CAPITOLO 3

Gli algoritmi logici per l'analisi funzionale

Introduzione

Nel precedente capitolo sono state descritte a livello visuale e "informale" le strutture matematiche mediante le quali rappresentare in maniera sistematica i vari elementi costituenti gli schemi di analisi funzionale. In particolare sono state descritte le strutture per la rappresentazione dei blocchi funzionali, quelle per la rappresentazione dei link funzionali ed è stata introdotta e commentata la particolare matrice di adiacenza strutturata mediante la quale rappresentare, attraverso i "bus", le connessioni esistenti tra i vari blocchi di una generica rete.

Il presente capitolo si occuperà di "tradurre" in maniera formale tutte le rappresentazioni strutturate presentate nel capitolo 2, operando, in particolare, una riproposizione in un linguaggio comprensibile al calcolatore di tutte le entità fin ora descritte solo in maniera visuale.

3.1 Descrizione delle modalità operative e definizioni fondamentali

L'obiettivo è riuscire a rappresentare al calcolatore tutte le entità necessarie alla realizzazione di uno studio funzionale, dando a queste "entità", delle valenze "fisiche" oltre che semplicemente formali. Dunque, rappresentare i blocchi funzionali e le connessioni esistenti tra questi, ovvero i link. Osservando tipici schemi di analisi funzionale, ed analizzando la letteratura a riguardo, l'idea si è concentrata verso i grafi, strutture matematiche formalmente molto simili alle reti funzionali. Di conseguenza è stato immediato associare il concetto di rappresentazione matematica dei grafi mediante matrici alla rappresentazione delle reti funzionali attraverso strutture matematiche simili. Queste valutazioni, trovano una concreta rappresentazione all'interno di un linguaggio di programmazione che renda effettive le fasi di "traduzione" formale degli schemi di rappresentazione di analisi funzionale in un codice orientato agli oggetti appropriato.

In riferimento a tali considerazioni, data la sua estrema potenzialità e versatilità, è stato scelto il linguaggio "C++" prima e successivamente "Matlab". Quest'ultimo, infatti, mette a disposizione una serie di strumenti mediante i quali la riproposizione delle idee teoriche poste nella fase di modellazione del problema di rappresentazione di reti funzionali diviene semplice e quasi puntuale, intendendo con puntuale la possibilità di ottenere una sorta di traduzione "uno-a-uno" dei vari elementi.

Al fine di ottenere la descrizione delle strutture elaborate mediante linguaggio di programmazione sono state utilizzate tutta una serie di funzioni "built-in" nei linguaggi di programmazione utilizzati: dapprima, infatti, è stato affrontato il problema attraverso il linguaggio C++, successivamente a causa della difficoltà di implementazione per le interfacce grafiche e per la definizione di molte funzioni, l'attenzione si è rivolta verso Matlab.

È noto che qualsiasi linguaggio di programmazione mette a disposizione dell'utente una serie di "tipi di dato" semplici. Ad esempio nel C++, per poter rappresentare numeri interi non decimali, il tipo di dato utilizzato è l'"*int*", mentre per poter definire delle stringhe di testo il tipo di dato da utilizzare è il "*char*". Sono numerosi i tipi di dato semplici messi a disposizione dal C++; ciò rende estremamente versatile e dinamica la programmazione. Nonostante ciò l'esistenza di molti tipi di dato non è ciò che rende speciale il linguaggio. Piuttosto la sua grande potenzialità si concretizza nel momento in cui si fa uso delle strutture dati interne. Le strutture sono dei "tipi di dato aggregati", ovvero dei tipi di dato che consentono il raggruppamento in un unico blocco di un certo numero di altri tipi elementari. La semplice definizione appena riportata è già sufficiente a far notare la forte analogia esistente tra i tipi di dato aggregati, (le strutture), e le definizioni strutturate fornite nel capitolo 2 riguardo gli elementi descrittivi della metodologia di analisi funzionale. Le potenzialità appena descritte, sono molto importanti nell'approccio alla programmazione di sistemi complessi, ma, allo stesso tempo, non si prestano per la tipologia di lavoro che si è intrapresa. La soluzione alternativa infatti, è l'utilizzo delle strutture ma nel linguaggio Matlab. L'idea base di raggruppare più informazioni all'interno di un "contenitore" unico, permette la gestione di una notevolissima quantità di dati, semplicemente interrogando le strutture stesse con i comandi disponibili nel linguaggio ed altri creati ad hoc.

Questa particolare tipologia di struttura permette di essere identificata e richiamata attraverso il nome assegnato al momento della creazione.

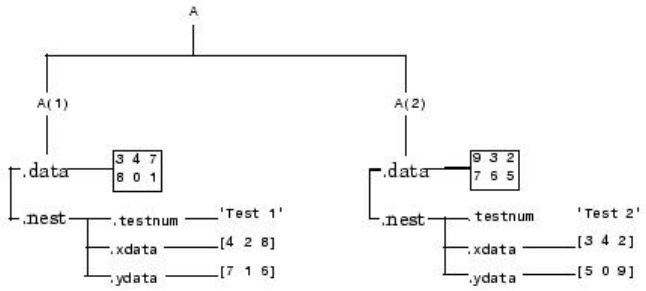

Figura 3-1: *Struttura generale in Matlab.*

Nell'esempio è possibile vedere come ogni struttura ha un campo nome che la identifica. Per richiamare un ramo della struttura, è sufficiente digitare il nome della struttura a cui si è interessati e in uscita si ottengono tutte le informazioni inserite al momento della creazione.

Per estendere l'uso di queste strutture dati e soprattutto per garantire una facilità di accesso ai dati da parte dell'utente, sono state create delle 'Structure's Cell'. Questo tipo di struttura dati permette di creare una "famiglia" di strutture, tra di loro indipendenti, con la possibilità di connettere le une con le altre.

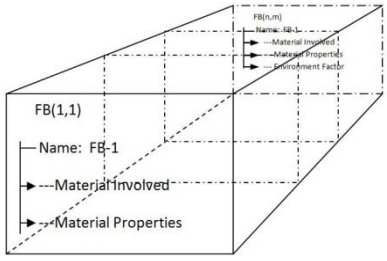

Figura 3-2: *Rappresentazione di una struttura multi-layer.*

Come è illustrato in figura, ogni layer ha un nome differente che all'interno del programma diventa identificativo del layer stesso. Ogni struttura ha dei campi indipendenti, fissati in numero di 5 come default del sistema, ma sostituibili e modificabili in funzione delle esigenze del progettista. All'inserimento dell'utente di un nuovo nodo, il programma abilita la creazione di una struttura dati. Successivamente alla creazione di almeno tre nodi ed al collegamento di una coppia di questi, viene generata una matrice di dimensioni pari al numero dei nodi creati, nella quale verranno conservati i collegamenti

(link) tra i vari nodi (Blocchi funzionali). Questa matrice sarà la matrice di adiacenza del grafo [20] [21].

