

# La progettazione del software

---

## Contents

---

- Passaggio dai requisiti alla realizzazione
- Il processo di design
- Strategie di design
- Attributi di qualità di un progetto
- Decomposizione del sistema
- Architetture software
- Design collaborativo
- Error Handling & Fault tolerance
- Convalida e verifica
- Standard IEEE 1016 for Software Design Description

## Progettazione e progetto

---

Il cliente desidera un nuovo sistema software, poiché non ne esiste uno che automatizzi un proprio processo, o perché il sistema esistente non lo soddisfa.

In ogni caso, il documento dei requisiti, ci dice tutto circa il problema che il sistema deve risolvere.

La fase di **progettazione** è il processo creativo di trasformazione del problema in una soluzione.

La descrizione della soluzione è chiamata **progetto**.

## Dai requisiti al progetto

---

Esempio: desideriamo realizzare una nuova casa che abbia:

- Una stanza per 3 bambini;
- Una stanza matrimoniale;
- Una cucina;
- Un bagno;
- Riscaldamento ed aria condizionata;
- ...Acqua corrente ed elettricità

*Questi sono dunque i nostri requisiti...*

## Dai requisiti al progetto

---

A questo punto, soltanto in fase di progettazione verrà deciso che:

- Le stanze da letto vanno collocate al primo piano;
- La cucina al pianterreno;
- Occorre massimizzare le dimensioni della camera dei bambini;
- La casa sarà in stile coloniale piuttosto che stile "ranch".

Ovviamente, esistono più progetti che soddisfano i requisiti specificati...

## Scelte di progetto

---

- Ad una stessa specifica corrispondono più progetti;
- Durante la fase di design vengono effettuate delle *scelte di progetto* che poi influenzano la realizzazione del sistema software;
- Le scelte di progetto devono poter cambiare senza influenzare radicalmente l'intero progetto;
- E' necessario saper anticipare i cambiamenti nei requisiti del software (design for change, Parnas, 1972).

## Progetto concettuale e tecnico

---

Nel trasformare dei requisiti in un sistema funzionante, occorre tener conto delle necessità sia del cliente, che del team di sviluppo. Per tale motivo, si passa da una vista concettuale ad una vista tecnica del sistema

- La vista concettuale dice al cliente esattamente **cosa** il sistema fa;
- La vista tecnica dice al team di sviluppo **come** occorre realizzare il sistema.



## Vista concettuale

---

### Caratteristiche:

- Scritta nel linguaggio del cliente (niente “gergo” tecnico);
- Descrive le funzioni del sistema;
- E’ indipendente dall’implementazione;
- E’ collegata al documento dei requisiti.

### Contenuti:

- Provenienza dei dati;
- Trasformazioni dei dati all’interno del sistema;
- Interfacce utente e descrizione dei reports
- Opzioni previste per gli utenti;
- Temporizzazione degli eventi;
- Descrizione di reports e schermate.

## Vista tecnica

---

### Contenuti:

- Descrizione delle componenti hardware e delle relative funzioni;
- Gerarchia e funzioni delle componenti software;
- Strutture dati.

## Processo di progettazione

---

- Comprensione del problema
  - Osservare il problema da diversi punti di vista
- Identificare le possibili soluzioni
  - Scegliere la più appropriata, anche in base all'esperienza dei progettisti e alle risorse
- Descrivere un'astrazione della soluzione
  - Notazioni grafiche, formali o descrittive
- Ripetere il processo per l'intero progetto.

