

- Lezione 30 - Esercizi

- Lezione 31

- ➔ Dodicesimo Blocco

Le misure, vengono usate in fase di progettazione preliminare e dettagliata.

Affronteremo in particolare:

- le *misure intermodulari* ➔ che permettono di quantificare la dipendenza tra moduli in base all'architettura software determinata in fase di progettazione preliminare.
- le *misure intramodulari* ➔ che invece quantificano i singoli moduli (nella fase di progettazione dettagliata)

Un modulo come sappiamo è una sequenza contigua di istruzioni, delimitata da alcuni elementi e che ha un certo identificatore (quando si pensa a un modulo si pensa a una parte di software che può esser compilata indipendentemente).

È importante saper misurare la soluzione architeturale in quanto tutte le decisioni che prendiamo in questa fase hanno impatto significativo sul software risultante, in particolare su attributi di qualità come facilità di implementazione, affidabilità, manutenibilità e riusabilità (NB: affidabilità importante non solo per software critico).

Le misure ci permettono quindi di avere un feedback che ci faccia capire se le caratteristiche del software fanno al caso nostro (soddisfano requisiti etc...).

Il concetto di modulo introdotto in fase di progettazione preliminare poi collegato in modo diretto (relazione 1 a 1) con i moduli a livello di codice.

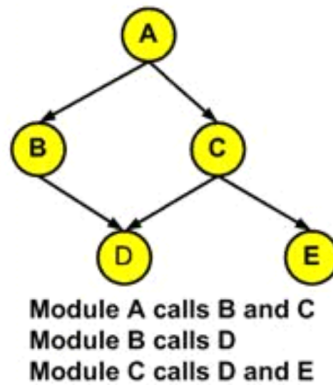
Le connessioni tra moduli identificate a livello di progettazione quindi tipicamente saranno riferimenti tra moduli a livello di codice, così come le interfacce di scambio dati identificate in fase di progettazione corrispondono alla condivisione dati a livello di codifica.

Torniamo ora al concetto di architettura software (anche chiamata structure chart).

Un'architettura software sappiamo essere un insieme di componenti che hanno tra loro relazioni di dipendenza.

Si può quindi concettualizzare l'architettura software usando un grafo, dove i nodi sono i moduli e gli archi le relazioni di dipendenza.

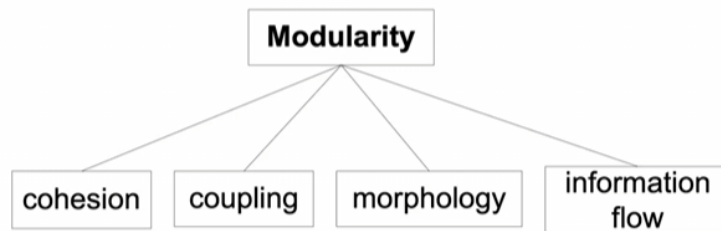
Le relazioni possono rappresentare diverse cose (es. chiamata di procedura, flusso di dati etc...).



Quando si definisce l'architettura dei moduli è importante definire il valore della modularità, che si ricorda essere il grado con cui un software è definito da componenti discrete tale che il cambiamento di una di esse comporta minimo impatto sulle altre componenti.

Si ricorda quindi che alta modularità è desiderabile, in quanto se si ha bassa modularità allora è più facile fare errori → difficile manutenzione, meno riusabili, meno affidabili etc... Sappiamo che per misurare la modularità si fa riferimento alle due metriche di Coesione e Coupling.

Per misurare in modo più completo la modularità introduciamo altre metriche che sono maggiormente legate alla struttura del grafo con cui rappresentiamo l'architettura dei moduli. Si aggiungono a cohesion e coupling le due metriche Morfologia e Information Flow.