Una struttura, in Matlab, è definita mediante la parola riservata "*struct*". Nel caso delle "*Structures Cells"*, la nomenclatura di dichiarazione è la seguente: "***s{i, j}.s1{ i1,j1}= name;"***

Nel momento in cui questa è definita bisogna associarle un label, ossia un nome (s,s1), mediante il quale è possibile identificarla, quindi richiamarla per effettuare le opportune modifiche. All'interno della struttura sono, invece, inseriti i cosiddetti "*membri*", i quali caratterizzeranno le "qualità" della struttura stessa, definendo, in particolare, ciò che potrà essere contenuto in essa. Vincolo essenziale per le *structures cells (SC)* è la definizione di un nome differente per ognuna di esse. Per far questo il programma valuta i nomi delle strutture inserite e ne propone uno differente, standard, modificabile. Tale nome se modificato, è sottoposto al controllo per valutare eventuali duplicati. Per quanto riguarda le forme di "tipo" contenibili da una struttura, non esistono limitazioni a riguardo; possono essere dei tipi semplici, ma anche delle strutture. Nel momento in cui una struttura ne ha un'altra tra i suoi membri, la stessa può essere definita "*struttura complessa*". È possibile, inoltre, inserire tra i membri della ***SC*** la struttura stessa; in questo caso essa si definisce *annidata, (nesting*) [22]. In figura 3.1 è riportata una tipica definizione di una struttura. Nell'immagine sono in evidenza gli elementi identificativi della

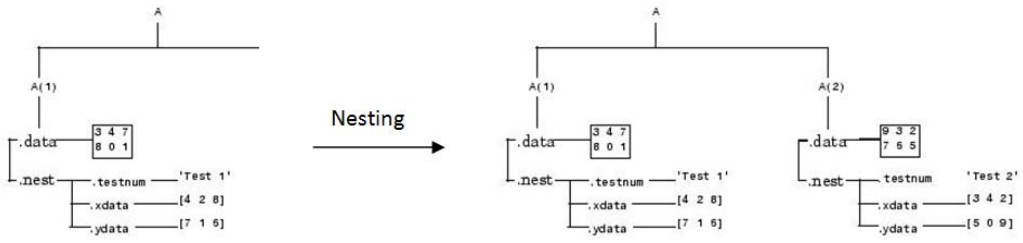

Figura 3-3: *Creazione di una struttura funzionale semplice.*

struttura, ovvero l'etichetta, i membri e le modalità di contenimento di questi ultimi all'interno della struttura stessa.

Per la definizione formale di una porzione di linguaggio, capace di costruire un modello di struttura funzionale, verrà utilizzato il nome di funzione (function).

3.2 La definizione degli elementi strutturati

Il primo passo verso la realizzazione del programma in grado di gestire una generica rete funzionale di prodotto è stato quello di descrivere tutti gli elementi strutturati definiti nel capitolo 2 mediante il tipo predefinito di stuttura dati citato: *struct*. In particolare sono stati definiti sei tipi di dato astratti: il "*Blocco_Funzionale*" ed il "*Blocco_Esterno*", mediante i quali definire i corrispettivi blocchi di analisi funzionale, ed i quattro link funzionali "*Link_Energia*", "*Link_Forza*", "*Link_Materiale*" e "*Link_Segnale*" . Associati a ciascuna delle suddette definizioni sono stati realizzate delle funzioni di inizializzazione ed utilizzazione delle strutture.

3.2.1 Il Blocco Funzionale

Si riporta di seguito il raffronto tra la definizione visuale della struttura di blocco funzionale definita nel capitolo 2 e la traduzione informatica del concetto:

Descrizione Visuale/Modello	Rappresentazione Informatica
	s{i,j}{i1,j1}={'Label'}; s{i,j}{i1,j1}={'Valore Incidenza'}; s{i,j}{i1,j1}={'Indice Incidenza'}; s{i,j}{i1,j1}={'Misura Energia'}; s{i,j}{i1,j1}={'Pozzo Sorgente'};

Figura 3-4: *Rappresentazione in linguaggio Matlab del blocco funzionale.*

Per la definizione del blocco funzionale è stata utilizzata una *SC* con i campi che accettano qualsiasi tipo di valore. Nel caso specifico *double*, in cui contenere valori numerici ("*valore_incidenza*", "*indice_incidenza*", "*label*", "*pozzo_sorgente*"), altri tre di tipo *char**, ovvero degli array di caratteri mediante i quali definire la funzione rappresentativa del blocco ("*funzione*"), l'unità di misura associata al valore di incidenza

("*misura*") e l'unità di misura associata al valore energetico contenuto nel campo "*pozzo_sorgente*" ("*misura_energia*").

La *SC* si comporta da puntatore al "Nodo_Blocco_Funzionale", denominato, appunto, Blocco_Funzionale. Il tipo complesso così definito è utilizzato all'interno del programma per definire elementi di tipo "Blocco_Funzionale".

La funzione che inizializza e "*costruisce*" il blocco funzionale riceve come parametro un blocco funzionale ed assegna ai campi di tipo double della struttura il valore "0", ai campi *char** la stringa "*vuoto*". L'inizializzazione delle strutture (e delle variabili in generale), anche non essendo strettamente necessaria, è una procedura fortemente consigliata per non incorrere in problemi di non corretta allocazione ed utilizzo della memoria virtuale utilizzata dal computer durante le fasi di esecuzione del programma.

La funzione così fatta da la possibilità all'utente di assegnare i valori desiderati alle varie variabili contenute nella struttura. La funzione generica di inserimento variabili è la seguente, fig(3.5)

```
function  [n a m p env punti_acquisiti_x...
    punti_acquisiti_y,vet_risp]=inserisci_dati_struttura(f,s,campi,matrice_nodi_i,...
    matrice_nodi_j,vet_risp,Nodes_List,Block_created)...
```

Figura 3-5: *Inserimento dati nel blocco funzionale.*

3.2.2 Il Blocco Esterno

La definizione del Blocco Esterno è sostanzialmente simile a quella vista per il blocco funzionale, anche se, per ovvie ragioni, molto più semplice. Presentando solo due campi (fig. 3.4), il blocco esterno è modellato a livello informatico mediante una struttura estremamente semplice, come tale è anche il metodo di completamento dei campi .

Analogamente a quanto visto per il Blocco Funzionale, viene definito un puntatore al Nodo_Blocco_Esterno, al momento del collegamento tra i due, che rappresenta la struttura vera e propria, costituita dai soli due campi "*label*" (un intero) e "*funzione*" (un array di caratteri); quest'ultimo è l'unico campo il cui completamento è assegnato all'utente. Il tutto viene svolto applicando la funzione

3.3 I link funzionali

Pur essendo teoricamente diversi dai blocchi funzionali, è stato osservato nel precedente capitolo come la definizione strutturata dei link funzionali fosse formalmente simile. Data la similitudine, è chiaro che anche per questi la rappresentazione informatica essenzialmente ricalchi quella già osservata per le due precedenti strutture. Si tratta sempre di collegare adeguatamente una *SC* ad un'altra, sfruttando la costruzione dei blocchi funzionali e la funzione di inserimento dei dati. Sostanzialmente, dunque, i concetti sono simili a quelli appena descritti nei precedenti paragrafi in più vi è la funzione di costruzione del link tra due nodi. In figura 3.6, è rappresentata una generico link tra due nodi.

```
1  Functional_Block_Name  A

2  Action_Developed  Prova

3  Material_involved

4  Material_Property

5  Environment_Impact_Factor

6  Force_Link
 0  1   0
0   0   0
0   0   0
```

