vari attributi, evitando la duplicazione.



TIPI DI ASSOCIAZIONI

Un'associazione diretta tra due entità X e Y si dice **TOTALE** quando il legame tra entità deve essere sempre presente e ad ogni elemento di X deve corrispondere almeno un elemento di Y, e la sua rappresentazione grafica è una linea tratteggiata.



Una persona può possedere un conto corrente (parzialità) e un conto corrente deve essere intestato ad almeno una persona (totalità).

Chiamiamo **CARDINALITÀ** di un'associazione tra X e Y la descrizione della molteplicità diretta dell'associazione e di quella inversa, dove per molteplicità intendiamo quante istanze di Y possono trovarsi in relazione con un'istanza di X e viceversa. Le associazioni si classificano in:

1) **UNO AD UNO (1:1)**: si verifica quando ad un'istanza di corrisponde una ed una sola istanza di Y e viceversa.



Associazione diretta: un dirigente scolastico dirige una scuola.

Associazione inversa: una scuola è diretta da un solo dirigente scolastico.

L'associazione diretta potrebbe essere anche parziale, se si considerano ad esempio i presidi in pensione.

2) **UNO A MOLTI (1:N)**: si verifica quando ad un'istanza di X possono corrispondere una o più istanze di Y, e ad ogni istanza di Y deve corrispondere una sola istanza di X.



Associazione diretta: una scuola ha in organico più personale docente.

Associazione inversa: una persona di segreteria lavora in una sola scuola.

3) **MOLTI A MOLTI (N:N)**: si verifica quando ad ogni istanza di X possono corrispondere una o più istanze di Y e viceversa



LA PROGETTAZIONE LOGICA: IL MODELLO RELAZIONALE

Il passo successivo nella progettazione concettuale di una base di dati è la progettazione logica, che consiste nel trasformare la rappresentazione ancora astratta in una più efficiente, detta SCHEMA LOGICO RELAZIONALE. In sostanza si tratta di convertire il diagramma E/R in un insieme di tabelle e nella definizione delle operazioni da compiere sullo schema. Definiamo schema di una relazione il nome della stessa e la lista dei suoi attributi, in questo modo:

Persona(Cognome:Stringa(30),Nome:Stringa(20),Età: Intero, Sesso:Booleano)

o anche:

Persona (Cognome, Nome, Età, Sesso)

Esempi:

1) Rappresentazione per elencazione:

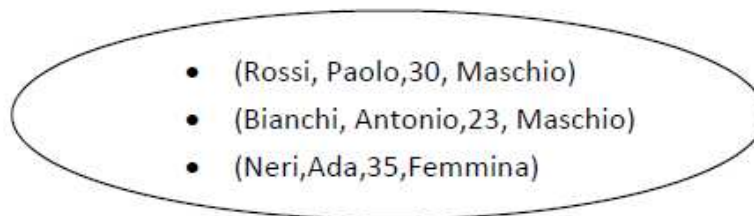
PERSONA (Cognome, Nome, Età, Sesso) = {
(Rossi, Paolo, 30; Maschio) *tupla*
(Bianchi, Antonio, 23, Maschio) *tupla*
(Neri, Ada, 35, Femmina) *tupla*

2) Rappresentazione mediante tabella:

PERSONA	COGNOME	NOME	ETÀ	SESSO
<i>Tuple (righe)</i>	Rossi	Paolo	30	Maschio
	Bianchi	Antonio	23	Maschio
	Neri	Ada	35	Femmina

colonne (attributi)

3) Rappresentazione insiemistica:



Per ogni relazione deve esistere una CHIAVE. In generale una relazione può ammettere diverse chiavi che si dicono CANDIDATE, ma tra queste ne viene scelta una, detta CHIAVE PRIMARIA, che viene sottolineata nello schema. I legami tra le relazioni si realizzano proprio usando tali chiavi.

DAL DIAGRAMMA E/R ALLO SCHEMA RELAZIONALE

Lo schema relazionale si ricava dal diagramma E/R applicando le regole di derivazione, per rappresentare:

1) Entità e attributi

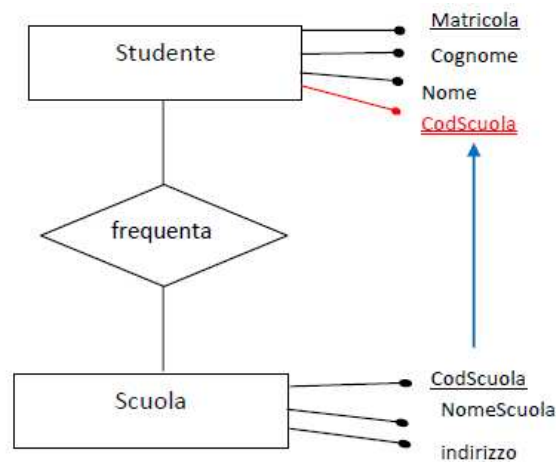
- 2) Associazioni 1:1, 1:N, N:N
- 3) Gerarchie di classi

In particolare possiamo affermare che:

- 1) Ogni entità diventa una relazione, cioè una tabella
- 2) Ogni attributo dell'entità lo diventa nella relazione ed è rappresentato mediante colonna
- 3) La chiave dell'entità diventa chiave nella relazione.