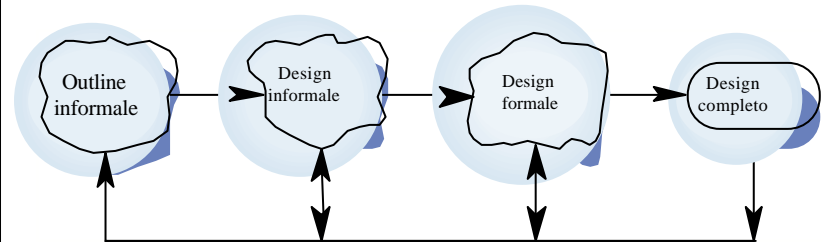
## Processo di progettazione

---

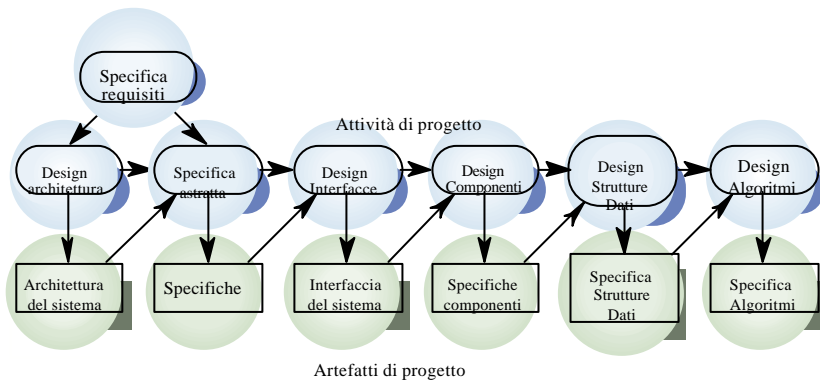
- Un progetto potrebbe essere modellato mediante un grafo diretto, costituito da entità in relazione tra loro;
- Il sistema dovrebbe essere descritto a differenti livelli di astrazione;
- Il processo di progettazione è costituito da fasi sovrapposte.

## Design informale -> Design formale

---



## Fasi del processo di progettazione

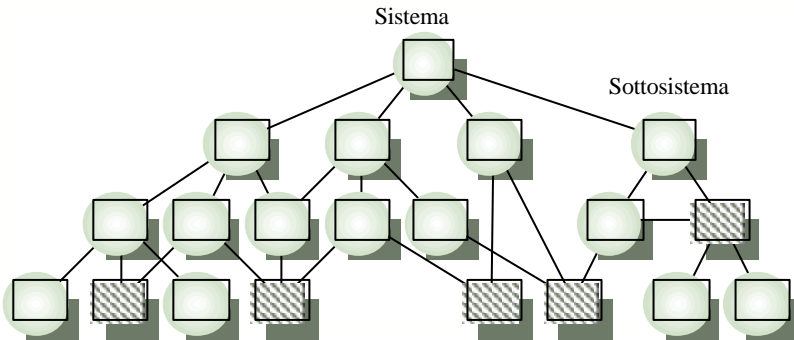


## Fasi del processo di progettazione

- *Design architetturale:* Identificazione dei sottosistemi
- *Specifica astratta:* Specifica dei sottosistemi
- *Design interfacce:* Descrizione delle interfacce dei sottosistemi
- *Design componenti:* Decomposizione dei sottosistemi in componenti
- *Design strutture dati:* Progettazione delle strutture atte a contenere i dati del problema
- *Design algoritmi:* Design delle soluzioni algoritmiche al problema

## Struttura gerarchica del progetto

---



## Progettazione top-down

---

- Si parte dai componenti più in alto nella gerarchia, per poi muoversi verso il basso;
- Nella pratica, la progettazione di un grosso sistema non è mai top-down;
  - Alcune parti del sistema sono progettate prima di altre;
  - Riutilizzo di esperienze (e componenti).

## Descrizione del progetto

---

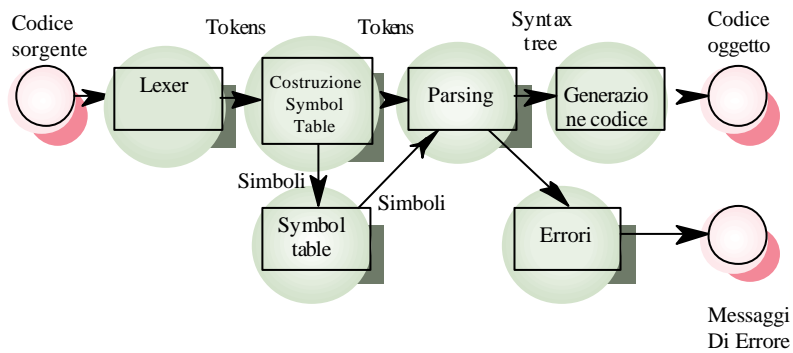
- *Notazioni grafiche*. Per visualizzare le relazioni tra i componenti;
- *Linguaggi di descrizione dei programmi*. Basati su linguaggi di programmazione, ma con maggiore flessibilità per descrivere concetti astratti;
- *Descrizioni informali*, espresse in linguaggio naturale;
- Tutte le notazioni sopra citate possono essere utilizzate nella progettazione di un grosso sistema software.

