UNIVERSITÀ DEGLI STUDI DI PISA

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea in Scienze dell'Informazione

TESI DI LAUREA

Implementazione in Java di strumenti per la simulazione di sistemi dinamici

Candidato: Lorenzo Cioni

Relatore:

Prof. Giorgio Gallo

Anno Accademico 2002/2003

A Daniela,
mi corazon y mi vida,
e a Yuri,
impertinente folletto
in vesti di gatto ...

Indice

Prefazione

Presentazione

Scopo della Tesi

Scopo della presente tesi è la progettazione e implementazione di un ambiente dedicato per:

1. la definizione di modelli grafici e matematici di sistemi,
2. la simulazione del comportamento dei modelli così definiti mediante la soluzione delle loro equazioni descrittive e
3. la visualizzazione del risultato di tali equazioni.

L'ambiente progettato e in parte implementato, detto $D(a)ySy\,Tool\,Box$[1], contiene, pertanto, un insieme di Tool che consentono la modellizzazione di sistemi dinamici, la simulazione del loro comportamento e la successiva visualizzazione degli andamenti nel tempo delle variabili descrittive dei diversi sistemi modellizzati.
La modellizzazione si traduce nella definizione di un modello grafico che rappresenta le relazioni fra le variabili caratteristiche del sistema, modello grafico cui viene fatto corrispondere un modello matematico ottenuto definendo un insieme di relazioni fra le varie variabili.
Si hanno pertanto:

1. una descrizione grafica o qualitativa di tipo grafico e
2. una descrizione quantitativa di tipo matematico.

La descrizione grafica può essere ottenuta utilizzando due strumenti distinti ma riducibili uno all'altro ovvero i diagrammi Causal-Loop (detti anche diagrammi CL) e i diagrammi di flusso (detti anche diagrammi FD).

[1] Il nome sta per Dynamical Systems Analysis Tool Box.

La descrizione matematica consiste in un insieme di equazioni che stabiliscono dei legami di causa-effetto fra le varie variabili.
Tali equazioni dipendono da una variabile tempo discretizzata.
Mediante la soluzione delle equazioni è possibile ottenere un certo numero di relazioni del tipo $y = x(t)$ che possono essere rappresentate utilizzando dei piani cartesiani in modo da avere l'evoluzione nel tempo delle variabili caratteristiche del modello.

Struttura della Tesi

Il primo capitolo contiene una breve introduzione teorica ai metodi e agli strumenti della System Dynamics, alle problematiche della Computer Simulation e della simulazione continua.
Il secondo capitolo affronta tematiche relative alla progettazione di editor grafici per la creazione e l'editing di grafi e di strutture analoghe oltre a tematiche relative alla visualizzazione di grandezze in funzione del tempo.
Nel terzo capitolo viene presentata la struttura astratta dell'ambiente di modellizzazione proposto, vengono presentati i singoli Tool, vengono descritte le loro interfacce e le loro funzionalità e vengono illustrate le relazioni esistenti fra i vari Tool.
L'analisi della struttura interna dei singoli Tool, incluse le strutture dati dinamiche utilizzate e le problematiche relative alla persistenza dei dati, viene affrontata nel capitolo quattro mentre il capitolo cinque contiene alcuni esempi di utilizzo dei vari Tool.
La tesi si chiude con un capitolo dedicato alle conclusioni in cui viene delineato il lavoro ancora da svolgere e con l'Appendice A.
L'Appendice A contiene la descrizione delle classi utilizzate, dei loro metodi e dei loro campi dati, e delle loro relazioni reciproche mentre i codici sorgente del software sviluppato per implementare l'ambiente oggetto della tesi sono memeorizzati nel CD allegato alla tesi stessa.

Note tipografiche

Il testo della tesi è stato composto e impaginato utilizzando il sistema di composizione LaTeX ([GMS94], [Lam94]) nella versione *MikTeX* per $Windows^{TM}$, utilizzando un *Personal Computer* modello *Presario*, marca *Compaq*, dotato di Sistema Operativo $Windows\ Millenium\ Edition^{TM}$.
Una volta composto il testo, i file in formato *.tex* sono stai convertiti in formato $PostScript^{TM}$ con il comando $dvips^{TM}$, visualizzati con il programma $GSview\ 4.3^{TM}$ per $Windows^{TM}$ e stampati con una stampante $LexmarkZ25$.
Per la *Bibliografia* è stato usato il package *BiBTex* ([GMS94] [Lam94]). Le figure, infine, sono state disegnate utilizzando il programma *Paint* per $Windows^{TM}$ e il programma *XPaint* per $Linux^{TM}$.

La converzione dei file contenenti le immagini dal formato *.gif* al formato *.eps*, in modo da poterle utilizzare all'interno dell'ambiente LaTeX, è stata ottenuta con i programmi *Paint Shop Pro* 7^{TM}, *Image Alchemy* 1.1^{TM} e *GSview* 4.3^{TM} per *Windows*^{TM}.

Ringraziamenti

La presente tesi, che chiude (almeno per ora) una carriera universitaria pluridecennale, deve tutto alla pazienza e all'amore. Desidero, pertanto, ringraziare il Professor Giorgio Gallo, paziente amichevole relatore della presente Tesi, e Daniela, *mi corazon y mi vida.*
Desidero, inoltre, ringraziare la Professoressa Maria Eugenia Occhiuto per il suo prezioso sostegno e i numerosi partecipanti al Newsgroup *it.comp.java* che hanno tollerato le mie spesso strampalate domande relative al linguaggio *Java*^{TM}.

Elenco delle tabelle

Elenco delle figure

Capitolo 1

System Dynamics e Computer Simulation

1.1 Introduzione

Scopo di questo capitolo è quello di presentare una rapida e concisa introduzione ai metodi e agli strumenti della System Dynamics, alle problematiche della Computer Simulation e della simulazione continua.
La System Dynamics fa uso di metodi di carattere generale la cui implementazione richiede l'uso di software progettato ad hoc che consenta la definizione di un modello del sistema sotto esame, la definizione di relazioni che ne descrivono il comportamento e la loro soluzione ottenuta utilizzando i metodi della programmazione discreta dal momento che le varie equazioni descrittive del modello sono scritte in funzione di una variabile tempo discretizzata.

1.2 La Simulazione dei Sistemi

1.2.1 Introduzione

La simulazione [Iaz75] rappresenta l'arte di costruire modelli matematici di sistemi naturali ed artificiali complessi, modelli che è possibile testare utilizzando un computer.
Concetti chiave della precedente definizione sono i concetti di *sistema* e di *modello* visti come oggetti del processo di simulazione.
Un sistema può essere definito come un insieme di componenti interagenti fra di loro in modo da dar luogo ad un comportamento osservabile.
Una volta stabilito quali elementi fanno parte del sistema e quali fanno parte del mondo esterno al sistema, è possibile studiare il comportamento del sistema utilizzando un modello del sistema cui si pongono delle domande, ovvero è possibile risolvere un problema relativo ad un sistema utilizzando un modello.

Il modello del sistema consiste in genere in una rappresentazione astratta o simbolica del sistema in esame.
Il modello contiene i componenti del sistema e le loro interazioni e rappresenta, in genere, una visione semplificata del sistema, dal momento che in esso sono rappresentati solo gli elementi significativi per il tipo di studio che si intende compiere sul sistema.
La risoluzione di un problema mediante un modello di un sistema la si ottiene nel rispetto di un certo numero di vincoli detti

- condizioni iniziali e
- condizioni al contorno.

Le **condizioni iniziali** definiscono le condizioni del sistema all'inizio dello studio mentre le **condizioni al contorno** sono i vincoli che il mondo esterno pone all'evoluzione del sistema ovvero ai cambiamenti di stato del modello del sistema oggetto di studio.
Scopo della simulazione è pertanto, dato un sistema, definirne un modello cui assegnare condizioni iniziali e al contorno e studiare l'evoluzione del modello in modo da ottenere, se esiste, una soluzione ad un dato problema.
Durante l'evoluzione del modello tutti i suoi componeneti mutano le loro condizioni istantanee ovvero il loro stato sebbene solo le condizioni rilevanti per lo studio del sistema entrino a far parte della definizione di uno stato. Un modello di un sistema non rappresenta il comportamento del sistema dal momento che il comportamento è descritto dalla successione degli stati del sistema, detta *storia degli stati*. Se il modello è stato definito in modo corretto, gli stati del sistema corrispondono agli stati del modello in modo che la successione degli stati del modello corrisponda alla successione degli stati del sistema.
Dato un sistema di cui si definisce un modello, si può usare il modello in un processo di simulazione per produrre una successione di stati che si suppone corrispondere alla successione di stati del sistema dato, in modo adeguato rispetto agli scopi per i quali la simulazione è stata effettuata.

1.2.2 Caratterizzazione dei sistemi

Il passo iniziale per la definizione di un modello di un sistema è la definizione dell'*interfaccia* fra il *sistema* e l'*ambiente esterno*.
L'ambiente esterno individua tutti gli elementi che non appartengono al sistema in esame ma che pongono vincoli (detti *condizioni al contorno*) sulla sua evoluzione.
I vincoli posti dall'ambiente esterno sul sistema non dipendono dall'azione del sistema sull'esterno. La definizione dell'interfaccia dipende dagli obbiettivi dello studio del sistema, dal momento che, in funzione di questi, certi elementi possono essere inclusi nel oppure esclusi dal sistema. Una volta individuato il sistema si

possono distinguere alcune grandezze caratteristiche soggette a variare nel tempo e dette *variabili*. Le funzioni che descrivono l'andamento nel tempo delle variabili sono dette *segnali*.
Le variabili caratteristiche di un sistema sono tali che l'evoluzione di alcune di esse dipende da quella di altre.
Si definiscono pertanto le *variabili di ingresso* o variabili indipendenti o cause, le *variabili interne* e le *variabili di uscita* o variabili dipendenti o effetti.
Le variabili di ingresso possono essere viste come le cause prime del comportamento del sistema. Le variabili interne e quelle di uscita si originano dall'interno del sistema e si differenziano per l'obbiettivo della loro azione: le prime agiscono su altri componenti interni al sistema mentre le seconde agiscono sull'ambiente esterno. Tali variabili sono dette *endogene*. Le variabili di ingresso sono dette *esogene* dal momento che si originano nell'ambiente esterno.
Le variabili esogene a loro volta si differenziano in *variabili manipolabili* e *variabili non manipolabili*: le prime sono variabili il cui andamento nel tempo può essere influenzato dall'interno del sistema mentre le seconde sono variabili il cui andamento nel tempo non può essere influenzato.
Di solito ([Mar81]) di un sistema si dà una rappresentazione utilizzando blocchi e collegamenti (ovvero archi orientati) fra blocchi. I blocchi possono rappresentare o il sistema nella sua interezza (modello a *black box*) oppure i vari elementi componenti il sistema.
Nel primo caso le connessioni in ingresso al blocco rappresentano le variabili esogene mentre le connessioni in uscita rappresentano le variabili di uscita.
Nel secondo caso sono rappresentate anche le variabili interne come collegamenti fra componenti interni del sistema.
Gli schemi a blocchi consentono di rappresentare un sistema complesso come composto da un certo numero di blocchi interconnessi. In tal modo è possibile evidenziare le relazioni di causa ed effetto (anche in presenza di anelli di *feedback*) fra le variabili caratteristiche del sistema. In un sistema si ha un anello di *feedback* tutte le volte che una catena di legami causa-effetto si chiude su se stessa in modo che un effetto diventi una delle cause di se stesso.
Nei diagrammi a blocchi in più è possibile rappresentare con facilità i collegamenti percorsi dai segnali all'interno del sistema e fra il sistema e il suo mondo esterno.

1.2.3 Caratterizzazione e classificazione dei modelli

Un modello[1] è, pertanto, una rappresentazione fisica o una descrizione simbolica di un sistema di cui si vuole studiare l'evoluzione e, in quanto tale, è una rappresentazione/descrizione approssimata del sistema in esame.
La rappresentazione fisica di un sistema dà luogo ad un modello fisico utilizzato

[1]Molte delle considerazioni che seguono valgono anche per i sistemi.

per descrivere per analogia il sistema in esame. Esempi di modelli fisici sono i modelli in scala, i modelli iconici ed i modelli analogici.
Questi ultimi sono caratterizzati da grandezze fisiche diverse ma le cui relazioni reciproche sono analoghe a quelle fra le grandezze fisiche caratteristiche del sistema in esame.
La descrizione simbolica di un sistema definisce un modello matematico. Diremo ([Mar81]) che di un sistema si ha un *modello matematico*(nel seguito solo *modello*) se sono note le equazioni che consentono di determinare gli andamenti nel tempo delle variabili interne e di uscita noti gli andamenti nel tempo delle variabili esogene.
Nei modelli matematici le variabili sono rappresentate in genere da numeri reali cui sono associate prefissate unità di misura mentre i segnali sono funzioni che legano i valori delle variabili alla variabile tempo, che può assumere valori reali oppure multipli interi di un intervallo di tempo T [2].
I modelli matematici possono essere classificati come *statici* oppure *dinamici*.
In un modello statico le variabili di ingresso non cambiano i loro valori per lunghi intervalli di tempo. In tal modo si hanno legami puramente algebrici fra le variabili di ingresso e le variabili interne e di uscita. Il sistema è supposto essere in uno stato detto di *regime stazionario* o di equilibrio in cui sono assenti fenomeni transitori. In tale condizione tutti i segnali assumono valori costanti.
È, in genere, possibile adottare un modello statico solo nel caso gli ingressi assumano valori costanti oppure variabili lentamente rispetto alle costanti di tempo del sistema in esame. Un modello statico è di solito costituito da una o più funzioni definite in modo analitico, grafico o tabellare.
I modelli matematici statici non danno informazioni sui *regimi transitori*, ovvero sugli andamenti nel tempo delle variabili dipendenti nel passaggio da un regime stazionario ad un altro.
Per ottenere tali informazioni è necessario far uso di modelli dinamici dal momento che tali modelli fanno uso di equazioni che descrivono sia i legami fra gli andamenti delle variabili sia i legami fra le variazioni di tali andamenti, questi ultimi utilizzando equazioni differenziali o alle differenze finite.
Nel caso si utilizzi un modello dinamico per lo studio di un sistema, l'analisi della risposta del sistema ad uno o più segnali detti *di eccitazione* viene fatta supponendo che il sistema sia inizialmente in una *condizione di equilibrio* o di quiete.
Una condizione di equilibrio è una condizione in cui tutte le variabili hanno valori costanti. In tale condizione le variabili di uscita non variano a meno che le variabili di ingresso non subiscano a loro volta delle variazioni. In alcuni casi le variabili di uscita possono variare anche in assenza di variazioni delle variabili di ingresso. In tal caso l'evoluzione del sistema dipende dal suo stato iniziale.
I modelli matematici possono inoltre essere classificati come *modelli lineari* e *modelli non lineari*. Un modello è detto essere lineare se soddisfa la proprietà di

[2] Altrove si è fatto uso, con significato identico, del simbolo Δt.

sovrapposizione degli effetti mentre è detto essere non lineare se non la soddisfa. Dato un sistema in una condizione di quiete il principio di sovrapposizione degli effetti stabilisce che:

1. se ad una causa x corrisponde un effetto y, ad una causa αx corrisponde l'effetto αy, $\forall \alpha \in \Re$;

2. se ad una causa x_1 corrisponde un effetto y_1 e ad una causa x_2 corrisponde un effetto y_2 , allora ad una causa $x_1 + x_2$ corrisponde l'effetto $y_1 + y_2$.

Molti sistemi ammettono modelli lineari se le variabili non assumono valori al di fuori di dati intervalli di valori detti intervalli di linearità, uno per ciascuna variabile. In molti casi è possibile, inoltre, usare modelli statici e dinamici lineari anche per lo studio di sistemi non lineari a patto di eseguire delle linearizzazioni locali approssimando con rette andamenti descritti da funzioni di ordine superiore.
Un modello, lineare o meno, si dice, infine, *stazionario* se soddisfa la proprietà di *invarianza nel tempo* o di *traslazione nel tempo di cause ed effetti.* Secondo tale proprietà, dato un sistema inizialmente in quiete, se ad una causa $x(t)$ corrisponde un effetto $y(t)$, $\forall$ T $\in \Re^+$ ad una causa $x(t-T)$ corrisponde un effetto $y(t-T)$.

1.2.4 Sistemi continui

I *sistemi continui* ([Iaz75]) sono sistemi che, ai fini dell'analisi, sono considerati caratterizzati da un flusso continuo di materiali (*flusso di materiali*) e di informazioni (*flusso di informazioni*).
I sistemi continui sono di solito descritti da modelli matematici caratterizzati da

1. equazioni differenziali,

2. equazioni alle differenze finite

che descrivono le leggi di variazione delle variabili nel tempo.
Ad un sistema continuo è pertanto associato un certo numero di equazioni differenziali o alle differenze finite. Le tecniche di risoluzione adottate sono di solito analitiche o numeriche ma è possibile far uso di tecniche di simulazione. La simulazione, nel caso di equazioni alle differenze finite viste anche come approssimazioni di equazioni differenziali, si traduce nel calcolo iterativo delle varie equazioni a partire da un istante iniziale e sulla base di un certo numero di condizioni iniziali.
Nei sistemi continui è necessario individuare, per prima cosa, gli elementi che costituiranno le *variabili di stato*, quelli che costituiranno le *variabili esogene*, i parametri e una relazione funzionale che specifica il comportamento del sistema. Tale passo ([Iaz75])rappresenta la *formulazione del modello* del sistema. Il passo

successivo è rappresentato dalla *risoluzione* ovvero dalla determinazione dei valori delle variabili di stato all'istante $t+\Delta t$ una volta che siano noti i valori di tutte le grandezze all'istante t e dalla ripetizione di tale determinazione dall'istante iniziale all'istante finale della simulazione, rappresentati da un istante 0 e da un istante $n\Delta t$.
Il risulato dell'elaborazione, che rappresenta il prodotto della simulazione, è costituito dagli andamenti nel tempo delle variabili di stato insieme a quelli delle variabili esogene ed eventualmente dei parametri.
I sistemi continui possono essere classificati come (cfr. la sezione 1.2.3) di tipo lineare o non lineare, di tipo stazionario o non stazionario. Nel caso dei sistemi lineari il principio di sovrapposizione degli effetti deve valere ([Iaz75]) sia rispetto alle condizioni iniziali (in assenza di ingressi) sia rispetto agli ingressi (quali che siano le condizioni iniziali) e stabilisce un legame fra le cause (variabili esogene e valori iniziali delle variabili di stato) e le variabili individuate (nel modello o nel sistema) come effetti.
I sistemi (ed i modelli) stazionari sono caratterizzati da parametri costanti nel tempo mentre i sistemi (ed i modelli) non stazionari sono caratterizzati da parametri variabili nel tempo.
Se un sistema continuo è di tipo lineare e stazionario lo si può descrivere con un certo numero di equazioni alle differenze finite lineari a coefficienti costanti.
I sistemi continui, inoltre, possono essere caratterizzati da anelli di *feedback* in cui una uscita, attraverso una trasformazione, viene riportata in ingresso (caso di un sistema con un solo ingresso e una sola uscita) e composta con questo. Definito il segnale errore come la differenza fra il segnale in ingresso e il segnale in uscita riportato in ingresso, il *feedback* si dice *negativo* se ad un aumento del secondo corrisponde una diminuzione dell'errore altrimenti si dice *positivo*. Nei casi di *feedback* positivo il sistema tende alla instabilità mentre, qualora il *feedback* sia negativo, il sistema, di solito, tende ad una posizione di equilibrio ovvero tende a raggiungere un obbiettivo (comportamento "goal seeking") sebbene, nel caso in cui la catena di ritorno sia caratterizzata da un guadagno eccessivo (cfr. la sezione 1.2.5), si possa avere instabilità anche in presenza di *feedback* negativo.
Lo studio dei sistemi continui lineari e stazionari si basa su una teoria ben fondata la cui trattazione esula dallo scopo della presente tesi (cfr. al proposito [Iaz75] e [Mar81]). In questo contesto ci si limita a notare come il problema di base dell'analisi di un sistema con le proprietà suddette sia quello di determinare il valore dell'uscita dato un ingresso applicato in un certo istante t_0 e date le condizioni iniziali. Tale problema lo si risolve, nell'ambito della *Teoria dei Sistemi*, determinando la risposta del sistema come somma della *risposta libera* e della *risposta forzata*. Per risposta libera si definisce la risposta del sistema in *evoluzione libera* ovvero in caso di ingressi nulli e condizioni iniziali non nulle mentre per risposta forzata si definisce la risposta del sistema in *evoluzione forzata* ovvero in presenza di segnali in ingresso e con le condizioni iniziali nulle. Un metodo simile può essere adottato per l'analisi dei sistemi modellizzati con le tecniche della *System*

Dynamics nei quali si può analizzare l'evoluzione del modello avendo fissato come costanti (anche nulle) tutte le variabili esogene in modo che il modello si porti in una situazione di equilibrio per poi applicare a certe variabili esogene dei segnali di ingresso di tipo particolare (tipicamente delle funzioni gradino unitario traslate nel tempo) che permettono di valutare l'evoluzione del sistema (e del modello) in presenza di sollecitazioni esterne.
Un metodo schematico per la descrizione dei sistemi continui ([Iaz75]) è quello della *System Dynamics* (cfr. la sezione 1.3).
Come sarà meglio illustrato nella sezione 1.3, la struttura di base di un modello è composta da elementi di accumulazione, detti *livelli*, collegati da archi orientati che rappresentano scambi di entità (informazioni o materiali) fra i livelli e sono detti $flussi$. I flussi sono regolati da equazioni che dipendono dai livelli cui sono collegati e da altre grandezze del modello. Il modello pone in relazione flussi e livelli e pertanto, poichè i flussi sono rappresentati dalla derivata di una variabile livello, rappresenta, in genere, un sistema di equazioni differenziali rappresentabili come equazioni alle differenze finite.
Nel caso, ad esempio, di una relazione (cfr. le sezioni 1.3 e 1.4) di proporzionalità diretta fra un livello $y(t)$ (ad esempio $Capitale$) e un flusso x entrante nel livello (ad esempio $interesseAnnuo$) in cui la costante di proporzionalità b è positiva (può essere il tasso di interesse) si possono scrivere le equazioni seguenti:

$$y(t + \Delta t) = y(t) + x(t) \times \Delta t \tag{1.1}$$

$$x(t) = b \times y(t) \tag{1.2}$$

dalle quali, con semplici passaggi e supponendo Δt tendente a 0, si ottiene la seguente equazione differenziale del primo ordine:

$$\frac{dy(t)}{dt} = b \times y(t) \tag{1.3}$$

la cui soluzione è $y(t) = y(0) \times e^{bt}$ in cui $y(0)$ è il valore all'istante iniziale del livello $y(t)$. A tale descrizione in termini di equazioni differenziali corrispondono due descrizioni pittoriche in termini sia di un diagramma CL sia di un diagramma FD (cfr. le sezioni 1.3 e 1.4). L'esempio visto caratterizza un anello con $feedback$ positivo in cui entrambe le variabili mostrano un andamento crescente nel tempo. In modo analogo si può caratterizzare un anello con $feedback$ negativo in cui le variabili tendono a zero oppure ad un valore obbiettivo. Il primo caso è descritto dalle equazioni seguenti (in cui a è una costante positiva che può rappresentare la "vita media" di un bene):

$$y(t + \Delta t) = y(t) - x(t) \times \Delta t \tag{1.4}$$

$$x(t) = \frac{1}{a} \times y(t) \tag{1.5}$$

dalle quali si può ottenere, con passaggi analoghi ai precedenti, la seguente equazione differenziale del primo ordine:

$$\frac{dy(t)}{dt} = -\frac{1}{a} \times y(t) \tag{1.6}$$

la cui soluzione è $y(t) = y(0) \times e^{\frac{-t}{a}}$ in cui $y(0)$ è il valore all'istante iniziale del livello $y(t)$. A tale equazione corrisponde un andamento del livello decrescente nel tempo. Come nel caso precedente, al modello matematico corrispondono due descrizioni pittoriche in termini di un diagramma CL e di un diagramma FD.
Nel caso, infine, in cui si voglia modellizzare un parcheggio in cui posono essere presenti al più Y auto e in cui, all'istante t, sono presenti $y(t)$ auto si può procedere come segue. Se si definisce con k la frequenza di arrivo ($k > 0$) delle auto, si ha un sistema in cui un flusso tende a far crescere un livello il quale a sua volta limita il valore del flusso fino a che, eventualmente, il parcheggio si riempie e il flusso si annulla. L'equazione che descrive l'andamento del flusso è, anche in questo caso, una equazione differenziale del primo ordine ed ha la seguente forma:

$$\frac{dy(t)}{dt} = k \times (Y \ - \ y(t)) \tag{1.7}$$

Tale equazione ha la seguente soluzione ([Iaz75]):

$$y(t) = Y \times (1 \ - \ e^{-kt}) \tag{1.8}$$

mentre l'equazione del flusso $x(t)$ è la seguente:

$$x(t) = k \times Y \times e^{-kt} \tag{1.9}$$

In qesto caso si ha una grandezza (il livello) che tende ad un valore limite (Y) e un'altra grandezza (il flusso) che tende a zero. Si può verificare ([Iaz75]) che tutti i sistemi modellizzati con un solo livello sono descrivibili da equazioni differenziali del primo ordine mentre, se si introducono livelli e relazioni fra flussi e livelli, si deve far ricorso a equazioni differenziali di ordine superiore. Ad esempio nel caso di tre livelli e tre flussi in relazione fra di loro si può dover ricorrere a tre equazioni differenziali (una del primo ordine, una del secondo ordine e una del terzo ordine) per descrivere gli andamenti dei tre livelli nel tempo dove, in ogni equazione compaiono solo la funzione che descrive il livello e le sue derivate. I metodi della *System Dynamics* rimpiazzano con rappresentazioni pittoriche, più intuitive e più facili da definire e caratterizzare, i sistemi di equazioni differenziali che sarebbe altrimenti necessario impostare per descrivere un sistema continuo.

1.2.5 Stabilità e sistemi con anelli di *feedback*

Un sistema ([Mar81]) soggetto ad una perturbazione in ingresso[3] invece di raggiungere una condizione di equilibrio può mostrare una risposta di ampiezza

[3]Una perturbazione è un segnale applicato ad uno degli ingressi di un sistema a partire, di solito, da una condizione di riposo.

crescente nel tempo. Un sistema (non necessariamente) lineare[4] che presenti un tale comportamento si dice *instabile*.
Il caso più semplice che si possa analizzare per discutere il concetto di *stabilità* è quello di un sistema con:

1. una variabile in ingresso $x(t)$ e
2. una variabile di uscita $y(t)$.

Si suppone che il sistema sia in una condizione di equilibrio all'istante $t = t_0$ e che venga perturbato mediante l'applicazione di un segnale non nullo e di durata limitata τ alla variabile di ingresso $x(t)$.
Il sistema in risposta a tale perturbazione può presentare tre comportamenti distinti che si traducono in:

1. una risposta limitata[5],
2. una risposta divergente,
3. una risposta convergente asintoticamente a zero.

Nel caso (1), che corrisponde ad un comportamento stabile, si ha che esiste una costante M tale che $|y(t)| \leq M$ per $t \geq t_0$.
Nel caso (2), che corrisponde ad un comportamento instabile, una tale costante non esiste, ovvero si può affermare che $\forall M \geq 0 \, \exists \, t \geq t_0$ tale che $|y(t)| \geq M$.
Nel caso (3), infine, si ha che esiste una costante M come nel caso (1) e in più si ha:

$$lim_{\, t \to +\infty} \, y(t) = 0$$

Il caso (3) è detto *asintoticamente stabile* o strettamente stabile ([Mar81]).
Per i sistemi lineari il comportamento del sistema sottoposto ad una perturbazione, dato che ad essi si applica il principio di sovrapposizione degli effetti, è indipendente dal punto di equilibrio in cui si trova il sistema al momento della perturbazione e dall'entità di questa per cui un sistema lineare è detto essere *stabile*, *instabile* oppure *asintoticamente stabile* se il suo comportamento in risposta ad una perturbazione è, rispettivamente, del tipo (1), (2) o (3).
Oltre alla *stabilità in presenza di una perturbazione* ([Mar81]), discussa nei paragrafi precedenti, si può fare riferimento alla *stabilità in presenza di ingresso limitato*.
Dato un sistema ad un solo ingresso ed una sola uscita che si trovi in uno stato di equilibrio caratterizzato da ingresso e uscita identicamente nulli, il sistema è detto essere stabile in presenza di un segnale di ingresso limitato se ad ogni segnale di

[4]Nel seguito della sezione faremo riferimento essenzialmente ai sistemi lineari stazionari, salvo avviso contrario.

[5]Tale caso comprende il caso della risposta costante.

ingresso $x(t)$ di ampiezza limitata (ovvero tale che $\exists M_x$ tale che $|x(t)| \leq M_x \, \forall t$) corrisponde un segnale di uscita $y(t)$ di ampiezza limitata (ovvero tale che $\exists M_y$ tale che $|y(t)| \leq M_y \, \forall t$). Anche la stabilità in presenza di ingresso limitato è indipendente dal punto di equilibrio del sistema e dal valore di M_x dal momento che, in forza del principio di sovrapposizione degli effetti, se le relazioni suddette sono soddisfatte per due valori M_x e M_y allora sono soddisfatte anche per i valori αM_x e $\alpha M_y \, \forall \, \alpha \geq 0$.
I concetti di stabilità in presenza di una perturbazione e di stabilità in presenza di ingresso limitato possono essere applicati all'analisi di sistemi caratterizzati dalla presenza di anelli di *feedback* , detti anche *sistemi in retroazione* ([Mar81]).
Il caso più semplice di modello di un sistema in retroazione è quello di uno schema a blocchi caratterizzato da due moduli e da un sommatore[6] collegati in modo da formare un singolo anello di *feedback* .
Il modello è caratterizzato da:

1. un certo numero di segnali;
2. i due moduli con le relative relazioni ingresso-uscita;
3. il sommatore e
4. le due catene di trasferimento del segnale: la catena diretta e la catena inversa.

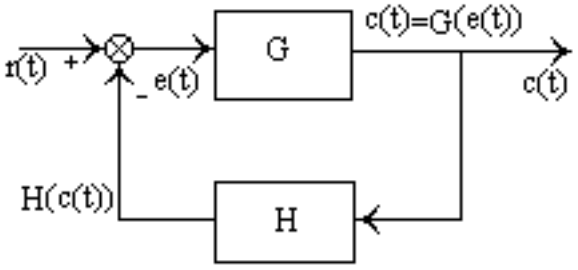

Figura 1.1: *Esempio di sistema in retroazione*

Nella figura 1.1 il segnale di ingresso è rappresentato dalla funzione $r(t)$ che, passando attraverso la catena diretta, determina il segnale di uscita $c(t)$. Il segnale $c(t)$ riportato in ingresso attraverso la catena inversa permette di generare, mediante il sommatore, il segnale errore $e(t) = r(t) - H(c(t))$ il quale, a sua volta, passando attraverso il modulo caratterizzato dalla funzione G,

[6]Un sommatore è un elemento che compone i segnali in ingresso in base ai segni presenti sui suoi archi in ingresso per cui può eseguire sia una somma sia una differenza sia una combinazione delle due operazioni sui segnali in ingresso.

permette di definire il segnale $c(t)$. Dal momento che il segnale errore è calcolato come differenza dei due segnali $r(t)$ e $c(t)$ l'anello di $feedback$ di figura 1.1 è un anello di $feedback$ negativo e il sistema tende a portarsi in uno stato di equilibrio in cui è $e(t) = 0$. Il raggiungimento dello stato di equilibrio può avvenire in presenza o assenza di oscillazioni smorzate attorno alla posizione di equilibrio.
L'analisi di tali sistemi può essere svolta determinando la *funzione di trasferimento* della catena diretta $G(s)$ e la *funzione di trasferimento* della catena inversa $H(s)$ in modo da definire la funzione di trasferimento del sistema nella cosiddetta forma minima ([Mar81]):

$$G_0(s) = \frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)} \tag{1.10}$$

in cui $R(s)$ è la $Trasformata\ di\ Laplace$ del segnale di ingresso e $C(s)$ è la $Trasformata\ di\ Laplace$ del segnale di uscita. È possibile, quindi, calcolare $c(t)$ antitrasformando $C(s)$ ottenuto come $G_0(s)R(s)$.
Lo studio della stabilità del sistema il cui modello è presentato in figura 1.1 si può ricondurre all'analisi della funzione di trasferimento in forma minima ovvero alla analisi dei suoi zeri e dei suoi poli[7]. Si può dimostrare ([Iaz75]) che un sistema lineare stazionario è stabile se e solo se tutti i poli della sua funzione di trasferimento hanno parte reale negativa in modo che i modi di evoluzione del sistema sono o esponenziali decrescenti o funzioni periodiche smorzate.
Nella equazione 1.10, $G(s)$ rappresenta la funzione di trasferimento del sistema in assenza di $feedback$ mentre $H(s)$ rappresenta il contributo della catena di $feedback$: se $H(s) \neq 0$ il sistema è caratterizzato da un $feedback$ per cui il guadagno del sistema varia del fattore $1 + G(s)H(s)$. In funzione del valore di $|1 + G(s)H(s)|$ si hanno i tre casi seguenti:

1. $|1 + G(s)H(s)| < 1$
2. $|1 + G(s)H(s)| = 0$
3. $|1 + G(s)H(s)| > 1$

cui corrispondono per il guadagno del sistema compreso l'anello di $feedback$ $G_0(s)$ i tre casi seguenti:

1. $G_0(s) > G(s)$
2. $G_0(s) = +\infty$
3. $G_0(s) < G(s)$

[7] Gli zeri sono le soluzioni della $G(s) = 0$ mentre i poli sono le soluzioni della $1+G(s)H(s) = 0$

Nel primo caso ([Iaz75]) il *feedback* è detto essere positivo e il sistema è detto rigenerativo, nel secondo caso il sistema è un oscillatore e si può avere un segnale il uscita anche senza nessun segnale in ingresso e nel terzo caso il *feedback* è negativo e il sistema è detto essere degenerativo.
Una verifica della stabilità di un sistema la si può fare utilizzando il denominatore della 1.10 ovvero $1 + G(s)H(s)$. Lo scopo è quello di verificare per quali valori di s si ha $|1 + G(s)H(s)| < 1$ in modo che il sistema sia stabile. Tale verifica di stabilità si basa sull'analisi del diagramma polare della funzione di trasferimento a catena aperta $G(s)H(s)$. In questa sede ci si limita ad osservare (*criterio di Nyquist*) che se la curva polare suddetta racchiude il punto $(-1, j0)$ allora il sistema è instabile.
Dato un sistema lineare stazionario e stabile si può diostrare che valogono le proprietà seguenti ([Iaz75]):

1. la risposta libera del sistema tende a zero per $t \to +\infty$ quali che siano le condizioni iniziali,

2. in assenza di segnali di ingresso il sistema ha un solo stato di equilibrio stabile in cui si porta a partire da un qualunque insieme di condizioni iniziali. Lo stato di equilibrio è detto di riposo e in tale stato le condizioni iniziali sono nulle.