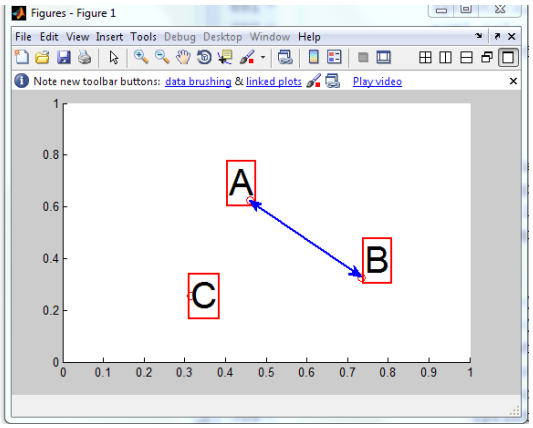

Figura 3-6: *Definizione del link di forza.*

Una volta creato il link di forza è necessario utilizzare i campi di modifica ed inserimento propri dei blocchi funzionali per editare i valori associati al link. In tal modo ogni nodo creato contiene tutte le informazioni che desidera l'utente, in più è dotato di puntatore al nodo ad esso collegato ed inoltre contiene le informazioni sulla tipologia di link. Con questo sistema si ottiene uno schema compatto del blocco funzionale, attorno al quale ruotano tutti gli altri blocchi connessi in modo biunivoco. Questa strategia di programmazione permette di snellire la creazione di funzioni e la possibilità di effettuare

errori sia di programmazione che di inserimento dati. Il discorso si ripete analogo per le altre tre tipologie di link.

3.4 Descrizione dei vari moduli del programma

Una volta definite le strutture matematiche mediante le quali rappresentare le entità costituenti gli studi di analisi funzionale è stato possibile definire la fase implementativa vera e propria del programma di gestione. L'idea è stata quella di suddividere l'applicativo in "moduli", ognuno dei quali contenente una serie di metodi attraverso i quali svolgere gruppi di operazioni della stessa natura. Il primo modulo che sarà discusso è il modulo denominato "Gestione_Blocchi".

Pur essendo stato l'ultimo ad essere realizzato in ordine di tempo, data la necessità di avere chiare le idee riguardo le altre strutture da implementare ed utilizzare (si fa riferimento alla matrice strutturata e alle modalità di gestione dei blocchi in relazione a quest'ultima), è il primo ad essere utilizzato nel momento in cui il programma è lanciato. Per questo motivo è utile definire subito le sue caratteristiche, per poi passare alla descrizione degli altri moduli.

3.4.1 Il modulo di gestione blocchi funzionali

Mediante il modulo di "Gestione_Blocchi" il programma riceve tutte le informazioni preliminari necessarie alla successiva caratterizzazione delle funzioni di definizione e gestione della matrice strutturata e degli algoritmi di verifica; inoltre, sempre all'interno di questo modulo, sono presenti delle funzioni mediante le quali svolgere tutte le operazioni di inizializzazione e definizione dei blocchi funzionali e dei blocchi di ambiente esterno.

La funzione *Inizia_sessione* assolve il compito di gestire l'inizio delle procedure. In particolare, dopo aver inviato in stampa sullo schermo del computer una serie di indicazioni utili all'utente al fine di svolgere correttamente le procedure di completamento interattivo, richiede allo stesso di specificare se continuare a costruire altri blocchi funzionali o meno. Il sistema non procede alle successive operazioni se non sono costruiti, solo nella prima fase, almeno tre blocchi funzionali. Tali informazioni vengono conservate all'interno delle due variabili "*n_b_t*" (numero blocchi totale) e "*n_b_e*" (numero blocchi

esterni), variabili passate per riferimento dunque automaticamente aggiornate dalle procedure interne al metodo (le variabili hanno ed i nomi hanno carattere simbolico rappresentativo).

Costruiti i blocchi si parte con la memorizzazione di tutte le informazioni sulla quantità dei blocchi creati e sul numero di blocchi costruiti. In tal modo è possibile raggiungere un qualsiasi nodo del sistema nel modo più veloce possibile grazie ad una matrice di memorizzazione sempre aggiornata. Tale matrice è costituita associando al numero (quantità) del blocco funzionale due indici i quali permettono una ricerca su vettori ordinati. Una volta eseguite le procedure, saranno disponibili due vettori contenenti l'insieme complessivo dei blocchi funzionali e dei blocchi di ambiente esterno completamente definiti. Si ricorda che, per evitare problemi di ambiguità, la procedura assegna automaticamente il label ai blocchi.

Per aggiungere altri blocchi o modificare gli elementi esistenti sono state implementate delle funzioni di supporto. Tali funzioni lavorano sulla matrice dei nodi appena definiti e consentono, rispettivamente, di aggiungere un ulteriore blocco funzionale in coda al corrispettivo vettore ed un ulteriore blocco di ambiente esterno in testa al corrispondente vettore. Inoltre i metodi aggiornano automaticamente i label dei nodi. Per ora è sufficiente sapere che, grazie a queste due ultime funzioni è possibile aggiungere nodi in qualsiasi momento durante le fasi di generazione della rete funzionale.

3.4.2 **Il blocco funzionale esploso**

Durante la stesura dell'analisi funzionale si ripercorre la rete creata in modo critico per epurarla da eventuali errori di costruzione e per ampliare delle sottofunzioni contenute in un particolare blocco funzionale. L'analisi funzionale, infatti, ha come proprietà intrinseca quella di evidenziare prima e studiare approfonditamente poi, la funzione di ogni componente di un sistema complesso. In prima battuta è difficile capire se quel particolare componente ha in se altre funzioni che meritano una stesura più dettagliata dell'analisi

funzionale. Da questo nasce il sistema che permette di entrare all'interno di un blocco funzionale complesso e scomporlo in altri più semplici. Guardando semplicemente alla teoria dei grafi è facile osservare che una volta definita una rete, la modifica di questa comporta una complessa rielaborazione. Ritornando all'analisi funzionale la problematica per rendere più fluida la gestione dell'analisi stessa, è stato costruito un sistema innovativo di gestione del singolo nodo. Sfruttando le celle di strutture *built-in* Matlab è stato possibile gestire *l'atomizzazione* dell'analisi funzionale senza ripresentare una nuova stesura della stessa. Il sistema di gestione del singolo blocco funzionale, legato direttamente alla matrice di adiacenza e alla matrice dei nodi, si trasforma grazie ad una funzione che si articola nei seguenti passi:

1) Richiesta del nodo da "esplodere",
2) Valutazione dei collegamenti con altri nodi,
3) Annullamento dei vecchi collegamenti,
4) Eliminazione del nodo,
5) Memorizzazione del nodo eliminato,
6) Sostituzione del nodo con i nuovi blocchi funzionali.

La schematizzazione in sei punti identifica i passi che il software compie per gestire una richiesta dell'utente nel caso di scomposizione del nodo in più nodi. Successivamente alla prima richiesta del nodo da esplodere, il programma interroga la matrice di adiacenza per valutare gli eventuali nodi collegati fra loro. Successivamente i collegamenti sono interrotti in quanto il nodo principale è sostituito ed i nuovi blocchi dovranno essere collegati in modo completamente diverso dal nodo singolo. Anche se il nodo master viene cancellato, all'interno del programma si trova un collettore di questi nodi capace di ritornare indietro per valutare ancora una volta una struttura non "esplosa". In questo modo coesistono due tipologie di analisi funzionale: struttura estesa e struttura contratta.
Creati i nuovi nodi sarà possibile effettuare tutte le operazioni di collegamento e modifica comuni a qualsiasi nodo della rete. Questa modifica si ripercuote sia sulla matrice di adiacenza, la quale sarà aggiornata secondo i nuovi nodi ed i nuovi link, sia sulla matrice dei nodi stessa, che in funzione di una matrice di appoggio dei nodi esplosa sarà anch'essa aggiornata.