## Strategie di progetto

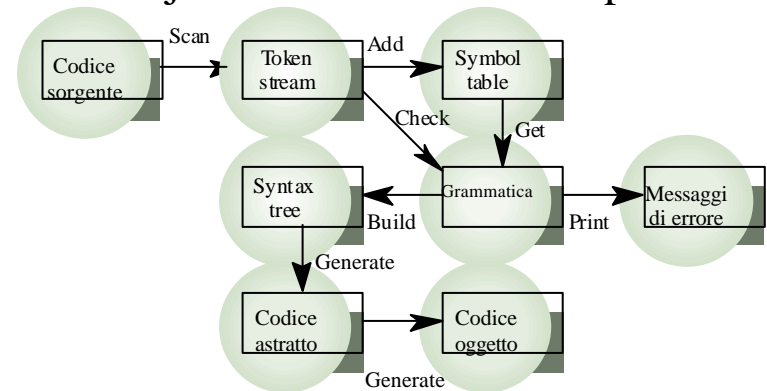
---

- Design funzionale
  - Il sistema è progettato da un punto di vista funzionale. Lo stato del sistema è rappresentato in maniera centralizzata e condiviso dalle funzioni che operano su di esso;
- Design object-oriented
  - Il sistema è visto come una collezione di oggetti interagenti, avente ognuno il proprio stato. Oggetti possono essere istanze di classi; essi comunicano mediante invocazione di metodi.

## Esempio: vista funzionale di un compilatore



## Vista object-oriented di un compilatore



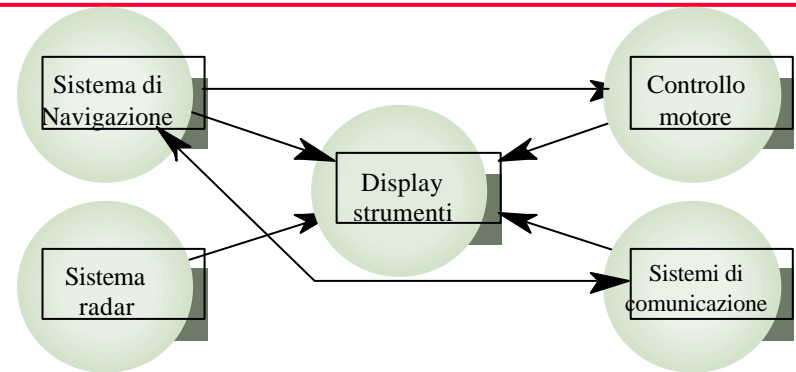
## Strategia mista

---

- Spesso gli approcci funzionale e object-oriented risultano essere complementari
- Sta ai progettisti la scelta della giusta combinazione per ciascun sottosistema

## Sottosistemi di un aereo

---



## Oggetti di alto livello

---

- Il sistema di navigazione
- Il sistema radar
- I sistemi di comunicazione
- Il display della strumentazione
- Il sistema di controllo motori
- ...

## Funzionalità del sistema (Livello sottosistemi)

---

- Visualizzazione tracciati radar (sottosistema radar)
- Compensazione velocità vento (sottosistema di navigazione)
- Riduzione potenza (sottosistema motore)
- Indicazione stato di emergenza (sottosistema strumentazione)
- Ricerca frequenza radiofaro (sistema di comunicazione)
- ...

## Oggetti di basso livello

---

- Stato motore
- Posizione aereo
- Altimetro
- ...