RAPPRESENTAZIONE DELLE ASSOCIAZIONI

In un'associazione di tipo 1:N tra due entità A e B, la chiave primaria di A diventa CHIAVE ESTERNA di B, cioè essa funge da puntatore logico alla tupla dell'altra relazione (A) alla quale è associata. Ricordiamo che non si può cancellare una tupla la cui chiave primaria compaia in almeno una tupla di altre relazioni come chiave esterna.

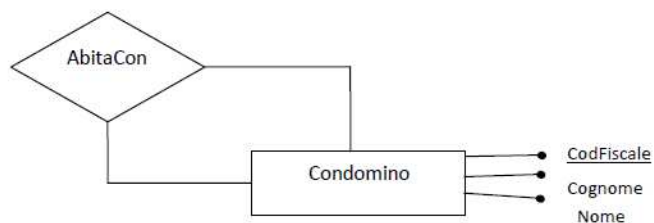


Il meccanismo delle chiavi esterne comporta di sicuro una duplicazione delle informazioni, per questo, spesso si preferisce usare una chiave detta ARTIFICIALE, costituita da un numero progressivo o contatore.

Le associazioni di tipo 1:1 sono un caso particolare delle 1:N e seguono le loro stesse regole, anche se generalmente si preferisce in questi casi riunire i due tipi di entità in un'unica relazione contenente tutti gli attributi dell'una e dell'altra entità.

Le associazioni di tipo N:N tra due entità A e B si crea mediante una nuova relazione avente almeno le chiavi primarie di A e di B, ed eventualmente anche attributi propri.

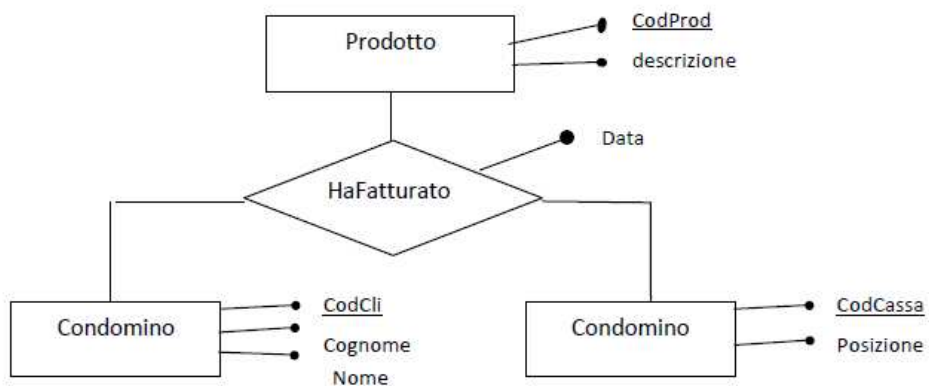
Un caso particolare di associazioni 1:N o N:N è quello in cui l'entità di partenza è uguale a quella di arrivo. In questo caso si opera allo stesso modo delle associazioni con più entità e si parla di associazione RICORSIVA.



Condomino (CodFisc; Cognome, Nome)

AbitaCon(CodFisc1, CodFisc2)

Esistono anche associazioni non binarie, ma esse vengono in genere trattate come quelle binarie.



Linguaggio SQL

Nell'[ingegneria del software](#), il termine **architettura multi-tier** o **architettura multi-strato** (spesso definita con l'espressione inglese **n-tier architecture**) indica un'[architettura software](#) di tipo [client-server](#) per [sistemi distribuiti](#), in cui le varie funzionalità del software sono logicamente separate ovvero suddivise su più strati o livelli software differenti in comunicazione tra loro- Ciascuno strato è in comunicazione diretta con quelli adiacenti ovvero richiede ed offre servizi allo strato adiacente in maniera concettualmente simile a quanto accade con le architetture di rete a strati.

Negli anni novanta si diffuse l'architettura Client-Server a due livelli, mentre negli anni 2000 l'impiego più diffuso di un'architettura multi-tier è **l'architettura a tre livelli**.