- Coesione → grado con cui un modulo individualmente realizza un task ben definito
- Coupling → grado di interdipendenza tra moduli  
 Si voleva massimizzare coesione e minimizzare coupling per ottenere elevato livello di modularità.
- Morfologia → misura la forma della structure chart cercando di capire qual è la forma migliore in funzione della modularità
- Information Flow → considera il flusso di informazioni tra moduli →  
 interconnessione tra moduli non solo dal punto di vista di scambio dati ma anche dal punto di vista del flusso di controllo (flusso di dati scambio di dati, flusso di controllo interdipendenza, misureremo ciò con tecniche che ci permetteranno di misurare fan-in e fan-out del modulo che sono rispettivamente quanto un modulo riceve in ingresso e invia in uscita)

Iniziamo dalla morfologia. Essa è caratterizzata da: Size (numero di nodi e archi), Depth (si vuole un grafo che sia quanto più simile a un albero possibile → depth come distanza radice-nodo), Width (massimo numero di nodi tra tutti i livelli), Edge-to-Node Ratio (misura di connettività nel grafo, ossia quanto è denso → rapporto archi nodi). Chiaramente per avere elevata modularità si vuole bassa connettività.

Vediamo un semplice esempio

Size:

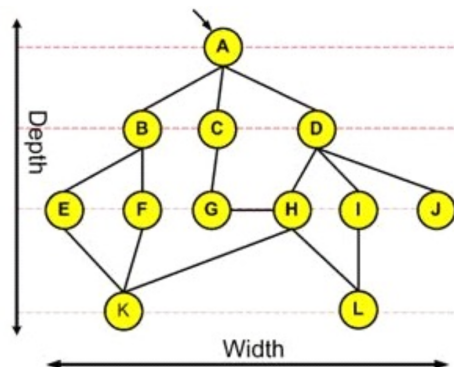
12 nodes

15 edges

Depth: 4

Width: 6

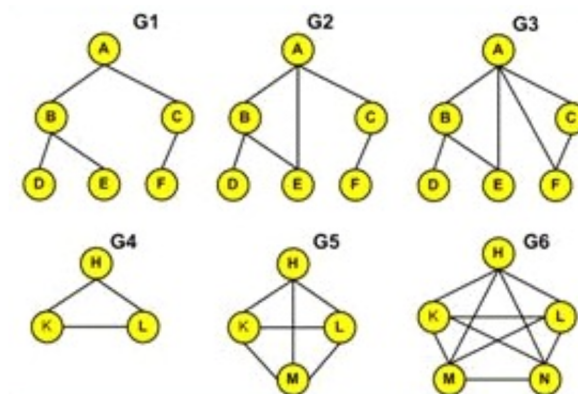
$e/n = 1.25$



Maggiormente è densa l'architettura, maggiori sono i legami di interdipendenza tra moduli. Un concetto importante per misurare la Morfologia è il concetto di Tree Impurity: esso misura quanto il grafo dell'architettura è lontano da un albero.

Sia  $m(G)$  quanto il grafo è diverso da un albero → minore  $m(G)$ , migliore la morfologia. Più sono i cicli, peggiore è la morfologia, in particolare non si vogliono ovviamente grafi fortemente connessi o clique.

Si vogliono evitare situazioni come G5 e G6 e avvicinarsi quanto più possibile a G1.



Come misurare quantitativamente  $m(G)$ ?

$$m(G) = \frac{\text{number of edges more than spanning tree}}{\text{maximum number of edges more than spanning tree}}$$

$$m(G) = \frac{2(e - n + 1)}{(n - 1)(n - 2)}$$

- Example

$$m(G_1) = 0 \quad m(G_2) = 0.1 \quad m(G_3) = 0.2$$

$$m(G_4) = 1 \quad m(G_5) = 1 \quad m(G_6) = 1$$

Spanning tree → albero che è sottografo di un grafo connesso e connette tutti i nodi.

Il numero massimo di archi per un grafo in più rispetto allo spanning tree è la differenza tra numero di archi di quel grafo con n nodi completo e quel grafo ma come spanning tree → n-1 archi spanning tree e  $(n(n-1))/2$  per clique, la differenza porta a  $(n-1)(n-2)$  (da qui il denominatore).

Sopra invece numero di archi tra spanning tree e grafo effettivo.

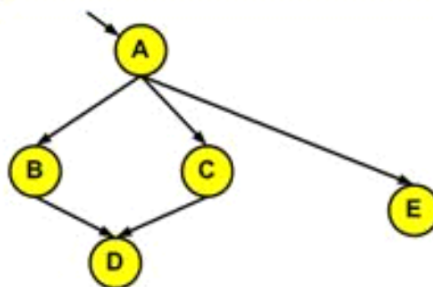
$M(G)$  varia tra 0 e 1.