## Qualità di un progetto

---

- La qualità di un progetto è spesso influenzata da priorità organizzative
- Un buon progetto dovrebbe essere il più efficiente, il più economico, il più manutenibile, il più affidabile, ecc.
- Le caratteristiche di qualità valgono sia per progetti function-oriented che per progetti object-oriented

## Coesione

---

- Misura l'omogeneità di un modulo, per avere un alto grado di coesione è necessario che tutto ciò che esso contiene faccia riferimento a caratteristiche comuni (ad esempio un modulo incapsula la struttura dati ed implementa le operazioni relative ad una tabella, un altro modulo si occupa della sua visualizzazione).
- Un componente dovrebbe implementare una singola funzione o entità logica
- La coesione è una componente desiderabile di un progetto, in quanto ogni cambiamento da effettuare è bene che sia localizzato su un singolo componente coeso;
- E' possibile identificare diversi livelli di coesione.

## Livelli di coesione

---

- Coesione incidentale (bassa)
  - Le parti di un componente sono semplicemente messe assieme
- Associazione logica (bassa)
  - I componenti che svolgono simili funzioni sono raggruppati
- Coesione temporale (bassa)
  - I componenti attivati ad istanti temporali vicini sono raggruppati
- Coesione procedurale (bassa)
  - Gli elementi di un componente sono raggruppati per creare una singola sequenza di controllo

## Livelli di coesione (II)

---

- Coesione comunicativa (medium)
  - Tutti gli elementi di un componente operano sugli stessi input e producono lo stesso output
- Coesione sequenziale (medium)
  - L'output di una parte di un componente è l'input di un'altra parte
- Coesione funzionale (strong)
  - Ciascuna parte del componente è necessaria all'esecuzione di una singola, specifica funzione
- Coesione a livello di oggetto (strong)
  - Ciascuna operazione consente l'accesso o la modifica di attributi dell'oggetto

## Coesione come attributo di un progetto

---

- E' spesso difficile classificare il livello di coesione
- Ereditare attributi da superclassi abbassa il livello di coesione
- Per comprendere un componente, le superclassi devono essere esaminate
- Program comprehension assistita da object browsers

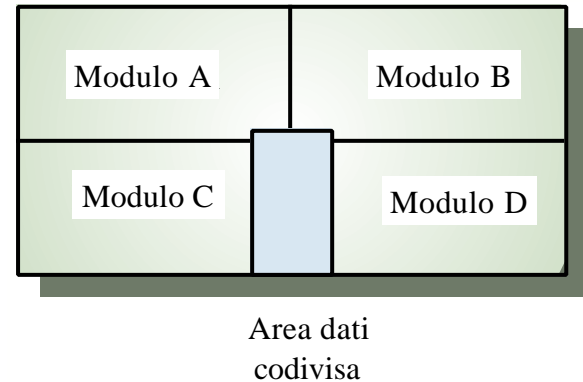
## Accoppiamento

---

- Misura il grado di interrelazione tra i moduli di un sistema software, minore è lo scambio di informazioni tra moduli minore è il grado di accoppiamento del sistema
- Basso accoppiamento indica che modifiche ad un componente difficilmente impattano su altri componenti
- Variabili condivise sono indice di alto accoppiamento
- Un basso accoppiamento può essere ottenuto mediante una decentralizzazione dello stato (es. oggetti) e mediante comunicazione tra componenti tramite parametri o scambio di messaggi