1.2.6 Schemi a blocchi e grafi di flusso

La figura 1.2 ([Mar81]) rappresenta il diagramma a blocchi di un semplice sistema in retroazione e il corrispondente grafo di flusso.

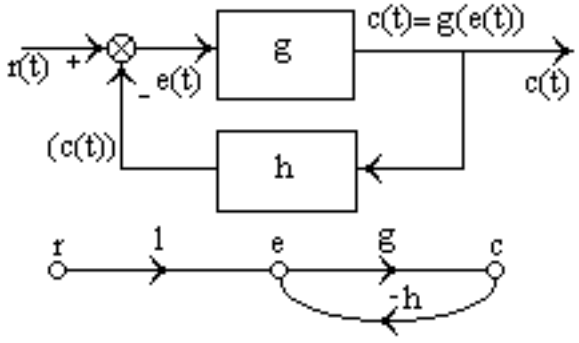

Figura 1.2: *Schema a blocchi e grafo di flusso corrispondente*

Un *grafo di flusso* rappresenta un modo alternativo agli schemi a blocchi per rappresentare graficamente sistemi complessi. Mediante i grafi di flusso un sistema viene rappresentato sotto forma di un grafo in cui i nodi rappresentano i segnali e gli archi orientati le trasformazioni eseguite sui segnali (con il termine 1 si indica

la trasforamzione identità ovvero tale che $1(s) = s$).
I nodi sono dei tipi seguenti:

1. nodi sorgente o indipendenti, privi di archi in ingresso,
2. nodi interni o dipendenti, con almeno un arco in ingresso

mentre agli archi sono associati i coefficienti che rappresentano le trasformazioni eseguite su ciascun arco.
Con riferimento alla figura 1.2, i nodi e e c sono interni mentre il nodo r è un nodo sorgente e gli archi sono caratterizzati, rispettivamente, da una trasformazione identità, dalla trasformazione g e dalla trasformazione $-h$.
Ad ogni grafo può essere associato un sistema di equazioni algebriche lineari in cui in nodi sorgente rappresentano i *termini noti*, i nodi dipendenti sono le *incognite* mentre gli archi rappresentano i *coefficienti*.
Nel caso della figura 1.2, il sistema di equazioni (che consente di esprimere la variabile di un nodo in funzione delle variabili dei nodi della stella entrante e dei coefficienti sugli archi relativi) ha la forma seguente:

$$e = r - hc \tag{1.11}$$

$$c = ge \tag{1.12}$$

e può essere espresso in forma compatta (sostituendo la 1.12 nella 1.11 ed effettuando alcuni semplici passaggi algebrici) come segue:

$$e = \frac{r}{1 + hg} \tag{1.13}$$

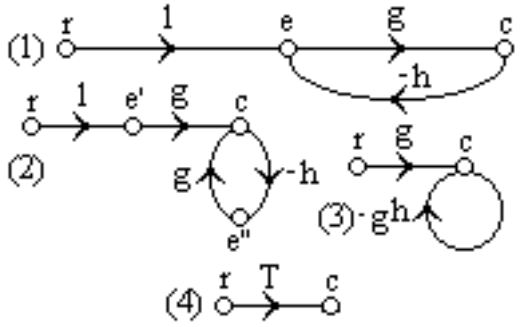

Figura 1.3: *Grafo di flusso e sua riduzione in forma minima*

Come risulta evidente dalle equazioni 1.11, 1.12 e 1.13 e come è illustrato dalla figura 1.3, un grafo di flusso può essere ridotto in *forma minima* utilizzando due procedimenti.

Dato un grafo G con un certo numero di nodi sorgente x_{0i} ($i \in [1, \ldots, n]$) e un certo numero di nodi dipendenti x_j ($j \in [1, \ldots, m]$) di cui ci interessano le variabili, lo si può ridurre in *forma minima* ottenendo un grafo $\overline{G}$ in cui sono presenti i soli nodi x_{0i} e x_j con un certo numero di archi che li collegano direttamente fra di loro.
La riduzione può essere effettuata:

1. per semplificazioni successive (cfr. la figura 1.3) applicando le regole di riduzione che saranno introdotte a breve in modo da rimuovere nodi e archi ritenuti superflui,

2. in modo diretto mediante un'analisi degli elementi topologici del grafo.

Il procedimento di riduzione per semplificazioni successive si basa sulla applicazione delle seguenti cinque regole:

1. riduzione di archi in parallelo ad un arco il cui coeficiente è dato dalla somma algebrica dei coefficienti dei singoli archi,

2. riduzione di archi in serie che formano un cammino orientato (ed eliminazione dei nodi interni) ad un arco il cui coeficiente è dato del prodotto dei coefficienti dei singoli archi,

3. eliminazione di un cappio (anello che si inizia e si chiude sullo stesso nodo) di coeficiente t con introduzione di un coeficiente moltiplicativo $1/(1-t)$ su tutti gli archi che terminano su quel nodo,

4. duplicazione di un nodo con duplicazione dei rami entranti e distribuzione dei rami uscenti in modo arbitrario fra i nodi duplicati,

5. duplicazione di un nodo con duplicazione dei rami uscenti e distribuzione dei rami entranti in modo arbitrario fra i nodi duplicati.

Tale procedimento si traduce nella rimozione di variabili e trasformazioni non ritenute significative e può trovare uso nell'ambito della simulazione di sistemi dinamici per l'eliminazione di variabili ausiliarie e la semplificazione dei diagrammi CL (cfr. la sezione 1.4). La riduzione illustrata nella figura 1.3 è stata ottenuta appplicando, nell'ordine, le regole (5), (2) e (3).
Il procedimento di riduzione diretta si basa sulla applicazione della *formula di Mason* e sfrutta , nel caso di sistemi lineari, il *principio di sovrapposizione degli effetti*. Sfruttando tale principio è possibile considerare separatamente l'effetto di ogni nodo sorgente su ogni nodo dipendente che si vuole mantenere nella forma minima in modo da ridursi a considerare forme minime con un solo nodo sorgente ed un solo nodo dipendente, come nel caso della figura 1.3. Per ulteriori dettagli su questa tecnica, più orientata all'uso nell'ambito dei controlli automatici, si rinvia a [Mar81].

1.3 La *System Dynamics*

1.3.1 Introduzione

I metodi della System Dynamics che verranno brevemente descritti nella presente sezione sono di tipo generale ma per poterli implementare è necessario fare uso di prodotti software ad hoc.
Da un punto di vista teorico, la *System Dynamics* ([Kir98]) si fonda sul cosiddetto "approccio sistemico"[8] e mira a descrivere il comportamento di sistemi complessi considerandoli composti di un gran numero di parti interagenti in modo da formare uno schema unificato.
Tale approccio richiede un cambiamento di prospettiva dal momento che non vengono esaminati più i singoli eventi e le relative cause ma vengono prese in esame complesse catene di cause ed effetti che, per essere di un qualche interesse, devono contenere anelli di $feedback$.
Secondo una visione tradizionale della Teoria dei Sistemi (cfr. la sezione 1.2), infatti, le spiegazioni del comportamento di un sistema vanno sempre ricercate in qualche evento esterno al sistema. L'approccio sistemico ribalta tale prospettiva e ipotizza che la *struttura interna* sia spesso più importante degli eventi esterni nella determinazione del comportamento di un sistema.
Considerando gli eventi come causa di comportamenti dei sistemi, in genere si arriva alla definizione di catene di eventi legati fra loro da relazioni di causa-effetto che raramente consentono di capire perché un sistema complesso si comporta in un certo modo, spesso controintuitivo, mentre considerando la struttura interna del sistema può essere più facile capire il comportamento mostrato e questo perché è la struttura interna che determina tale comportamento.

1.3.2 Gli "andamenti tipici"

L'analisi del comportamento dei sistemi ha come punto di partenza la definizione di un certo numero di andamenti (o schemi di comportamento) tipici che possono essere seguiti dalle variabili caratteristiche del sistema. Tale approccio si basa sull'assunzione che gli andamenti tipici sono riscontrabili in molte situazioni in cui la struttura del sistema è nota per cui, dato un certo andamento delle variabili, è possibile inferire la struttura interna del sistema o, meglio, dato un certo andamento delle variabili, è possibile ricercare nel sistema una struttura che è capace di produrre quel comportamento.
Gli andamenti tipici (cfr. la figura 1.4) che saranno brevemente esaminati nel seguito per poi venire associati (nella sezione 1.3.3) a strutture interne dei sistemi sono i seguenti:

1. crescita/decrescita esponenziale,

[8] Il termine inglese corrispondente è *systems thinking*.

2. asintotico,
3. crescita a S,
4. oscillatorio.

Tali andamenti possono essere osservati sia singolarmente sia in combinazione fra di loro. Ad esempio non è raro osservare andamenti asintotici o di crescita esponenziale o di crescita a S con sovrapposte delle oscillazioni smorzate[9] o meno.

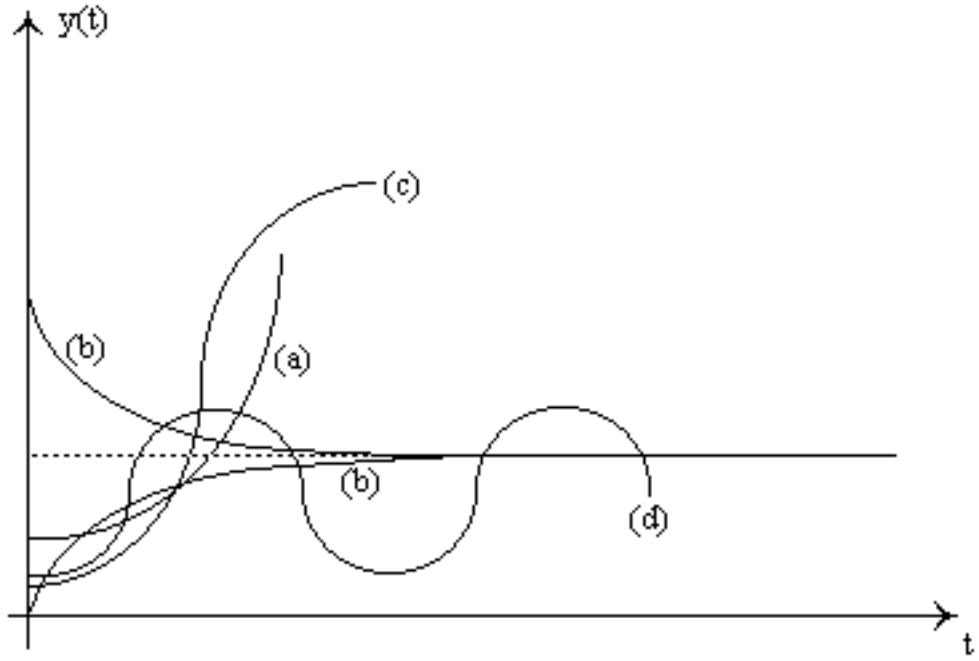

Figura 1.4: *Esempi di andamenti tipici*

Nel caso di crescita (cfr. il tracciato etichettato come (a) in figura 1.4) o decrescita esponenziale la variabile di cui si studia l'andamento assume, con una velocità di variazione crescente nel tempo, valori crescenti o decrescenti a partire da un valore iniziale.
Il modello matematico di tali andamenti è costituito, rispettivamente, dalle funzioni ([BS91])[10]:

$$y(t) = Ae^{\alpha t/T} \; con \; \alpha \; > \; 0 \; e \; T > 0 \tag{1.14}$$

[9] Una oscillazione si dice *smorzata* se la sua ampiezza tende a zero con il passare del tempo.
[10] Le funzioni utilizzate nel seguito per rapprersentare gli "andamenti tipici" sono ovviamente delle approssimazioni ideali degli andamenti riscontrati nella pratica.

$$y(t) = Ae^{-\alpha t/T} \; con \; \alpha \; > \; 0 \; e \; T > 0 \tag{1.15}$$

In entrambe le equazioni il valore A rappresenta il valore iniziale (ovvero il valore assunto dalla funzione per $t = 0$).
L'equazione 1.14 descrive un andamento rapidamente crescente nel tempo. Per apprezzare la sua rapidità è possibile calcolare il valore della derivata della funzione in $t = 0$ e due grandezze quali la *costante di tempo* e il *tempo di raddoppio*[11]. Il valore della costante di tempo τ lo si ottiene imponendo che il valore dell'esponente dell'esponenziale sia uguale a 1 per cui si ha:

$$\tau = \frac{T}{\alpha} \tag{1.16}$$

Il valore del tempo di raddoppio lo si ottiene risolvendo l'equazione $y(t) = 2y(0)$ ottenendo il valore:

$$\tau_d = \frac{T}{\alpha} \ln 2 \; = \tau \; \ln 2 \tag{1.17}$$

Il valore della derivata in $t = 0$, che permette di individuare la retta tangente alla curva nell'origine, è dato dalla relazione seguente:

$$y'(0) = \frac{A\alpha}{T} = \frac{\alpha}{\tau} \tag{1.18}$$

Da tali relazioni si vede come i valori della costante di tempo, del tempo di raddoppio e della tangente nell'origine dipendano dai valori di α e T: tenendo α costante, a bassi valori di T corrispondono andamenti rapidamente crescenti (bassi valori di τ e di τ_d) mentre ad elevati valori di T corrispondono andamenti lentamente crescenti (elevati valori di τ e di τ_d). Andamenti opposti si ottengono tenendo costante T e facendo variare α.
L'equazione 1.15 descrive, invece, un andamento rapidamente decrescente nel tempo che può essere descritto sfruttando relazioni analoghe alle 1.16, 1.17 e 1.18 solo che invece che di tempo di raddoppio si parla di *tempo di dimezzamento*[12] definito sempre dalla 1.17 e la tangente nell'origine ha pendenza opposta e perciò negativa.
In questo caso dopo un tempo pari a τ la funzione si è ridotta ad un valore pari a 0.368 del valore iniziale. Anche in questo caso i valori di α e di T influenzano la velocità di variazione della variabile verso il valore di regime (in questo caso 0).
Nel caso di una evoluzione asintotica dei valori di una variabile (cfr. i tracciati etichettati come (b) in figura 1.4), due sono gli andamenti possibili:

1. il valore iniziale della variabile all'istante $t = 0$, A_0, è maggiore del valore a regime A (ovvero, idealmente, a $t = +\infty$)
   oppure

[11] Il termine inglese corrispondente è *doubling time*.
[12] Il termine inglese corrispondente è *halving time*.

2. il valore iniziale della variabile all'istante $t = 0$, A_0 (può essere anche $A_0 = 0$), è inferiore al valore a regime A (ovvero, idealmente, a $t = +\infty$)

Il primo caso è assimilabile al caso descritto dall'equazione 1.15 e infatti lo si può descrivere con una equazione del tipo:

$$y(t) = A + (A_0 - A)e^{-\alpha t/T} \ \ con\, \alpha \ > \ 0\, e\, T > 0 \tag{1.19}$$

mentre il secondo caso (considerando un valore iniziale nullo[13]) può essere descritto da una equazione del tipo:

$$y(t) = A(1 - e^{-\alpha t/T}) \ con\, \alpha \ > \ 0\, e\, T > 0 \tag{1.21}$$

Anche in questo caso si può definire la costante di tempo τ ed è possibile valutare il valore della tangente, rispettivamente, in $(0, A_0)$ e $(0, 0)$ svolgendo considerazioni analoghe alle precedenti.
Nel caso di una crescita a S (cfr. il tracciato etichettato come (c) in figura 1.4) la variabile mostra su un intervallo $[0, t_0]$ una crescita esponenziale dal valore 0 ad un valore A_1 seguita da una evoluzione asintotica che, nel caso ideale, fa in modo che la variabile raggiunga un valore costante A_2.
L'andamento può essere descritto dalla equazione seguente, in cui $u(t)$ rappresenta la funzione gradino unitario[14]:

$$y(t) = Ae^{\alpha t/T}(u(t) - u(t - t_0)) + A_1(1 + (A_2 - A_1)(1 - e^{-\beta(t-t_0)/T'}))u(t - t_0) \tag{1.23}$$

con $\alpha > 0$, $\beta > 0$, $T > 0$, $T' > 0$ e $A_0 = Ae^{\alpha t_0/T}$.
Nel caso di un andamento oscillatorio (cfr. il tracciato etichettato come (d) in figura 1.4) la variabile mostra un andamento che fluttua attorno ad un livello A_0. Tale andamento può mostrare una maggiore o minore periodicità e una maggiore o minore regolarità nel tempo.
Il caso più semplice è quello di un andamento periodico semplice descrivibile da una equazione come la seguente[15]:

$$y(t) = A\cos(2\pi t/T + \varphi) \tag{1.24}$$

[13] Le equazioni 1.19 e 1.21 sono scrivibili entrambe nella forma

$$y(t) = A + (A_0 - A)e^{-\alpha t/T} \tag{1.20}$$

il motivo della differenziazione è di tipo espositivo.

[14] La funzione gradino unitario è una funzione descritta dalle relazioni seguenti:

$$u(t) = 0\, \forall\, t < 0, \ \ u(t) = 1\, \forall t \geq 0 \tag{1.22}$$

mentre la funzione $u(t - t_0)$ è la stessa funzione traslata in t_0.

[15] Si ricorda che un segnale è detto essere periodico se esiste un valore T_0 tale che $y(t) = y(t + T_0)$.

Nell'equazione 1.24 il valore φ rappresenta la fase della variabile ovvero il valore dell'angolo all'istante iniziale. Il segnale descritto dalla 1.24 varia in ampiezza fra i valori A e $-A$ ed ha un andamento periodico di periodo $T > 0$.
Una variante degna di interesse dei segnali periodici sono i *segnali periodici smorzati* ovvero i segnali periodici la cui ampiezza tende a decrescere nel tempo fino ad annullarsi.
Un modello di tali segnali è costituito dalla seguente equazione ([BS91]):

$$y(t) = Ae^{(-\alpha t/T')}cos(2\pi t/T + \varphi) \tag{1.25}$$

(con $T > 0$) in cui all'oscillazione rappresentata dalla funzione *coseno* si sovrappone lo smorzamento imposto dall'esponenziale la cui ampiezza decresce nel tempo fino ad annullarsi.

1.3.3 Anelli di *feedback* e diagrammi CL

Allo scopo di descrivere le varie strutture dei sistemi che causano gli andamenti tipici descritti nella sezione 1.3.2 è necessario introdurre una notazione che permetta di rappresentare graficamente le relazioni di causa-effetto fra i vari elementi di un sistema.
Tale notazione fa uso di *etichette* (dette anche variabili) e di *archi orientati*: le prime individuano le variabili descrittive della struttura del sistema mentre i secondi individuano le relazioni di causa-effetto suddette.
Mediante tali elementi grafici (ed altri che saranno introdotti a breve) è possibile costruire diagrammi CL descrittivi dei vari sistemi. I diagrammi così costruiti possono contenere i cosiddetti anelli di feeback (o anelli causali, in inglese *causal loop*) ovvero ([Kir98]) sequenze chiuse di cause ed effetti: in presenza di un anello di $feedback$ si ha che un elemento del diagramma influenza se stesso attraverso la cosiddetta catena di reazione (cfr. anche la sezione 1.2.5). Gli elementi dei diagrammi che non fanno parte di anelli di $feedback$ e appartengono a catene di cause-effetti che non si chiudono su se stesse fanno parte dei cosiddetti *anelli aperti* (o *open loop*)[16].
Oltre agli elementi visti, per poter analizzare in modo efficace la struttura di un sistema è necessario conoscere qualcosa di più in merito alle relazioni causa-effetto fra i vari elementi dei diagrammi.
Dati due elementi A e B tali che A è la causa e B l'effetto (e pertanto esiste un arco orientato da A a B detto *legame causale*) è necessario stabilire se tale relazione è di proporzionalità diretta o inversa.
Nel primo caso una variazione di A (aumento o diminuzione) causa una variazione dello stesso segno di B per cui a fianco della freccia dell'arco orientato da A a B compare un segno $+$.
Nel secondo caso ad una variazione di A (aumento o diminuzione) corrisponde

[16]Si parla al proposito, rispettivamente, di approccio *closed loop* e di approccio *open loop*.

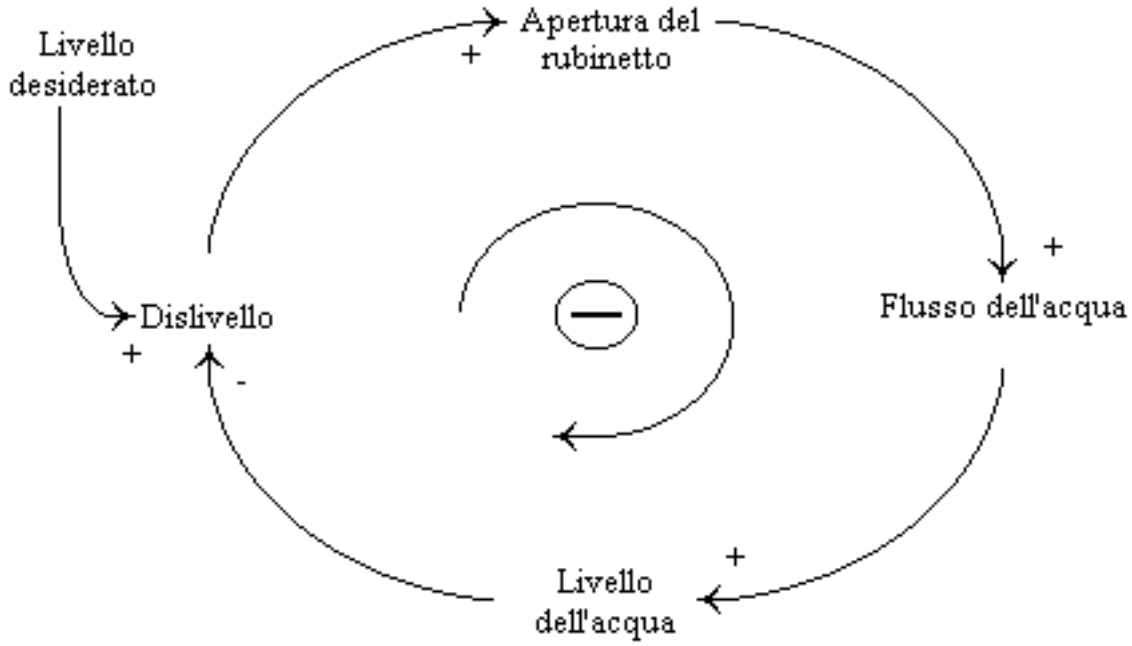

Figura 1.5: *Archi con segno in diagrammi CL*

una variazione di segno opposto di B (rispettivamente, diminuzione o aumento) per cui a fianco della freccia dell'arco orientato da A a B compare un segno $-$.
La figura 1.5, tratta da [Kir98] con minimi adattamenti, descrive un *sistema* per il riempimento di un bicchiere di acqua ovvero il *processo* di riempimento di un bicchiere di acqua.
Sebbene nel seguito i termini *sistema* e *processo* saranno usati come sinonimi, il primo pone l'accento sui moduli mentre il secondo pone l'accento sulle attività.
L'analisi del diagramma di figura 1.5 può iniziare da uno qualunque dei suoi elementi. Supponendo, per semplicità, che il bicchiere sia inizialmente vuoto, si può partire dal "livello dell'acqua" osservando che tanto più questo è basso tanto maggiore è il valore del dislivello (ovvero della differenza fra tale livello e il "livello desiderato") da cui il segno $-$ sul corrispondente legame causale. Proseguendo è facile intuire un legame di proporzionalità diretta fra l'entità del dislivello e la posizione del rubinetto (più o meno aperto) e fra questo e l'entità del "flusso dell'acqua" che a sua volta influenza il valore del "livello dell'acqua" con una relazione di proporzionalità diretta.
Dal momento che il processo descritto tende ad annullare il valore del "dislivello" in modo da causare la chiusura del rubinetto e l'annullamento del flusso, l'anello di *feedback* è un anello di *feedback negativo* che tende a portare il sistema descritto nella posizione di equilibrio di "bicchiere riempito fino al livello desiderato".

Il diagramma di figura 1.5 segnala tale fatto inserendo un segno − circolettato ($\ominus$) al centro dell'anello di *feedback* . Una regola pratica che permette di determinare se un anello di *feedback* è di tipo positivo oppure negativo è la seguente: un anello di *feedback* è di tipo *positivo* (e tale fatto viene indicato mettendo un segno + circolettato, $\oplus$, al suo centro) se contiene un numero pari di legami causali di segno negativo mentre è di tipo *negativo* (e tale fatto viene indicato mettendo un segno − circolettato, $\ominus$, al suo centro) se contiene un numero dispari di legami causali di segno negativo.
Si fa notare come non tutti gli elementi della figura 1.5 sono descritti da variabili collegate a formare un anello di *feedback* e influenzabili dalla evoluzione del sistema: la figura contiene, infatti, un elemento che fissa un valore indipendente dalla dinamica del sistema ma che la condiziona pesantemente, tale elemento è rappresentato dalla *variabile esogena* "livello desiderato".
Oltre alla presenza di tale variabile esogena si fanno notare le caratteristiche di altre due variabili: il "flusso dell'acqua" e il "livello dell'acqua". Come sarà argomentato in modo più approfondito nella sezione 1.3.4, la prima rappresenta una grandezza caratterizzata da una velocità mentre la seconda è una grandezza cui può essere associato un livello di accumulo. La prima, pertanto, è una una variabile di tipo flusso (nel seguito solo *flusso*) mentre la seconda è una variabile di tipo livello (nel seguito solo *livello*).

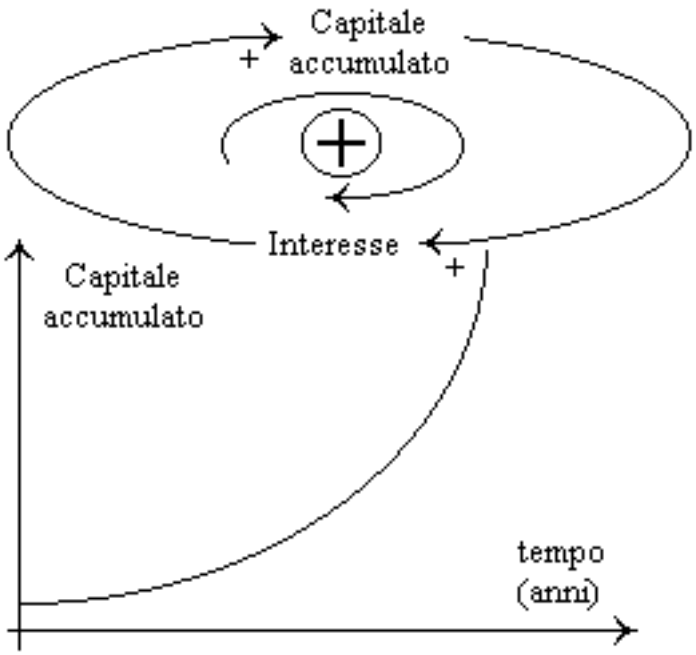

Figura 1.6: *Anello di feedback positivo e andamento di una delle variabili*

La figura 1.6 (tratta da [Kir98]) illustra un anello di *feedback* positivo. Sinonimi possibili sono *ciclo virtuoso*, se il fenomeno descritto ha una valenza positiva, e *ciclo vizioso* se, viceversa, ha una valenza negativa.
Se nel caso di anelli di *feedback* negativo il sistema tende a portarsi, tranne che in situazioni particolari[17], verso uno stato di equilibrio, nel caso si abbia un anello di fedback positivo di solito si assiste a crescite esponenziali di una o più variabili dell'anello. Nel caso di figura 1.6, ad esempio, se si inizia l'analisi dalla variabile "capitale accumulato" si ha che quanto più questo è elevato tanto maggiore (a parità di tasso di interesse[18]) sarà l'entità dell'"interesse" annuo che, a sua volta, possiede un legame di proporzionalità diretta (qualora venga mantenuto sul conto corrente e non sia prelevato) col "capitale accumulato": in questo caso si ha un anello di *feedback* positivo nel quale entrambe le variabili ("capitale accumulato", cfr. la figura 1.6, e "interesse") mostrano una evoluzione di tipo esponenziale.
Come risulta dalla figura 1.5, un anello di *feedback* negativo fa si che un sistema tenda a portarsi vero una condizione di equilibrio. In quel caso la condizione di equilibrio è rappresentata da uno stato finale (rubinetto chiuso, flusso dell'acqua nullo e bicchiere riempito fino al livello desiderato) raggiunto a partire da uno stato iniziale in cui il valore iniziale della variabile di interesse (in questo caso "livello dell'acqua") è inferiore al valore in condizione di equilibrio. Un altro esempio è quello di un sistema di riscaldamento ideale in cui la temperatura iniziale T_0 è inferiore a quella desiderata T_f.
Si parla, in tali casi, di *comportamento asintotico dal basso*: la variabile di interesse raggiunge il valore finale a partire da un valore iniziale inferiore, in un tempo teoricamente infinito.
Se, invece, il valore iniziale della variabile di interesse è superiore al valore in condizione di equilibrio (come accade in un sistema di raffreddamento ideale in cui la temperatura iniziale T_0 è superiore a quella desiderata T_f) si parla di *comportamento asintotico dall'alto*: la variabile di interesse raggiunge il valore finale a partire da un valore iniziale superiore, in un tempo teoricamente infinito.
Entrambi gli andamenti sono illustrati nella figura 1.4 dalle curve etichettate (b).
Nel caso in cui un anello di *feedback* negativo contiene dei ritardi di entità non trascurabile l'evoluzione invece che di tipo asintotico può essere di tipo oscillatorio.
Una ipotesi soggiacente all'evoluzione di tipo asintotico è infatti che la correzione (rappresentata in genere dal vincolo causale di segno negativo) agisca istantaneamente in modo da non essere mai di entità eccessiva ripetto ai valori correnti delle variabili su cui agisce.

[17] In generale non è possibile, infatti, affermare che in presenza di un anello di *feedback* negativo il sistema da questi modellizzato evolve sempre verso uno stato di equilibrio.

[18] Come dovrebbe essere chiaro già da questo esempio e come sarà chiaro da altri esempi, spesso nei diagrammi sono presenti ipotesi soggiacenti che, almeno per una maggiore leggibilità, dovrebbero essere espresse mediante variabili, in genere esogene.

Nei casi in cui si ha una evoluzione di tipo oscillatorio le alternative possibili sono due e quale delle due sia seguita dipende dalla struttura del sistema e dal valore dei ritardi e delle variabili: nel primo caso il sistema può oscillare indefinitamente attorno alla posizione di quilibrio mentre nel secondo l'ampiezza delle oscillazioni decresce nel tempo e le variabili riescono ad assumere valori tipici della posizione di equlibrio.
In [Kir98] viene presentato un esempio di andamento oscillatorio causato dal ritardo fra la rilevazione dell'entità reale della domanda di un prodotto e l'acquisizione di tale dato da parte del sistema produttivo. In tal caso la produzione prosegue per un certo periodo in modo da superare la domanda per poi scendere al di sotto della domanda a causa dei ritardi con cui l'esaurimento delle scorte viene comunicato al sistema produttivo. Atri esempi possono essere i seguenti:

1. il sistema (semplificato) che descrive il legame fra il tasso di immigrazione e il numero di opportunità di lavoro disponibili,

2. un sistema termostatico.