Un altro per misurare sempre la Morfologia è l'Internal Reuse.

Essa indica il grado con cui i moduli sono riusati all'interno dello stesso prodotto.

Per misurarlo si usa  $r(G)$  che è  $e - n + 1$ .

Meno  $r(G)$  significa meno riuso → migliore morfologia in quanto minor interdipendenza



Module D is reused by  
modules B and C

**Example:**

$$r(G1) = 0 \quad r(G2) = 1$$

$$r(G3) = 2 \quad r(G4) = 1$$

$$r(G5) = 3 \quad r(G6) = 6$$

Questa tecnica di misurazione tuttavia non tiene conto del numero di chiamate fatte da un modulo ad un altro (non so D quante volte viene chiamato da B e C) e inoltre non si tiene

conto della dimensione dei moduli → bisogna utilizzare oltre che questa misura anche Tree Impurity per capire la morfologia.

Per approfondire come i moduli dipendano effettivamente uno dall'altro dobbiamo misurare l'Information Flow.

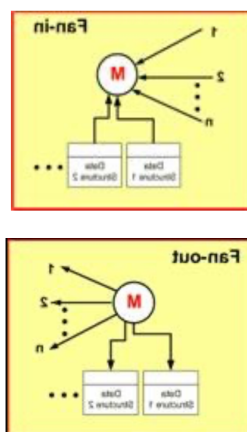
Questa misura assume che la complessità di un modulo dipenda da due fattori principali: la complessità intrinseca del modulo (il suo codice) e la complessità della sua interfaccia (ossia quanto è aperto all'ambiente circostante, quanto è dipendente da altri moduli o altri moduli dipendono da lui).

Il livello totale di information flow in un sistema è attributo intermodulare, tuttavia è possibile anche misurare l'information flow tra un singolo modulo e il resto del sistema come quindi attributo intramodulare.

Per misurare l'information flow si devono contare le connessioni tra un modulo con il resto dei moduli del sistema (fan-in e fan-out di un modulo), inoltre si assume che le misure di information flow sono basate su flussi di informazioni sia locali (ossia legati a un modulo che chiama un altro modulo se diretto o il valore di ritorno di un metodo invocato se indiretto) e globali (quando l'informazione tra moduli è condivisa e accessibile a tutti). L'obiettivo dell'information flow è misurare quanto i moduli dipendano dagli altri, quindi può essere utilizzato per individuare le parti critiche del sistema e capire se la nostra progettazione può essere pericolosa in termini di affidabilità, manutenibilità, riusabilità etc...

- Fan-In di un modulo M: rappresenta il numero di flussi locali (diretti e indiretti) che terminano nel modulo M in aggiunta al numero di flussi globali (strutture dati globali utilizzate dal modulo M) → tutto ciò che entra nel modulo sia in termini di valori di ritorno che in termini di dati presi da strutture dati globali fa parte del Fan-in del modulo.

-Fan-Out: rappresenta viceversa tutto ciò che “esce del modulo”: il numero di flussi locali (diretti o indiretti) che partono dal modulo M più il numero di aggiornamenti che M fa su strutture dati globali.



È importante misurarli in quanto se un modulo ha alto fan-out allora esso probabilmente influenza molti altri moduli, viceversa elevato fan-in indica che il modulo dipende da molti altri moduli.

Un modulo con elevato fan in e fan out è sicuramente un modulo che è al centro del sistema, mentre uno con basso fan in e fan out alla periferia del sistema.

L'obiettivo è ovviamente ridurre il fan in e fan out per ogni modulo in quanto se elevato allora indica un modulo complesso che potrebbe portare a errori in quanto il modulo può eseguire più di una funzione → 1) se devo modificarlo devo modificare molti altri moduli e inoltre 2) se voglio modificare una funzione devo guardare in più punti, non ho raffinato bene i moduli affinché ognuno svolga un unico compito ben preciso.

Si introduce quindi la misura dell'Information Flow.