3.4.3 Il modulo Matrice

È sicuramente il modulo più complesso dell'intero programma e, nello stesso tempo, è il "cuore" del programma stesso. Nel precedente capitolo è stata ampiamente trattata la problematica di definizione delle reti di analisi funzionali mediante grafi; in particolare è stata evidenziata l'analogia esistente tra i grafi e le reti, dunque la possibilità di descrivere e analizzare schemi di analisi funzionale utilizzando algoritmi che lavorassero, appunto, su matrici. Ovviamente tali matrici non sono semplici collezioni di numeri ma ognuno degli elementi è una collezione di strutture ad estensione n-dimensionale, dunque una struttura di strutture ad estensione n-dimensionale. Le due dimensioni "planari" corrispondono al numero dei blocchi costituenti la struttura, le altre dimensioni sono utilizzate per la procedura di esplosione del nodo.

L'analogia tra reti funzionali e grafi è nota, ma non ha mai avuto un forte legame. In letteratura, infatti, si studiano i grafi per le varie problematiche già citate, mentre l'analisi funzionale è frutto di uno studio a più settoriale. A tal proposito nasce l'idea di unire le due letterature e di fondare una nuova architettura di approccio al problema ingegneristico sfruttando le potenzialità offerte da due sistemi di analisi che, anche se molto simili sono sostanzialmente differenti.

3.4.4 Il modulo "Matrice": la creazione della matrice

È noto dal capitolo 2 che ogni elemento della matrice di adiacenza, demandata alla rappresentazione delle interconnessioni tra i vari blocchi funzionali facenti parte di una generica rete, deve contenere una collezione di link forza, segnale, materiale ed energia. Per definire a livello informatico un elemento di questo tipo è stata utilizzata una struttura complessa, i cui membri sono simili a dei puntatori a strutture, in particolare alle strutture definenti le quattro tipologie di link. Seguendo una metodologia di rappresentazione analoga a quella utilizzata per descrivere le strutture destinate alla rappresentazione informatica dei link, di seguito si riporta l'idea di traduzione proposta ed utilizzata al fine

di generare la struttura mediante la quale definire il generico elemento della matrice di adiacenza.

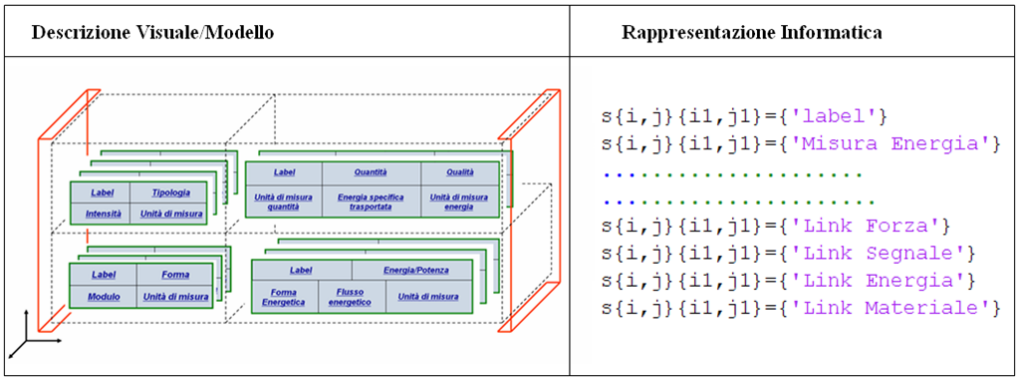

Figura 3-7: *Rappresentazione informatica dell'elemento matrice.*

In particolare una volta costruiti i nodi e definiti le tipologie di link la matrice di adiacenza si aggiorna secondo i collegamenti effettuati dall'utente. Il link creato con il suo valore lo si trova all'interno dello stesso nodo. La struttura contente tutte le informazioni sul nodo, infatti, ne contiene anche delle altre che riguardano sia la tipologia di link sia la connessione del nodo rispetto agli altri. Si ha la possibilità di gestire quattro celle di strutture (dalle dimensioni non definite, e ciò è esattamente quello che si prevedeva per una struttura dati scalabile.), mediante le quali gestire totalmente i collegamenti nodali. A seconda delle occorrenze, il generico elemento matrice potrà contenere un numero diverso di link dello stesso tipo.

In questo modo è stato definito il prototipo della matrice strutturata. Ovviamente, per avere una matrice vera e propria è stato necessario predisporre una funzione di creazione. Al suo interno sono state implementate delle funzioni di definizione dei link funzionali, presentati precedentemente.

La definizione strutturata dell'elemento matrice non comporta una sua reale creazione, ma solo una costruzione potenziale. Quando si creano dei nodi, a sua volta, si crea in background una matrice pari al numero dei nodi creati. Tale matrice è "elastica" ovvero in grado di adattarsi ad una suddivisone del singolo nodo in più nodi. Le funzioni complementari invece, assegnano caratteristiche concrete a ciascun elemento della matrice, il tutto sfruttando le funzioni per la gestione dei nodi. La procedura generata attraverso la funzione consente di entrare in ciascuno dei quattro campi dell'elemento e definire il

numero e le caratteristiche dei link corrispondenti, ciò utilizzando le funzioni di definizione e modifica della tipologia di link, (simili a quelle per i nodi).

Il metodo, per funzionare, ha bisogno di una serie di parametri che vanno dai nomi dei nodi che si creano, di cui si vuole inizializzare un link, la posizione del nodo all'interno della matrice dei nodi, i valori dei link, etc.

La funzione di inizializzazione della matrice include la definizione degli elementi diagonali della matrice. Considerando che, all'interno degli schemi funzionali non esistano loop, quella tipologia di link che genera e termina su di uno stesso blocco funzionale, è evidente che gli elementi diagonali della matrice di adiacenza debbano essere tutti colmati con degli "0". Ogni elemento diagonale è, infatti rappresentativo del collegamento esistente tra un generico nodo "*i*" e se stesso. Il metodo è stato pensato per facilitare il riempimento dei campi degli elementi diagonali.

Di seguito è presentato il diagramma di flusso descrittivo dell'algoritmo di creazione e definizione della matrice.