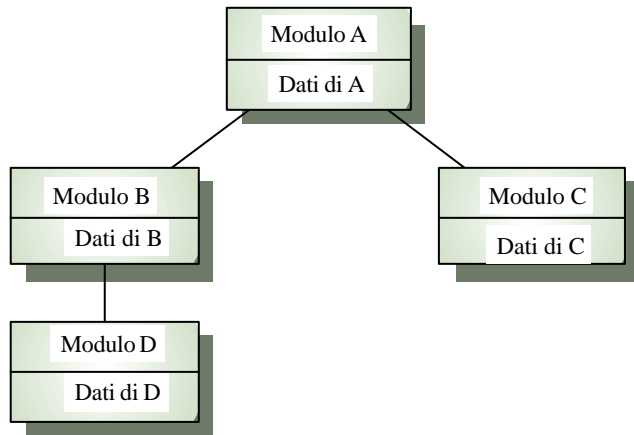
## Alto accoppiamento

---



## Basso accoppiamento

---



La progettazione

33

## Accoppiamento e ereditarietà

---

- I sistemi object-oriented sono caratterizzati da basso accoppiamento per via dell'assenza di dati condivisi e grazie alla comunicazione tra oggetti mediante scambio di messaggi
- Però, una classe è accoppiata alla sua superclasse: modifiche agli attributi o ai metodi delle superclassi si propagano alle sottoclassi. Tali modifiche devono essere accuratamente controllate

La progettazione

34

## Comprensibilità

---

- E' legata a diverse caratteristiche di un componente
  - *Coesione*. Può il componente essere compreso in maniera autonoma?
  - *Naming*. I nomi usati sono significativi?
  - *Documentazione*. Il progetto è ben documentato?
  - *Complessità*. Quanto sono complessi gli algoritmi utilizzati?
- In maniera informale, una alta complessità indica molte relazioni tra parti differenti del design, per cui la comprensione risulta difficile
- Molte metriche di qualità di un progetto sono orientate alle misure di complessità, sebbene il loro uso risulta limitato

## Modificabilità - Adattabilità

---

- E' essenziale saper anticipare il cambiamento durante la progettazione: ciò si ha mediante la cosiddetta *progettazione per il cambiamento* (Parnas, 1972).
- Le spinte alla modifica di un sistema software provengono da fattori dipendenti sia dall'ambiente esterno che dall'ambiente di sviluppo stesso:
  - **Cambiamenti nel sistema di calcolo per cui il prodotto viene sviluppato;**
  - **Evoluzione delle specifiche dei requisiti;**
  - **Cambiamenti per il miglioramento delle prestazioni;**
  - **Inadeguatezza delle specifiche dei requisiti;**
  - **Evoluzione per motivi di mercato**

## Modificabilità - Adattabilità

---

- Un progetto è facilmente manutenibile se:
  - L'accoppiamento è basso
  - Il progetto è ben documentato e la documentazione è aggiornata
  - Esiste una corrispondenza tra i diversi livelli del progetto (visibilità del progetto)
  - Ciascun componente è altamente coeso
- Per adattare un progetto, deve essere possibile tracciare i collegamenti tra i componenti di un progetto, in maniera tale da poter analizzare le conseguenze dei cambiamenti

## Adattabilità ed ereditarietà

---

- L'ereditarietà migliora significativamente il livello di adattabilità. I componenti possono essere adattati senza modifiche derivando una sotto classe e introducendo in essa le opportune modifiche
- Però, al momento in cui la profondità di una gerarchia di ereditarietà cresce, essa diventa enormemente complessa, per cui va periodicamente rivista e ristrutturata

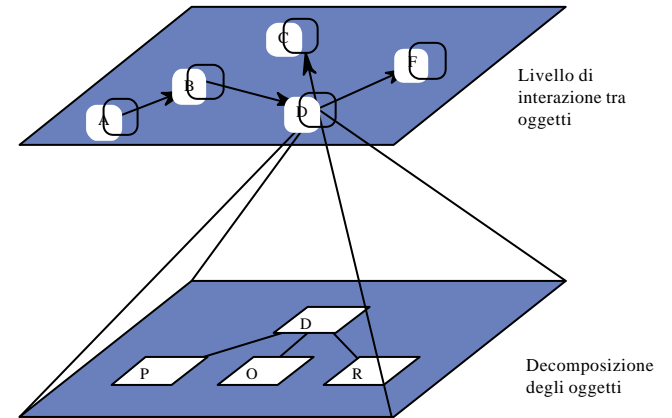
## Affidabilità

---

- Un alto grado di affidabilità è un elemento sostanziale per un prodotto software
- Una buona progettazione aumenta l'affidabilità complessiva
- Una opportuna suddivisione in sottosistemi del sistema complessivo predispone in minor grado all'errore durante la realizzazione;
- Importante soprattutto per applicazioni in tempo reale per sistemi critici;