Nel primo caso si ha un legame di proporzionalità diretta fra il numero di opportunità di lavoro disponibili e il tasso di immigrazione e un legame di proporzionalità inversa fra questo e il numero di opportunità di lavoro disponibili, dato che gli immigrati occupano tali opportunità di lavoro. Se esiste un ritardo con cui i potenziali immigrati vengono a conoscenza del numero reale di opportunità di lavoro si può avere un andamento oscillatorio del tasso di immigrazione ovvero si possono avere periodi di "sovra-immigrazione" seguiti da periodi di "sotto-immigrazione", considerando come livello di equilibrio un valore del tasso di immigrazione tale da consentire la saturazione del numero di opportunità di lavoro effettivamente disponibili.
Nel secondo caso, partendo da una condizione in cui $T_0 < T_f$, si ha una azione iniziale di riscaldamento che può essere eccessiva (a causa sia del ritardo con cui un termometro presente nel sistema regola tale azione sia della sensibilità di tale termometro [19]) e pertanto può innalzare la temperatura del sistema oltre quella desiderata. A tale azione deve seguire, pertanto, una azione di raffreddamento che, a sua volta, può essere eccessiva (per motivi analoghi ai precedenti) per cui la temperatura del sistema scende al di sotto di quella desiderata e così via. In tal modo la temperatura effettiva del sistema oscilla fra due valori estremi T_{min} e T_{max} attorno al valore atteso T_A e tale oscillazione può proseguire indefinitamente nel tempo.
Combinando fra di loro in vari modi anelli di $feedback$ positivo ed anelli di $feedback$ negativo ideali (ovvero privi di ritardi) si possono ottenere andamenti dei tipi più disparati. Il caso più semplice ([Kir98]) è quello di un anello di $feedback$ positivo che condivide una variabile (di cui si vuole monitorare l'andamento) con un anello di $feedback$ negativo.

[19] Si fa notare che un termometro reale reagisce a variazioni finite di temperatura.

L'anello di *feedback* positivo può dare inizialmente luogo ad una crescita esponenziale di tale variabile fino a che questa non raggiunge un valore a partire dal quale entra in gioco l'anello di *feedback* negativo che fa in modo che la variabile si assesti su un valore di equilibrio. In tal modo la variabile mostra una crescita a S: la crescita esponenziale del tratto iniziale viene frenata da limiti fisici del sistema (esaurimento di risorse, saturazione di un ambiente o altre simili) e si trasforma in un comportamento asintotico, in genere dal basso.
Sulla base degli esempi visti verrano a questo punto svolte alcune considerazioni di carattere generale in merito ai diagrammi CL ([Kir98]).
Il punto di partenza per la costruzione di un diagramma CL è rappresentato dalla individuazione degli *eventi* che si ritengono significativi per la comprensione della struttura di un sistema, eventi cui verranno associate delle variabili e dei quali si cerca di determinare l'andamento caratteristico (anche solo da un punto di vista qualitativo) in modo da poterlo associare ad una o più strutture "paradigmatiche" (ovvero uno o più anelli di *feedback* positivo e negativo anche interagenti fra loro).
Una volta individuati gli eventi che faranno parte di un diagramma CL, a ciascuno di essi si associa una grandezza variabile nel tempo (ovvero una *variabile*) mentre le azioni sono rappresentate dai legami causali fra gli elementi che dovrebbero essere scelti in modo da descrivere un legame causa-effetto più che una successione temporale.
Durante la costruzione di un diagramma CL a partire da un insieme di variabili è necessario avere presente la possibilità che altre variabili debbano essere introdotte per rappresentare sia nuove relazioni causa-effetto sia influenze del mondo esterno, sotto forma di variabili esogene. Le variabili esogene sono di solito presenti nel caso di anelli di *feedback* negativo per fissare il valore di regime di una variabile e per permettere di calcolare la differenza istantanea fra tale valore e il valore corrente, differenza che determina l'evoluzione del sistema verso lo stato di equilibrio.
In molti casi è necessario introdurre uno o più *ritardi* all'interno di un diagramma CL in modo da spiegare la differenza fra il valore reale di una variabile e il valore percepito. L'introduzione dei ritardi permette sia di modellizzare i sistemi in modo più realistico (nessun sistema fisico mostra una risposta istantanea ad una sollecitazione) sia di spiegare alcuni comportamenti controintuitivi dei sistemi,come si è visto nel caso degli andamenti oscillatori.
Infine esistono due requisiti che possono essere in conflitto fra di loro: il primo richiede che i legami causali fra le variabili non siano troppo complessi, mentre il secondo richiede che i diagrammi, pur nella loro semplicità, siano in grado di evidenziare la struttura interna che può dare luogo al comportamento osservato.

1.3.4 I diagrammi FD

I diagrammi CL consentono di descrivere le relazioni di causa-effetto che esistono fra le variabili descrittive di un sistema dinamico essenzialmente da un punto di vista qualitativo. In tali modelli manca, infatti, ogni informazione sia sulla velocità di variazione delle variabili sia su quali siano le relazioni matematiche che governano effettivamente l'evoluzione delle variabili.
Per una migliore caratterizzazione dei sistemi, inoltre, vedremo come sia necesario introdurre delle distinzioni sia sulle variabili sia sui legami causali fra queste. Per far ciò sia alle variabili sia ai legami causali sarà associato il concetto di *tipo*.

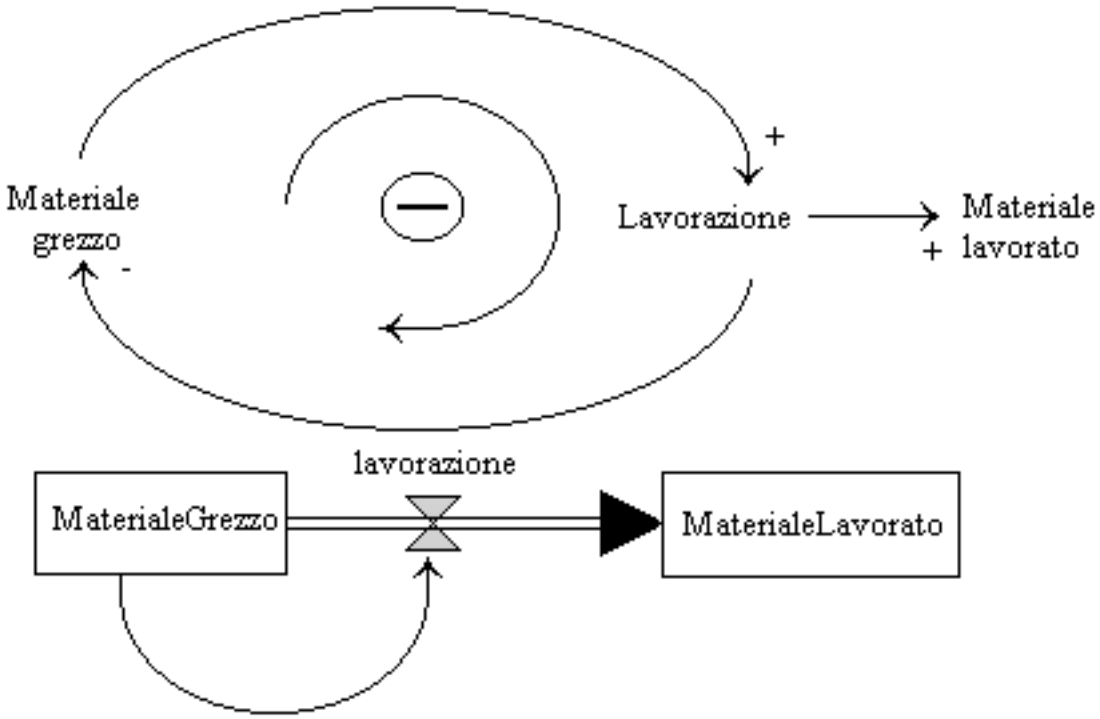

Figura 1.7: *Dai diagrammi CL ai diagrammi FD*

La figura 1.7[20] ([Kir98]) contiene (in alto) il diagramma CL di un sistema di trasformazione (molto semplificato) di una certa quantità di "materiale grezzo" in "materiale lavorato" mediante una non meglio specificata "lavorazione": tanto maggiore è la quantità di materiale grezzo e tanto maggiore può essere la lavorazione; a sua volta tanto più questa è elevata tanto minore è la scorta di materiale

[20]Nelle figure illustrative dei diagrammi FD viene usata la notazione utilizzata da package commerciali per la simulazione di Sistemi Dinamici quali *Vensim*.

grezzo residuo e tanto maggiore è la quantità di materiale lavorato. Da tali considerazioni discendono i segni sui legami causali fra le variabili nel diagramma CL. Si ha un anello di $feedback$ negativo che tende mantenere il sistema in una condizione di equilibrio ed eventualmente a portare il sistema nello stato di "materiale grezzo esaurito, lavorazione ferma e materiale lavorato pari alla quantità totale di materiale grezzo".
Ciò che, purtroppo, manca nel diagramma CL è sia una qualche indicazione della velocità con cui il "materiale grezzo" viene trasformato in "materiale lavorato" sia una qualunque indicazione sull'andamento nel tempo di tale trasformazione. Tali informazioni, invece, possono essere molto importanti sia in tutti i casi in cui è necessario dover monitorare con precisione le quantità correntemente disponibili di materiale grezzo e di materiale lavorato sia, ad esempio, in tutti i casi in cui il materiale grezzo o quello lavorato (o entrambi) sono deperibili e presentano problemi di stoccaggio e spedizione.
Al fine di introdurre nei modelli tali informazioni (ed altre di cui diremo fra poco) è necessario passare ai diagrammi FD.
La figura 1.7 contiene, in basso, il diagramma FD corrispondente al diagramma CL precedentemente descritto.
Il diagramma si basa su una notazione che utilizza un certo numero di tipi per individuare le variabili (la figura ne contiene solo due, altri saranno introdotti in seguito) e due tipi per i legami causali.
Le variabili presenti nella figura 1.7 possono essere

1. di tipo flusso,
2. di tipo livello

mentre i legami causali possono individuare

1. un flusso di materiali,
2. un flusso di informazioni[21].

I diagrammi FD sfruttano una notazione grafica, che consente di distinguere sia le variabili sia i legami causali per tipo, per sviluppare modelli quantitativi dei sistemi e consentono, come i diagrammi CL, di stabilire relazioni fra variabili che variano nel tempo.
Tornando alla figura 1.7, le variabili "materiale grezzo" e "materiale lavorato" individuano quantità che si accumulano nel tempo e pertanto sono di tipo livello e sono rappresentate come racchiuse in scatole rettangolari mentre la variabile "lavorazione" è rappresentata a fianco di una valvola ed è pertanto di tipo flusso[22]: un $flusso$ è una variabile var cui si può associare una velocità, ovvero un

[21] Nell'ottica della *system dynamics* tali elementi sono i componenti dei sistemi di cui si vuole esaminare il comportamento.

[22] Nel seguito verrà omessa la specificazione "di tipo" per le variabili per cui le variabili viste saranno dette semplicemente flussi o livelli, a seconda dei casi.

rapporto $\frac{\Delta var}{\Delta t}$, mentre un *livello* è una variabile che rappresenta l'accumularsi di una risorsa o di un bene o di una sostanza e corrisponde ad una *variabile di stato* della teoria dei sistemi.
Oltre alle variabili, il diagramma CL della figura 1.7 contiene i legami causali fra "materiale grezzo" e "lavorazione", fra "lavorazione" e "materiale lavorato" e fra "lavorazione" e "materiale grezzo".
In corrispondenza di tali legami il diagramma FD presenta un flusso di materiali fra i due livelli, flusso controllato dalla variabile flusso "lavorazione", e un flusso di informazione fra la variabile livello "materiale grezzo" e la variabile flusso "lavorazione": nel diagramma FD è rappresentata una trasformazione di "materiale grezzo" in "materiale lavorato" governata dalla relazione (di tipo informativo) esistente fra la quantità di "materiale grezzo" e la rapidità di lavorazione.
Il diagramma FD è, pertanto, caratterizzato, oltre che dalle variabili, da *flussi di materiale* e da *flussi di informazione*: i primi sono rappresentati come dei "tubi" su cui si innestano le "valvole" di controllo, i secondi sono rappresentati con archi orientati a tratto sottile che ne evidenziano il carattere "immateriale".
La caratteristica fondamentale dei flussi di materiale è quella di essere punto-punto e conservativi. Un flusso di materiale può pertanto collegare fra loro solo due nodi (punto-punto) e deve, inoltre, soddisfare le equazioni di conservazione ai nodi[23] attraverso cui fluisce e ciò rende necessaria l'introduzione di due tipi particolari di nodi che diremo

1. sorgenti
2. pozzi

in modo da poter modellizzare sia il punto di partenza sia il punto di arrivo di un flusso conservativo, di solito localizzati nel *mondo esterno* rispetto al sistema che si sta modellizzando. I flussi di materiale sono perciò caratterizzati da *nodi sorgente*, che modellizzano i punti di ingresso del materiale nel sistema, da nodi di controllo e di accumulo (rispettivamente *flussi* e *livelli*) e da nodi *nodi pozzo*, che modellizzano i punti di uscita del materiale dal sistema.
Il diagramma FD raffigurato in basso nella figura 1.7 è caratterizzato da un flusso di informazioni dal livello "materiale grezzo" al flusso "lavorazione" che sta ad indicare che il primo esercita una certa influenza sul secondo ovvero che l'informazione relativa alla quantità di materiale grezzo ne condiziona la lavorazione e, quindi, la trasformazione in materiale lavorato. Anche se non risulta evidente dalla figura 1.7 i flussi di informazioni possono essere di tipo *uno-a-molti* e non sono conservativi.
Un flusso di informazioni può, infatti, nascere da qualsiasi elemento di un diagramma, da cui il suo non essere conservativo, e può influenzare un numero

[23] L'equazione di conservazione del flusso ad un nodo impone che la differenza fra il flusso uscente e il flusso entrante sia pari all'accumulo nel nodo. I nodi privi di accumulo sono detti *nodi di transito* mentre quelli con accumulo, nel nostro caso, sono detti *livelli*.

qualunque di altri elementi, da cui la sua caratteristica uno-a-molti.
In questa e nelle precedenti sezioni sono stati introdotti due strumenti per la modellizzazione dei Sistemi Dinamaici e cioè i diagrammi CL e i diagrammi FD. Sebbene i secondi abbiano un maggiore potere descrittivo rispetto ai primi le relative *rappresentazioni grafiche* mancano ancora di *caratteristiche quantitative* che permettano di caratterizzare l'evoluzione dei sistemi. Ritornando al caso della figura 1.7, il diagramma FD non contiene indicazioni in merito alla legge di variazione del "materiale grezzo" né il legame esistente fra la quantità di materiale grezzo e il flusso di lavorazione né indicazioni in merito ai valori iniziali del livelli per cui è necessario arricchire il diagramma con caratteristiche quantitative che compaiono sotto forma di *equazioni* e/o di *valori iniziali*.
L'approccio adottato nella impostazione delle caratteristiche quantitative si basa sulle seguenti ipotesi ([Kir98]):

1. i flussi (di materiali e di informazioni) sono *continui*,
2. i flussi sono di tipo *deterministico*.

Un flusso è considerato essere continuo ([Kir98]) se il materiale o l'informazione che fluisce può essere suddivisa in quantità piccole a piacere, sia rispetto al flusso sia rispetto al periodo di tempo durante il quale il flusso viene monitorato. Tale assunzione di solito permette di ottenere risultati abbastanza accurati anche in casi in cui è palesemente falsa e, in più, semplifica notevolmente sia la costruzione del modello sia la sua soluzione.
Un flusso, invece, è detto essere deterministico se è completamente specificato dai valori delle variabili dei nodi alle sue estremità.
Partendo da tali ipotesi vediamo come sia possibile associare sia ai flussi sia ai livelli le rispettive equazioni descrittive dei loro andamenti nel tempo.
Nel caso dei *livelli* è necessario specificare i rispettivi valori iniziali oltre alla dipendenza dai flussi di ingresso e di uscita dal nodo mentre per i *flussi* è necessario specificare le rispettive equazioni matematiche che ne descrivono gli andamenti.
Applicando tali considerazioni al caso della figura 1.7 e considerando che all'istante $t > t_0$ generico la quantità di "materiale grezzo" è pari alla sua quantità iniziale (ovvero all'istante t_0) meno la quantità che è stata lavorata nell'intervallo $[t_0, t]$ è possibile scrivere la seguente equazione[24]:

$$MaterialeGrezzo(t) = MaterialeGrezzo(t_0) - \int_{t_0}^{t} lavorazione(\tau)d\tau \quad (1.26)$$

[24]La variabile MaterialeGrezzo è associata alla quantità "materiale grezzo". Nel seguito vedremo altri esempi di associazioni di questo tipo in cui i nomi delle variabili non contengono spazi. Livelli e flussi sono caratterizzati da unità di misura di cui si dirà nella sezione 2.3.5.

Una relazione analoga può essere scritta anche per il livello "materiale lavorato" (nel caso in cui all'istante iniziale t_0 sia $materiale\ lavorato = 0$):

$$MaterialeLavorato(t) = \int_{t_0}^{t} lavorazione(\tau)d\tau \tag{1.27}$$

Le equazioni 1.26 e 1.27 sono esempi particolari della relazione generale che regola il comportamento di un livello e che può essere espressa come segue:

$$Livello(t) = Livello(t_0) + mat_{in}(t) - mat_{out}(t) \tag{1.28}$$

in cui

$$mat_{in}(t) = \int_{t_0}^{t} \phi_{in}(\tau)d\tau \tag{1.29}$$

e

$$mat_{out}(t) = \int_{t_0}^{t} \phi_{out}(\tau)d\tau \tag{1.30}$$

se ϕ_{in} e ϕ_{out} sono i valori istantanei dei flussi in ingresso al/in uscita dal livello.
Nel caso dei livelli le equazioni descrittive hanno una struttura invariante per cui l'unica cosa che è necessario specificare è il valore iniziale assunto da ciascun livello, dato che le indicazioni in merito ai flussi si ricavano dalla struttura delle connessioni del livello con gli altri elementi del diagramma.
Nel caso dei flussi è necessario, invece, impostare le equazioni descrittive, una per ciascun flusso, mentre non ha senso specificare i valori iniziali.
Nel caso di figura 1.7 si ha un solo flusso, rappresentato dalla variabile "lavorazione", che dipende da un solo livello, ovvero "materiale grezzo", cui si può associare la seguente equazione:

$$lavorazione(t) = \alpha \times MaterialeGrezzo(t) \tag{1.31}$$

se la quantità di prodotto lavorato è una frazione α della quantità di "materiale grezzo" disponibile[25].
Le equazioni 1.26, 1.27 e 1.31 rappresentano la descrizione quantitativa associata al diagramma FD della figura 1.7 ovvero il suo *modello matematico*.
La soluzione di tale modello richiede che vengano specificate altre grandezze quali:

1. istante di inizio della valutazione delle relazioni del modello (ovvero valore di t_0;

2. istante di fine della valutazione delle relazioni del modello (ovvero valore di t_{max};

[25]Le unità di misura delle variabili possono essere [*ton*] per MaterialeGrezzo e MaterialeLavorato e [$ton/giorno$] per lavorazione.

3. il valore dell'intervallo di tempo fra due valutazioni successive ovvero il valore del parametro T già introdotto nella *Presentazione*.

Risolvendo tali equazioni con metodi diretti oppure, caso più comune nel campo della simulazione di sistemi dinamici, con metodi iterativi (cfr. la sezione 1.4) si ottengono i valori delle variabili *MaterialeGrezzo*, *MaterialeLavorato* e *lavorazione* nell'intervallo $[t_0, t_{max}]$. Tali valori disponibili sotto forma di tabelle vengono rappresentati mediante diagrammi cartesiani (cfr. la sezione 2.3). Come risulta anche dall'esempio della figura 1.7, flussi e livelli sono sufficienti per la costruzione dei diagrammi FD. Considerazioni legate alla leggibilità dei diagrammi e alla natura conservativa dei flussi di materiali hanno portato alla introduzione di *sorgenti* e *pozzi* ma è possibile introdurre altre variabili per aumentare ulteriormente la leggibilità dei diagrammi.

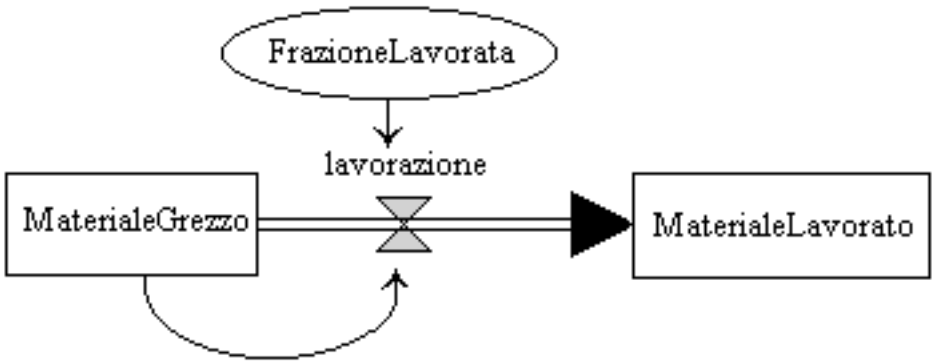

Figura 1.8: *Variabili ausiliarie in diagrammi FD*

Ad esempio la costante α dell'equazione 1.31 potrebbe essere associata ad una variabile ausiliaria costante $FrazioneLavorata$ (cfr. la figura 1.8): l'introduzione di tale variabile permette di chiarire la struttura del modello e di evidenziare la presenza di un fattore esterno nel legame fra due variabili (nel caso in esame "MaterialeGrezzo" e "lavorazione").
Le variabili ausiliarie possono essere di tipo costante, come in questo caso, oppure possono essere associate a funzioni predefinite del tempo (ad esempio una funzione gradino unitario, una funzione esponenziale oppure una funzione periodica).

1.3.5 Le *strutture di base*

Si passa ora ([Kir98]) all'esame di alcune strutture paradigmatiche di diagrammi FD (dette *strutture di base*) che danno origine agli andamenti tipici illustrati dalla figura 1.4, iniziando dalla struttura più semplice, ovvero un diagramma contenente un solo anello di $feedback$ positivo, diagramma illustrato nella figura

1.9 e che da luogo ad una crescita (o decrescita) esponenziale.

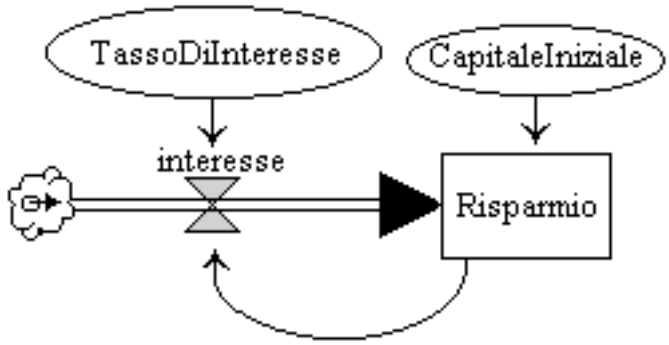

Figura 1.9: *Diagramma FD con un solo anello di feedback : evoluzione esponenziale*

In tale figura viene rappresentato il diagramma FD corrispondente al diagramma CL della figura 1.6, diagramma cui sono state aggiunte la variabile esogena $TassoDiInteresse$ e la variabile $CapitaleIniziale$: il valore della prima influenza la rapidità di incremento della variabile *Risparmio* (un livello) agendo sul valore della variabile *interesse* (un flusso); il valore della seconda stabilisce se il sistema è in una condizione di equilibrio ($CapitaleIniziale = 0$) oppure se il ciclo porta ad una crescita esponenziale positiva ($CapitaleIniziale > 0$, *ciclo virtuoso*) o negativa ($CapitaleIniziale < 0$, *ciclo vizioso*).
Nel caso del *ciclo virtuoso* si ha una crescita esponenziale del *Risparmio* dal momento che, a parità di $TassoDiInteresse$, tanto maggiore è il *Risparmio* tanto più elevato è l'*interesse* che, sommandosi al *Risparmio*, ne causa un aumento e così via.
Una volta tracciato il diagramma FD con le necessarie variabili è necessario stabilire i legami fra tali variabili ovvero è necessario impostare le equazioni del modello matematico la cui soluzione ci permetterà di ricavare gli andamenti nel tempo delle variabili significative (in questo caso le variabili sono il livello *Risparmio* e il flusso *interesse*).
Le equazioni necessarie sono le seguenti:

$$TassoDiInteresse = 0.05 \tag{1.32}$$

$$interesse(t) = TassoDiInteresse * Risparmio(t-1) \tag{1.33}$$

$$Risparmio(t) = Risparmio(t-1) + interesse(t) \tag{1.34}$$

$$CapitaleIniziale = 100 \tag{1.35}$$

$$Risparmio(t_0) = CapitaleIniziale \tag{1.36}$$

Oltre a tali equazioni è necessario impostare un certo numero di relazioni per le grandezze che pilotano la simulazione, ovvero la soluzione iterativa delle equazioni 1.33 e 1.34. Tali grandezze sono gli istanti di inizio (t_0) e di fine (t_{max}) della simulazione e il *passo di simulazione* ovvero l'ampiezza dell'intervallo di tempo fra due valutazioni successive delle 1.33 e 1.34. In tali equazioni si è indicato con $t-1$ l'istante precedente a quello corrente ma tale valore deve essere esplicitato per permettere la valutazione effettiva delle equazioni.
Per maggiori dettagli e la presentazione di alcune tecniche di risoluzione si rimanda alla sezione 1.4.
Un'altro andamento tipico ottenibile da un diagramma FD con un solo anello di *feedback* è illustrato nella figura 1.10 ([Kir98]).

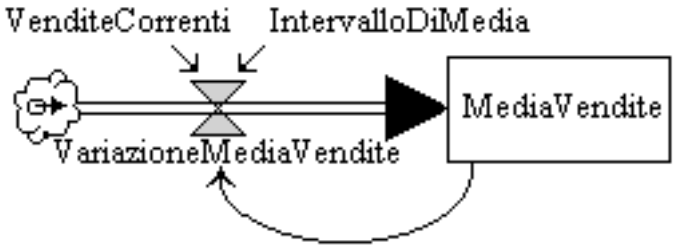

Figura 1.10: *Diagramma FD con un solo anello di feedback : evoluzione di tipo asintotico*

In questo caso si ha una variabile livello ($MediaVendite$) che "insegue" il valore di un'altra variabile di riferimento ($VenditeCorrenti$): come sarà illustrato più in dettaglio nella sezione 1.4, se la variabile $VenditeCorrenti$ passa da un valore costante $v1$ ad un altro $v2$, la variabile $MediaVendite$ tende al secondo valore con una evoluzione di tipo asintotico dal basso (se $v1 < v2$) o dall'alto (se $v1 > v2$).
In questo caso il sistema modellizzato viene testato con un segnale di ingresso che serve per studiare la risposta del sistema e, pertanto, per avere informazioni sulle sue caratteristiche, informazioni di tipo semantico e non sintattico.
Il segnale di prova è caratterizzato dalla seguente equazione[26]:

$$VenditeCorrenti(t) = VenditeCorrenti(t_0) + \Delta Vendite \times u(t - t_\Delta) \quad (1.37)$$

in cui t_Δ rappresenta l'istante di variazione del livello delle vendite e $\Delta Vendite$ rappresenta l'entità di tale variazione rispetto al valore delle vendite nell'intervallo $[t_0, t_\Delta)$ (VenditeCorrenti(t_0)).
Un ruolo importante è svolto nel modello dalla variabile $IntervalloDiMedia$ che determina la rapidità con cui la variabile $MediaVendite$ tende ad assumere lo

[26]La funzione $u(t - t_\Delta)$ è la funzione gradino unitario traslata in t_Δ, funzione definita nella sezione 1.3.2 alla quale si rimanda.

stesso valore della variabile $VenditeCorrenti$.
Inizialmente il sistema è in una condizione di equilibrio in cui:

$$MediaVendite(t) = VenditeCorrenti(t) \quad t \in [t_0, t_\Delta) \tag{1.38}$$

All'istante t_Δ al sistema viene applicata la perturbazione rappresentata dalla funzione a gradino (cfr. la 1.37) che fa si che il sistema subisca una evoluzione la cui velocità è regolata dal valore della variabile $IntervalloDiMedia$: tanto più tale valore è alto tanto più lentamente varia la variabile $MediaVendite$ e tanto più lungo risulta il transitorio, ovvero il tempo necessario perchè il sistema raggiunga la nuova condizione di equilibrio in cui vale ancora la 1.38 con i valori stabiliti dalla 1.37.

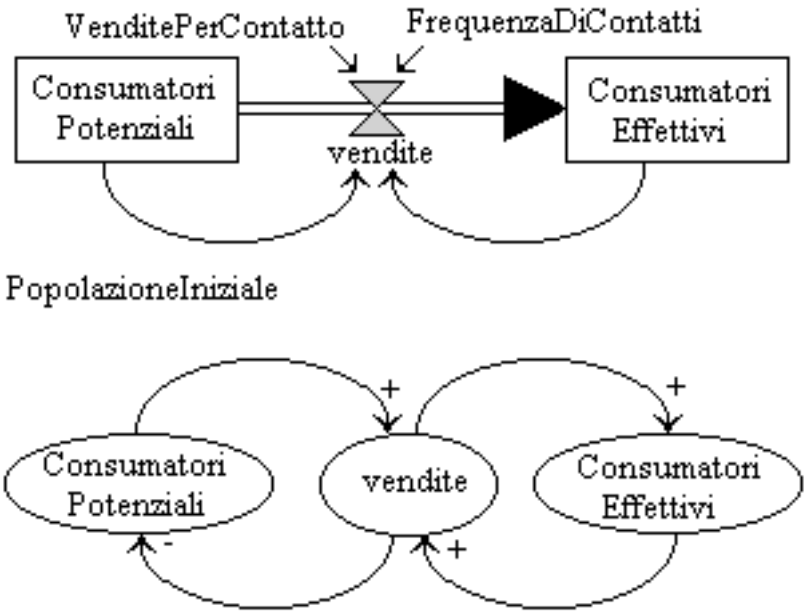

Figura 1.11: *Esempio di diagrammi FD e CL con "crescita a S" di una delle variabili*

Oltre alla crescita esponenziale e alle evoluzioni di tipo asintotico (dall'alto o dal basso) altri andamenti tipici sono la crescita a S e le evoluzioni di tipo oscillatorio (cfr. la figura 1.4). Anche tali andamenti sono riconducibili a strutture tipiche dei diagrammi FD, strutture più complesse di quelle viste sinora, dal momento che gli andamenti che devono essere descritti hanno una complessità maggiore dei precedenti.
È possibile, infatti, vedere, la crescita a S ([RAD$^+$83]) come ottenuta dalla successione di una crescita esponenziale seguita da una crescita asintotica (dal basso).

Una variabile che mostra una evoluzione nel tempo di questo tipo è una variabile che appartiene (utilizzando un diagramma CL) sia ad un ciclo di $feedback$ positivo sia ad un ciclo di $feedback$ negativo (cfr. la figura 1.11): inizialmente il primo ciclo è il ciclo dominante e causa una crescita esponenziale della variabile fino a che non prende il sopravvento il secondo che trasforma la crescita esponenziale in una crescita asintotica.
In modo analogo, un andamento oscillatorio può essere descritto, con una certa approssimazione, come una successione di crescite e decrescite a S, dove per decrescita a S si intende un andamento speculare (rispetto al valore asintotico) della crescita a S.
Sebbene un modello matematico degli andamenti oscillatori possa essere, infatti, la funzione $Acos(\omega t + \varphi)$, i fenomeni oscillatori reali non sono rappresentabili da tale equazione. Il loro tratto distintivo ([RAD+83]) è essenzialmente quello di essere caratterizzati da successioni di crescite verso un valore massimo e di decrescite verso un valore minimo con più o meno lunghi periodi di permanenza a tali valori estremi (o in loro intorni). Da tali considerazioni discende la possibilità di utilizzare l'approssimazione suddetta.
La figura 1.11 contiene un diagramma FD e il corrispondente diagramma CL (semplificato in quanto privo dei legami con le variabili esogene) in cui una delle variabili (e precisamente $ConsumatoriEffettivi$) mostra una crescita a S mentre un'altra ($ConsumatoriPotenziali$) mostra una decrescita a S speculare alla precedente.
Dal diagramma CL si può vedere come i $ConsumatoriPotenziali$ si trasformino in $ConsumatoriEffettivi$ grazie alle vendite e come il processo prosegua fino a che tutti i $ConsumatoriPotenziali$ non si sono trasformati in $ConsumatoriEffettivi$. Se $PopolazioneIniziale$ indica il numero iniziale di possibili consumatori (il cosiddetto "bacino di utenza") lo stato iniziale è individuato dalle seguenti relazioni:

$$ConsumatoriPotenziali = PopolazioneIniziale - ConsumatoriEffettivi = 490 \tag{1.39}$$

$$ConsumatoriEffettivi = 10 \tag{1.40}$$

mentre lo stato finale (in condizioni ideali) è individuato dalle seguenti relazioni:

$$ConsumatoriPotenziali = 0 \tag{1.41}$$

$$ConsumatoriEffettivi = PopolazioneIniziale = 500 \tag{1.42}$$

$$vendite = 0 \tag{1.43}$$

Il modello proposto dalla figura 1.11 si basa infatti sui seguenti assunti:

1. esiste una $PopolazioneIniziale$ di consumatori potenziali di un prodotto;
2. alcuni di loro acquisicono il prodotto in modo "autonomo" (cfr. la 1.40);

3. tali $ConsumatoriEffettivi$ entrando in contatto con gli altri $ConsumatoriPotenziali$ ne convertono un certo numero in nuovi $ConsumatoriEffettivi$.