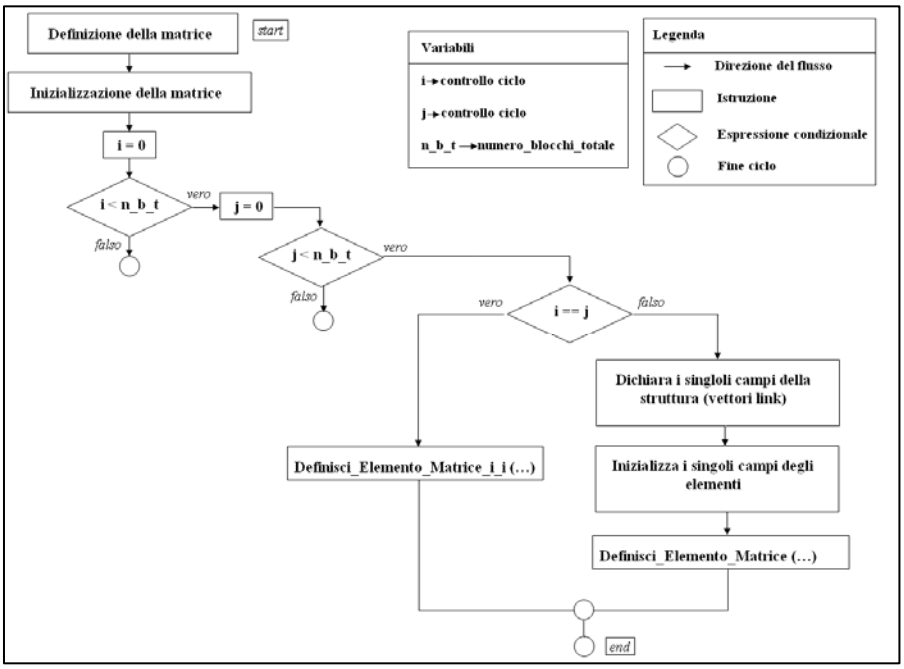

Figura 3-8: *Diagramma di flusso per la creazione della matrice generale.*

Dopo una prima fase di dichiarazione ed inizializzazione della matrice (fasi "obbligatorie" nelle normali procedure di programmazione), mediante due istruzioni "*for*" vengono passati al vaglio tutti gli elementi, pari, in numerosità, al prodotto (numero_blocchi_totale x numero_blocchi_totale). Mediante un'istruzione "*if-else*" si discrimina tra gli elementi diagonali e gli elementi fuori diagonale. Per i primi (ovvero quando i == j) è semplicemente applicata la funzione di definizione matrice. Per i secondi viene svolta "esternamente" la dichiarazione e l'inizializzazione, dopodiché la procedura prevede l'applicazione del metodo discusso in precedenza. Raggiunto il limite di blocchi, dunque definite tutte le connessioni esistenti tra essi la funzione termina. Osservando il codice reale del metodo si riscontrano altre linee di comando non commentate mediante il diagramma di flusso. Tali operazioni sono "relazioni ausiliarie" che non hanno diretta correlazione con l'algoritmo; sono servite unicamente a definire strutture ed elementi di supporto necessarie alla "gestione interna" del metodo di creazione della matrice. La creazione della matrice, in abbinamento alle precedenti definizioni dei blocchi funzionali e dei blocchi di ambiente esterno, di fatto sancisce la completa caratterizzazione informatica di tutti gli elementi della rete funzionale. In pratica, avere a disposizione la matrice dei link

e le definizione dei nodi vuol dire aver tradotto in codice l'intera struttura di analisi funzionale (fig. 3.9)

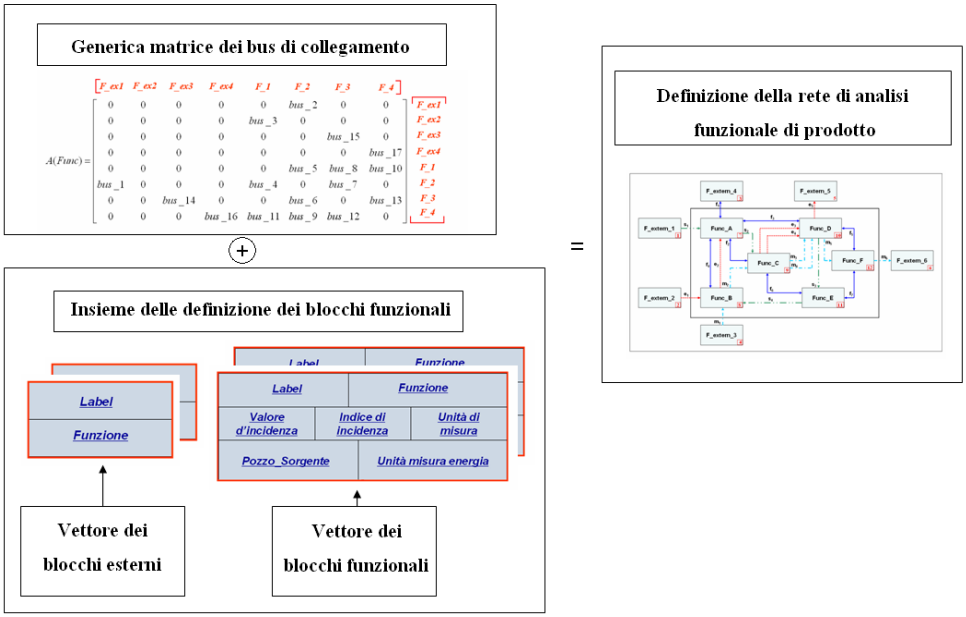

Figura 3-9: *Assemblaggio del codice di programmazione.*

Avendo a disposizione la rappresentazione completa della rete di analisi funzionale, a questo punto non rimane altro che definire, su di essa, le verifiche fisiche necessarie alla sua validazione. La verifica delle connessioni avviene automaticamente, il programma, infatti, all'interno della funzione di creazione e gestione dei link di forza è predisposto per il controllo dei collegamenti. Grazie anche alla disponibilità di un'interfaccia grafica che permette la visualizzazione dei nodi collegati è agevole verificare visualmente le connessioni. In ogni caso, per prevenire errori o il ricollegare nodi già dipendenti il software riconosce la struttura e avverte l'utente impedendogli di riformulare un collegamento esistente. Allo stesso tempo non è possibile lasciare nodi scollegati poiché il sistema avverte l'utente della non completezza della rete. Il software è comunque in grado di garantire l'inserimento di link ulteriori e di apportare modifiche agli stessi.

La costruzione della matrice di adiacenza si estende, quindi, alle quattro tipologie di link individuati all'interno dell'analisi funzionale. In particolare ogni collegamento nodale è caratterizzato da una o più tipologie di link, a tal proposito si ha un'estensione n-dimensionale delle matrici di adiacenza che raccolgono tutti i link utilizzati per le connessioni nodali. La funzione che permette il collegamento tra i nodi compie diverse

azioni in funzione dei parametri ricevuti. In particolare prima di creare sia una locazione di memoria, che un collegamento fisico visualizzabile come in fig. 3.10

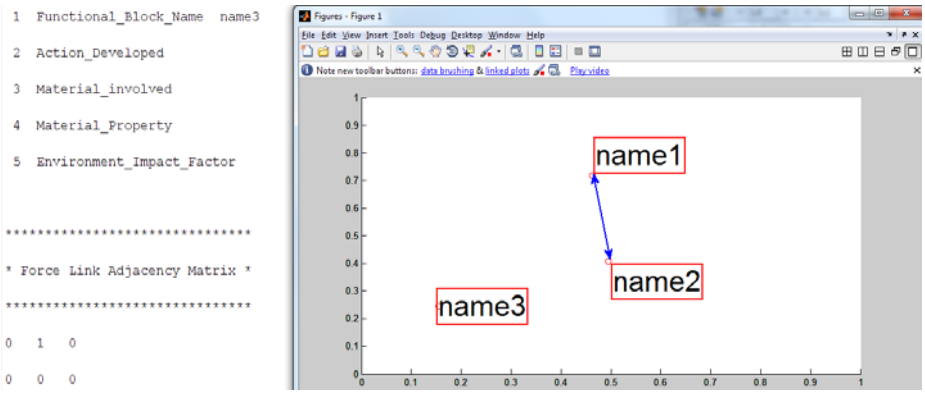

Figura 3-10: *Link di forza tra due nodi generici.*

il programma esegue delle verifiche in funzione degli input che riceve. La prima verifica è