## Tracciabilità del progetto

---



## Decomposizione del sistema

---

- La progettazione di un sistema software è un'attività che deriva dal fatto che la complessità intrinseca di un sistema software è tutt'altro che banale.
- Non rimane dunque che dominare tale complessità: decomporre cioè il problema in più sottoproblemi, facendo in modo che la "somma" delle complessità dei singoli sottoproblemi sia inferiore alla complessità del problema totale.
- La decomposizione risulta, inoltre, particolarmente utile per poter consentire la realizzazione delle singole parti da parte di gruppi diversi di programmatori operanti in parallelo.

## Decomposizione del sistema (II)

---

La progettazione consiste nel determinare l'insieme di componenti e di interfacce tra i componenti soddisfacente i requisiti (DeMarco, 1982).

Secondo Wasserman, è possibile decomporre il sistema in 5 modi:

- Decomposizione modulare;
- Decomposizione data-oriented;
- Decomposizione event-oriented;
- Decomposizione basata su input e output del sistema;
- Object-oriented design.

## Moduli – definizioni (I)

---

Un modulo è una unità software che mette a disposizione risorse e servizi computazionali. Esso si compone di una Interfaccia e di una Sezione Implementativa.

- **Interfaccia:** definisce risorse e servizi messi a disposizione dei “clienti”. Essa è completamente visibile ai clienti
- **Sezione Implementativa:** implementa le risorse ed i servizi messi a disposizione del cliente. Essa è occultata, cioè non è accessibile ai clienti.

## Moduli – definizioni (II)

---

- Un modulo può essere compilato separatamente.
- Un modulo può essere incluso, e quindi impiegato, nell’ambito di svariati e differenti sistemi software.
- I servizi e le risorse offerte da un modulo possono essere utilizzate da un ‘cliente’ inferenziandole e/o istanziandole.

## Moduli – caratteristiche

---

Attraverso un modulo è possibile implementare:

- **Astrazioni funzionali:** il modulo mette a disposizione, tramite la sua interfaccia, una serie di funzionalità. Seguendo il principio dell'Information Hiding è necessario annullare gli effetti collaterali e non utilizzare variabili globali.
- **Data encapsulation:** incapsulare una struttura dati regolamentandone l'accesso attraverso le sole operazioni ammesse su di essa. (N.B. una struttura dati non è incapsulabile utilizzando unicamente procedure, se ne perderebbe lo stato).

## Relazioni tra Moduli - 1

---

Due moduli  $m_i$  ed  $m_j$  sono in relazione **USA** ( $m_i$  USA  $m_j$ ) se, affinché il modulo  $m_i$  risulti conforme alle sue specifiche, è necessario che anche  $m_j$  sia conforme alle proprie specifiche.

Due estremi:

- ciascun modulo usa tutti gli altri.
- nessun modulo usa alcun altro modulo del sistema.

Nel primo caso si registra un elevato grado di accoppiamento. Nel secondo caso l'accoppiamento è nullo, ciò può essere indice di gravi disfunzioni quali, ad esempio, la spropositata duplicazione di codice.

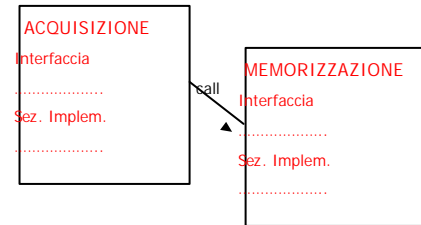
## Relazioni tra Moduli - 2

La relazione **COMPONENTE\_DI** deriva dalla scomposizione top-down (raffinamento) di moduli in sottomoduli. Essa dà luogo ad una struttura gerarchica dei moduli di un sistema che può essere rappresentata graficamente attraverso un albero.