Al punto 3 nel modello corrispondono, infatti, le equazioni seguenti:

$$vendite = FrequenzaDiContatti \times VenditePerContatto \tag{1.44}$$
$$\times ConsumatoriEffettivi \times ConsumatoriPotenziali$$

$$ConsumatoriEffettivi = 10 + \int vendite\, dt \tag{1.45}$$

$$ConsumatoriPotenziali = 490 - \int vendite\, dt \tag{1.46}$$

la cui valutazione è possibile date le seguenti relazioni per le variabili esogene:

$$VenditePerContatto = 0.1 \tag{1.47}$$

(ovvero su 10 $ConsumatoriPotenziali$ contattati da ognuno dei $ConsumatoriEffettivi$ 1 effettua l'acquisto e si trasforma in consumatore effettivo)

$$FrequenzaDiContatti = 0.02 \tag{1.48}$$

(rappresenta il numero di contatti per unità di tempo nel caso in cui i gruppi di $ConsumatoriPotenziali$ e di $ConsumatoriEffettivi$ contengano un elemento ciascuno).

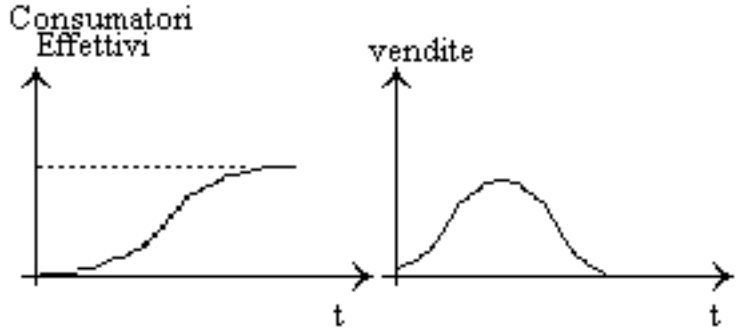

Figura 1.12: *Andamenti delle variabili "ConsumatoriEffettivi" e "vendite"*

Le equazioni 1.45 e 1.46, in questo contesto, hanno il solo compito di illustrare la trasformazione dei $ConsumatoriPotenziali$ in $ConsumatoriEffettivi$ per mezzo delle *vendite* e verranno esaminate più in dettaglio nella sezione 1.4.
Si fa notare che se si ha uno stato iniziale con:

$$ConsumatoriEffettivi = 0 \tag{1.49}$$

il flusso di vendite si annulla (cfr. la 1.44) per cui il sistema si mantiene nella condizione iniziale di equilibrio e non si ha nessuna evoluzione verso lo stato finale rappresentato dalle equazioni 1.41 e 1.42.
La figura 1.12 mostra l'evoluzione a S della variabile $ConsumatoriEffettivi$ (un livello) e quello della variabile *vendite* (un flusso). Il valore asintotico della variabile $ConsumatoriEffettivi$ corrisponde al valore della variabile $PopolazioneIniziale$ mentre la variabile *vendite* mostra una evoluzione a campana in cui il valore per $t = 0$ corrisponde al tasso di vendite associato al valore iniziale della variabile $ConsumatoriEffettivi$.
Nel caso in cui sia i $ConsumatoriEffettivi$ sia i $ConsumatoriPotenziali$ perdono, dopo qualche tempo, interesse ad un prodotto la variabile $ConsumatoriEffettivi$ arriva ad un massimo e poi decresce mentre la variabile *vendite* decresce più rapidamente. Questo comportamento, nelle ipotesi del modello, è dovuto al fatto che i $ConsumatoriEffettivi$, perdendo interesse, diradano i contatti con i $ConsumatoriPotenziali$ ed in più questi perdono interesse per conto loro.
Per modellizzare questi fenomeni si deve modificare la figura 1.11 introducendo gli elementi presentati nella figura 1.13.

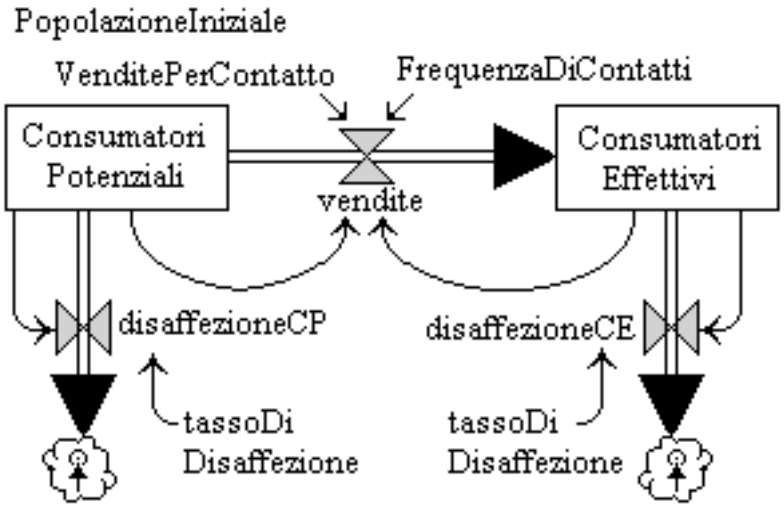

Figura 1.13: *Esempio di diagramma FD con "crescita a S" seguita da declino di una delle variabili*

In questo caso si hanno due flussi in uscita da $ConsumatoriEffettivi$ e $ConsumatoriPotenziali$ pilotati da una variabile esogena che diremo $tassoDiDisaffezione$ che fanno si che non tutti i $ConsumatoriPotenziali$ diventino $ConsumatoriEffettivi$ e che il numero di contatti fra i membri delle due classi si riduca in modo che la variabile $ConsumatoriEffettivi$ tenda a decresce-

re una volta raggiunto il massimo e la variabile *vendite* decresca più velocemente. La decrescita dei livelli $ConsumatoriEffettivi$ e $ConsumatoriPotenziali$ è giustificata dalla presenza di due flussi in uscita in più in modo che le equazioni dei livelli ora siano le seguenti:

$$ConsumatoriEffettivi = 10 + \int vendite\, dt - \int disaffezioneCE\, dt \quad (1.50)$$

$$ConsumatoriPotenziali = 490 - \int vendite\, dt - \int disaffezioneCP\, dt \quad (1.51)$$

I modelli visti sinora non sono stati caratterizzati da oscillazioni degli andamenti delle variabili dal momento che si è supposto che tutte le variazioni avvengano istantaneamente e siano propagate istantaneamente senza ritardi.
In presenza di ritardi si possono avere oscillazioni a livello di variabili chiave del modello (tipicamente *livelli* o *flussi*). Perchè un processo (ovvero l'evoluzione di un modello) mostri un andamento oscillatorio devono essere presenti almeno due livelli ([Kir98]) mentre il grado di oscillazione dipende dalla entità dei ritardi nel senso che tanto maggiori sono i ritardi tanto più ampie possono essere le oscillazioni, a parità di altre cause.
Nel caso le variabili del modello mostrino andamenti oscillatori si parla di andamenti ciclici delle variabili, che assumono periodicamente gli stessi valori, e di lunghezza di un ciclo. La lunghezza T_c del ciclo alla quale un processo oscilla in risposta ad una funzione in ingresso costituita da una funzione gradino unitario si dice una *risonanza* del processo mentre il suo inverso $1/T_c$ si dice *frequenza di risonanza*. In genere ([Kir98]) i processi sono caratterizzati da risposte di ampiezza maggiore in presenza di sollecitazioni in ingresso la cui frequenza è prossima alla frequenza di risonanza. In tali casi si dice che il sistema entra in risonanza e può mostrare un comportamento instabile nel senso che l'ampiezza delle variazioni è maggiore di quella che si avrebbe in assenza di risonanza e può essere crescente nel tempo.
Un esempio di sistema in cui una variabile può avere un andamento oscillatorio in presenza di ritardi è illustrato nella figura 1.14 in cui viene presentato un modello semplificato della immigrazione in area urbana (la variabile $FlussoMigratorio$) in presenza di opportunità di lavoro (la variabile *OpportunitàDiLavoro*).
In assenza di ritardi il diagramma CL si può leggere come segue: tanto maggiori sono le *OpportunitàDiLavoro* tanto più elevato è il $FlussoMigratorio$ e viceversa mentre tanto più elevato è il $FlussoMigratorio$ tanto minori sono le *OpportunitàDiLavoro*, che sono saturate dai nuovi arrivati, e viceversa.
Il sistema tende ad un equilibrio fra le due grandezze nel senso che in assenza di *OpportunitàDiLavoro* non si ha $FlussoMigratorio$ mentre, se le *OpportunitàDiLavoro* crescono/decrescono si ha un $FlussoMigratorio$ crescente/decrecscente. Se le *OpportunitàDiLavoro* assumono valori negativi (nel caso una industria si trasferisca in un'altra città) si arriva ad avere un

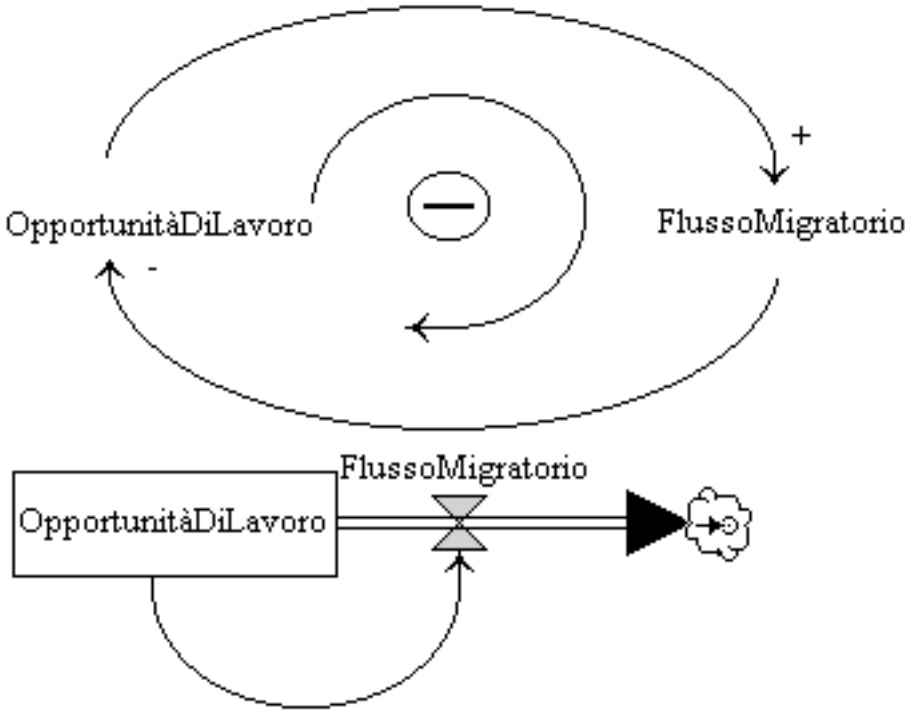

Figura 1.14: *Esempio di sistema con feedback negativo con possibili oscillazioni*

FlussoMigratorio in uscita dall'area urbana.
Le *OpportunitàDiLavoro* nel presente modello sono supposte variare anche in funzione di altri legami non illustrati nei diagrammi: nell'esempio si suppone che le *OpportunitàDiLavoro* varino per una qualche ragione attivando un legame di proporzionalità diretta con il *FlussoMigratorio* che, a parità di altre condizioni, influenza in un senso o in un altro le *OpportunitàDiLavoro*.
Se si introduce un ritardo nel legame fra le *OpportunitàDiLavoro* e il *FlussoMigratorio* si possono avere periodi si sovraimmigrazione e di sottoimmigrazione in modo da ottenere un andamento oscillatorio (eventualmente smorzato) della variabile *FlussoMigratorio*.

1.3.6 Lo sviluppo di un modello

Un *modello* viene usato per descrivere o un *sistema esistente* o un sistema *in fase di progettazione*. Nel primo caso si vuole elaborare un modello caratterizzato da un certo numero di variabili, correlate con grandezze misurabili sul sistema esistente, in modo che il comportamento delle variabili sia analogo al comportamento delle grandezze del sistema modellizzato. Nel secondo caso si elabora un modello in modo da prevedere il comportamento del sistema e valutare se il

comportamento soddisfa le specifiche del progetto.
Lo sviluppo di un modello su cui verranno eseguite delle simulazioni è, in entrambi i casi, un processo:

1. iterativo,

2. incrementale.

Lo sviluppo è *incrementale* perchè di solito si parte da una struttura minima che viene arricchita sulla base dei risultati delle simulazioni ed è *iterativo* perchè ad ogni stadio dello sviluppo si può dover tornare indietro qualora le modifiche introdotte abbiano dato luogo a comportamenti difformi da quelli osservati o prescritti dalle specifiche.
Ad ogni stadio dello sviluppo del modello, dopo aver assegnato le equazioni alle variabili del modello ed aver impostato i valori iniziali, si passa a simulare il modello (ovvero se ne risolvono le equazioni) e si valuta se il modello descrive con sufficiente accuratezza il comportamento osservato in un sistema esistente oppure mostra un comportamento conforme, entro certi limiti, alle specifiche, nel caso di un sistema in fase di progettazione.
Lo sviluppo di un modello parte dalla definizione dei confini del sistema e delle sue relazioni con il mondo esterno, caratterizzate da un certo numero di variabili dette *esogene*. Nel caso di un diagramma FD, a questo livello si definiscono le variabili "sorgente" e "pozzo" che rappresentano le origini e le destinazioni dei flussi di materiali al di fuori del modello.
Il passo successivo è rappresentato dalla individuazione delle variabili descrittive principali del modello, di solito *livelli* e *flussi*.
Fatto questo si individuano per prima cosa i flussi conservativi (o di materiali) che legano i livelli fra di loro, oppure con sorgenti e pozzi. Un livello può avere, infatti, flussi in ingresso (che ne causano un aumento) e flussi in uscita (che ne causano una diminuzione) e per ogni livello deve essere soddisfatta una equazione di bilancio.
Il passo successivo prevede l'introduzione dei flussi non conservativi (ovvero di tipo informativo). Lo scopo è quello di sfruttare le strutture di base, il cui comportamento paradigmatico è noto, e loro combinazioni al fine di ottenere il comportamento desiderato. A questo stadio si collegano ai flussi le variabili esogene.
Il modello, oltre a contenere gli elementi suddetti, ne può contenere altri, detti *variabili ausiliarie*, con il duplice scopo di

1. arricchire il modello da un punto di vista informativo,

2. aumentare la flessibilità del modello.

Una variabile ausiliaria (come sarà illustrato dagli esempi della sezione 1.4) può, infatti, rendere più chiaro il legame esistente fra un livello ed un flusso, nel senso che lo può caratterizzare in modo più compiuto, oppure permette di rendere

più accessibili le costanti caratteristiche delle equazioni dei livelli e dei flussi. In questo modo è possibile modificare i parametri del modello da una simulazione ad un'altra e passare da condizioni di equilibrio a condizioni di crescita o di decadimento, a seconda dei casi.
Oltre ai flussi, ai livelli, alle variabili esogene (che definiscono l'influenza del mondo esterno sul modello) ed alle variabili ausiliarie, un modello può contenere anche dei ritardi che tengono conto del tempo finito di propagazione nel modello di materiali ed informazioni.
Una volta che la struttura del modello è stata definita è necessario assegnare a ciascuna delle variabili una equazione ed una unità di misura in modo che il modello sia strutturalmente e dimensionalmente corretto, ovvero che le singole equazioni siano state scritte correttamente e che le unità di misura di una variabile siano congruenti con quelle delle altre variabili con cui interagisce.
Una volta che il modello è stato verificato è possibile, partendo da uno stato iniziale, simularlo, ovvero risolvere le equazioni associate alle variabili, ed ottenere gli andamenti nel tempo delle variabili.
Il modello, a questo punto, deve essere *validato* ovvero deve essere accettato oppure no. Il modello viene accettato se gli andamenti delle varabili sono, entro un certo margine, quelli attesi altrimenti viene rifiutato.
Nel caso un modello venga rifiutato, in alcuni casi lo si può modificare introducendo nuovi legami fra le variabili oppure eliminando legami esistenti che hanno dimostrato dare luogo a comportamenti indesiderati. Le modifiche possono comportare la definizione di variabili ausiliarie aggiuntive che permettono di caratterizzare in modo diverso i legami fra flussi e livelli. In ogni caso le modifiche non dovrebbero interessare la struttura di base del modello, quella relativa ai flussi conservativi: qualora si dovesse intervenire a tale livello ciò sarebbe indice di gravi pecche nel modello che renderebbero necessario reimpostare da zero il modello.
Una volta modificato il modello è necessario ripetere la simulazione. Le modifiche introdotte sono accettate qualora il modello abbia un comportamento più vicino a quello atteso, altrimenti sono rifiutate. Lo studio del modello, mediante simulazioni ripetute, avviene di solito partendo da una *condizione di equilibrio* (o di *stato stazionario*) in cui i livelli non variano (ovvero per ogni livello si ha un equilibrio fra il flusso in ingresso e il flusso in uscita). Un passo necessario, sotto questa ipotesi, è pertanto la definizione di uno stato di equlibrio per il modello. Ad esempio, nel caso di un modello di una popolazione, si può partire da una condizione in cui il *tasso di natalità* e la *vita media* si equivalgono in modo che il livello della *Popolazione* sia costante per poi passare a verificare cosa accade al modello se si modificano i valori delle variabili ausiliarie *tassoDiNatalità* e *vitaMedia*.
Partendo, pertanto, da uno stato di equilibrio si applicano delle sollecitazioni esterne, mediante variabili esogene che mostrano delle discontinuità, in modo da causare una evoluzione del modello. Il motivo per cui si parte da una situazione

di equilibrio e non da una situazione di non equilibrio (in cui anche in assenza di ingressi si hanno variazioni a livello di qualcuna delle variabili) è perché siamo interessati a vedere come il modello si comporta in presenza di variazioni delle variabili esogene (o di ingresso). In caso contrario può essere difficile separare le variazioni dei valori delle variabili legate alle variazioni delle variabili esogene da quelle legate alla mancanza di uno stato di equlibrio iniziale.

1.4 La *Computer Simulation* per la modellizzazione dei Sistemi Dinamici

1.4.1 Introduzione

Nella presente sezione[27] vengono riprese ed estese le considerazioni svolte nella sezione 1.3. Lo scopo è quello di approfondire la descrizione del legame esistente fra i diagrammi CL e i diagrammi FD e presentare le problematiche relative alla definizione delle equazioni descrittive di un diagramma FD e alla loro risoluzione.
Un diagramma CL, cui è possibile associare un grafo orientato i cui nodi e archi sono in corrispondenza biunivoca con gli elementi del diagramma CL, per poter essere trasformato in un diagramma FD deve essere "tipizzato": ai singoli nodi deve essere assegnato un tipo

$$\tau \in \mathcal{T} = \{level, rate, source, sink, delay, constant, auxiliary\} \tag{1.52}$$

mentre per ognuno degli archi è necessario specificare il segno ("+" o "-") e il tipo ("Materials", nel caso il flusso sull'arco sia di tipo conservativo, o "Information", in caso contrario). Si ricorda che il segno rappresenta il tipo di relazione esistente fra le grandezze agli estremi dell'arco: si ha il segno "+" se tale relazione è di proporzionalità diretta e il segno "-" se la relazione è, invece, di proporzionalità inversa.
Un diagramma FD, cui è possibile associare un multigrafo orientato le cui caratteristiche saranno esaminate a breve, oltre ad essere caratterizzato da elementi tipizzati è caratterizzato da un certo numero di equazioni, una per ognuno degli elementi del diagramma, la cui soluzione è l'obbiettivo dell'analisi del modello rappresentato dal diagramma FD.
Le equazioni sono associate a:

1. livelli,
2. flussi

[27] Il contenuto della presente sezione 1.4, compresi molti degli esempi, deriva in gran parte da ([RAD+83]).

e possono contenere *costanti* e/o *variabili esogene*, ovvero elementi i cui valori sono imposti dal *mondo esterno* sul *sistema modellizzato*.
Nel seguito verranno presentati alcuni esempi di diagrammi CL per ciascuno dei quali:

1. viene presentato e discusso il corrispondente diagramma FD,
2. vengono presentate le relative equazioni descrittive,
3. vengono discusse le problematiche relative alla risoluzione di tali equazioni.

Sebbene i diagrammi FD possano essere tracciati indipendentemente dai diagrammi CL, si è scelto di partire dai diagrammi CL da cui derivare i relativi diagrammi FD in modo da illustrare in pratica gli stessi concetti affrontati per via teorica nelle sottosezioni della sezione 1.3.
Prima di passare alla presentazione degli esempi, si ritiene, tuttavia, di far notare quanto segue. Le equazioni che saranno introdotte sono *equazioni alle differenze finite* scritte in funzione di una variabile $n \in \mathcal{N}$ sottintendendo la variabile T, che rappresenta l'ampiezza dell'intervallo di tempo fra due "osservazioni" consecutive, dove con il termine "osservazione" si definisce una operazione di calcolo delle equazioni per un particolare valore di n.
Se n assume valori in un intervallo $[n_{init}, n_{end}]$ si hanno, pertanto, $n_{end} - n_{init} + 1$ "osservazioni" delle quali quella per $n = n_{init}$ è quella iniziale che si basa sui valori iniziali dei flussi e dei livelli mentre quella per $n = n_{end}$ è quella finale che determina lo stato finale del modello.
Le equazioni alle differenze finite[28] hanno per i *livelli* la forma generale 1.53[29]:

$$livello(n) = livello(n-1) + \phi_{in}(n-1, n) - \phi_{out}(n-1, n) \tag{1.53}$$

in cui $\phi_{in}(n-1, n)$ e $\phi_{out}(n-1, n)$ rappresentano i contributi dei $flussi$ in ingresso e in uscita fra gli istanti $n-1$ e n ovvero su un intervallo di ampiezza T.
Per i flussi vedremo come la forma generale possa variare in funzione del tipo di legame esistente fra un flusso e il livello associato. A questo punto ci preme solo di far notare che:

1. un flusso può contribuire o al riempimento o allo svuotamento di un livello,
2. i livelli possono essere modificati solo per mezzo di flussi,
3. per il calcolo di un livello è necessario conoscerne il valore iniziale.

[28] Le equazioni che compaiono nel testo della presente sezione hanno solo scopo descrittivo e mirano ad esemplificare i legami fra le varie grandezze senza rappresentare necessariamente l'algoritmo usato per la loro risoluzione. Per ulteriori considerazioni al proposito si rimanda alla sezione 1.5.

[29] La notazione $\phi_{in}(n-1, n)$ non individua una funzione di due variabili $n-1$ ed n ma una funzione ad una variabile che assume un dato valore fra due istanti di tempo successivi. Nel seguito verranno usate notazioni simili con analogo significato.

1.4.2 Caso 1: *Anello singolo, feedabck positivo/feedback negativo*

La figura 1.15 rappresenta la tipica situazione di un modello con un solo anello in cui il *feedback* è di tipo positivo. Nel diagramma CL si hanno due variabili "itnerne" (*Interessi* e *Capitale*) ed una variabile esogena (*TassoDiInterese*).

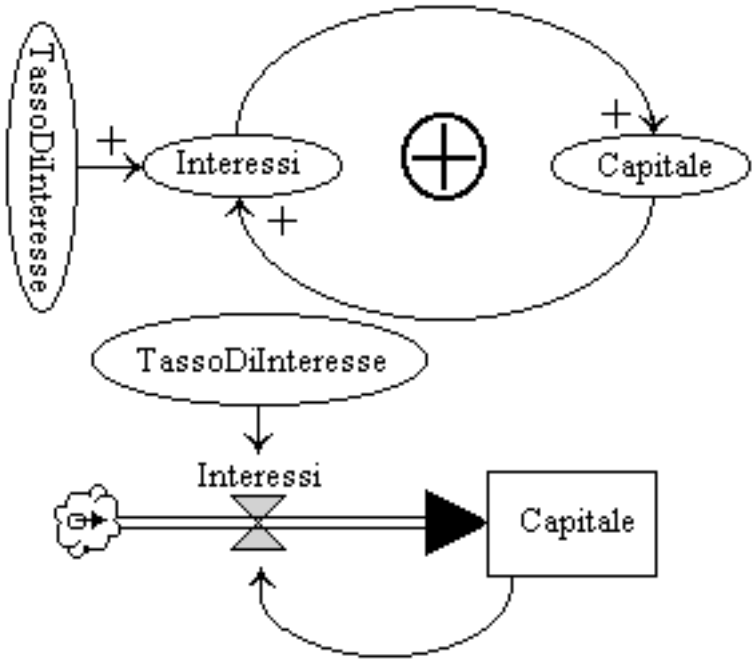

Figura 1.15: *Singolo anello, feedabck positivo*

Il legame fra le variabili "interne" è di proporzionalità diretta: tanto più alta è la quota di *Capitale* tanto maggiore, a parità di *TassoDiInteresse*, risulta il flusso di *Ineressi* che sua volta causa un incremento del *Capitale*. Si ha un *feedback* positivo in cui le due variabili coinvolte si incrementano a vicenda con un effetto detto *a valanga*. Oltre a tali legami si ha un legame di proporzionalità diretta fra le varaibili *Interessi* e *TassoDiInteresse*. I legami di proporzionalità diretta sono rappresentati da un segno "+" a fianco dell'arco orientato che rappresenta il legame. In questo caso il modello descrive il cosiddetto *ciclo virtuoso* ma può descrivere anche il ciclo con andamento complementare detto *ciclo vizioso*[30]: se il valore iniziale di *Capitale* è negativo (come accade se si modellizza un prestito e non un deposito) anche la variabile *Interessi* assume valori negativi e, mancando nel modello una qualche forma di deposito, la situazione debitoria diventa ben

[30]Se il valore iniziale di *Capitale* è nullo il sistema è in equilibrio *instabile* dal momento che un piccolo scostamento da tale valore da origine ad un comportamento divergente.

presto insostenibile.
Anche da questa analisi sommaria risulta evidente che:

1. la variabile *Capitale*, dal momento che descrive una quantità che si accumula, è un *livello*,

2. la variabile *Interessi* rappresenta un *flusso* dal momento che la si misura in [*valuta/tempo*],

3. la variabile esogena *TassoDiInteresse* la si misura come un numero puro moltiplicato per una frequenza, ad esempio 5% all'anno,

4. al legame fra la variabile *Interessi* e la variabile *Capitale* viene associato il tipo "materials" mentre a quello fra la variabile *Capitale* e la variabile *Interessi* viene associato il tipo "information".

Basandosi su tali considerazioni si può facilmente capire come trasformare il diagramma CL nel corrispondente diagramma FD (cfr. la figura 1.15 in cui compare un simbolo di "source" che modellizza la provenienza degli interessi come non specificata nel modello).
Una volta che sia stato tracciato il diagramma FD è possibile associare alle variabili le equazioni descrittive. In questo caso è necessario scrivere due equazioni, una associata alla variabile *livello* e una associata alla variabile *flusso*:

$$Capitale(n) = Capitale(n-1) + Interessi(n-1, n) \times T \tag{1.54}$$

$$Interessi(n, n+1) = Capitale(n) \times TassoDiInteresse \tag{1.55}$$

in cui T rappresenta l'ampiezza, in unità di tempo, dell'intervallo $[n-1, n]$ ovvero la distanza fra due "osservazioni" successive.
Per il calcolo di tali equazioni è necessario stabilire un valore per la variabile esogena *TassoDiInteresse* e un valore iniziale[31] per la variabile *livello* ovvero $Capitale(0)$. La conoscenza di $Capitale(0)$ ci consente di ricavare, dalla 1.55, il valore iniziale del flusso come:

$$Interessi(0, 1) = Capitale(0) \times TassoDiInteresse \tag{1.56}$$

La variabile esogena *TassoDiInteresse*, in questi casi, può essere descritta da espressioni semplici oppure complesse. Nel primo caso si può avere:

$$TassoDiInteresse = k \tag{1.57}$$

[31] Negli esempi, per semplicità, si farà l'assunzione che sia $n_{init} = 0$ e $n_{end} = N$.

mentre nel secondo caso si può arrivare ad avere espressioni condizionali del tipo:

$$
\begin{aligned}
&if(Capitale(n) \geq 1000)\\
&then\\
&TassoDiInteresse = k_1;\\
&else\\
&TassoDiInteresse = k_2;
\end{aligned}
$$

Dopo aver scritto le equazioni (ad esempio le 1.54 e 1.55) è necessario risolverle in modo da ottenere, nel caso in esame, due *array* di valori, rispettivamente, $Capitale[i]$ e $Interessi[i]$ in cui $i \in [0, N]$. Tali array di valori possono essere poi rappresentati sotto forma di tabelle oppure sotto forma di grafici su piani cartesiani (cfr. la sezione 2.3).
Un passo preliminare necessario per risolvere le equazioni 1.54 e 1.55 è quello di stabilire le relazioni esistenti fra tali equazioni. Il caso della figura 1.15 è un caso molto semplice e un esame delle equazioni 1.54 e 1.55 permette di evidenziare come la prima sia contenuta nella seconda in modo che fra le due esiste un ordinamento totale che consente, partendo dai valori iniziali, di valutare ciclicamente le due equazioni 1.54 e 1.55 per il numero di passi N prefissato.
Stabilito l'ordinamento e supponendo che

$$TassoDiInteresse = 10\% \tag{1.58}$$

$$Capitale(0) = 1000 \tag{1.59}$$

si può calcolare

$$Interessi(0,1) = 1000 \times 0,1 = 100 \tag{1.60}$$

in modo da ottenere, applicando ripetutamente le equazioni le 1.54 e 1.55, la tabella 1.1. La tabella 1.1 permette di apprezzare la rapidità della crescita di variabili all'interno di un anello con $feedback$ positivo.
È facile infatti vedere che il tempo di raddoppio del $Capitale$ si colloca grosso modo al settimo anno mentre al dodicesimo anno il *capitale* è più che triplicato (vale 3160.20) al quindicesimo quaduplicato (vale 4206.23) e così via.
La figura 1.16 presenta una situazione diversa da quella della figura 1.15: la figura 1.16 rapresenta, infatti, un tentativo di descrivere un parcheggio durante il periodo di tempo in cui questo si rimpie. Il parcheggio ha una capienza massima in numero di auto, N_{max}, ed è inizialmente, ovvero all'istante 0 della nostra osservazione, vuoto. Il flusso di auto (misurato in $[numero\_auto/tempo]$) riempie il parcheggio.
Si hanno le due variabili $Arrivi$ (un flusso) e $InPark$ (un livello) e i legami fra le due sono quelli indicati dal diagramma CL della figura 1.16: tanto più elevato è il flusso di auto tanto maggiore è il numero di auto nel parcheggio. Tale numero, a sua volta, tende a ridurre il flusso di auto in ingresso al parcheggio fino a che

Anno	Interessi	Capitale
0	-	1000
1	100	1100
2	110	1210
3	121	1331
4	133.1	1464.1
5	146.41	1610.51
6	161.05	1771.56
7	177.16	1948.72
8	194.87	2143.59
9	214.36	2357.95
10	235.79	2611.74

Tabella 1.1: Tabella esplicativa della relazione *Interessi/Capitale*

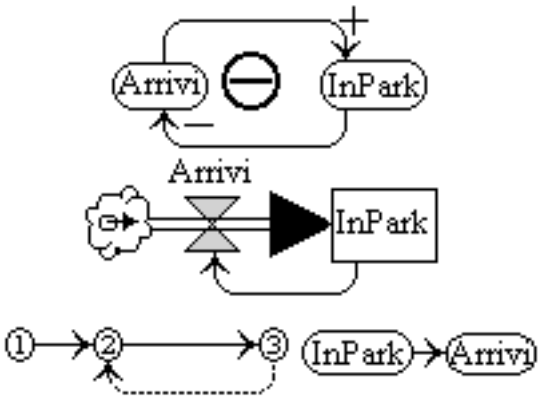

Figura 1.16: *Singolo anello, feedabck negativo*

questo non si annulla e il parcheggio si porta nella situazione di regime in cui tutti i posti sono occupati.
La figura 1.16 presenta anche il corrispondente diagramma FD in cui il simbolo di "source" modellizza la sorgente delle auto in arrivo al parcheggio. La figura presenta, in basso a sinistra, la struttura dei legami fra le variabili e soggiacente al diagramma FD, detta "multigrafo associato"[32]. Si fa notare che l'arco disegnato a tratto in tale diagramma individua un flusso di informazioni (non conservativo) mentre gli altri individuano flussi di materiali (ovvero flussi conservativi). Questa convenzione sarà mantenuta nelle successive figure della presente sezione. Nel

[32]In questo caso si ha un grafo vero e proprio. La figura 1.16 presenta, in basso a destra, anche il cosiddetto "grafo delle dipendenze" di cui si dirà a breve e che sarà utilizzato per definire i legami fra le equazioni di un diagramma FD.

multigrafo di figura 1.16 con il numero 1 si identifica la "source", con il numero 2 il "flusso" e con il numero 3 il "livello".
Dopo avere definito i diagrammi CL e FD è necessario scrivere le equazioni descrittive delle variabili del modello. In questo caso le equazioni hanno la forma seguente:

$$InPark(n) = InPark(n-1) + Arrivi(n-1, n) \times T \tag{1.61}$$

$$Arrivi(n, n+1) = \varphi(InPark(n)) \tag{1.62}$$

La forma della funzione $\varphi(InPark(n))$ nella equazione 1.62 dipende dal comportamento che si vuole modellizzare. Nella forma (1.62), l'equazione si limita a stabilire che il numero di auto in arrivo al parcheggio fra gli istanti n ed $n+1$ dipende dal numero di auto presenti nel parcheggio all'istante n.
In ogni caso, da un esame delle equazioni 1.61 e 1.62 si vede come il calcolo del *livello* all'istante n debba precedere il calcolo del $flusso$ fra gli istanti n e $n+1$. Questo spiega il *grafo delle dipendenze*, illustrato in basso a destra nella figura 1.16: all'istante 0 il valore di $InPark(0)$ è necessario per il calcolo del $flusso$ *Arrivi* fra gli istanti 0 e 1 e lo stesso vale $\forall n > 0$.
Una forma possibile per l'equazione 1.62 è la seguente:

$$Arrivi(n, n+1) = K \times e^{-\alpha InPark(n)} \tag{1.63}$$

Nella 1.63 la funzione $e^{-\alpha InPark(n)}$ stabilisce un legame di proporzionalità inversa fra il livello e il flusso mentre il parametro K deve avere una forma tale da garantire che se $InPark(n) = N_{max}$ (con N_{max} che indica la massima capienza del parcheggio) allora $Arrivi(n, n+1) = 0$. Una possibile soluzione è porre $K = N_{max} - InPark(n)$ da cui discende che a parcheggio vuoto il flusso assume valore massimo.