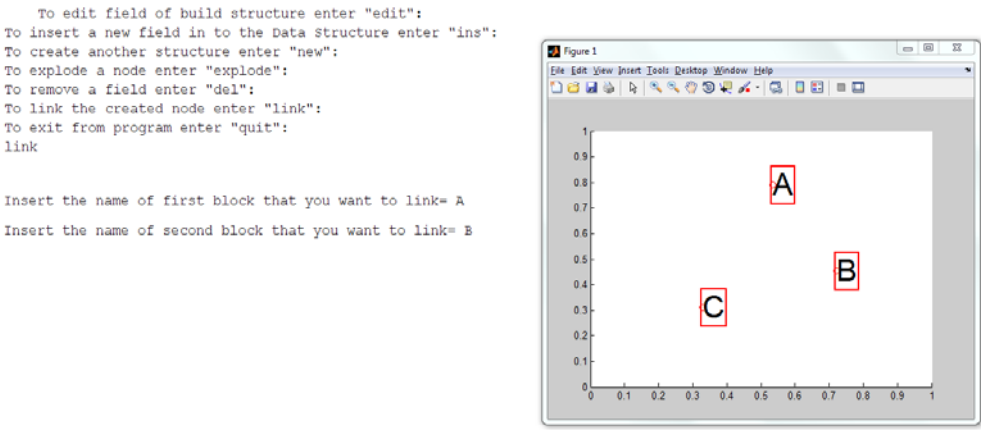

Figura 3-11: *Verifica dell'esistenza del nodo.*

quella della presenza o meno del nodo. Essendo disponibile una zona grafica dove visualizzare la rete funzionale completa, per un componente semplice è facile individuare nomi e collegamenti, mentre con l'infittirsi della rete consegue una difficoltà oggettiva sull'identificazione immediata dei nodi collegati e sul nome dei nodi stessi. A tal proposito il sistema che riceve in input la matrice dei nodi ed eventualmente dei nodi esplosi, chiede quali di questi si intende connettere verificandone l'esistenza. Se la verifica di esistenza da esito positivo si procede con la richiesta del secondo nodo da connettere altrimenti il sistema chiede di correggere il nome immesso.

Individuati i nodi da collegare si prosegue con la funzione "link" (fig. 3.11) che collega sia fisicamente, visualizzando in grafica il collegamento, che al livello di memoria i due nodi scelti. Definite le proprietà del link, la matrice si presenta in questa forma:

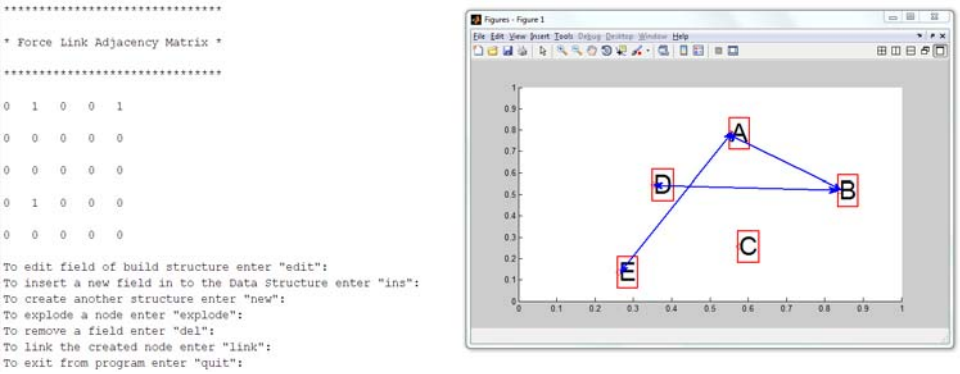

Figura 3-12: *Matrice di adiacenza per i link di forza.*

Come si può notare dalla grafica proposta dal sistema è importante notare come il legame tra grafi ed analisi funzionale è molto stretto. In particolare la matrice di adiacenza descrive la storia delle connessioni conservandone le proprietà per la rete. Altresì è facile notare come i valori positivi della matrice di adiacenza riflettono il collegamento tra il primo ed il secondo nodo, tra il quarto ed il secondo e tra il quinto ed il primo. Teoricamente i link di forza dovrebbero essere bidirezionali, in funzione dello scambio di forze tra due componenti. Nel caso proposto, poiché in versione "*testing*" è stato gestito al livello matriciale un caso più semplice. I link di forza, infatti, risultano essere monodirezionali ed il grafo associato rappresenta un grafo orientato per i nodi A, B e D collegati in modo diretto, mentre il nodo E ed il nodo A sono collegati in modo indiretto. Questa caratteristica si evince dall'elemento che si trova nella parte triangolare inferiore della matrice; mentre se il legame tra il primo e l'ultimo nodo fosse stato diretto ovvero se avesse seguito verso degli altri link, la matrice sarebbe stata nulla nel triangolo inferiore.

3.5 **La matrice di adiacenza e la rete funzionale estesa**

L'analisi funzionale di un generico prodotto non è immediata. È flessibile ma richiede più interventi di affinamento, più termini di riflessione per divenire una vera e propria schematizzazione puntuale di un prodotto industriale. Successivamente alla stesura di una rete di analisi funzionale è necessario ripercorre i cammini, verificare che tutti i nodi abbiano una funzione ben definita ed in fine svolgere un'analisi critica del lavoro svolto. L'analisi deve essere eseguita per determinare, appunto, che ogni blocco funzionale sia

stato ampliato o meglio che la funzione che quel nodo rappresenta sia effettivamente "atomica", non scomponibile, quindi, in altre sottofunzioni. Nel caso in cui, dopo la verifica dei nodi, risultasse indispensabile la suddivisione di un nodo in altri più dettagliati è necessario ricorre all'atomizzazione del nodo. A questa problematica il sistema si comporta in modo molto flessibile automatizzando gli step che dovrebbe compiere l'utente nel momento in cui un nodo deve essere soppresso insieme ai suoi collegamenti, per rimpiazzarlo con altri nodi contenenti informazioni più dettagliate sul componente in analisi.

Quando si desidera approfondire le funzionalità di un sistema meccanico o meccatronico, il software, nelle varie voci del menù offre la funzione "*esplodi nodo"*. Alla richiesta dell'utente di gestire un nodo ampliato si genera immediatamente una matrice che tiene conto dei nodi esplosi ed allo stesso tempo viene modificata la matrice di adiacenza originale. Si parte quindi da una rete funzionale con diversi collegamenti, quindi una matrice di adiacenza associata al grafo principale, dove il sistema in background scioglie i legami tra gli altri nodi e quello che si esplode, scrive sulla matrice dei nodi esplosi le informazioni che si riferiscono alla creazione di tali nodi, infine prepara la matrice di adiacenza originaria eliminando i termini che rappresentano i collegamenti tra i nodi restanti e quello eliminato. In seguito la stessa matrice di adiacenza è ridimensionata rispetto al nodo eliminato in attesa che sia ripristinato un collegamento con gli altri nodi.