Tra tutti i moduli che partecipano ad una relazione **COMPONENTE\_DI** solo quelli che non hanno alcun altro modulo componente (le foglie dell'albero) hanno un'esistenza fisica effettiva nel sistema finale. I moduli intermedi sono contenitori logici di moduli, sarebbe piuttosto complicato giungere alla comprensione di tutti e soli i moduli foglia

## Relazioni tra Moduli - 3

La realizzazione fisica di una relazione **USA** non necessariamente si realizza tramite una chiamata di procedura. Infatti un accesso ad informazioni condivise o anche uno scambio di messaggi. E' poi anche vero che una chiamata di procedura non necessariamente implica una relazione **USA**.



**ACQUISIZIONE:** recupera dati da un dispositivo esterno.

**MEMORIZZAZIONE:** memorizza i dati in una tabella.

Il buon funzionamento di **ACQUISIZIONE** non è influenzato dal fatto che la procedura di **MEMORIZZAZIONE** chiamata funzioni correttamente.

**Quindi, in questo caso, non sussiste la relazione USA.**

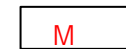
## Structure Charts

Le Structure Charts costituiscono una metodologia adatta al progetto di sistemi software mediante la decomposizione funzionale.

Le caratteristiche principali delle Structure Charts sono:

- rappresentazione grafica della struttura modulare di un sistema;
- struttura gerarchica;
- relazioni tra moduli;
- flussi principali di dati e controlli;
- principali decisioni ed iterazioni che regolano l'attivazione dei moduli.

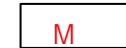
## Notazione di base - 1



Modulo di Nome **M**

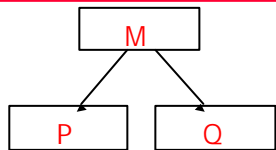


Il Modulo **M** Attiva il Modulo **N**

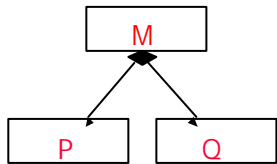


Il Modulo **M** Attiva il Modulo **N** passandogli il dato **P** ed il controllo **F**. Il Modulo **M** ritorna il dato **Q** ed il controllo **L**.

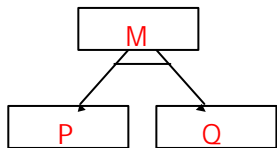
## Notazione di base - 2



Il Modulo **M** Attiva i Moduli **P** e **Q**.

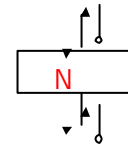


**M** attiva il modulo **P** o il modulo **Q**, in maniera mutuamente esclusiva in base al verificarsi di un condizione. (Selezione di Moduli)



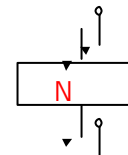
**M** attiva ciclicamente i moduli **P** e **Q**. (Iterazione di Moduli)

## Notazione di base - 3



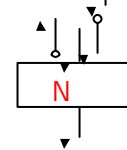
### Moduli Afferenti.

Ottengono informazioni dai subordinati e le passano a quelli superiori. Possono, eventualmente, modificare i dati trasferiti.



### Moduli Efferenti.

Ricevono le informazioni dai superiori e le passano a quelli subordinati. Possono, eventualmente, modificare i dati trasferiti.

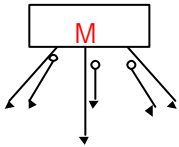


### Moduli di Trasformazione.

Ricevono le informazioni dai superiori a cui le ritornano dopo averle trasformate.

## Notazione di base - 4

---



### Moduli di Coordinamento.

Controllano e gestiscono l'attivazione di moduli a loro subordinati, passando loro informazioni

### Sequenza di Attivazione.

In strutture "complesse" la sequenza di attivazione prevede che si attivino prima i moduli a livelli gerarchici superiori, quindi gli altri. Non esiste alcun ordinamento particolare tra i moduli appartenenti allo stesso livello.

## Punti chiave

---

- Il design è un processo creativo
- Le attività di progetto includono il design dell'architettura, la specifica del sistema, la progettazione dei componenti e delle strutture dati, la progettazione degli algoritmi
- La decomposizione funzionale considera il sistema come insieme di unità funzionali
- La decomposizione object-oriented considera il sistema come insieme di oggetti