1.4.3 Caso 2: *Anelli multipli, feedabck positivo/feedback negativo*

Dopo aver esaminato con qualche dettaglio i modelli di sistemi caratterizzati da un solo anello con $feedback$ sia positivo (cfr. la figura 1.15) e negativo (cfr. la figura 1.16) in questa sezione vengono presentati alcuni esempi più complessi di modelli con due anelli.
Il primo esempio illustra il caso di un modello con due anelli entrambi con $feedback$ negativo; il secondo presenta un modello un pò più complesso con due anelli, uno con $feedback$ negativo e uno con $feedback$ negativo, mentre l'ultimo presenta un modello con due anelli entrambi con $feedback$ positivo.
In tutti i casi in cui si hanno più anelli nello stesso modello si devono tenere presenti i casi in cui uno degli anelli domina sugli altri imponendo al modello un particolare andamento delle variabili. Nel seguito verranno mostrati alcuni esempi di questo fenomeno.

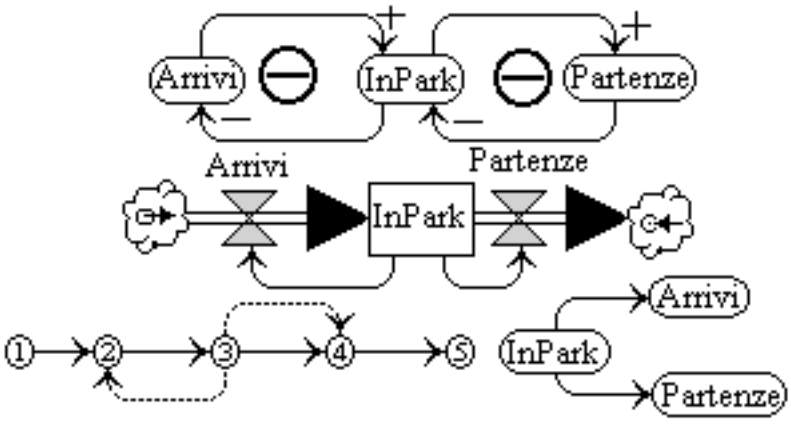

Figura 1.17: *Due anelli, feedabck negativo*

Il primo esempio è quello della figura 1.17 in cui si presenta una estensione dell'esempio della figura 1.16: in questo caso il modello descrive il comportamento dinamico di un parcheggio in cui le auto arrivano e partono. Il diagramma CL contiene due anelli con $feedback$ negativo. Le variabili coninvolte sono le seguenti:

1. Arrivi,
2. InPark,
3. Partenze

tutte evidenziate nella figura 1.17 in cui non compaiono le variabili esogene che invece sono presenti nelle equazioni descrittive del modello (cfr. oltre).
Il diagramma CL ha la seguente interpretazione: tanto maggiore è il flusso di *Arrivi* tanto più alto è il numero di auto nel parcheggio (conteggiato dalla variabile $InPark$) e tanto più questo numero è elevato tanto minore è il flusso degli *Arrivi* (primo anello); tanto più alto è il numero di auto nel parcheggio tanto maggiore è il flusso delle *Partenze* e tanto più questo è elevato e tanto minore è il numero di auto nel parcheggio (secondo anello).
Dalla discussione precedente si evince facilmente che:

1. la variablile $InPark$ è un livello,
2. le variabili *Arrivi* e *Partenze* sono flussi

per cui è facile ricavare il diagramma FD della figura 1.17 e il multigrafo associato. Il diagramma FD contiene anche la "sorgente" e il "pozzo" delle auto che modellizzano i flussi delle auto in ingresso e in uscita dal modello, in pratica il "mondo esterno".
I flussi di materiali sono due: quello fra i nodi 1 (*source*) e 3 (*livello*) tende a

riempire la variabile *livello* mentre quello fra i nodi 3 e 5 (*sink*) tende a svuotarla. Oltre a questi si hanno due flussi di informazioni che legano la variabile livello alle variabili flusso.
Le equazioni descrittive del modello che ne condizionano il comportamento che, in questo caso, può essere molto complesso sono le seguenti:

$$InPark(n) = InPark(n-1) + Arrivi(n-1, n) \times T - Partenze(n-1, n) \times T \quad (1.64)$$

$$Arrivi(n, n+1) = K \times e^{-\alpha InPark(n)} \quad (1.65)$$

$$Partenze(n, n+1) = InPark(n) \times TD \quad (1.66)$$

L'equazione 1.66 contiene una variabile esogena TD (tasso di deflusso) che non dipende dalla situazione del parcheggio ma dalle abitudini degli utenti abituali. Una ulteriore variabile esogena (TA, tasso di arrivo) può essere fatta comparire anche nel parametro k della 1.65 in modo da tenere conto di carateristiche del flusso in ingresso dipendente sempre dalle abitudini degli utenti del parcheggio modellizzato.
La figura 1.17 contiene anche il grafo delle dipendenze che stabilisce l'ordine in cui devono essere valutate le equazioni del modello in funzione dei legami reciproci. È evidente come la valutazione della 1.64, a partire da un valore iniziale $InPark(0)$, debba precedere la valutazione delle 1.65 e 1.66.
Il prossimo esempio è quello presentato in figura 1.18 che descrive, in modo semplificato, la dinamica, all'interno di una popolazione, dei legami fra il tasso di natalità (*Nascite*), il numero di bambini (*Bambini*), il tasso di maturazione (*Maturazione*) da bambini ad adulti (intesi come individui in grado di riprodursi) e la relazione fra in numero di adulti (*Adulti*) e il tasso di natalità. Nel modello è stata introdotta anche una variabile che rappresenta il tasso di mortalità degli adulti (*Morti*) e sono indicati i legami di tale variabile con la variabile che rappresenta il numero degli adulti. Il modello è molto semplificato in quanto considera solo il *tasso di mortalità* degli adulti e trascura quello dei bambini per cui tutti i bambini riescono a diventare adulti.
La figura illustra il diagramma CL, di facile lettura, il corrispondente diagramma FD con il multigrafo associato. Il diagramma CL presenta tre anelli, due con *feedback* negativo (uno in cui il tasso di maturazione tende a tenere stabile il numero dei bambini e l'altro in cui il tasso di mortalità tende a tenere stabile il numero degli adulti) e uno con *feedback* positivo,in cui la quantità di adulti tende ad far crescere il tasso di natalità.
Anche da un esame superficiale del diagramma CL è facile capire che si hanno:

1. tre variabili flusso (*Nascite*, *Maturazione* e *Morti*),
2. due variabili livello, *Bambini* e *Adulti*,
3. tre variabili esogene che diremo rispettivamente *tasso di natalità* (TN), *tasso di maturazione sessuale* (TMS) e *tasso di mortalità* (TM), di ovvio significato.

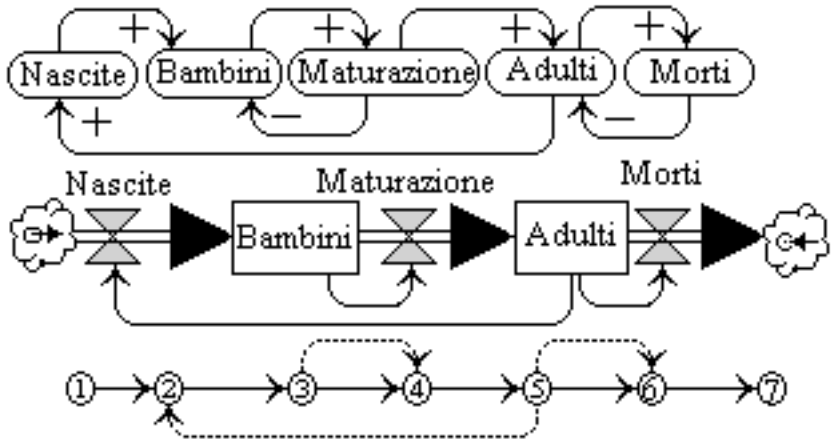

Figura 1.18: *Tre anelli: due con feedback negativo, uno con feedback positivo*

Le equazioni descrittive del modello sono le seguenti:

$$Bambini(n) = Bambini(n-1) + Nascite(n-1, n) \times T - Maturazione(n-1, n) \times T \tag{1.67}$$

$$Adulti(n) = Adulti(n-1) + Maturazione(n-1, n) \times T - Morti(n-1, n) \times T \tag{1.68}$$

$$Nascite(n, n+1) = Adulti(n) \times TN \tag{1.69}$$

$$Maturazione(n, n+1) = Bambini(n) \times TMS \tag{1.70}$$

$$Morti(n, n+1) = Adulti(n) \times TM \tag{1.71}$$

La conoscenza delle condizioni iniziali dei livelli, $Adulti(0)$ e $Bambini(0)$, permette il calcolo dei valori iniziali dei flussi:

$$Morti(0, 1) = Adulti(0) \times TM \tag{1.72}$$

$$Maturazione(0, 1) = Bambini(0) \times TN \tag{1.73}$$

$$Nascite(0, 1) = Adulti(0) \times TMS \tag{1.74}$$

in base ai quali si innesca il calcolo iterativo delle equazioni 1.67 ... 1.71, per il numero di passi desiderato. Dato che ognuno dei flussi dipende da un solo livello, il grafo delle dipendenze, non riportato in figura 1.18 è, in questo caso, una foresta di due componenti connesse. Nella prima, la valutazione della 1.67 precede ad ogni passo quella della 1.70 mentre, nella seconda, la valutazione della 1.68 precede ad ogni passo quella della 1.69 e della 1.71.
L'ultimo esempio di questa sezione è quello di un sistema il cui diagramma CL contiene due anelli entrambi con *feedback* positivo ed è illustrato in figura 1.19.
Il diagramma CL della figura 1.19 modellizza un livello L che ha due cause di

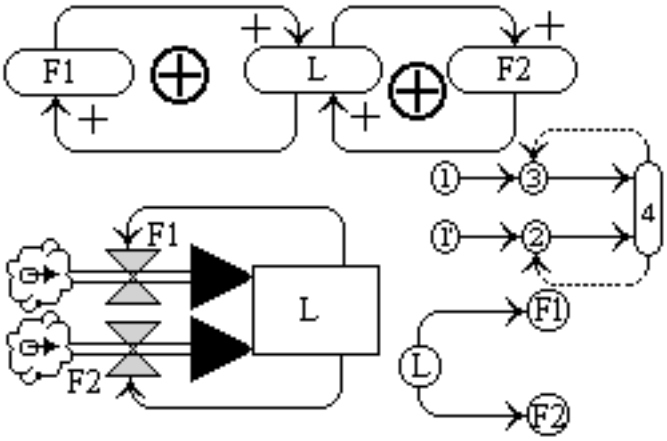

Figura 1.19: *Due anelli, feedabck positivo*

accrescimento indipendenti, regolate dai due flussi $F1$ e $F2$ che dipendono in qualche modo dal valore della variabile L.
I legami fra le variabili sono tutti di proporzionalità diretta ed è agevole ricavare sia il diagramma FD sia il multigrafo associato (anche in questo caso si tratta di un grafo) e il grafo delle dipendenze, tutti presenti in figura 1.19.
Le equazioni, infine, sono le seguenti:

$$L(n) = L(n-1) + F1(n-1,n) \times T + F2(n-1,n) \times T \tag{1.75}$$

$$F1(n,n+1) = K1 \times L(n) \tag{1.76}$$

$$F2(n,n+1) = K1 \times L(n) \tag{1.77}$$

in cui $K1$ e $K2$ sono, in generale, due variabili esogene non rappresentate nella figura 1.19.

1.4.4 Caso 3: *Anelli con più di due elementi per anello, anelli sovrapposti, feedabck positivo/feedback negativo*

Gli esempi esaminati nelle sezioni 1.4.2 e 1.4.3 sono caratterizzati da diagrammi CL con un singolo anello con *feedback* positivo o negativo oppure da al più due anelli. In questi casi, fissato un nodo di partenza, percorrendo l'anello a partire da tale nodo fino a tornarvi (e trascurando le variabili esogene) si attraversano elementi che sono alternativamente un flusso e un livello o viceversa. Da questa caratteristica discendono la relativa facilità con cui si sono ricavati i diagrammi FD e si sono scritte, caso per caso, le equazioni descrittive.
Nei diagrammi visti compaiono, infatti, solamente *flussi*, *livelli* e *variabili esogene* e le relazioni fra questi sono molto semplici:

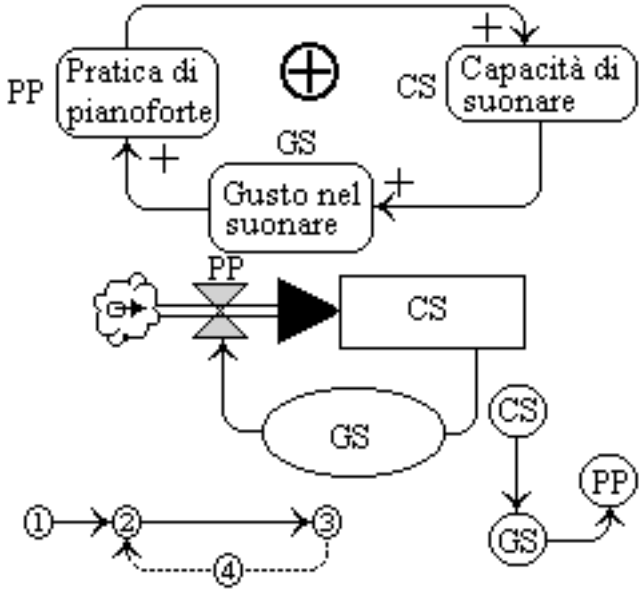

Figura 1.20: *Anello singolo con tre elementi, feedback positivo*

1. le variabili esogene influenzano solo i flussi con legami di proporzionalità diretta o inversa,
2. i flussi contribuiscono a riempire i livelli (flussi in ingresso) o a svuotarli (flussi in uscita),
3. i livelli influenzano uno o più flussi con relazioni di proporzionalità diretta o inversa.

Nella presente sezione vedremo alcuni casi in cui la stesura del diagramma CL non presenta problemi mentre la sua traduzione in diagramma FD non è possibile se non si introduce un altro tipo di variabili che può giocare il ruolo di livello o di flusso, a seconda dei casi, e che diremo *variabili ausiliarie*.
La figura 1.20 contiene un diagramma CL che modellizza il mecanismo di apprendimento del pianoforte e contiene le tre variabili:

1. Pratica di Pianoforte, nel seguito PP
2. Capacità di suonare, nel seguito CS,
3. Gusto nel suonare, nel seguito GS.

Il diagramma CL si può leggere come segue, partendo da PP: quanto maggiore è PP tanto maggiore è CS che a sua volta causa un incremento di pari segno di GS che causa, infine, un aumento di PP. Simmetricamente una diminuzione di PP causa una diminuzione di CS che a sua volta provoca una riduzione di

GS con, infine, una nuova riduzione di PP: si ha in questo caso un anello con $feedback$ positivo in cui compaiono tre elementi collegati fra loro come illustrato in figura 1.20.
Se fossimo vincolati ad usare solo flussi, livelli e variabili esogene ci troveremmo in difficoltà nel definire il corrispondente diagramma FD dato che è facile capire come PP sia un $flusso$ e CS sia un $livello$ mentre non si riesce ad attribuire nessuno dei tre tipi suddetti alla variabile GS in modo da catturarne la natura e consentirne l'inserimento nel diagramma FD.
L'unica soluzione è definire GS come una *variabile ausiliaria* che, in questo caso almeno, gioca il ruolo di un livello (in effetti il "Gusto di suonare" ha le caratteristiche di un livello).
L'uso di tale tipo di variabile (rappresentato come un ovale nei diagrammi FD) ci permette di ricavare il diagramma FD cui sono associati il multigrafo (in questo caso un grafo tout court) e il grafo delle dipendenze, tutti riportati in figura 1.20.
Le equazioni associate al diagramma FD sono le seguenti:

$$CS(n) = CS(n-1) + K \times PP(n-1, n) \times T \tag{1.78}$$

$$GS(n) = K_1 \times CS(n) \tag{1.79}$$

$$PP(n, n+1) = K_2 \times GS(n) \tag{1.80}$$

Da tali equazioni, e da altre che saranno introdotto nel seguito della sezione, risulta evidente il ruolo giocato dalla variabile ausiliaria GS: il suo ruolo è quello di elemento di arricchimento del modello, che potrebbe essere definito anche senza tale variabile perdendo però in espressività. È facile vedere come sia possibile eliminare completamente la variabile ausiliaria GS osservando che le 1.78 e 1.79 sono calcolate per lo stesso valore di n per cui si può sostituire la 1.79 nella 1.80 ottenendo due equazioni in cui non compare la variabile ausiliaria. Un altro esempio è riportato in figura 1.21 in cui compaiono tre variabili (*Livello di inquinamento*, nel seguito LI, *Quantità di controlli*, nel seguito QC, e $Preoccupazione$, nel seguito P) messe in relazione fra di loro da legami di proporzionalità diretta e inversa in modo da creare un anello con $feedback$ negativo: scopo del modello è quello di spiegare come il livello di inquinamento sia tenuto costante (sia pure con alcune oscillazioni) a seguito di controlli sollecitati dalla preoccupazione delle persone per l'inquinamento stesso.
Da una analisi del diagramma CL e della sua descrizione si può dedurre che:

1. la variabile LI è un livello,
2. la variabile QI è un flusso,
3. la variabile P è di tipo ausiliario ed ha un comportamento simile ad un livello,
4. fra le variabili LI e P e fra P e QC la relazione è di proporzionalità diretta,

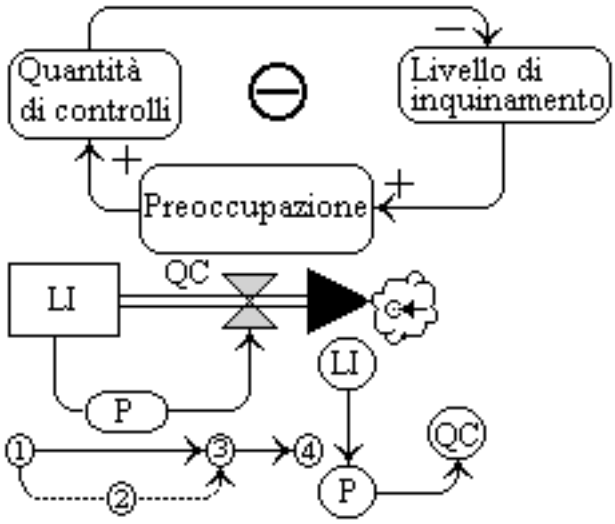

Figura 1.21: *Anello singolo con tre elementi, feedback negativo*

5. fra le variabili QC e LI si ha una relazione di proporzionalità inversa,
6. il diagramma CL è caratterizzato da un *feedback* negativo.

La figura 1.21 contiene il diagramma FD, in cui la variabile ausiliaria è rappresentata racchiusa in un ovale, il [*multi*]grafo associato e il grafo delle dipendenze utilizzabile per il calcolo delle equazioni seguenti (di nuovo K, K_1 e K_2 sono variabili esogene non rappresentate in figura 1.21):

$$LI(n) = LI(n-1) - K \times QC(n-1,n) \times T \tag{1.81}$$

$$P(n) = K_1 \times LI(n) \tag{1.82}$$

$$QC(n,n+1) = K_2 \times P(n) \tag{1.83}$$

Le variabili esogene hanno, in questo come in tutti gli altri esempi, due funzioni:

1. una "dimensionale",
2. una "comportamentale".

La prima funzione viene svolta dalle variabili esogene grazie al fatto che sono caratterizzate dalle opportune dimensioni fisiche in modo da rendere dimensionalmente consistenti le varie equazioni.
La seconda modellizza il legame esistente fra un flusso e alcune possibili cause di variazione del flusso indipendenti dalla variazione del livello ad esso associato. Nel caso dell'equazione 1.82, ad esempio, la variabile esogena K_1 potrebbe essere una misura di fattori che influenzano la variabile P *a parità di valore di* LI quali:

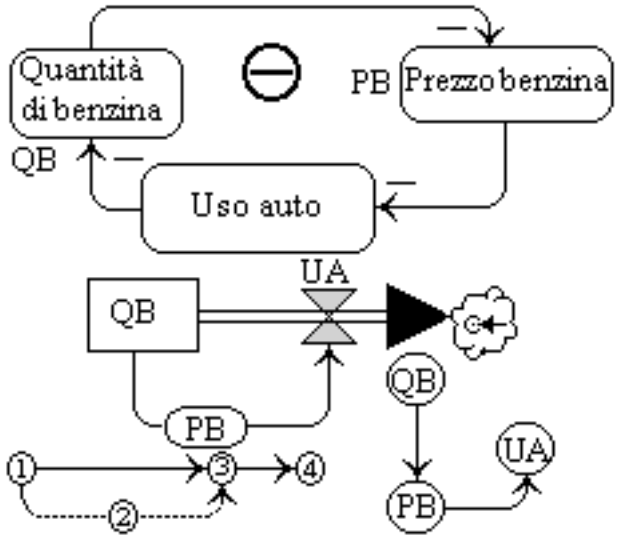

Figura 1.22: *Anello singolo con tre elementi, feedback negativo*

periodo dell'anno, situazione economica, situazione climatica e altre. Tali legami fra le variabili esogene e le variabili flusso, se riportati nei diagrammi CL e nei diagrammi FD, devono essere modellizzati come si è visto, ad esempio, nel caso della figura 1.15.
La figura 1.22 rappresenta una lieve variante strutturale della figura 1.21. Nella figura 1.22 compaiono:

1. il flusso UA,
2. il livello QB,
3. la variabile ausiliaria PB che può essere vista come la causa indiretta dell'uso (o del non uso) dell'auto e che gioca un ruolo simile ad un livello.

La figura contiene il diagramma CL, il corrispondente diagramma FD e il grafo delle dipendenze. Le equazioni descrittive possono assumere le seguenti forme (K, K_1, K_2 e K_3 sono variabili esogene):

$$QB(n) = QB(n-1) - K \times UA(n-1, n) \times T \tag{1.84}$$

$$PB(n) = K_1 \times e^{-\alpha \times QB(n)} \tag{1.85}$$

$$UA(n, n+1) = \frac{K_2}{PB(n) + K_3} \tag{1.86}$$

La variabile K_1 tiene conto di variazioni di prezzo a parità di disponibilità di benzina (ovvero in assenza di variazioni del livello QB) mentre le variabili K_2 e K_3 definiscono l'uso dell'auto (cioè il valore del flusso UA) anche nel caso che il

prezzo della benzina tenda PB (per assurdo) a zero.
L'esempio della figura 1.23 mostra, infine, un diagramma CL con due anelli interdipendenti, uno con $feedback$ negativo e uno con $feedback$ positivo.
Le variabili in gioco, indicate per semplicità con acronimi nella figura 1.23, sono le seguenti:

1. *Bisogno di Nuove Autostrade* o BNA, variabile ausiliaria;
2. *Autostrade Costruite* o AC, flusso;
3. *Numero di Autostrade* o NA, livello;
4. *Attrattiva della Guida in Autostrada* o AGA, variabile ausiliaria;
5. *Numero di Ingorghi* o NI, variabile ausiliaria.

I due anelli possono essere interpretati come segue: un aumento di BNA causa un aumento di AC che a sua volta causa un incremento a catena di NA, di AGA e di NI. Parallelamente l'incremento di AC si traduce nella diminuzione di NI. Si hanno in questo caso due effetti contrastanti su una stessa variabile che agisce sulle sue proprie cause mediante due anelli di $feedback$, uno positivo e uno negativo.
La figura 1.23 contiene il diagramma FD e il [multi]grafo associati. Il grafo delle dipendenze può essere agevolmente ricavato osservando le equazioni descrittive del modello che si riportano qui di seguito.

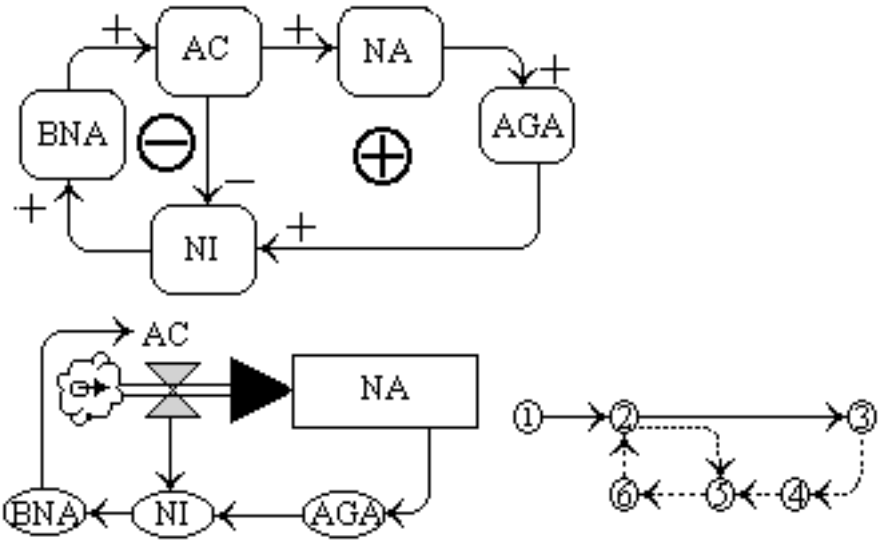

Figura 1.23: *Anelli sovrapposti, feedback positivo e negativo*

$$NA(n) = NA(n-1) + K_0 \times AC(n-1, n) \times T \tag{1.87}$$

$$AGA(n) = K_1 \times NA(n) \tag{1.88}$$

$$NI(n) = K_2 \times \frac{\alpha \times AGA(n)}{\beta \times AC(n-1,n)} \tag{1.89}$$

$$BNA(n) = K_3 \times NI(n) \tag{1.90}$$

$$AC(n,n+1) = K_4 \times BNA(n) \tag{1.91}$$

Nelle equazioni 1.87 ... 1.91 compaiono le variabili esogene K_i $i \in \{0,1,2,3,4\}$ e le costanti α e β. L'equazione 1.87 rappresenta l'andamento tipico di un livello (NA) con un solo flusso in ingresso (AC). Le 1.88, 1.89 e 1.90 descrivono gli andamenti di variabili ausiliarie, valutate tutte allo stesso istante n, in funzione del livello (la 1.88), del flusso e di un'altra variabile ausiliaria (la 1.89) o di solo una variabile ausiliaria (la 1.90). La 1.91, infine, definisce l'andamento del flusso AC in funzione di una variabile ausiliaria (BNA).

1.5 I metodi di risoluzione

Dagli esempi presentati nella sezione 1.4 e nella sezione 1.3 risulta evidente come la simulazione di un modello richieda la risoluzione di un certo numero di equazioni associate a *livelli*, a *flussi* ed, eventualmente, a *variabili ausiliarie*. La soluzione delle equazioni relative ai flussi e alle variabili ausiliarie richiede, di solito, l'esecuzione di semplici operazioni algebriche oltre alla valutazione di una o più funzioni mentre la soluzione delle equazioni relative ai livelli richiede l'esecuzione di operazioni di integrazione ([Kir98], [PTVF92]).
Il metodo più semplice che può essere usato per l'esecuzione di tali operazioni di integrazione è il *metodo di Eulero* mentre metodi di integrazione più sofisticati sono i metodi di $Runge - Kutta$.
Sia il metodo di Eulero sia i metodi di Runge-Kutta ([PTVF92]) rappresentano metodi numerici per la risoluzione di equazioni differenziali ordinarie scritte come equazioni alle differenze finite. Il loro ambito di applicazione è quello della risoluzione di problemi di valore iniziale. Per probelma di valore iniziale si intende un problema in cui, a partire dal valore di una variabile y_i in un istante iniziale x_0, si determina il valore della variabile o in un punto finale x_n o in una successione di punti x_i con $i \in [0, \ldots, n]$.

1.5.1 Il metodo di Eulero

Il *metodo di Eulero* ([Kir98]), come viene di solito implementato nei package commerciali per la Simulazione di sistemi Dinamici, consiste nella esecuzione dei seguenti passi:

1. si fissa il parametro tempo "t" al valore iniziale, di solito 0,

2. si inizializzano i livelli su cui si vuole eseguire il calcolo al loro valore iniziale,

3. si calcola la velocità di variazione di ciascun livello per il valore corrente di "t" calcolando i valori dei flussi in ingresso al livello e in uscita dal livello all'istante "t",

4. si suppone che tale velocità di variazione sia costante nell'intervallo di tempo $[t, t+T]$, dove T rappresenta l'intervallo di tempo fra due valutazioni successive delle equazioni del modello, e si calcola il valore di ogni livello all'istante $t+T$ utilizzando la seguente equazione

$$Livello(t+T) = Livello(t) + T \times flusso(t) \tag{1.92}$$

   in cui $flusso(t)$ rappresenta la velocità di variazione calcolata al passo precedente,

5. si pone t=t+T;

6. si ripetono i passi 3,4 e 5 fino a che non si arriva all'istante di fine della simulazione $n \times T$ dopo n passi di simulazione.

Si fa notare che il passo 4 viene ripetuto per tutti i livelli presenti nel modello e che la computazione termina dopo un numero finito di passi fissato al momento della creazione del modello.
Il valore scelto per la variabile T, che assume il nome di $TIME\,STEP$, influenza l'accuratezza del metodo di Eulero dal momento che il metodo presuppone la costanza dei flussi in ingresso e in uscita da un livello su intervalli di ampiezza T. Una regola pratica per la scelta del valore del $TIME\,STEP$ è quella di prendere come valore un valore inferiore ad un terzo della più piccola costante del modello che definisce un valore di tempo. Un modo per verificare se il valore scelto è corretto è quello di dimezzarlo e di rieseguire la simulazione con il nuovo valore. Se i risultati delle due simulazioni (quella per T e quella per $T/2$) non mostrano variazioni significative allora il valore scelto per T è adeguato. Nonostante si adottino tali cautele nella scelta del valore di T il metodo di Eulero può dare luogo a risultati inaccurati nel caso un processo presenti oscillazioni significative. Metodi di integrazione più accurati sono i metodi di Runge-Kutta che si basano sull'idea di valutare la velocità di variazione dei flussi in ingresso e in uscita da un livello nel tempo e usare tali informazioni per migliorare il calcolo del valore di un livello all'isante $t+T$ sulla base del valore all'istante t. Il pregio di tali metodi è quello di riuscire ad ottenere una elevata accuratezza senza il sovraccarico computazionale che si ha nel caso si usi il metodo di Eulero con valori molto piccoli della variabile T.

1.5.2 I metodi di Runge-Kutta

La formula che realizza il metodo di Eulero può essere scritta nella forma seguente ([PTVF92]):

$$y_{n+1} = y_n \ + \ h\, f(x_n, y_n) \tag{1.93}$$

in cui h rappresenta il passo di incremento della variabile indipendente (ovvero si ha $x_{n+1} = x_n + h$) e $f(x_n, y_n)$ rappresenta il valore della derivata nel punto iniziale dell'intervallo e cioè in (x_n, y_n). La derivata nel punto iniziale di ogni intervallo viene estrapolata in modo da definire il valore nel punto finale dell'intervallo ovvero (x_{n+1}, y_{n+1}). Il valore h rappresenta il quanto di incremento della variabile indipendente x e coincide con la variabile *TIME STEP* introdotta nella sezione 1.5.1 nei casi in cui la variabile indipendente sia la variabile tempo t.
La formula 1.92 può, nonostante i suoi difetti, essere usata come base per lo sviluppo di metodi più accurati e più stabili, quali i metodi di Runge-Kutta del secondo e del quarto ordine.
La 1.92 può, infatti, essere usata, in un metodo del secondo ordine[33] come un modo per valutare un valore approssimato nel punto di mezzo dell'intervallo (ovvero in $x_n + \frac{h}{2}$) in base al quale valutare il valore della variabile dipendente y nel punto x_{n+1}.
Le equazioni corrispondenti sono le seguenti ([PTVF92]):

$$k_1 = \ h\, f(x_n, y_n) \tag{1.94}$$

$$k_2 = \ h\, f(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \tag{1.95}$$

$$y_{n+1} = y_n \ + \ k_2 \tag{1.96}$$

e definiscono un metodo di Runge-Kutta del secondo ordine (con un errore $O(h^3)$) detto anche del punto di mezzo. In tali equazioni k_1 rappresenta la derivata nel punto iniziale x_n di ciascun intervallo di ampiezza h e k_2 rappresenta il valore della derivata nel punto di mezzo di tale intervallo ovvero in $x_n + \frac{h}{2}$.
Per ottenere un metodo con una accuratezza maggiore[34] si può ricorrere a valutazioni ulteriori e ripetute della derivata ovvero del termine $f(x, y)$. Nel caso del metodo di Runge-Kutta del quarto ordine (in cui l'errore è $O(h^5)$) si hanno quattro valutazioni della derivata: una nel punto iniziale x_n, due nel punto di mezzo $x_n + \frac{h}{2}$ e una nel punto finale $x_n + h$.
Le equazioni che descrivono il metodo con le quattro valutazioni della derivata sono le seguenti:

$$k_1 = \ h\, f(x_n, y_n) \tag{1.97}$$

[33] Un metodo è detto essere di ordine n-esimo se il termine che individua l'errore in una espansione in serie di potenze è $O(h^{n+1})$.