La visualizzazione di una struttura funzionale contratta ed estesa è possibile visualizzarla nella seguente immagine:

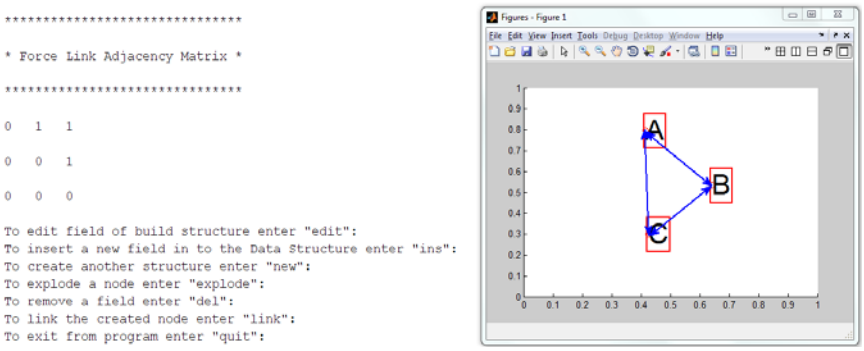

Figura 3-13: *Rete funzionale contratta.*

Nella precedente figura è stata realizzata una rete funzionale con tre nodi, la lista nodi associata si presenta in questo modo:

```
Nodes_List =

     1     1
     2     1
     3     1
```

Figura 3- 14: *Lista nodi struttura contratta.*

La prima fila di valori presenti nella "Nodes_List" identifica l'ordine gerarchico della creazione dei nodi. La seconda fila indica il layer di appartenenza dei nodi. La matrice di adiacenza si forma leggendo le informazioni da questa lista di nodi creati per definire le dimensioni.

Quando l'utente espande la funzione rappresentata da un qualsiasi blocco funzionale, il sistema genera questa configurazione:

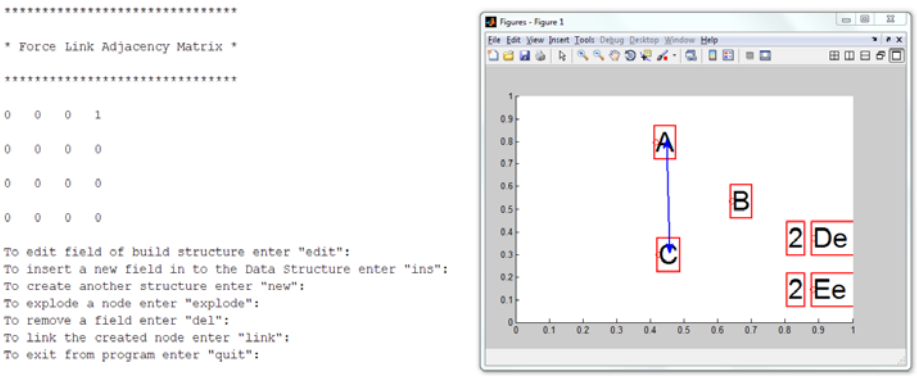

Figura 3-15: *Lista nodi struttura estesa.*

L'immagine precedente si riferisce all'espansione del blocco funzionale "B" mostrato nel diagramma funzionale in figura 3.14. Come si può notare nella grafica all'interno dell'area della struttura funzionale, troviamo i nodi che non hanno subito variazioni ed il nodo "B"ancorato al grafico solo per una migliore comprensione, con i suoi "figli". I nodi così detti "esplosi" sono evidenziati con il numero 2, che sta ad indicare l'appartenenza al blocco espanso B, e denominati "Dex, Eex". Ovviamente in relazione al controllo sui nomi non è possibile inserire un nome identico a quello del blocco funzionale eliminato o ad altri presenti nella rete. Questa tecnica di gestione dei nomi permette la suddivisione in layer del programma, infatti, la possibilità di avere sempre a disposizione una rete funzionale attualizzata in diversi istanti della creazione, quindi espansa o contratta, giustifica la scelta delle restrizioni applicate.

Di notevole importanza è la modifica subita dalla matrice di adiacenza. È possibile confrontare il comportamento della matrice prima e dopo la suddivisione del nodo "B": nel primo caso la matrice è quadrata di lato 3 con gli elementi triangolari superiori maggiori di zero, nel secondo caso la matrice è sempre quadrata ma di lato 4. Il numero di elementi presenti all'interno della matrice indica il numero effettivo di membri che compongono la rete funzionale e che potenzialmente possono essere connessi. Si riconosce infine che, dopo l'espansione nodale del blocco "B" gli altri due blocchi funzionali "A" e "B" che all'interno della rete contratta erano collegati e rappresentati dal valore "1" nella prima matrice di adiacenza, dopo l'applicazione della funzione esplodi nodi sono ancora presenti, questo perché le connessioni tra i vari nodi non si elidono nel caso di passaggi a diverse forme di rete funzionale.

La matrice di adiacenza, quindi, si modella in funzione delle operazioni effettuate dall'utente sulla rete. Allorquando l'utente chiede al sistema di espandere la funzione di un nodo, il programma congela la matrice di adiacenza nelle condizioni precedenti la richiesta, poi lascia agire l'utente nella suddivisione del nodo in alti sottonodi, ed infine valuta il numero di blocchi creati così da poter creare altre righe e colonne per quanti blocchi sono stati immessi nel sistema. In seguito elimina il blocco funzionale non più presente a quel livello quindi elimina una riga ed una colonna. Per quanto riguarda l'inserimento delle nuove righe e colonne, corrispondenti ai nuovi "sottonodi" creati, nella matrice di adiacenza, il programma non si limita ad aumentare le dimensioni della matrice semplicemente inserendo in coda i nuovi elementi, bensì entra all'interno della riga e della colonna appartenenti al nodo che si desidera espandere e da quel punto che le nuove

locazioni di memoria sono inserite. Questo metodo rende più semplici le operazioni di sistemazione dei dati all'interno della matrice e soprattutto permette di controllare i dati in modo più efficiente.

3.5 Costruzione dell'interfaccia grafica

L'analisi funzionale ed in particolare il sistema automatico di generazione dell'analisi funzionale hanno il compito di supportare il progettista in tutte le fasi dello sviluppo prodotto. Questo implica che nelle fasi più delicate, come quella del conceptual design, dove effettivamente si inserisce lo studio effettuato in questo lavoro, il software di supporto sia utilizzabile nel modo più pratico ed efficiente possibile. La creatività del progettista ed allo stesso tempo la possibilità di immagazzinare informazioni, devono essere esaltate e supportate dallo strumento e non affannate dalla difficoltà di utilizzarlo. Per queste ragioni che è stata implementata un'interfaccia grafica ad hoc in grado di semplificare tutte le operazioni che prima erano demandate ad un software in parte grafico ed in parte testuale.

Il lavoro è stato svolto integralmente in Matlab® sfruttando le Graphical User Interface "GUI" built-in all'interno del linguaggio. La programmazione ha subito notevoli passi avanti anche se con un impegno maggiore dovuto alla riprogettazione del software. La possibilità di gestire tutti i comandi attraverso un'interfaccia dedicata permette al progettista di compiere il minimo sforzo per la gestione della rete funzionale. Anche se la programmazione del software è stata modificata radicalmente i vantaggi sono stati notevoli.

Innanzitutto la gestione delle funzioni che prima erano testuali, con l'ausilio della grafica è stato possibile associare ad ogni tasto un evento. Da questo evento scaturiscono le dipendenze per le altre operazioni. Un esempio di interfaccia grafica è riportato in figura nella pagina successiva.