- Information flow (IF) for module  $M_i$  [Henry-Kafura, 1981]:

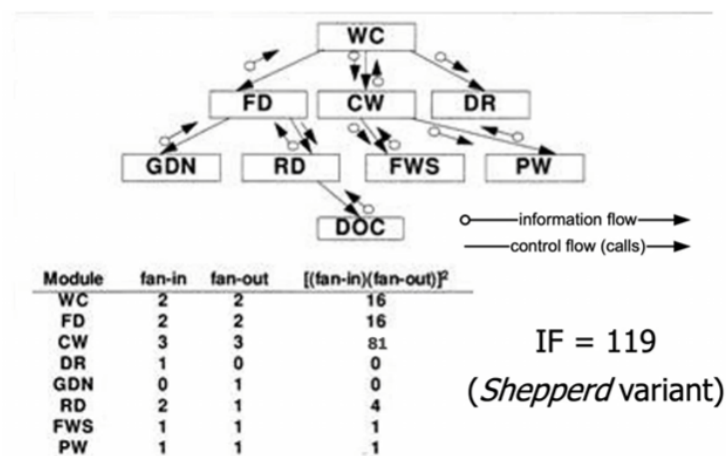
$$IF(M_i) = [fan-in(M_i) \times fan-out(M_i)]^2$$

- IF for a system with  $n$  modules is:

$$IF = \sum_{i=1}^n IF(M_i)$$

Questa misura tiene conto dell'information flow sia come flusso dati che come flusso di controllo (quindi si contano anche le chiamate di un modulo a un altro senza effettivo scambio di dati).

Esiste una variante di information flow measure, la Shepperd, che si limita al flusso di dati. Vediamo un esempio di questa misura con la variante Shepperd. Abbiamo una buona structure chart (in quanto ad albero). Ogni arco rappresenta chiamate fatte tra moduli (flusso di controllo) mentre le frecce con pallino flusso di dati.



(autoesplicativo, vedi frecce con pallino fuori e dentro per contare fan in e out per poi calcolare la shepperd measure dove si conta solo flusso di dati).

Quindi in generale non possiamo misurare direttamente la modularità, ma per capire il livello dobbiamo misurare i suoi 4 sottoattributi che invece sono direttamente misurabili. Quando si passa alla parte di progettazione dettagliata ci si focalizza di più sulla complessità del singolo modulo, ora introduciamo delle misure strutturali (fanno riferimento alla struttura del modulo) che valgono sia in fase di progettazione dettagliata che in fase di implementazione (la soluzione è trovata in fase di progettazione dettagliata, in fase di implementazione solo traduzione della soluzione in linguaggio specifico).

Quando si parla di misure strutturali come detto si fa riferimento alla struttura del modulo, tre principali componenti:

- Struttura del flusso di controllo → sequenza di istruzioni del programma
- Flusso di dati → si tiene traccia dei dati creati e gestiti dal modulo
- Struttura dei dati → organizzazione dei dati indipendenti dal modulo

È importante usare queste misure perché importanti in fase di reverse engineering (manutenibilità), testing (un passaggio importante è path coverage in questo senso, ossia cercare di percorrere tutti i possibili percorsi di esecuzione per rilevare eventuali errori), ristrutturazione codice, data flow analysis (si ricorda in questo senso l'importanza di modelli come il data flow diagram)...

Come rappresentare la struttura del singolo modulo? Attraverso un grafo del flusso di controllo. Una volta rappresentata la struttura potremo definirne la complessità usando la complessità ciclomatica, misura strutturale che vedremo si può fare sia a livello di codice che di modulo come grafo di flusso.

Come si esercita il flusso di controllo in un modulo?

Si definiscono delle strutture di controllo di base (BCS Basic Control Structure), che sono null'altro che i meccanismi essenziali di control flow usati per costruire la struttura logica del programma. Tre tipi di BCS:

- Sequenza (serie di istruzioni senza che vi siano altre BCS ad influire su di esse),
- Selezione (if, then, else...),
- Iterazione (while, repeat until ...).

Esistono anche strutture di controllo avanzate ACS che permettono di introdurre nuovi concetti come chiamata di funzione/procedura, ricorsione, interrupt e concorrenza.

Vedremo come queste strutture di controllo di base vengono combinate per definire la struttura complessiva del modulo.