[34] Si trascurano in questa sede considerazioni in merito al legame esistente fra l'ordine di un metodo e la sua accuratezza per le quali si rimanda a [PTVF92].

$$k_2 = h\, f(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \tag{1.98}$$

$$k_3 = h\, f(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \tag{1.99}$$

$$k_4 = h\, f(x_n + h, y_n + k_3) \tag{1.100}$$

$$y_{n+1} = y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \tag{1.101}$$

Dai quattro valori della derivata nei tre punti suddetti (e cioè k_1, k_2, k_3 e k_4) viene calcolato il valore y_{n+1} nel punto x_{n+1}.
I metodi di Runge-Kutta qui illustrati ([PTVF92]), come il metodo di Eulero, considerano ogni passo in una sequenza di passi allo stesso modo, nel senso che non usano i valori precedenti di una soluzione per il calcolo dei valori successivi. La motivazione di questo modo di procedere risiede nel fatto che ogni punto lungo una traiettoria descritta da una equazione differenziale ordinaria può essere preso come punto iniziale per la sua risoluzione. Da un punto di vista algoritmico ciò si traduce nell'uso di un valore costante per il passo h.
L'uso di un valore costante per il passo h è indicato nel caso in cui sia necessario ottenere i valori di una funzione in punti equidistanti senza una accuratezza particolarmente elevata e cioè, ad esempio, in tutti i casi in cui si deve produrre il grafico di una funzione soluzione di una equazione differenziale. In tali casi è sufficiente applicare il metodo scelto a partire da un valore iniziale x_0 ad un valore finale $x_n = x_0 + n \times h$.
Nel caso sia necessario ottenere soluzioni con una accuratezza molto elevata oppure in tutti i casi in cui l'uso di un valore costante per il passo h dá luogo a soluzioni inaccurate si può ricorrere a varianti dell'algoritmo che prevedono un controllo adattivo del valore di h.
In tali casi si agisce sul valore di h, partendo da un valore iniziale e riducendo tale valore fino a che l'accuratezza con cui si ottiene la soluzione non è quella desiderata con il minimo sforzo computazionale. Per ulteriori dettagli su queste tecniche si rimanda a [PTVF92].
I metodi di Runge-Kutta, come il metodo di Eulero di cui rappresentano un raffinamento, sono metodi iterativi per cui la loro applicazione alla valutazione dell'andamento di una variabile di tipo *livello* richiede l'esecuzione ripetuta di un certo numero di passi simili a quelli illustrati nella sezione 1.5.1: l'unica variante è che, in questi casi, la valutazione di cui al passo (4) invece che una equazione come la 1.92 richiede la soluzione di equazioni che hanno la forma delle 1.94 ... 1.96 (metodo del secondo ordine) oppure delle 1.97 ... 1.102 (metodo del quarto ordine).

1.6 Considerazioni finali

Il presente Capitolo contiene una introduzione, inevitabilmente incompleta, a discipline vaste e complesse quali la *Teoria dei Sistemi*, la *System Dynamics* e la *Computer Simulation*. In questa sezione verranno esaminati brevemente e in modo molto parziale alcuni degli argomenti cui si è solo accennato nelle sezioni precedenti del Capitolo.

1.6.1 I ritardi

I *ritardi* sono inevitabilmente presenti in tutti i sistemi fisici e in tutti i processi gestionali ovvero in tutti i modelli descritti con i metodi della *System Dynamics*. Le loro cause sono essenzialmente le inerzie dei sistemi che impiegano tempo a reagire a delle sollecitazioni ma anche caratteristiche proprie dei sistemi modellizzati.
All'interno di un sistema (e del modello corrispondente), i ritardi possono essere presenti sia sui *flussi di materiali* (ritardi di materiali) sia sui *flussi di informazioni* (ritardi di informazioni). I ritardi del primo tipo sono relativi al tempo necessario per la elaborazione di grandezze fisiche mentre i ritardi del secondo tipo dipendono dal tempo che intercorre fra la conoscenza di una informazione e il suo utilizzo pratico. Nel caso dei flussi di materiali il tipo più semplice di ritardo è detto di tipo *pipeline*. In questo caso il materiale entra in una estremità del ritardo e fuoriesce dall'altra dopo un predefinito periodo di tempo come se fluisse lungo una pipeline. Un ritardo di tale tipo può essere presente fra due livelli e caratterizza il ritardo con cui si ha il trasferimento di materiale da un livello ad un altro causato, ad esempio, dai tempi finiti di trasporto da una località ad un'altra. Per la modellizzazione si usano funzioni che rispettano la caratteristica del flusso di materiali di essere conservativi e che permettono di specificare l'entità del ritardo (che può essere pilotata da una variabile esogena) e la grandezza che deve essere influenzata dal ritardo.
In molti casi il ritardo di tipo pipeline (caratterizzato da un ritardo costante) rappresenta un modo impreciso di descrivere il ritardo presente in un flusso conservativo che può essere caratterizzato da una variazione del tempo di ritardo per i singoli elementi del flusso. In tali casi si fa uso di ritardi di tipo esponenziale, di solito del terzo ordine, che permettono di modellizzare situazioni in cui si ha una variazione della velocità del flusso attraverso il ritardo.
Nel caso di un ritardo di tipo pipeline se in ingresso si ha una funzione gradino unitario

$$u(t) = 0 \, per \, t \leq 0 \, u(t) = 1 \, per \, t > 0 \tag{1.102}$$

in uscita si ha la stessa funzione traslata di un tempo Δ pari al ritardo introdotto dalla pipeline

$$u(t - \Delta) = 0 \, per \, t \leq \Delta \, u(t) = 1 \, per \, t > \Delta \tag{1.103}$$

Nel caso di un ritardo di tipo esponenziale si ha che in presenza di una funzione del tipo (1.102) l'uscita dal ritardo raggiunge il valore di regime in modo graduale. La figura 1.24 illustra un ritardo di tipo pipeline (in alto) e un ritardo di tipo esponenziale (in basso).

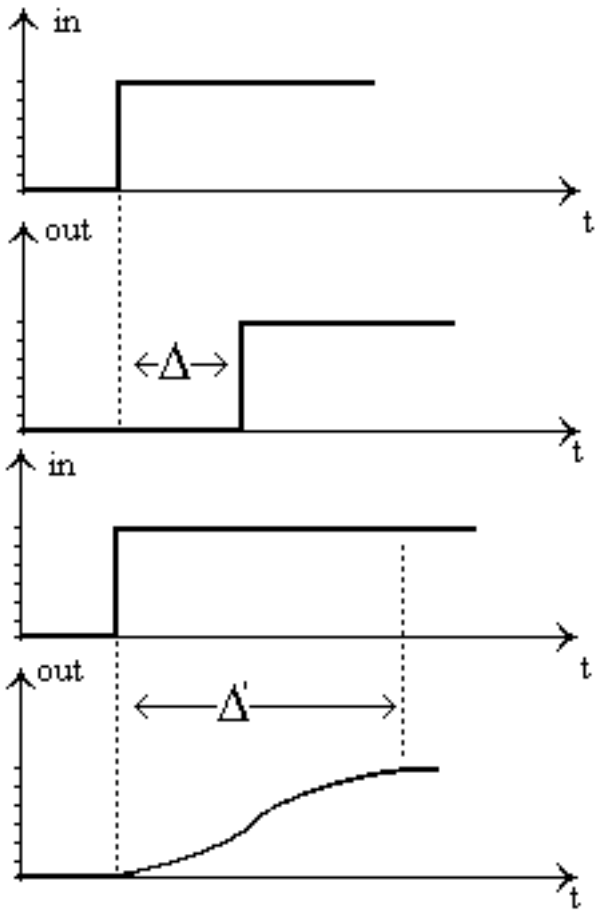

Figura 1.24: *Ritardo pipeline e ritardo esponenziale*

La figura 1.24 illustra, in alto, un segnale di tipo gradino unitario in ingresso (*in*) ad un ritardo di tipo *pipeline* e il corrispondente segnale in uscita (*out*) che riproduce l'andamento del segnale in ingresso con un ritardo Δ. La stessa figura illustra, in basso, un ritardo di tipo *esponenziale* in cui il segnale di uscita raggiunge il valore a regime del segnale in ingresso in modo graduale e con un ritardo pari a Δ'.
I ritardi possono essere presenti anche sui *flussi di informazioni* oltre che sui flussi di materiali dato che anche la trasmissione di informazioni può essere ritardata. Per la modellizzazione di tali ritardi si possono usare le funzioni prima introdotte in

cui però non si ha il vincolo della conservatività per cui alcune delle informazioni ritardate possono essere dimenticate se il ritardo varia nel tempo ([Kir98]).

1.6.2 Le non linearità

Un processo è detto avere un comportamento lineare se per esso vale il principio di sovrapposizione degli effetti altrimenti è detto essere *non lineare*. Le *non linearità* sono presenti nella maggior parte dei sistemi fisici per i quali un'analisi lineare è possibile solo ricorrendo a delle approssimazioni.
La trattazione delle non linerità, data la vastità e la complessità dell'argomento, esula dagli scopi della presente Tesi. In questa sede ci si limita, pertanto, ad illustrare, in modo non esaustivo, alcuni metodi molto semplici che permettono di introdurre le non linearità nei modelli utilizzati per la simulazione di Sistemi Dinamici.
Volendo modellizzare sistemi caratterizzati dalla presenza di *non linearità* utilizzando gli strumenti della *System Dyamics* è necessario introdurre una notazione che consenta la definizione di legami non lineari fra le variabili del modello.
Gli strumenti di solito disponibili nei package commerciali sono i seguenti:

1. le istruzioni "IF THEN ELSE",
2. le funzioni definite come tabelle,
3. le funzioni definite come grafici.

Tali strumenti permettono di modellizzare legami di tipo non lineare fra un *livello* e un *flusso* in esso incidente (ovvero un flusso in ingresso o un flusso in uscita). Nel caso (1) si usa l'istruzione condizionale per definire, imponendo ua condizione sul valore del livello, due andamenti diversi di un flusso. Ad esempio, riprendendo il caso della figura 1.6, si può voler descrivere una situazione in cui se il "capitale accumulato" ($Capitale$) è inferiore ad un certo valore ($ValoreSoglia$) si applica un certo tasso di interesse ($tassoB$) mentre se è superiore si applica un altro tasso di interesse ($tassoA$). In questo caso si può usare un legame fra il livello ($Capitale$) e il flusso *interesse* di tipo non lineare descritto dalla seguente istruzione:

```
IF (Capitale < ValoreSoglia) THEN
  interesse= Capitale*tassoB
ELSE
  interesse= tassoB*ValoreSoglia+ (Capitale-ValoreSoglia)*tassoA
```

In questo caso si ottiene una risposta non lineare definita su due intervalli di valori della variabile $Capitale$. Utilizzando istruzioni "IF THEN ELSE" annidate si possono descrivere situazioni più complesse.
Nei casi (2) e (3), invece, si ricorre ad una tabella oppure ad un grafico per

definire una relazione fra due variabili. Nel caso (2) si crea una tabella che stabilisce un legame fra i valori della variabile *Capitale* e la variabile *interesse*. A tale scopo si introducono coppie di valori nella tabella, il primo dei quali rappresenta il valore di *Capitale* e il secondo quello di *interesse* in modo da definire una spezzata che stabilisce il legame fra le due variabili. Dato un valore di *Capitale* compreso fra due valori della tabella il corrispondente valore di *interesse* lo si ricava con una operazione di estrapolazione lineare. Nel caso (3), infine, invece che mediante una tabella la funzione viene caratterizzata fissando coppie di punti in modo da definire una spezzata che rappresenta il legame fra le due variabili. Entrambi i metodi portano, comunque, alla definizione di relazioni continue fra le due variabili ovvero di spezzate i cui segmenti hanno pendenze diverse ma sono raccordati. L'uso di tabelle o di graficiè da preferire al'uso dell'istruzione "IF THEN ELSE" in tutti i casi in cui si dovrebbero annidare numerose istruzioni, fatto che potrebbe portare a istruzioni complesse da scrivere e difficili da interpretare.

Capitolo 2

La progettazione di editor grafici e la visualizzazione di grandezze variabili nel tempo

2.1 Introduzione

Il presente capitolo si propone di illustrare alcuni problemi di carattere generale relativi al progetto ed alla implementazione di editor grafici ed alla visualizzazione di grandezze variabili in funzione del tempo.
L'analisi delle problematiche relative al progetto ed alla implementazione di editor grafici sarà svolta limitatamente agli editor per la creazione e la manipolazione di grafi (detti *editor grafici orientati ai grafi*) con lo scopo di illustrare quali caratteristiche deve presentare un editor di tale tipo per consentire ad un utente la creazione e la modifica di grafi, il loro salvataggio in strutture persistenti il cui contenuto deve poter essere visualizzato in modo coerente ogni volta che sia necessario.
I grafi sono strutture dati utilizzabili per comunicare informazioni in forma pittorica in molti ambiti applicativi. Dato un grafo, i suoi nodi possono rappresentare gli oggetti di una applicazione mentre gli archi rappresentano le relazioni esistenti fra tali oggetti.
Un editor grafico orientato ai grafi è un tool interattivo che consente di rappresentare i grafi un forma pittorica e mette a disposizione dell'utente un certo numero di operazioni per l'editing dei grafi.
Una differenza sostanziale che c'è fra un editor grafico orientato ai grafi e un *programma di grafica di tipo general purpose* consiste nel fatto che il secondo tratta gli oggetti tracciati dall'utente come entità autonome e quindi come punti e linee piuttosto che come nodi ed archi, mentre il primo li considera come elementi di una struttura dati complessa caratterizzati da relazioni reciproche. Una conseguenza ovvia è che, nel primo caso, una operazione di spostamento di un nodo

si riflette su tutti gli archi incidenti nel nodo mentre nel secondo caso ciò non accade e gli elementi che rappresentano gli archi devono essere spostati a mano. L'ambiente $D(a)ySy\,Tool\,Box$ ha come scopo la determinazione della soluzione di un certo numero di equazioni e la visualizzazione delle soluzioni sotto forma di $graficicartesiani$ in cui i valori assunti dalle variabili caratteristiche del modello sono rappresentati in funzione del tempo.
Il presente capitolo contiene, pertanto, una breve discussione delle problematiche relative alla visualizzazione di grandezze variabili in funzione del tempo (fra le quali: scelta della scala dell'asse dei tempi, scelta della scala dell'asse delle ordinate e visualizzazione di più grafici sovrapposti) ed una illustrazione delle soluzioni adottate nel presente contesto.

2.2 La progettazione di editor grafici orientati ai grafi

2.2.1 Introduzione

Un editor grafico orientato ai grafi[1] è un tool interattivo che consente la rappresentazione dei grafi in forma pittorica e ne consente l'editing in tempo reale da parte dell'utente.

Figura 2.1: *Interazione fra un editor grafico e una applicazione*

Le operazioni di editing prevedono l'aggiunta, la rimozione e la modifica sia dei nodi sia degli archi del grafo (cfr. la sezione 2.2.2) e una interrogazione delle informazioni ad essi associate. Le modifiche apportate dall'utente ad un grafo si riflettono sulla rappresentazione pittorica del grafo e si traducono in azioni significative per l'applicazione che utilizza la struttura dati grafo per rappresentare i propri dati (cfr. la figura 2.1). Un editor grafico orientato ai grafi combina, pertanto, una modalità di tipo generale di rappresentazione dell'informazione (i grafi) con un modello di interazione di tipo generale (un editor).

[1] Il contenuto della presente sezione 2.2 deriva in gran parte da ([Pau93]).

Il tracciamento dei grafi, ovvero il posizionamento di nodi e archi, può avvenire in modo *automatico* oppure *manuale*.
La modalità di tracciamento manuale consente il cosiddetto *tracciamento incrementale* dei grafi: l'utente crea i grafi posizionando nodi e archi in posizioni arbitrarie e non ci sono strumenti per garantire layout[2] di qualità accettabile.
La modalità di tracciamento automatico fa di solito uso di *algoritmi di layout* guidati dalle cosiddette *estetiche di layout* ovvero da vincoli di tipo estetico che descrivono quali attributi di un grafo sono significativi e devono essere presi in considerazione al momento del posizionamento di nodi e archi in modo da produrre un layout che rispecchi certe proprietà topologiche del grafo. Le estetiche di layout possono richiedere, ad esempio, che venga minimizzato il numero degli incroci fra gli archi del grafo oppure che vengano evidenziate proprietà topologiche di un grafo quali la planarità, la simmetria e la struttura gerarchica.
Gli algoritmi di layout, tuttavia, non possono tenere conto con facilità dei vincoli posti sia dall'utente sia dall'applicazione dal momento che:

1. tendono a soddisfare solo le estetiche di layout relative alla struttura del grafo,

2. non hanno nessuna conoscenza della semantica del grafo ovvero del significato di nodi e archi per l'applicazione che usa tale forma di rappresentazione dei propri dati.

Pertanto, se è necessario che vincoli posti dall'utente e dall'applicazione siano soddisfatti, è necessario prevedere un gestore ad hoc che ne garantisca il soddisfacimento.
Altre problematiche relative agli editor grafici orientati ai grafi che saranno brevemente esaminate nel seguito riguardano la stabilità dei layout, l'astrazione grafica, la persistenza e l'estendibilità.
La *stabilità* può essere definita come una caratteristica di un editor grafico orientato ai grafi che garantisce che il layout di un grafo non cambi in maniera eccessiva in seguito ad operazioni di editing sul grafo stesso.
L'*astrazione grafica* (cfr. la sezione 2.2.5) viene di solito implementata fornendo all'utente strumenti che consentono di evidenziare sottografi del grafo corrente mediante l'impostazione di relazioni che devono essere soddisfatte dai nodi e/o dagli archi.
La *persistenza* indica la capacità di un editor di salvare le strutture dati dei grafi in strutture permanenti quali i file. Le informazioni memorizzate in file sono ,di solito:

1. informazioni relative alla struttura del grafo,

2. attributi di nodi e archi,

[2]Letteralmente il termine *layout* significa *disposizione delle parti*. Nel seguito verrà usato il termine inglese, di uso corente in ambito informatico.

3. informazioni di tipo pittorico, necessarie al posizionamento dei nodi e al tracciamento degli archi.

L'*estendibilità* è una misura della capacità di un editor di modificare la rappresentazione dei grafi in funzione dell'applicazione e di interagire correttamente con l'applicazione traducendo le operazioni su nodi e archi nella semantica di questa.

2.2.2 I grafi: alcune definizioni

Un grafo rappresenta una struttura dati astratta descrivibile in funzione di due insiemi, un insieme di nodi $N = \{i; i = 1, .., n\}$ ed un insieme di archi A, i cui elementi sono coppie di elementi distinti di N per cui un grafo, per definizione, non contiene *cappi* ovvero archi aventi per estremi lo stesso nodo.
Se gli estremi di un arco (u, v) costituiscono una coppia ordinata, l'arco si dice orientato e differisce dall'arco di estremi (v, u).
Se gli estremi di un arco (u, v) sono una coppia non ordinata, l'arco si dice non orientato e coincide con l'arco di estremi (v, u).
Dato un arco orientato (u, v) il nodo u è detto *coda* mentre il nodo v è detto *testa* dell'arco e, inoltre, il nodo u è detto essere il predecessore di v che, a sua volta, è il successore di u. Se tutti gli archi sono orientati il grafo è detto orientato (o digrafo) mentre se tutti gli archi non sono orientati il grafo è detto essere non orientato.
Dal momento che un arco non orientato può essere visto come una coppia di archi orientati in direzioni opposte si può affermare che i grafi orientati sono una generalizzazione di quelli non orientati.

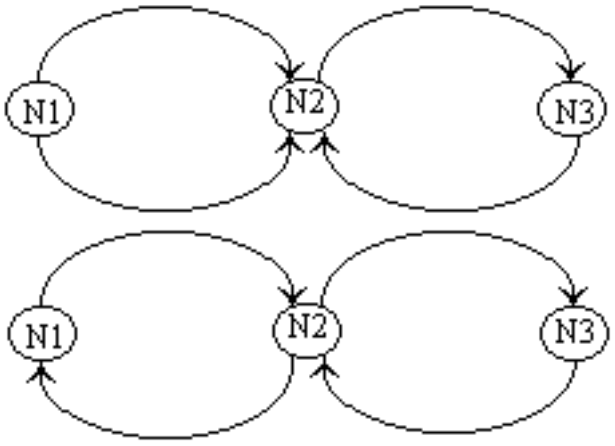

Figura 2.2: *Multigrafi e grafi (cfr. la figura 1.11)*

Oltre che dall'orientamento dei suoi archi, un grafo è caratterizzato dal numero

di archi equiorientati che possono essere presenti fra due nodi. Se fra due nodi u e v può essere presente al più un arco si parla di grafo tout court mentre se fra due nodi u e v possono essere presenti più archi si parla di *multigrafo*.
Nel seguito ci limiteremo a considerare due tipologie di grafi (cfr. la figura 2.2):

1. grafi orientati in cui fra due nodi u e v vi può essere al più un arco (v, u),
2. multigrafi, ovvero grafi in cui fra due nodi u e v vi possono essere più archi.

I grafi del primo tipo saranno utilizzati per la costruzione dei diagrammi CL mentre i grafi del secondo tipo saranno utilizzati per la costruzione dei diagrammi FD. In questo caso dati due nodi fra cui possono esistere più archi equiorientati vedremo che gli archi saranno di tipi distinti, al più due.
Introducendo il concetto di molteplicità di un arco[3] è possibile ricondurre i multigrafi ai grafi per cui nel seguito faremo riferimento solo ai grafi del primo tipo.
Dato, pertanto, un grafo orientato $G = (N, A)$ e dato un nodo $i \in N$ si introducono i concetti di *grado entrante* d_in_i, di *grado uscente* d_out_i, di *stella entrante* (BS_i) e di *stella uscente* (FS_i) del nodo i come segue:

$$BS_i = \{j \in N \mid (j, i) \in A\}$$
$$FS_i = \{j \in N \mid (i, j) \in A\}$$
$$d\_in_i = \#BS_i$$
$$d\_out_i = \#FS_i$$

In considerazione del fatto che gli elementi di N possono essere considerati rappresentare gli elementi di una applicazione o gli elementi di un dato dominio, mentre gli elementi di A rappresentano le relazioni esistenti fra tali elementi, la stella entrante consente di definire una relazione 1 a m fra un nodo e i nodi della sua BS_i (se $m = |BS_i|$) mentre la stella uscente consente di definire quali sono i nodi $j \in N$ influenzati da un dato nodo i e in che modo.
In un grafo orientato G si definisce un *cammino* C tra un nodo i_s detto *sorgente* e un nodo i_t detto *destinazione* una successione di nodi e archi che permettono di passare da i_s a i_t.
Se tutti gli archi di un cammino C sono orientati da i_s a i_t il cammino si dice orientato. Una sottosequenza di C è detta *sottocammino*. Se $i_s = i_t$ il cammino è detto essere un ciclo (eventualmente orientato). Un ciclo è semplice se non contiene cicli come sottocammini propri. Il concetto di *ciclo* è stato utilizzato per individuare gli anelli di feedback nei modelli dei Sistemi Dinamici (cfr. il Capitolo 1).

[3] La molteplicità di un arco è un numero intero n che rappresenta il numero di volte che un arco (u, v) è presente nel grafo fra una coppia di nodi u e v ([Ore76]). È evidente che per i grafi è $n = 1$.

Dato, infine, un grafo $G = (N, A)$ è possibile considerare un sottoinsieme dell'insieme dei nodi $N' \subseteq N$ oppure un sottoinsieme dell'insieme degli archi $A' \subseteq A$ in modo da evidenziare una porzione del grafo avente specifiche caratteristiche, ad esempio i cui nodi o i cui archi sono di un certo tipo.
In funzione del modo in cui la porzione di grafo viene individuata si parla, rispettivamente, di *grafo parziale*, di *grafo indotto* oppure di *sottografo* sebbene nel seguito, per economia di notazione, si userà solamente la dizione generica di *sottografo*.
Dato, pertanto, il grafo $G = (N, A)$ si ha che:

1. un suo grafo parziale è il grafo $G' = (N, A')$ con $A' \subseteq A$;
2. un suo grafo indotto è il grafo $G'' = (N'', A'')$ con $N'' \subseteq N$ e $A'' = \{(i, j) \in A : i, j \in N''\}$
3. un suo sottografo è il grafo $G^* = (N^*, A^*)$ con $A^* \subseteq A$ e N^* che contiene tutti i nodi che sono estremi deli archi di A^*

Dato un grafo $G = (N, A)$ è possibile associare agli archi ed ai nodi delle grandezze sia numeriche sia di altri tipi, ad esempio tipi, etichette o funzioni. In letteratura si trova il termine *rete* ogni volta che ai nodi e/o agli archi di un grafo sono associati dei valori numerici detti *pesi*. Tale termine sarà usato anche nella presente tesi tutte le volte che sarà necessario evidenziare la differenza fra una struttura topologica a grafo e la struttura più ricca di informazioni prodotta da una certa applicazione.

2.2.3 La caratterizzazione di un editor grafico orientato ai grafi

Un editor grafico orientato ai grafi [Pau93](nel seguito solo editor grafico) è un tool interattivo che presenta all'utente i grafi sotto forma di una rappresentazione pittorica e consente l'esecuzione su di essi di operazioni di editing che si riflettono sulla rappresentazione pittorica in tempo reale.[4]
Un editor è, pertanto, caratterizzato da una interfaccia utente che consente di visualizzare sia grafi di grosse dimensioni, mediante aree scrollabili, sia sottografi e di eseguire operazioni di aggiunta, rimozione e modifica di oggetti. Gli oggetti sono le entità rappresentabili dall'editor e che l'utente può selezionare ovvero nodi e archi.
Ogni oggetto ha associati degli attributi che l'utente può interrogare e modificare e che sono utilizzabili anche per modificare l'aspetto visivo dell'oggetto e il suo ruolo nel contesto dell'applicazione.
Gli editor grafici orientati ai grafi possono essere classificati come:

[4]Esistono altri strumenti, detti *browser grafici*, che consentono solo la visualizzazione dei grafi senza consentirne la modifica.

1. editor di tipo "special purpose" ovvero editor progettati per lavorare in combinazione con una particolare applicazione;

2. editor di tipo "general purpose" utilizzabili per l'editing di strutture di tipo grafo ma privi di una interfaccia verso una applicazione;

3. editor estendibili ovvero editor che mettono a disposizione dell'utente sia un editor di tipo "general purpose" sia una interfaccia verso una applicazione che traduce le operazioni eseguite su un grafo nella semantica dell'applicazione.

Sebbene sia possibile ottenere rappresentazioni pittoriche di grafi utilizzando, invece di un editor special purpose, un programma di grafica di tipo "general purpose" di solito si preferisce far ricorso a un editor special purpose dal momento che questi possiede, in genere, sia gli strumenti per interfacciarsi con una applicazione sia le informazioni per trattare i grafi come strutture dati astratte. In tal modo le operazioni sui nodi (spostamento e rimozione, ad esempio) si riflettono automaticamente sugli archi incidenti ed in più il programma possiede le informazioni necessarie per il tracciamento del layout (cfr. la sezione 2.2.4).

2.2.4 Algoritmi e vincoli di layout

Un *algoritmo di layout* è un algoritmo che, dato un grafo $G = (N, A)$, determina sia la posizione dei nodi del grafo su una superfice di visualizzazione (che nel seguito diremo *canvas*) sia il percorso degli archi fra i vari nodi. L'uso di un algoritmo di layout permette il tracciamento automatico di un grafo ed evita all'utente la fatica di posizionare manualmente nodi ed archi. È pertanto consigliato nel caso di grafi di grosse dimensioni. Inoltre si deve fare ricorso ad algoritmi di layout in tutti i casi in cui la struttura di un grafo viene acquisita dall'esterno (cfr. la sezione 2.2.7).
Lo scopo ultimo di ogni algoritmo di layout è quello di produrre delle rappresentazioni pittoriche dei grafi che siano chiare e facili da capire. Tale scopo viene di solito conseguito mediante la applicazione di un certo numero di *criteri estetici*. È possibile, inoltre, porre un certo numero di restrizioni sul layout dette *vincoli di layout*. Tali vincoli sono, di solito, relativi alle modalità di posizionamento dei nodi e di tracciamento degli archi.
Gli algoritmi di layout ([Pau93]) possono essere classificati in base ai seguenti criteri:

1. standard grafici,

2. estetica,

3. vincoli di layout.

Nel primo caso il layout è determinato dalla presenza di una griglia che definisce due standard: il primo, detto della griglia, impone che i nodi del grafo siano posizionati nei nodi della griglia e che gli archi siano tracciati lungo le maglie della griglia mentre il secondo, detto delle linee rette, permette un posizionamento libero dei nodi mentre gli archi devono essere tracciati come linee rette fra coppie di nodi.
L'estetica di un grafo è una misura della bontà della sua rappresentazione pittorica. La scelta di una estetica piuttosto che un'altra dipende da vari fattori fra cui: il tipo di grafo, il tipo di applicazione associata al grafo e il gusto dell'utente. Alcuni dei criteri estetici di solito adottati sono i seguenti ([Pau93]):

minimizzare gli incroci fra archi ovvero posizionare gli archi in modo da disegnare un grafo il più possibile planare;

evidenziare la gerarchia ovvero orientare gli archi in modo da evidenziare la struttura gerarchica presente nel grafo;

massimizzare la simmetria;

minimizzare l'area occupata dalla rappresentazione pittorica del grafo;

minimizzare la larghezza dell'area totale usata dal layout del grafo;

minimizzare l'arco più lungo;

minimizzare la lunghezza totale degli archi;

uniformare la lunghezza degli archi ovvero usare archi grosso modo della stessa lunghezza;

uniformare la distribuzione dei nodi;

minimizzare le curvature degli archi;

allineare i livelli in modo da mettere nodi di uguale profondità sullo stesso livello

e altri di minore importanza relativi al posizionamento dei nodi padri rispetto ai figli e viceversa e alla rappresentazione isomorfa o meno dei sottografi di uguale struttura.
Un algoritmo di layout può cercare di soddisfare più criteri estetici contemporaneamente ma ciò può portare all'insorgere di conflitti. Ad esempio la richiesta di tracciare un grafo con un layout gerarchico (il caso tipico è quello di una struttura ad albero) si scontra, in generale, con la richiesta di minimizzare gli incroci fra gli archi, richiesta che sarebbe possibile soddisfare ricorrendo ad un layout planare in cui la struttura gerarchica, però, in genere va perduta.
I vincoli di layout sono restrizioni poste al layout di solito dalla semantica del

grafo ma è possibile imporre dei vincoli di tipo generale relativi alla posizione di uno o più nodi nel layout del grafo ([Pau93]). Tali vincoli, detti generici, permettono di imporre l'alineamento di un gruppo di nodi, la definizione di cluster di nodi, il posizionamento di uno o più nodi al centro oppure sul bordo del layout e il posizionamento relativo di un nodo rispetto ad un altro.
I vincoli di layout, allo stesso modo dei criteri estetici, possono entrare in conflitto tra di loro. Una possibile soluzione è quella di assegnare un valore di priorità che ne individui l'importanza relativa in modo da poter cercare di soddisfare i vincoli in ordine decrescente di priorità e in assenza di conflitti.
In funzione della tipologia del grafo di cui devono tracciare il layout, gli algoritmi di layout possono di deafult privilegiare certi criteri estetici rispetto ad altri. Nel caso di *grafi planari*, importanti per applicazioni di teoria dei grafi e in tutti i casi di archi bidirezionali (quali le reti di comunicazione e, in genere, i percorsi stradali), l'obbiettivo estetico più ovvio è quello di produrre layout privi di incroci fra archi in modo da evidenziare la planarità.
Nel caso di *grafi non orientati* si tende a far uso di criteri estetici di tipo generale (quali: minimizzare l'area, massimizzare la simmetria, minimizzare il numero di incroci fra archi oppure usare distribuzioni uniformi di nodi e/o archi) dal momento che si hanno poche informazioni in merito al tracciamento di un grafo.
Nel caso di *alberi*, utilizzati per alberi sintattici e di decisione, i criteri estetici da adottare sono quelli che evidenziano la truttura e pertanto: evidenziare la gerarchia, allineare i livelli spaziandoli fra di loro e posizionare correttamente i nodi padri rispetto ai figli.
Nel caso di *grafi orientati*, utilizzati ad esempio nel caso degli automi a stati finiti, i criteri estetici sono quelli di evidenziare l'orientamento degli archi, allineare i nodi di uguale rango, minimizzare l'area occupata dal grafo e minimizzare il numero di incroci fra archi.
Come visto nei paragrafi precedenti, gli algoritmi di layout si limitano a posizionare nodi e archi di un grafo in un layout essenzialmente in base a criteri estetici, dal momento che non hanno informazioni in merito alla semantica del grafo ovvero non conoscono il significato di nodi e archi nel dominio dell'applicazione che usa la struttura dati a grafo per rappresentare i propri dati.
Ciò che un algoritmo di layout si limita a fare è, dato un grafo, produrne una rappresentazione pittorica che sia la più chiara e comprensibile.
In molti casi è necessario che sia l'utente sia l'applicazione siano in grado di porre dei vincoli sul layout del grafo detti, come visto più sopra, vincoli semantici e vincoli generici. L'introduzione di tali vincoli richiede la presenza di un modulo software ad hoc detto *gestore dei vincoli* che si deve occupare di gestire i vincoli suddetti e di integrarli nei vari algoritmi di layout (che in genere devono essere modificati allo scopo).
I compiti di un tale gestore sono essenzialmente i seguenti:

1. mantenere una lista di vincoli caratterizzati da una prorità, alla quale i vincoli possono essere aggiunti, dalla quale possono essere rimossi, e controllare lo stato globale dei vincoli,

2. mantenere consistente l'insieme dei vincoli ovvero evitare che siano presenti vincoli fra loro in conflitto.