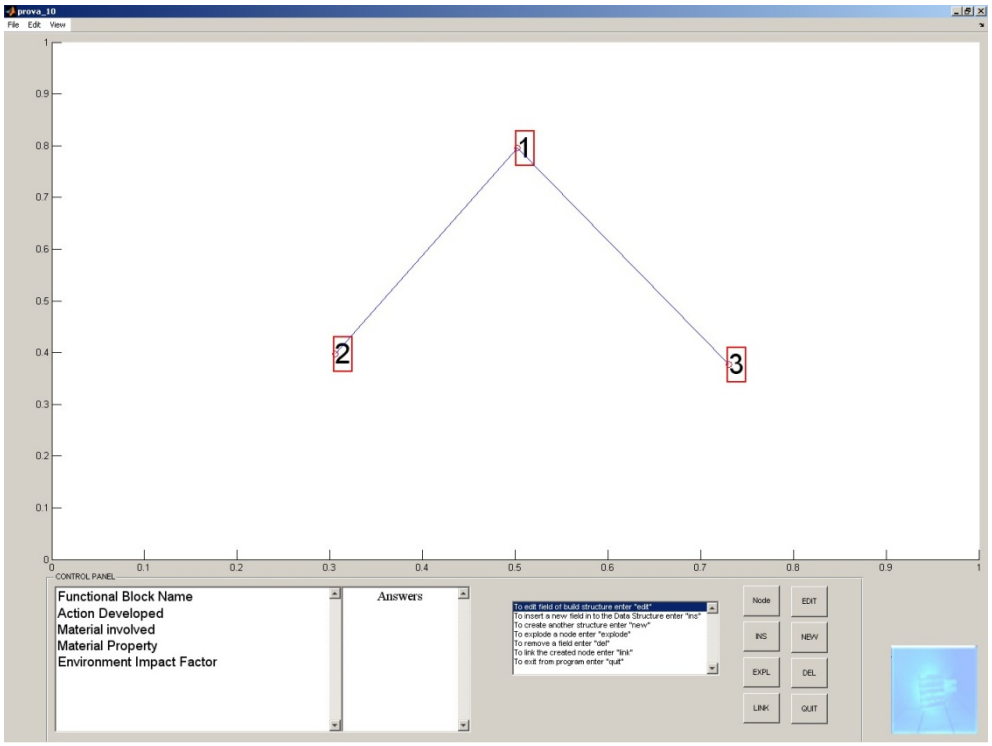

Figura 3-16: *Interfaccia grafica rete funzionale.*

L'interfaccia creata mette a disposizione dell'utente tutte le funzioni in un pannello di controllo di facile lettura e interazione. Tutte le voci relative ai campi standard, per un classico blocco funzionale, sono riportate in un riquadro dedicato associato allo spazio per completare ciascun campo. Nel riquadro contiguo ai pulsanti standard sono riportate le informazioni su ciascuna funzione associata a questi.

La particolarità della programmazione attraverso interfacce grafiche è la circolarità delle variabili. All'interno della programmazione standard perché una funzione sia in grado di manipolare dei parametri è necessario passarli attraverso il metodo standard del linguaggio, come è possibile vedere dalla figura successiva:

```
function  [lung_i lung_j Block_created s...
    punti_acquisiti_x...
    punti_acquisiti_y pts Nodes_List,Nodes_Exploded,...
    matrice_nodi_i,matrice_nodi_j,campi,vet_risp]=creanodo(s,campi,matrice_nodi_i,...
    matrice_nodi_j,...
    f,Block_created,pts,Nodes_List,Nodes_Exploded,vet_risp)...
```

Figura 3-17: *Passaggio parametri in programmazione standard.*

La metodologia di programmazione con interfaccia grafica permette ai parametri di circolare fin quando non sono modificati da altre funzioni. Questo significa che le funzioni demandate alla gestione di un evento, hanno soltanto i parametri standard "`hObject, eventdata, handles`". Questi parametri particolari gestiscono gli oggetti presenti

all'interno dell'interfaccia grafica, gli eventi associati ad un pulsante ed appunto gli "handles", ovvero i parametri circolanti all'interno del programma. In pratica se l'utente fa una richiesta di creazione nodo la funzione demandata a questa operazione si presenta in questo modo:

```
% --- Executes on button press in build_node.
function build_node_Callback(hObject, eventdata, handles)
```

Figura 3-18: *Passaggio parametri in modalità GUI.*

La funzione "build node" inizializza delle variabili interne alla funzione stessa ed il parametro "eventdata" le gestisce tutte e le rende disponibili a qualsiasi altra funzione che ne richieda l'uso. In questo modo il programma si velocizza ed allo stesso tempo si riduce il consumo di memoria.

Di seguito si riportano altre interfacce grafiche realizzate come avanzamento dell'analisi funzionale inizializzata precedentemente.

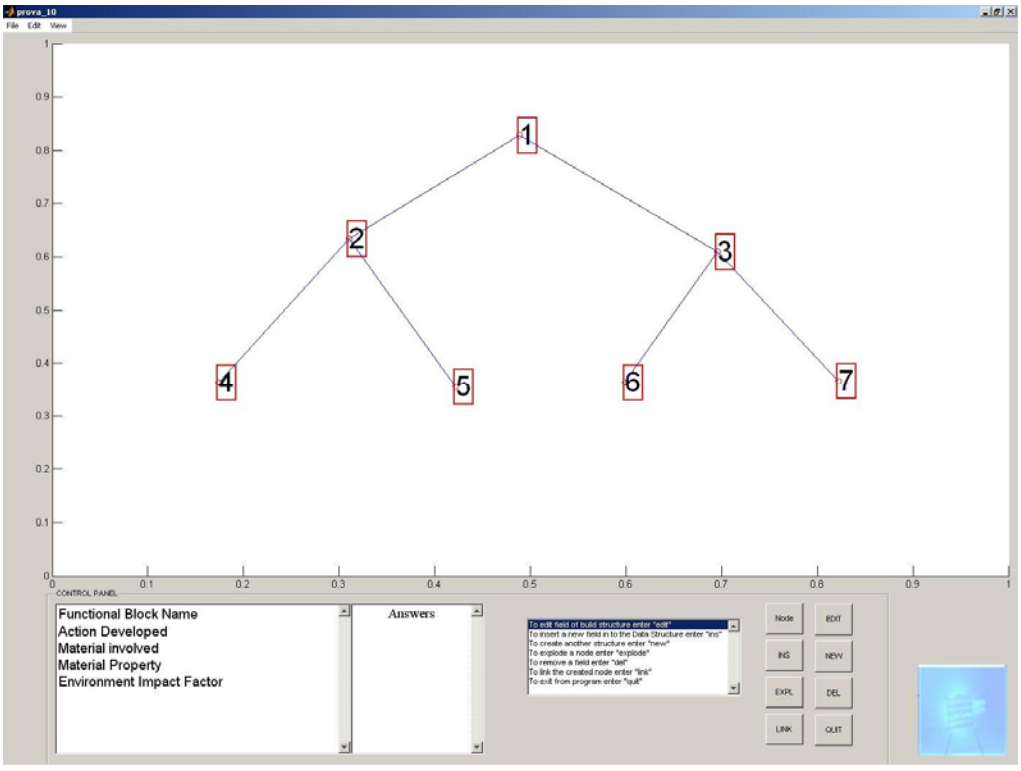

Figura 3-19: *Analisi funzionale ampliata.*