Three-tier è un'architettura client-server in cui l'interfaccia utente, i processi logico funzionali ("regole aziendali"), l'archiviazione informatica dei dati e l'accesso ai dati sono sviluppate e mantenute come moduli indipendenti, la maggior parte delle volte su piattaforme separate.

L'ARCHITETTURA A TRE LIVELLI

Il three-tier è un modello di architettura software e allo stesso tempo uno schema di progettazione software. Oltre ai vantaggi abituali di software modulare con interfacce ben definite, l'architettura three-tier è destinata a consentire a qualsiasi dei tre livelli di essere aggiornati o sostituiti indipendentemente dal cambiamento di requisiti o tecnologia. Ad esempio, un cambiamento di sistema operativo nel livello di presentazione interesserebbe solo il codice di interfaccia utente.

In genere, l'interfaccia utente viene eseguita su un desktop PC o workstation e utilizza un'interfaccia utente grafica standard, la logica di processo funzionale può essere costituita da uno o più moduli separati in esecuzione su una workstation o applicazioni server, e un RDBMS in un database server o mainframe contiene i dati di archiviazione logica del computer. Il livello intermedio può essere anche multi-tier

Three-tier architecture ha i seguenti tre livelli:

Livello di presentazione

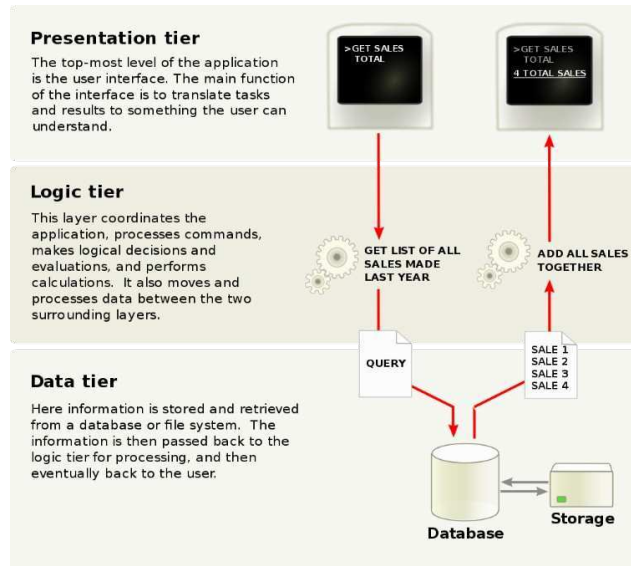
Questo è il livello più alto dell'applicazione. Il livello di presentazione mostra le informazioni relative a servizi come merce online, acquisti, e i contenuti del carrello della spesa. Comunica con altri livelli attraverso i risultati di output al livello browser/client e tutti gli altri livelli della rete.

Livello applicazione (business logic, la logica di primo livello, l'accesso ai dati di secondo livello, o di livello intermedio)

La logica di primo livello viene tirato fuori dal livello di presentazione e, come suo proprio livello, controlla la funzionalità di un'applicazione eseguendo elaborazioni dettagliate.

Livello dati

Questo livello è costituito da server database. Qui le informazioni vengono memorizzate e recuperate. Questo livello mantiene i dati neutrali e indipendenti da applicazioni server o da logica di business. Fornendo informazioni del proprio livello inoltre migliora la scalabilità e le prestazioni.



EasyPHP

EasyPHP è una piattaforma di sviluppo web di tipo WAMP che permette di far funzionare localmente un server web (quindi senza connettersi ad un server esterno) basato sull'interprete PHP. Lanciato nel 1999, è stato il primo pacchetto WAMP esistente.

EasyPHP non è solo un software, ma un ambiente di sviluppo comprendente un server web Apache, un server di database MySQL, un interprete di script PHP e un amministratore di database MySQL con interfaccia grafica chiamato phpMyAdmin. Tutti questi componenti sono installati insieme per avere in un volta sola tutto il necessario per iniziare lo sviluppo locale di siti web in PHP.

EasyPHP dispone di un'interfaccia d'amministrazione che permette di gestire gli utenti, l'avvio e lo spegnimento dei server.

Il server Apache crea automaticamente di default un dominio virtuale (in locale) all'indirizzo di localhost (<http://127.0.0.1>).

EasyPHP può essere utilizzato come applicazione portatile, per esempio attraverso una chiave USB.

Componenti

I componenti di base sono:

- **Microsoft Windows:** il sistema operativo;
- **Apache:** il web server;
- **MySQL:** il database management system (o database server) con SQLite e relativi tool grafici;
- **PHP:** il linguaggio di scripting.

Capitolo 7

Sistemi multimediali

6.1) Rappresentazione digitale dei diversi tipi di informazione: Il suono.