Nel caso in cui l'insieme dei vincoli sia inconsistente è compito del gestore risolvere il problema disattivando alcuni dei vincoli, partendo da quelli a più bassa priorità, fino ad ottenere un insieme consistente.
Un'altra caratteristica degli algoritmi di layout che può essere fonte di problemi è che molti di tali algoritmi non considerano il layout corrente al momento di tracciarne uno nuovo (a seguito di inserimenti e/o rimozioni di nodi ed archi) in modo che il nuovo può differire radicalmente dal precedente fino al punto che la porzione di grafo che l'utente stava editando può trovarsi al di fuori della vista corrente.
Tale caratteristica può disorientare l'utente e al proposito si parla di *stabilità di layout* come di una misura della entità del cambiamento di un grafo in seguito ad operazioni di editing. Una elevata stabilità la si ottiene minimizzando le differenze fra layout consecutivi sotto il vincolo dei vari criteri estetici visti nei paragrafi precedenti.

2.2.5 Astrazione grafica

Le *astrazioni grafiche* forniscono rappresentazioni alternative o supplementari utilizzabili per ottenere una migliore comprensione di un grafo. Le si dice astrazioni perchè consentono di nascondere dettagli ritenuti non rilevanti per l'utente. A tale scopo le astrazioni grafiche mettono a disposizione dell'utente una vasta gamma di possibilità che vanno dalla visualizzazione a livello di nodi e archi fino a quella del'intero layout in cui i dettagli relativi ai singoli nodi e archi non sono più visibili.
Le problematiche relative alla astrazione grafica possono essere affrontate da due punti di vista che sono quello di definizione di una astrazione e quello di rappresentazione della astrazione.
Per definizione di una astrazione si intendono gli strumenti che consentono all'utente di specificare i sottografi da utilizzare come astrazioni grafiche.
Di solito i meccanismi di astrazione per sottografi sono definiti manualmente dall'utente sebbene sia possibile fare uso di tecniche di clustering per la definizione dei sottografi da usare per l'astrazione grafica. Tali tecniche prevedono l'uso di riferimenti incrociati e il partizionamento di un grafo in più sottografi relativamente indipendenti che sono tracciati separatamente ma la loro trattazione esula dagli scopi della presente tesi. Per rappresentazione di una astrazione si intendono, invece, i modi in cui una astrazione grafica viene presentata all'utente.
Strumenti tipici utilizzabili per la rappresentazione delle astrazioni grafiche sono:

1. le viste, ovvero l'uso di finestre separate per la visualizzazione di sottografi,

2. la strutturazione gerarchica ovvero la possibilità di passare da un livello di astrazione ad un altro in una gerarchia di livelli,

3. la restrizione ovvero la possibilità di variare la porzione di grafo correntemente visualizzata,

4. gli ausili per la navigazione ovvero, sopratutto nel caso di grafi molto complessi, un certo numero di strumenti che consentono all'utente di mantenere il proprio orientamento nella struttura del grafo. Alcuni esempi di ausili per la navigazione sono il *focusing* e la *panoramica di layout*. Il primo prevede la possibilità di definire un oggetto del grafo che diventa il centro della rappresentazione pittorica mentre il secondo prevede la definizione di una finestra contenente la rappresentazione schematica dell'intero grafo con evidenziata la porzione correntemente visualizzata.

Basati su tali strumenti se ne hanno altri due che saranno esaminati più in dettaglio nel seguito, ovvero:

1. il sottografo di astrazione con o senza viste separate,

2. il concentramento di archi.

Un *sottografo di astrazione* è un sottografo cui sono associate una o più rappresentazioni grafiche fra le quali l'utente può sceglierne una in modo da controllare il livello di dettaglio della rappresentazione del sottografo.
I sottografi di astrazione possono essere utilizzati sia per nascondere alla vista porzioni di un grafo perchè non interessanti oppure per evitare un eccessivo affollamento di un layout sia per consentire la visualizzazione del solo sottografo mediante l'uso di viste separate (vedi oltre).
In entrambi i casi l'obbiettivo è quello di focalizzare l'attenzione dell'utente su certe porzioni di un grafo.
Di solito un sottografo di astrazione viene raffigurato come un nodo all'interno del grafo di partenza, ovvero viene visualizzato nel contesto del grafo cui appartiene e viene posizionato dagli algoritmi di layout come ogni altro nodo, sebbene per la sua visualizzazione effettiva si abbiano più possibilità ovvero:

1. lo si visualizza come un nodo normale associandogli una icona ad hoc che lo individua come una astrazione, in questo caso la struttura del sottografo non è visibile e si parla di vista di tipo "black box",

2. lo si visualizza come un nodo più grande degli altri in modo da poter rappresentare al suo interno il layout del sottografo.

Nel secondo caso è necessario decidere come tracciare gli archi che connettono i nodi interni al sottografo con i restanti nodi del grafo. Si hanno due possibilità: nella prima gli archi di connessione si fermano sul confine del nodo astrazione (ovvero la struttura interna è visibile ma non è collegata con quella esterna) mentre nella seconda gli archi connettono i nodi esterni ai nodi dell'astrazione per cui la struttura interna è visibile ed è noto come sia collegata con il resto del grafo. Nei due casi si parla, rispettivamente, di vista di tipo "scatola grigia" e di vista di tipo "scatola bianca" e l'editor deve disporre di comandi per consentire all'utente di passare dall'una all'altra con facilità.
È possibile prendere la decisione di tracciare il layout del sottografo di astrazione in modo del tutto indipendente da quello del resto del grafo. Se ciò da un lato può accellerare il processo di tracciamento del grafo perché solo i sottografi interessati da un cambiamento devono essere ritracciati dall'altro può dare luogo, soprattutto nel caso della vista di tipo "scatola bianca", data l'indipendenza dei layout, ad un eccessivo numero di incroci fra archi.
La scelta di rappresentare un sottografo di astrazione come un nodo nel contesto del grafo cui appartiene permette di costruire una gerarchia stretta di astrazioni per sottografi in cui un nodo può appartenere ad uno ed un solo sottografo.
Un'altra possibilità è quella di rappresentare un sottografo di astrazione in una vista separata, ovvero in una rappresentazione visualizzata in una finestra separata sullo schermo, indipendente da quella in cui viene visualizzato il grafo nel suo complesso.
Una vista separata può essere usata sia per aprire una sessione di editing autonoma di un sottografo di astrazione (ma le variazione apportate alla vista non si ripercuotono automaticamente sul grafo di partenza) oppure può essere usata, nel caso ad esempio si abbia un grafo con archi di tipi diversi, per visualizzare tutti gli archi di un certo tipo.
In entrambi i casi (visualizzazione nel contesto del grafo oppure in una vista separata) si hanno vari metodi in base ai quali è possibile individuare un sottografo di astrazione a partire da un grafo dato considerando che, salvo diversa specificazione, un sottografo è di solito individuato specificando un sottoinsieme dei nodi del grafo di partenza mentre gli archi del sottografo sono tutti gli archi del grafo i cui estremi appartengono al sottoinsieme scelto.[5]
I metodi di uso corrente sono i seguenti:

1. selezione individuale: l'utente seleziona un sottografo manualmente scegliendone i nodi uno ad uno, compresi eventuali nodi astrazione visualizzati in contesto di grafo;

2. selezione per area: l'utente seleziona un sottografo manualmente scegliendo i nodi contenuti in una porzione dello schermo, compresi eventuali nodi

[5]Un sottografo può essere specificato anche selezionando un sottoinsieme dell'insieme degli archi per cui i nodi del sottografo sono tutti quei nodi che sono estremi degli archi scelti.

astrazione visualizzati in contesto di grafo;

3. selezione per tipo: l'utente specifica uno o più tipi e i nodi e gli archi che soddisfano la specifica di tipo individuano il sottografo;

4. selezione per nome: l'utente definisce una espressione regolare e i nodi il cui nome soddisfa tale espressione individuano il sottografo;

5. selezione per chiusura transitiva: l'utente specifica un nodo e i suoi predecessori (o successori) per individuare un sottografo, in più può specificare che solo archi di un certo tipo facciano parte del sottografo;

6. selezione per lettura da file: l'utente può leggere la descrizione di un sottografo da un file e, in base a questa, creare un sottografo;

7. selezione funzionale: l'utente può definire un sottografo applicando una funzione che, dato un grafo, ne restituisce un sottografo.

In molte aree applicative capita spesso di avere grafi con un gran numero di archi rispetto al numero dei nodi.[6] In tali casi la rappresentazione di un grafo può contenere un numero di archi così elevato da essere in pratica inutile come ausilio visivo per la comprensione delle relazioni fra i nodi.
Una possibile soluzione consiste nell'individuare nel grafo dato $G = (N, A)$ eventuali sottografi bipartiti completi in modo da sostituirli con rappresentazioni equivalenti che consentono di ridurre sia il numero degli archi sia gli incroci fra questi, aumentando in questo modo la leggibilità del grafo.[7]
Il procedimento, di cui si danno solo brevi cenni, consiste nel passare da sottografi bipartiti completi $B = (N_1, N_2, A)$ a sottografi tripartiti equivalenti $T = (N_1, EC, N_2, A')$ ottenuti inserendo un livello ulteriore contenente il solo nodo EC (detto *concentratore di archi*[8] e caratterizzato da una icona ad hoc che permette di distinguerlo dagli altri nodi) fra i due livelli preesistenti in modo che sia $A' = (N_1 \times EC) \cup (EC \times N_2)$.
In tal modo il nodo EC riceve archi da tutti i nodi di N_1 e invia archi a tutti i nodi di N_2 per cui se $\#N_1 = n_1$ e $\#N_2 = n_2$ allora B ha $n_1 + n_2$ nodi e $n_1 \times n_2$ archi mentre T ha $n_1 + n_2 + 1$ nodi e $n_1 + n_2$ archi.
È evidente che tale trasformazione è possibile solo se la relazione che esiste fra

[6] In un grafo orientato con n nodi si posono avere fino a $n(n-1)$ archi. Nel caso dei multigrafi tale numero deve tenere conto della molteplicità dei vari archi.

[7] Si ricorda che un *grafo bipartito* è un grafo $G = (N_1, N_2, A)$ in cui l'insieme dei nodi N può essere ripartito nei due sottoinsiemi N_1 e N_2 (tali che $N_1 \cup N_2 = N$ e che $N_1 \cap N_2 = \emptyset$) e ogni arco di A unisce un nodo di N_1 con uno di N_2. La rappesentazione equivalente è ottenuta con un *grafo tripartito* ovvero in cui l'insieme dei nodi N può essere ripartito nei tre sottoinsiemi N_1, N_2 e N_3 e in cui non si hanno archi fra nodi dello stesso sottoinsieme. Un grafo, inoltre, è detto essere *completo* se esiste un arco fra ogni coppia di nodi e, infine, un *grafo bipartito completo* è un grafo bipartito in cui è $A = N_1 \times N_2$.

[8] Da cui il nome di *concentramento di archi.*

i nodi di B è la stessa per tutti i nodi e, inoltre, che tale trasformazione non conserva eventuali informazioni associate ai singoli archi ma, tutte le volte che è applicabile, riesce ad aumentare notevolmente la leggibilità dei grafi.
Dato un grafo G, il primo passo è quello di individuare i possibili concentramenti di archi per poi passare a convertirli nelle relative rappresentazioni equivalenti. Un possibile modo di procedere, che riduce di molto la complessità computazionale della ricerca, è quello di limitarla a sottografi bipartiti $B = (N_1, N_2, A')$ di G in cui N_1 è l'insieme dei nodi di G di un certo livello, N_2 è l'insieme dei nodi successori di N_1 e $A' \subset A$ è l'insieme degli archi fra N_1 e N_2.
Lo scopo di tutti gli algoritmi di *concentramento di archi* è quello di minimizzare il numero di archi nel grafo trasformato senza perdita di informazioni.
Per una trattazione più approfondita e una soluzione approssimata del problema si rimanda a ([Pau93]).

2.2.6 Le operazioni di editing ed il soddisfacimento dei vincoli topologici ed applicativi

Le operazioni di editing vengono eseguite a livello di rappresentazione pittorica e consentono di interagire direttamente con questa e indirettamente con la struttura del grafo sottostante e con l'insieme delle relazioni 1 a m fra un nodo i ed i nodi della sua BS_i.
A livello di rappresentazione pittorica l'editor consente:

1. l'aggiunta, la rimozione ed il posizionamento delle icone;
2. l'aggiunta, la rimozione ed il posizionamento degli elementi di connessione;
3. la customizzazione delle icone (modifica dell'etichetta) e degli elementi di connessione (modifica dell'orientamento);
4. l'interrogazione delle icone e degli elementi di connessione.

Tali operazioni a livello di rappresentazione pittorica sarebbero sempre possibili dal momento che la rappresentazione pittorica non possiede informazioni né in merito alla struttura del grafo sottotante né in merito al dominio applicativo.
L'introduzione di vincoli topologici ed applicativi permette di abilitare o meno le singole operazioni ed inoltre fa in modo che la rappresentazione pittorica sia sempre allineata con la struttura del grafo corrispondente: l'utente interagisce, infatti, con la rappresentazione pittorica manipolando indirettamente il grafo ed eseguendo su di esso operazioni che possono riflettersi a loro volta sulla rappresentazione pittorica e sull'insieme delle relazioni 1 a m fra i nodi del grafo.
Alcuni vincoli topologici relativi alle operazioni di aggiunta di archi, ad esempio, sono i seguenti:

1. un arco può esistere solo fra nodi distinti,

2. un arco può essere aggiunto fra due nodi solo se fra questi non esiste già un arco con lo stesso orientamento.

L'aggiunta di un elemento di connessione deve sottostare al soddisfacimento dei vincoli suddetti e si traduce nell'aggiunta di un arco (i, j) fra i nodi i e j e nell'aggiornamento di FS_i (aggiunta di j) e BS_j (aggiunta di i).
Le operazioni per la rimozione di icone e di elementi di connessione non sono soggette direttamente a vincoli topologici ma richiedono l'aggiornamento della rappresentazione pittorica cui corrisponde un aggiornamento del grafo che può riflettersi in un ulteriore aggiornamento della rappresentazione pittorica.
Ad esempio la rimozione di un elemento di connessione si traduce nella rimozione dell'arco (i, j) corrispondente e nell'aggiornamento di FS_i (rimozione di j) e BS_j (rimozione di i).
La rimozione di un'icona si traduce nella rimozione del nodo $i \in N$ corrispondente e nella rimozione degli archi (k, i) e (i, k) tali che $k \in N$, incidenti nel nodo.
Tale rimozione si traduce nell'aggiornamento delle FS_k e BS_k e nell'aggiornamento della rappresentazione pittorica da cui devono essere rimossi gli elementi di connessione corrispondenti.
Se il fatto di considerare una struttura a grafo permette di introdurre i vincoli topologici l'introduzione di una tipizzazione dei nodi permette di introdurre i vincoli cosiddetti applicativi.
Se, infatti, le icone (e i nodi corrispondenti) sono tipizzate in funzione del dominio dell'applicazione è possibile definire un insieme di vincoli che impediscono il tracciamento di un elemento di connessione fra due icone di tipi dati e pertanto impediscono l'aggiunta di un arco fra i nodi corrispondenti.
Il dominio dell'applicazione permette di definire a tale scopo due insiemi che diremo dei tipi T e dei vincoli V. Tali insiemi sono definiti dalle relazioni seguenti:

$T = \{t_i\ ,\ i\ =\ 1,\ \ldots,\ n\}$
$V = \{v_{i,j}\ \ i,\ j = 1,\ \ldots,\ n\}$

dove:

$v_{i,j}\ =\ 1$ se $\exists\ R$ tale che $t_i\ R\ t_j$ mentre $v_{i,j}\ =\ 0$ se $\neg\exists\ R$ tale che $t_i R t_j$.

Nelle espressionini precedenti R rappresenta una relazione fra tipi ovvero fra elementi che soddisfano le varie definizioni di tipo.

2.2.7 Persistenza dei dati ed estendibilità

Come visto nella sezione 2.2.3, un editor grafico è un tool interattivo per la modifica di strutture dati per cui è necessario che sia caratterizzato da strumenti che consentano di salvare lo stato corrente della sessione di editing e delle strut-

ture dati in modo da poterle ripristinare in un qualunque momento. L'editor deve avere la possibilità di salvare tutte le sue strutture dati significative su una memoria non volatile sotto forma di una *rappresentazione esterna* che deve poter essere ricarcata dall'editor stesso in qualunque momento e con comandi semplici e intuitivi.
In tal modo le stutture dati create dall'editor possono essere mantenute oltre la durata di una sessione di editing permettendo di ottenere la cosiddetta *persistenza delle strutture dati* ovvero la memorizzazione persistente dei dati.
A tale fine sono disponibili essenzialmente due alternative: la prima fa uso di un data base mentre la seconda fa uso di file.
La prima soluzione è da preferire in tutti i casi in cui sia necessario dare una strutturazione logica ai dati e disporre dei servizi offerti da un data base.
La seconda soluzione, che si appoggia, per la gestione dei dati, sul file system del Sistema Operativo ospite, è la più adatta nel caso ci si voglia limitare a salvare lo stato di sessioni di editing: in questo caso le singole sessioni vengono salvate sotto forma di file la cui gestione è ottenuta inserendo nel codice dell'editor comandi propri del Sistema Operativo ospite.
In entrambi i casi un requisito essenziale è che la implementazione della persistenza dei dati sia trasparente all'utente.
Il modo più semplice per salvare in una rappresentazione esterna lo stato corrente di un grafo e di una sessione di editing è quello di fare uso di un linguaggio che deve essere caratterizzato dalle seguenti proprietà:

1. deve essere *completo* in modo che la rappresentazione esterna sia in grado di registrare tutte le informazioni significative relative sia al grafo sia alla sessione di editing con particolare riferimento
   (a) alla struttura del grafo (nodi, archi e astrazioni grafiche),
   (b) agli attributi che influenzano la visualizzazione di un grafo (scelta dell'algoritmo di layout, dei vincoli di layout e valori degli attributi per la visualizzazione di nodi ed archi),
   (c) allo stato della sessione di editing con particolare riferimento alla posizione e alla dimensione delle varie finestre di editing, alla posizione delle barre di scorrimento all'interno delle varie finestre e allo stato dei nodi correntemente visualizzati;
2. deve essere *estendibile* ovvero gli attributi dipendenti dalla applicazione dovrebbero essere rappresentati in modo simile agli altri attributi dei grafi;
3. deve essere *modulare* in modo da consentire di spezzare la rappresentazione esterna in moduli indipendenti e di estrarre la descrizione di un sottografo in modo da poterla usare indipendentemente da quella del resto del grafo;
4. deve essere *editabile* in modo che la rappresentazione esterna sia in formato testo e pertanto facilmente leggibile ed editabile;

5. deve essere *flessibile* in modo da non dettare un ordinamente rigido nella struttura della rappresentazione esterna nella quale intere porzioni, in certi casi, dovrebbero poter mancare e

6. deve essere *indipendente* sia dall'hardware sia dal Sistema Operativo ospite.

Data l'importanza di realizzare la persistenza dei dati sono stati proposti molti metodi fra i quali si segnalano i seguenti [Pau93]:

metodo dell'istantanea: il metodo prevede il salvataggio in memoria non volatile dello stato istantaneo dell'editor e delle strutture dati, è un metodo non portatile (è strettamente "machine dependant") e non modulare (lo stato viene salvato nella sua interezza);

metodo dell'appiattimento testuale: il metodo prevede la codifica della struttura dei grafi in una rappresentazione "piatta" contenuta in file di in formato testo, direttamente leggibili ed editabili (con tutti i rischi di inconsistenza che ciò comporta);

metodo degli spazi di indirizzamento multipli: in questo caso il sistema gestisce una tabella che contiene una entry per ognuna delle strutture dati che si vogliono persistenti; la tabella contiene le corrispondenze fra strutture dati e loro indirizzi, sia nello spazio di indirizzamento della memoria centrale sia nello spazio di indirizzamento su disco, e il sistema gestisce il trasferimento delle strutture dati fra il disco e la memoria centrale in modo del tutto trasparente all'utente (da cui il nome di *persistenza trasparente*);

metodo "a database": invece di fare uso di file per memorizzare i dati e/o lo stato dell'editor questi viene fatto interfacciare con un database in modo da consentire, fra le altre cose, la condivisione dei dati fra programmi e utenti.

Oltre ad essere in grado di gestire la *persistenza dei dati* un editor grafico dovrebbe essere il più possibile *estendibile*.
La estendibilità è una caratteristica dei programmi che consente loro di adattarsi ad un'ampia varietà di applicazioni. Un programma è detto esssere estendibile se

1. è *customizzabile* ovvero se l'utente può specificare i valori degli attributi di caratteristiche esistenti;

2. è *ampliabile* ovvero se l'utente può introdurre nuove caratteristiche al tool in modo che questi sia in grado di affrontare nuove ed impreviste (in fase di progetto) situazioni.

2.2.8 Soluzioni implementative

L'ambiente *D(a)ySy Tool Box* è caratterizzato da due editor grafici per la creazione e la modifica, rispettivamente, di diagrammi CL e di diagrammi FD.
Il primo di tali editor (cfr. la sezione 3.3) consente la creazione di grafi orientati mentre il secondo (cfr. la sezione 3.4) consente la creazione di multigrafi orientati in cui fra coppie di nodi u e v possono essere presenti fino a due archi con lo stesso orientamento (u, v).
Ad un $[multi]grafo$ corrisponde una rappresentazione pittorica caratterizzata da etichette ed elementi di connessione, nel caso dei diagrammi CL, e da icone, etichette ed elementi di connessione, nel caso dei diagrammi FD.
Nel caso di entrambi gli editor, l'utente istanzia e posiziona manualmente i nodi (rispettivamente etichette o coppie etichetta e icona) e traccia gli elementi di connessione utilizzando il mouse[9]
I nodi sono selezionati da un insieme di icone dipendente dall'editor e sono istanziati e posizionati con una tecnica di *select and drop*: l'utente seleziona l'icona e ne crea un'istanza con un click del mouse in un punto della superfice di tracciamento (cfr. al proposito la sezione 4.3.2). I nodi possono essere spostati sulla superfice di tracciamento e lo spostamento fa in modo che ogni nodo si trascini dietro gli elementi di connessione in esso incidenti.
Per il tracciamento degli elementi di connessione l'utente può, in entrambi gli editor, utilizzare sia segmenti di retta, sia spezzate, sia archi.
Le tecniche utilizzabili per il tracciamento degli elementi di connessione sono due: *point and click* e *drag and drop*. Nel primo caso l'utente, per eseguire il tracciamento, seleziona con semplici click del mouse il punto iniziale e finale di un segmento o di un arco e tutti i punti (iniziale, finale e intermedi) di una spezzata.
Nel secondo caso l'utente preme il pulsante del mouse nel punto di origine (cfr. oltre) della connessione e lo rilascia o nel punto di destinazione o in uno dei punti intermedi di una spezzata.
Gli editor mantengono in *tempo reale* la *consistenza* del grafo corrispondente alla rappresentazione pittorica implementando così i *vincoli topologici* ed *applicativi* mentre non sono stati imposti vincoli di layout e ciò (unito al posizionamento e al tracciamento manuali) può dare origine a layout "brutti".
I vincoli topologici sono stati ridotti al minimo e si riducono a tre vincoli sul tracciamento degli elementi di connessione e ad un vincolo sulla rimozione di nodi.
Il primo vincolo stabilisce che è possibile inserire un elemento di connessione se e

[9]Salvo avviso contrario il pulsante del mouse cui si fa riferimento è il sinistro. Per selezione di un elemento si intende la pressione e il rilascio di tale pulsante con il cursore posizionato sull'elemento mentre il trascinamento lo si ottiene spostando il cursore tenendo premuto il pulsante sinistro del mouse. Se il cursore è posizionato su un elemento al momento in cui viene premuto si ottiene lo spostamento dell'elemento puntato. Il rilascio del pulsante coincide con il posizionaento dell'elemnto.

solo se il suo punto iniziale e il suo punto terminale sono in prossimità dell'area occupata da un nodo origine, nel primo caso, e destinazione, nel secondo.
Il secondo e il terzo vincolo, cui nel caso di diagrammi FD si aggiunge un vincolo applicativo, stabiliscono, rispettivamente, che si possa tracciare un elemento di connessione fra due nodi oppure se ne possa invertire l'orientamento se e solo se la presenza di tale elemento è ammessa (vincolo applicativo, vedi oltre) e se non ne esiste già uno con lo stesso orientamento fra la stessa coppia di nodi (vincolo topologico).
Il vincolo sulla rimozione di un nodo, infine, stabilisce che la rimozione di un nodo si traduca nella rimozione di tutti gli archi in esso incidenti.
I vincoli applicativi sono attivi solo nel caso dei diagrammi FD e stabiliscono se un elemento di connessione di un certo tipo può essere presente o meno fra due nodi di cui sono noti i tipi. Per una definizione dei tipi dei nodi e degli elementi di connessione si rimanda alle sezioni 3.3 e 3.4.
I meccanismi di astrazione, implementati in entrambi gli editor, sono riconducibili alla possibilità di selezionare e visualizzare (in finestre dedicate di sola visualizzazione) sottodiagrammi (o sottografi) ovvero porzioni delle rappresentazioni pittoriche individuate sulla base di certi criteri.
Nel caso dei diagrammi CL l'utente può selezionare un sottodiagramma del diagramma corrente nei modi seguenti:

1. selezionando un sottoinsieme degli archi sulla base del tipo o del segno (cfr. la sezione 3.3),
2. selezionando un sottoinsieme dei nodi sulla base del tipo (cfr. la sezione 3.3),
3. selezionando nodi e archi di un anello segnato (cfr. la sezione 3.3).

Nel caso dei diagrammi FD l'utente può selezionare un sottodiagramma del diagramma corrente nei modi seguenti:

1. selezionando un sottoinsieme degli archi sulla base del tipo (cfr. la sezione 3.4),
2. selezionando un sottoinsieme dei nodi sulla base del tipo (cfr. la sezione 3.4).

Entrambi gli editor, infine, possiedono comandi per assicurare la persistenza delle strutture dati che l'utente crea mediante il loro utilizzo. Le strutture dati sono rappresentate dai dati relativi alle rappresentazioni pittoriche e dai corrispondenti grafi. I meccanismi per la loro memorizzazione all'interno di file sono di due tipi (per ulteriori dettagli vedi le sezioni 3.3 e 3.4):

1. il primo tipo prevede che le strutture dati siano memorizzate utilizzando le istanze delle classi usate per la loro definizione, ovvero sotto forma di *stream di oggetti*,

2. il secondo tipo prevede che le strutture dati siano memorizzate utilizzando stringhe di testo formattate e suddivise in campi, ovvero sotto forma di *stringhe di testo delimitato*.

Il formato testuale è stato introdotto in modo da consenire una esportabilità delle strutture dati prodotte dagli editor verso altri domini applicativi.

2.3 La visualizzazione di grandezze variabili nel tempo

2.3.1 Introduzione

Una grandezza variabile nel tempo può essere descritta in funzione del parametro tempo variabile con continuità oppure secondo multipli interi di un quanto di tempo T.
Nel primo caso, che diremo continuo, la grandezza viene di solito descritta mediante una espressione matematica esplicita più o meno complessa del tipo $y = f(t)$ mentre nel secondo caso, che diremo (per contrapposizione) discreto, la grandezza può essere descritta in due modi:

1. mediante una espressione matematica, come nel caso precedente,

2. mediante una tabella di valori.

Nel primo caso si usa una espressione del tipo $y = f(nT)$ mentre nel secondo caso la grandezza viene descritta da un insieme di coppie $(nT, f(nT))$ dove n varia in un sottoinsieme di N di solito della forma $[0, k_{max}]$ (vedi oltre).
Una volta stabilita la forma della grandezza da rappresentare prima di rappresentarla effettivamente devono essere operate della scelte sia in merito alla forma da dare all'asse dei tempi sia in merito a come rappresentare i valori della funzione sull'asse delle ordinate, nel caso di una rappresentazione in coordinate cartesiane.
Tali scelte sono complicate dalla necessità di rappresentare più grandezze sullo stesso piano cartesiano in modo da poter operare confronti diretti fra gli andamenti delle diverse grandezze.
Nel caso continuo, il tempo viene rappresentato sull'asse delle ascisse mentre i valori
$y = f(t)$
sono rappresentati sull'asse delle ordinate in modo da consentire la definizione di una curva continua.
Nel caso discreto, il tempo viene rappresentato sull'asse delle ascisse mentre i valori
$y(n) = f(nT)$
sono rappresentati sull'asse delle ordinate in modo da consentire la definizione

di un insieme di punti che descrivono l'andamento nel tempo della grandezza in oggetto, punti che possono essere rappresentati isolati oppure raccordati in vari modi.
La rappresentazione del tempo sull'asse delle ascisse può essere fatta in scala lineare oppure normalizzata mentre la rappresentazione della variabile dipendente sull'asse delle ordinate può essere fatta:

1. in scala lineare,
2. in scala logaritmica,
3. normalizzata.

Le varie modalità di rappresentazione saranno brevemente esaminate nei paragrafi che seguono.

2.3.2 La rappresentazione dell'asse dei tempi

La scala sull'asse dei tempi rappresenta la modalità di evoluzione del tempo dall'istante di inizio all'istante di fine dell'evoluzione. L'analisi dell'evoluzione di un sistema dinamico, infatti, prevede che, a partire da una condizione di equilibrio, il sistema venga sottoposto ad una perturbazione che dà luogo ad una evoluzione delle variabili caratteristiche del sistema a partire da valori in condizione di riposo. Tale evoluzione viene osservata per un certo periodo di tempo in modo che risultano definiti due istanti, detti di inizio e di fine dell'evoluzione.
Se l'insieme dei valori delle variabili all'istante 0 rappresenta lo *stato iniziale* del sistema, l'insieme dei valori delle variabili all'istante di fine dell'evoluzione rappresenta il suo *stato finale*, mentre l'insieme di tali valori in un istante intermedio è detto *stato corrente*.

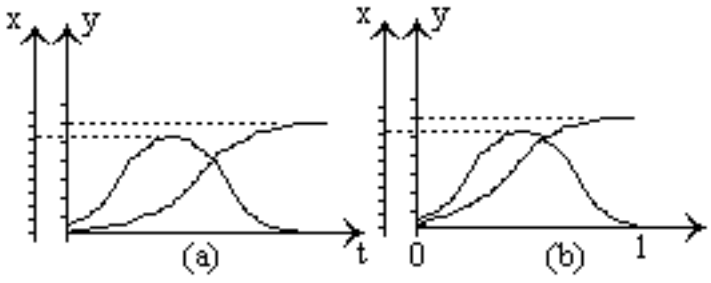

Figura 2.3: *Tempo assoluto e tempo normalizzato*

L'istante di inizio dell'osservazione è detto convenzionalmente istante 0 mentre per l'istante di fine osservazione si hanno tre possibilità, due delle quali sono illustrate nella figura 2.3.
L'istante di fine osservazione:

1. può assumere un valore arbitrario nel caso di *tempo assoluto continuo*,

2. assume il valore $k_{max}T$, dove k_{max} rappresenta il numero massimo di intervalli di tempo T nei quali si valutano le equazioni descrittive delle variabili del sistema prima di interrompere l'osservazione, nel caso di *tempo assoluto discretizzato*,

3. assume il valore 1 nel caso di *tempo normalizzato*.

La figura 2.3 illustra, infatti, gli andamenti sovrapposti di due variabili X e Y descrittive di due fenomeni aventi durate diverse e che non sono necessariamente fra loro in una relazione di causa-effetto.
La figura 2.3-(a) rappresenta gli andamenti sovrapposti delle due variabili, ciascun andamento con la propria durata, in una scala di tempo assoluto continuo[10] mentre la figura 2.3-(b) rappresenta gli andamenti delle due variabili sovrapposti con le durate normalizzate rispetto all'unità: nel primo caso l'asse dei tempi è soddiviso in multipli dell'unità di tempo scelta per la rappresentazione (secondi, minuti, ore e così via) mentre nel secondo caso l'asse dei tempi è suddiviso in frazioni dell'unità.
L'uso di una scala normalizzata dei tempi consente un agevole confronto di fenomeni aventi diverse durate ovvero aventi istanti di fine osservazione distinti e, pertanto, trova scarsa applicazione nell'analisi dei sistemi dinamici per i quali l'osservazione è pilotata da un solo orologio di riferimento. Vedremo nella sezione 2.3.3 come una tecnica analoga sia applicabile all'asse delle ordinate in modo da consentire un più agevole confronto di andamenti caratterizzati da diversi valori massimi e minimi.
La scala dell'asse dei tempi rappresenta, pertanto, sia la modalità di evoluzione sia la velocità di evoluzione delle variabili e quindi dello stato del sistema.

2.3.3 La rappresentazione sull'asse delle ordinate

L'asse delle ordinate può essere caratterizzato da scale di diversi tipi, in funzione dell'intervallo dei valori assunti da una variabile. Qualora una variabile assuma, ad esempio, valori compresi fra $-V_{min}$ e V_{max} (cfr. la figura 2.4(a)) è possibile fare uso

1. di una scala lineare,

2. di una scala lineare normalizzata.

Qualora, invece, una variabile assuma, ad esempio, valori compresi fra V_{min} e V_{max}, con $V_{min} > 1$, (cfr. la figura 2.4(b)) è possibile fare uso

1. di una scala lineare,

[10] Considerazioni simili valgono nel caso del tempo assoluto discretizzato.

2. di una scala logaritmica,

3. di una scala lineare normalizzata.

Nel caso si usi una scala lineare vengono rapresentati i valori effettivamente assunti dalla variabile mentre, qualora si faccia uso di una scala logaritmica o di una scala normalizzata, i valori assunti dalla variabile sono soggetti ad una trasformazione prima di venire rappresentati.
Nel primo caso viene calcolato il *logaritmo* in base e del valore della variabile mentre, nel secondo caso, l'intervallo dei valori assunti dalla variabile viene mappato, con una trasformazione lineare, sull'intervallo $[0, 1]$.
Sia che la variabile assuma valori compresi fra $-V_{min}$ e V_{max} sia che assuma valori compresi fra V_{min} e V_{max} al primo valore viene fatto corrispondere lo 0 della scala normalizzata mentre al secondo viene fatto corrispondere il valore 1 di tale scala mentre un valore y compreso fra gli estremi della variabile corrisponde al valore

$$x = \frac{|y|}{\Delta V} \tag{2.1}$$

nella scala normalizzata, in cui ΔV vale $V_{max} + V_{min}$ o $V_{max} - V_{min}$ a seconda dei casi.

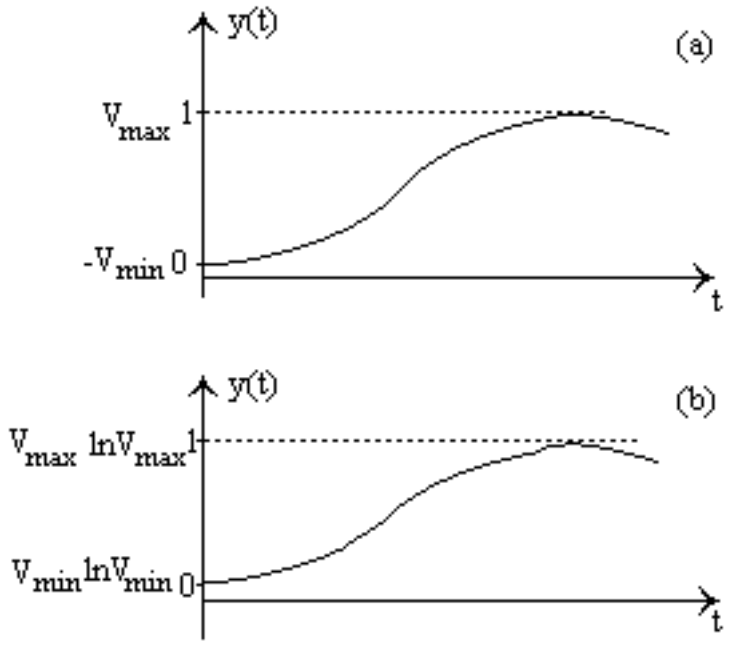

Figura 2.4: *Scala lineare e normalizzata (a), scala lineare, logaritmica e normalizzata (b)*

Nel caso si usi una scala logaritmica (cfr. la figura 2.4 (b)) gli estremi della scala

sono, pertanto, rappresentati dai valori $\ln V_{min}$ e $\ln V_{max}$ mentre, se si usa una scala normalizzata, gli estremi della scala sono rappresentati dai valori 0 e 1 (cfr. le figure 2.4 (a) e 2.4 (b)).
L'uso di una scala invece di un altra dipende da vari fattori fra i quali si ritengono degni di nota i seguenti:

1. l'intervallo dei valori da rappresentare,

2. la necessità di confrontare fra loro intervalli di valori molto diversi.

Qualora, infatti, i valori assunti da una variabile coprano un intervallo $[V_{min}, V_{max}]$ molto ampio, l'uso di una scala lineare può creare problemi di leggibilità della scala e del grafico a cui essa si riferisce. Nel caso in cui h sia l'altezza massima utile (ovvero utilizzabile per la visualizzazione dei grafici) di una finestra sullo schermo ad essa deve corrispondere, infatti, un intervallo di valori di ampiezza pari a $V_{min} - V_{max}$: a ciascun pixel corrisponde un intervallo pari a $(V_{min} - V_{max})/h$ ciò può dare luogo a grafici troppo compatti e, perciò, poco leggibili.
Una soluzione possibile è quella di ricorrere a multipli dell'unità di misura (ovvero, ad esempio, usare tonnellate invece di chili) in modo da ridurre l'intervallo di variazione della variabile di un fattore costante ma una soluzione più generale è quella di fare uso di una scala logaritmica.
In questo caso l'intervallo di variazione della variabile è $[\ln V_{min}, \ln V_{max}]$ e tale intervallo può essere agevolmente rappresentato nello spazio disponibile. Oltre a consentire la rappresentazione di intervalli molto ampi, la scala logaritmica (grazie all'andamento della funzione logaritmo) presenta il vantaggio di evidenziare le variazioni nell'intorno dell'origine rispetto alle altre.
Dovendo, infine, rappresentare variabili che hanno intervalli di variazione molto diversi in modo da confrontare fra loro gli andamenti, ricorrendo (anche) a rappresentazioni sovrapposte (cfr. la sezione 2.3.4 e le figure 2.3 e 2.7) una soluzione è quella di fare uso di una scala normalizzata. In tale modo, indipendentemente dall'intervallo di variazione reale, le variabili assumono valori compresi nell'intervallo chiuso $[0, 1]$ e ciò ne facilita il confronto.
Una volta fissata la scala con cui rappresentare i valori di una variabile $y(nT)$[11] è necessario stabilire il modo in cui i valori sono rappresentati sul piano cartesiano. Le modalità disponibili sono illustrate nella figura 2.5.
Nel caso della *rappresentazione puntiforme* viene rappresentato un marker ad hoc in corrispondenza di ognuna delle coppie $(nT, y(nT))$ ovvero nessuna ipotesi viene fatta sull'evoluzione della variabile fra tali punti.
Nel caso della *interpolazione lineare* i punti $(nT, y(nT))$ e $((n+1)T, y((n+1)T))$ sono raccordati da segmenti di retta. In questo caso si ipotizza una evoluzione lineare della variabile fra due istanti di osservazione successivi.

[11] Nel caso in cui la variabile sia descritta da una relazione del tipo $y(t)$ (caso continuo) si definisce una curva continua ovvero una curva definita $\forall t \in [t_0, t_{max}]$.

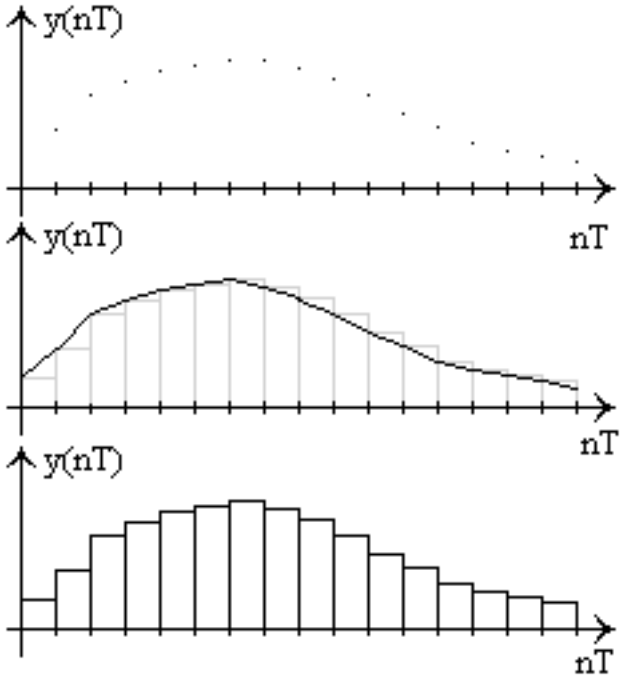

Figura 2.5: *Rappresentazione puntiforme, con interpolazione lineare, a gradini*

Nel caso della *rappresentazione a gradini* si ipotizza che sull'intervalo $[nT, (n+1)T]$ la variabile mantenga un valore costante e pari a quello assunto nel punto nT.

2.3.4 Rappresentazioni sovrapposte

Come illustrato dalle figure 2.3, 2.6 e 2.7, in molti casi è necessario rappresentare gli andamenti di più variabili su una stessa superfice di rappresentazione in modo da confrontarli sia da un punto di vista qualitativo sia da un punto di vista quantitativo.

Per poter rappresentare più andamenti su una stessa superfice di rappresentazione è necessario definire gli assi cartesiani in modo da rendere significativa la rappresentazione.

La figura 2.6 illustra una possibile soluzione ovvero il caso di due variabili rappresentate usando lo stesso riferimento temporale ma ciascuna con il proprio asse delle ordinate su cui è rappresentato l'intervallo di variazione della variabile suddiviso negli opportuni sottomultipli.

La figura 2.7 illustra l'altra soluzione ovvero il caso di due variabili rappresentate usando lo stesso riferimento temporale e una scala normalizzata delle ordinate[12].

[12]Nella figura 2.7 la scala delle ordinate è suddivisa in valori percentuali invece che in frazioni dell'unità ma il fatto è ininfluente.

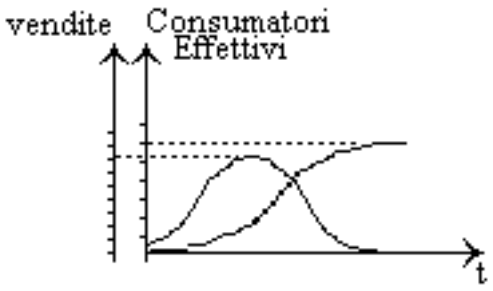

Figura 2.6: *Rappresentazioni sovrapposte, scale lineari (o logaritmiche)*

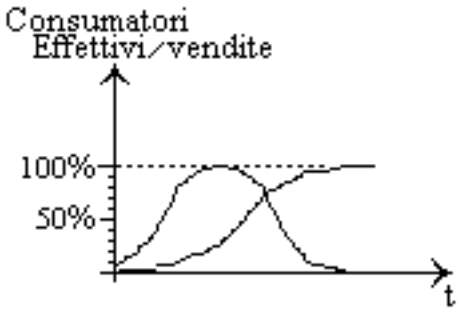

Figura 2.7: *Rappresentazioni sovrapposte, scale normalizzate*

Nel secondo caso quali che siano gli intervalli di variazione delle singole variabili è agevole eseguire fra queste dei confronti quantitativi. Nel caso della figura 2.7 è possibile, infatti, affermare che all'istante n_0T la variabile *ConsumatoriEffettivi* ha raggiunto e.g. il 30% del suo valore massimo mentre l'altra variabile *vendite* ha raggiunto e.g. il 10% del suo valore massimo.
Nel caso di rappresentazioni sovrapposte, allo scopo di aumentare la leggibilità dei grafici si possono inoltre usare sia colori distinti per i diversi grafici e per la superfice di visualizzazione sia stili diversi per il tracciamento di ciascun grafico. Per ulteriori dettagli vedi la sezione 2.3.6.

2.3.5 Le unità di misura

Le grandezze di cui si vuole visualizzare l'andamento nel tempo solo raramente sono adimensionali mentre, nella maggior parte dei casi, sono caratterizzate da ben precise unità di misura.
Nel nostro caso, ad esempio, si hanno grandezze dei seguenti tipi: *livelli*, *flussi*, *variabili ausiliarie* e *variabili esogene*.

Le prime sono caratterizzate da unità di misura del tipo $[kg]$, $[l]$, ovvero da unità di misura in cui non compare l'unità di misura $[tempo]$ a denominatore dell'espressione dimensionale.
Le seconde sono caratterizzate dalla presenza della unità di misura $[tempo]$ a denominatore delle espressioni dimensionali per cui si hanno espressioni, assimilabili ad una velocità, quali $[kg/sec]$ o $[l/ora]$. Le *variabili ausiliarie* e *variabili esogene* sono riconducibili ai due casi precedenti. Le unità di misura possono rendere ardua la visualizzazione sovrapposta di più grandezze sullo stesso riferimento cartesiano essenzialmente per due motivi:

1. estetici,
2. dimensionali.

I motivi estetici sono legati alle differenze fra gli intervalli di variazione di due o più grandezze che si vogliono rappresentare sovrapposte. Data, ad esempio, una grandezza A che varia da 0 ad un valore A_{max} ed una grandezza B che varia fra due valori B_{min} e B_{max}, nei casi seguenti:

1. $A_{max} << B_{min}$
2. $B_{min} \leq A_{max} << B_{max}$

la visualizzazione sovrapposta può avere poco significato.
I motivi estetici sono legati alle differenze fra le unità di misura di grandezze che si vogliono rappresentare sovrapposte e che non sono fra di loro commensurabili. Una soluzione che si può adottare è quella di ricorrere ad una scala normalizzata in modo da riportare tutti gli intervalli di variazione all'intervallo adimensionale $[0, 1]$. In questo modo i problemi di natura *estetica* e *dimensionale* sono evidentemente risolti.
Un altro problema può sorgere in tutti i casi in cui le grandezze da visualizzare si riferiscono a fenomeni che hanno durate anche molto diverse. In tali casi è come se si avessero più variabili tempo indipendenti. Volendo confrontare gli andamenti di tali grandezze è possibile utilizzare una scala normalizzata dei tempi in modo da rappresentare gli andamenti in funzione delle loro durate percentuali. In tal modo tutti gli andamenti hanno la stessa "ampiezza temporale" e possono essere facilmente confrontati.
Nel nostro caso non si è fatto ricorso ad una scala normalizzata dal momento che tutte le variabili in gioco evolvono con la stessa temporizzazione da un istante 0 iniziale ad uno finale dato da $N_{max} \times T$ in cui T rappresenta la distanza fra due osservazioni successive.

2.3.6 Soluzioni implementative

L'ambiente $D(a)ySyTool\,Box$ è caratterizzato da un modulo, detto *Display*, per la visualizzazione degli andamenti nel tempo delle variabili caratteristiche dei

diagrammi FD.
Tale modulo, che sarà descritto più in dettaglio nei Capitoli 4 e 5, consente all'utente di visualizzare un certo numero di grafici su piani cartesiani disgiunti o sovrapposti.
Per la visualizzazione dei grafici il modulo *Display* definisce un set di *settaggi globali*, validi per tutte le superfici di visualizzazione e per tutti i grafici, settaggi che possono essere specializzati

1. per l'inseme dei grafici rappresentati su una superfice di visualizzazione,
2. per il singolo grafico

secondo modalità che saranno illustrate in dettaglio nella sezione 3.5.
I settaggi relativi alla singola superfice di visualizzazione sono detti *settaggi locali* mentre quelli relativi al singolo grafico sono detti *settaggi puntuali*.
I settaggi permettono di definire:

1. *la scala dell'asse dei tempi*,
2. *la scala dell'asse delle ordinate*,
3. *la modalità di rappresentazione*,
4. *il colore*,
5. *lo stile di tracciamento*.

La scala dell'asse dei tempi prevede l'uso di un tempo assoluto (cfr. la sezione 2.3.2) continuo (ovvero il tempo può assumere valori t in un sottoinsieme di $\Re$, $t \in [t_0, t_{max}]$) oppure discretizzato (ovvero il tempo può assumere valori nT con $n \in [N_0, N_{max}]$).
La scala dell'asse delle ordinate può essere di tipo *lineare*, *logaritmico* o *normalizzato* (cfr. la sezione 2.3.3) mentre la modalità di rappresentazione stabilisce se i grafici sono rappresentati su una stessa superfice di visualizzazione (si parla di *rappresentazioni sovrapposte*) o sono rappresentati su superfici di visualizzazione distinte (si parla di *rappresentazioni disgiunte*).
Nel caso di rappresentazioni sovrapposte l'uso del colore facilita la leggibilità dei grafici che possono essere tracciati usando le tre modalità descritte nella sezione 2.3.3: rappresentazione puntiforme, con interpolazione lineare e a gradini.
I *settaggi globali* definiscono gli elementi 1-5, i *settaggi locali* permettono di modificare gli elementi 1, 2, 3 e 5 mentre i *settaggi puntuali* consentono di modificare per ciascun grafico solo gli elementi 2 e 4. Per maggiori dettagli si rimanda alla sezione 3.5.

Capitolo 3

D(a)ySy ToolBox : la struttura astratta

3.1 Introduzione

Il presente Capitolo contiene una breve descrizione "lato utente" dei singoli Tool che compongono l'ambiente *D(a)ySy Tool Box*. Di ciascun Tool (cfr. la Tabella 3.1) viene descritta l'*interfaccia grafica*, vengono presentati i vari comandi e gli *elementi ausiliari* quali *finestre di dialogo* e *finestre "secondarie"*. Alla descrizione della *struttura interna* dei singoli Tool è dedicato il Capitolo 4 mentre il Capitolo 5 contiene accenni alla semantica dei vari Tool.
L'ambiente *D(a)ySy Tool Box* consente la simulazione del comportamento dei Sistemi Dinamici mettendo a disposizione dell'utente un certo numero di Tool autonomi, cioè utilizzabili uno indipendentemente dagli altri, ma interagenti dal momento che esiste un Tool ad hoc, detto *TopLevel*, che ne consente una gestione integrata.
I Tool che fanno parte dell'ambiente *D(a)ySy ToolBox* sono elencati nella Tabella 3.1.

Nella tabella 3.1 è riportato l'elenco dei vari Tool (di ciascuno dei quali viene specificata la *funzionalità principale* e vengono indicati quali sono gli altri Tool con cui interagisce più o meno direttamente (campo *Comunicazione*)) mentre la figura 3.1 rappresenta uno dei possibili flussi logici delle operazioni fra i vari Tool dell'ambiente *D(a)ySy Tool Box*.
I punti di partenza possibili sono, infatti, sia i diagrammi CL (creati e modificati mediante il Tool *Causal Loop Graphic Editor*, cfr. la sezione 3.3) sia i diagrammi FD (creati e modificati mediante il Tool *Flow Diagram Graphic Editor*, cfr. la sezione 3.4) : nel primo caso, prima di poter scrivere le equazioni descrittive del modello, è necessario operare una conversione da un diagramma CL al corrispondente diagramma FD.
Per poter operare la conversione è necessario arricchire gli elementi dei diagrammi

Nome del Tool	*Funzionalità principale*	*Comunicazione*
TopLevel	gestione multi-threaded delle altre applicazioni	CLGE, FDGE e convertitori
Causal Loop Graphic Editor (CLGE)	editing di diagrammi Causal Loop	FDGE via CLtoFD
Flow Diagram Graphic Editor (FDGE)	editing di diagrammi Flow Diagram	CLGE via FDtoCL, Resolver,Display
Resolver	risoluzione di equazioni	FDGE
Display	visualizzazione di grafici	FDGE, Resolver
CLtoFD	conversione da CL a FD	FDGE
FDtoCL	conversione da FD a CL	CLGE

Tabella 3.1: *Elenco dei Tool*

Figura 3.1: *Possibile flusso logico delle operazioni nell'ambiente D(a)ySy Tool Box*

CL di informazioni, inutili nel caso dei diagrammi CL, relative al tipo sia delle variabili sia delle relazioni fra di queste (cfr. le sezioni 3.4 e 3.7).
Utilizzando il Tool *Flow Diagram Graphic Editor* è possibile definire un insieme di equazioni che danno le relazioni fra le varie variabili del modello.
Una volta scritte le equazioni è necessario assegnare i necessari valori iniziali a tutte le variabili in esse presenti prima di risolverle (ovvero eseguire la *simulazione*) a cui può seguire la visualizzazione dei risultati.
La figura 3.2 stabilisce le associazioni fra le operazioni della figura 3.1 e i Tool che costituiscono l'ambiente *D(a)ySy Tool Box*.
Tale figura, inoltre, stabilisce un legame bidirezionale fra i due editor nel senso che indica come sia possibile definire un diagramma CL da cui estrarre poi il corrispondente diagramma FD ma come sia altresì possibile partire dalla defini-

Figura 3.2: *Relazioni logiche fra operazioni e moduli dell'ambiente D(a)ySy Tool Box*

zione di un diagramma FD da cui estrarre il corrispondente diagramma CL. A tale scopo il *Causal Loop Graphic Editor* interagisce con il convertitore $CLtoFD$ mentre il *Flow Diagram Graphic Editor* interagisce con il convertitore $FDtoCL$ (cfr. la Tabella 3.1).

3.2 *TopLevel*

Il Tool *TopLevel* svolge il compito di Tool di coordinamento, dal momento che consente all'utente di accedere a tutti gli altri Tool dell'ambiente nella loro versione "slave", sebbene ciascuno caratterizzato da una propria *interfaccia grafica* (cfr. oltre). Si fa notare, infatti, come tutti i Tool siano eseguibili indipendentemente da *TopLevel* come Tool "stand alone" (cfr. le singole sezioni dei Tool).
La figura 3.3 presenta l'interfaccia del Tool *TopLevel* attraverso la quale l'utente può accedere a tutti gli altri Tool dell'ambiente *D(a)ySy Tool Box* mentre la Tabella 3.2 elenca gli elementi (pulsanti e voci di menù, ad esclusione del menù $Pop-up$, il cui utilizzo sarà esaminato a breve) di tale interfaccia, di ciascuno dei quali la tabella contiene una breve descrizione. I due menù di *TopLevel* (*Application* e *Convert*) permettono, rispettivamente:

1. di creare una istanza del Tool *Causal Loop Graphic Editor* oppure del Tool *Flow Diagram Graphic Editor*,

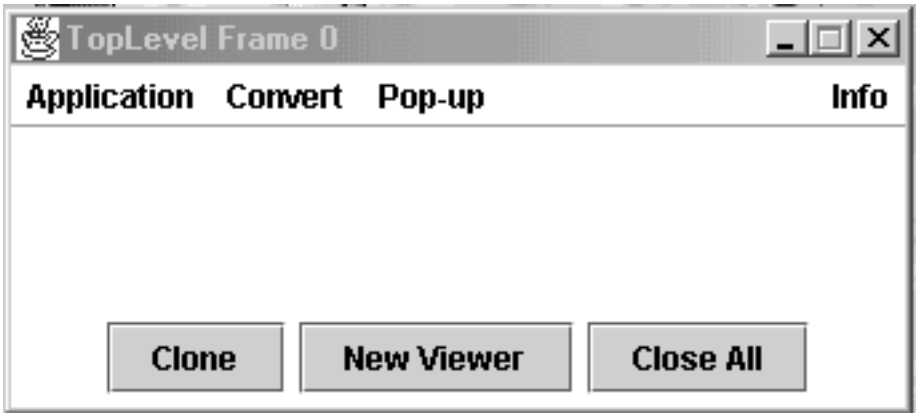

Figura 3.3: *TopLevel*

Elemento	*Tipo*	*Contenitore*	*Funzione*
Open CL editor	Voce di menù	Menù *Application*	istanzia CLGE
Open FD editor	Voce di menù	Menù *Application*	istanzia FDGE
CL to FD	Voce di menù	Menù *Convert*	istanzia CLtoFD
FD to CL	Voce di menù	Menù *Convert*	istanzia FDtoCL
Clone	pulsante	*frame*	crea un clone di *TopLevel*
New Viewer	pulsante	*frame*	crea un *viewer*
Close All	pulsante	*frame*	chiude tutti i *viewer* di un clone

Tabella 3.2: *Elementi dell'interfaccia di TopLevel*

2. di creare una istanza del convertitore da diagrammi CL a diagrammi FD (CltoFD) oppure una istanza del convertitore da diagrammi FD a diagrammi CL (FDtoCL).

Le istanze di tali Tool così create sono eseguite ciascuna in un *thread* separato e l'unica cosa che le distingue dalla loro versione "stand alone" è che la loro terminazione si traduce nella terminazione del *thread* al cui interno sono eseguite ma non ha nessun effetto sulla esecuzione del processo *TopLevel*. Le versioni "stand alone" sono applicazioni autocontenute ed autonome la cui terminazione causa la terminazione del relativo processo.
I pulsanti *Clone*, *New Viewer* e *Close All*, presenti in basso sulla finestra dell'interfaccia (finestra indicata con il termine di *frame* nella tabella 3.2), consentono rispettivamente:

1. di clonare il frame principale creando una nuova istanza di *TopLevel*,

2. di creare un nuovo frame per la visualizzazione di rappresentazioni pittoriche (detto *Viewer*, cfr. la figura 3.4),

3. di chiudere tutti i *frame* aperti a seguito di successive pressioni del pulsante *New Viewer*.

Il codice delle classi che compongono *TopLevel* contiene dei controlli interni che impediscono all'utente di creare più di due copie del *frame* principale e, per ciascun frame, di creare più di cinque *Viewer*: i frame principali sono numerati 0, 1 e 2 e vengono creati in posizioni fisse sullo schermo (cfr. la figura 3.4) mentre i singoli *Viewer* vengono creati al di sotto del *frame* principale corrispondente e sono etichettati come *Viewer Window m.n*, dove m individua il *frame* e n può assumere un valore compreso fra 1 e 5 ed individua il singolo *Viewer*.
Il menù $Pop-up$ (cfr. la figura 3.4) viene inizializzato dinamicamente e contiene gli identificativi dei *Viewer* creati mediante il pulsante *New Viewer* e associati ad uno dei *frame*. Il suo scopo è quello di consentire all'utente di portare in primo piano il *Viewer* desiderato.
Ogni *Viewer*, infine, rappresenta un *frame ausiliario* di visualizzazione della rappresentazione pittorica di un grafo. A tale scopo è caratterizzato da un insieme molto semplice di comandi raggruppati in un menù con due sole voci (a parte la voce *Info*[1]):

1. *File*,

2. *Execution*.

[1]La voce *Info*, comune a tutti i Tool, consente di accedere a due finestre ausiliarie, una con informazioni relative al prodotto e al suo autore e l'altra destinata, in futuro (crf. il Capitolo 6), a contenere un *help in linea*. Tali finestre non saranno descritte nè in questo capitolo nè nei successivi Capitoli 4 e 5.

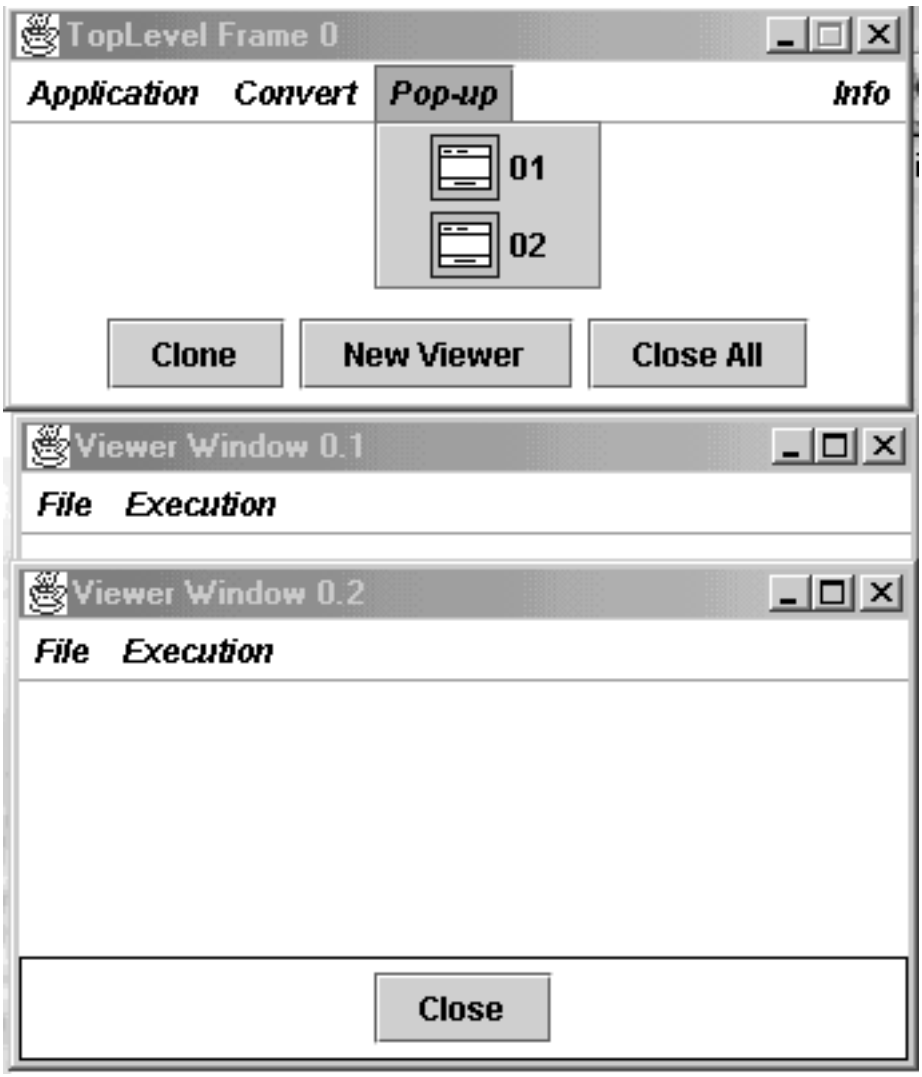

Figura 3.4: *TopLevel e due Viewer*

Lo scopo dei *Viewer* è, essenzialmente, quello di rendere il Tool *TopLevel* in una certa misura "autocontenuto": se l'utente intende lavorare su diagrammi CL o FD già inizializzati[2], senza apportare a questi nessuna modifica, allora può limitarsi ad utilizzare i *Viewer*, uno per diagramma, fino ad un massimo di 15 contemporaneamente attivi. All'interno di ciascun *Viewer* l'utente può visualizzare un diagramma CL o un diagramma FD.
La voce *File* contiene allo scopo due voci:

1. Open in viewer...
2. Import in viewer...

Tali voci consentono all'utente di accedere alle descrizione di diagrammi CL o FD contenute, rispettivamente, in file in formato oggetto o in formato testo e che coincidono con i comandi *Open* e *Import* di cui alle sezioni 3.3.4 e 3.4.2 cui si rimamnda.
La voce *Execution*, infine, qualora l'utente visualizzi nel viewer un diagramma FD consente una interazione diretta con il Tool *Resolver* per cui si rimanda alla sezione 3.6.

3.3 *Causal Loop Graphic Editor*

3.3.1 Introduzione

L'Editor Grafico per i diagrammi CL (il Tool *Causal Loop Graphic Editor*) si presenta all'utente con un'interfaccia (mostrata nella figura 3.5) caratterizzata da un menù (in alto nella figura 3.5) e da due superfici vuote, una bianca e una grigia (parzialmente coperte dalle voci del menù nella figura 3.5).
La superfice bianca è destinata a contenere un certo numero di pulsanti che possono apparire o scomparire in funzione dello stato interno dell'editor. I pulsanti sono caratterizzati da icone dal significato intuitivo, significato che sarà spiegato a breve.
La superfice grigia viene occupata (su richiesta dell'utente, cfr. oltre) dal *canvas* sul quale l'utente può tracciare e/o modificare i *diagrammi CL* e sul quale compaiono alcuni menù flottanti che consentono una manipolazione diretta degli oggetti su di esso presenti.
Il menù è di tipo contestuale dal momento che le sue voci sono abilitate (voci in colore *nero* in figura 3.5) oppure disabilitate (voci in colore *grigio*, dette *dimmed*, in figura 3.5) in funzione dello stato interno dell'editor.
Nel caso di figura 3.5 l'editor è nello stato iniziale per cui all'utente sono accessibili solo le voci che consentono di:

[2]Un diagramma si dice inizializzato se tutte le grandezze tipiche dei nodi e degli elementi di connessione/archi sono state inizializzate correttamente. Per maggiori dettagli nei diversi casi si rimanda alle sezioni 3.3 e 3.4.

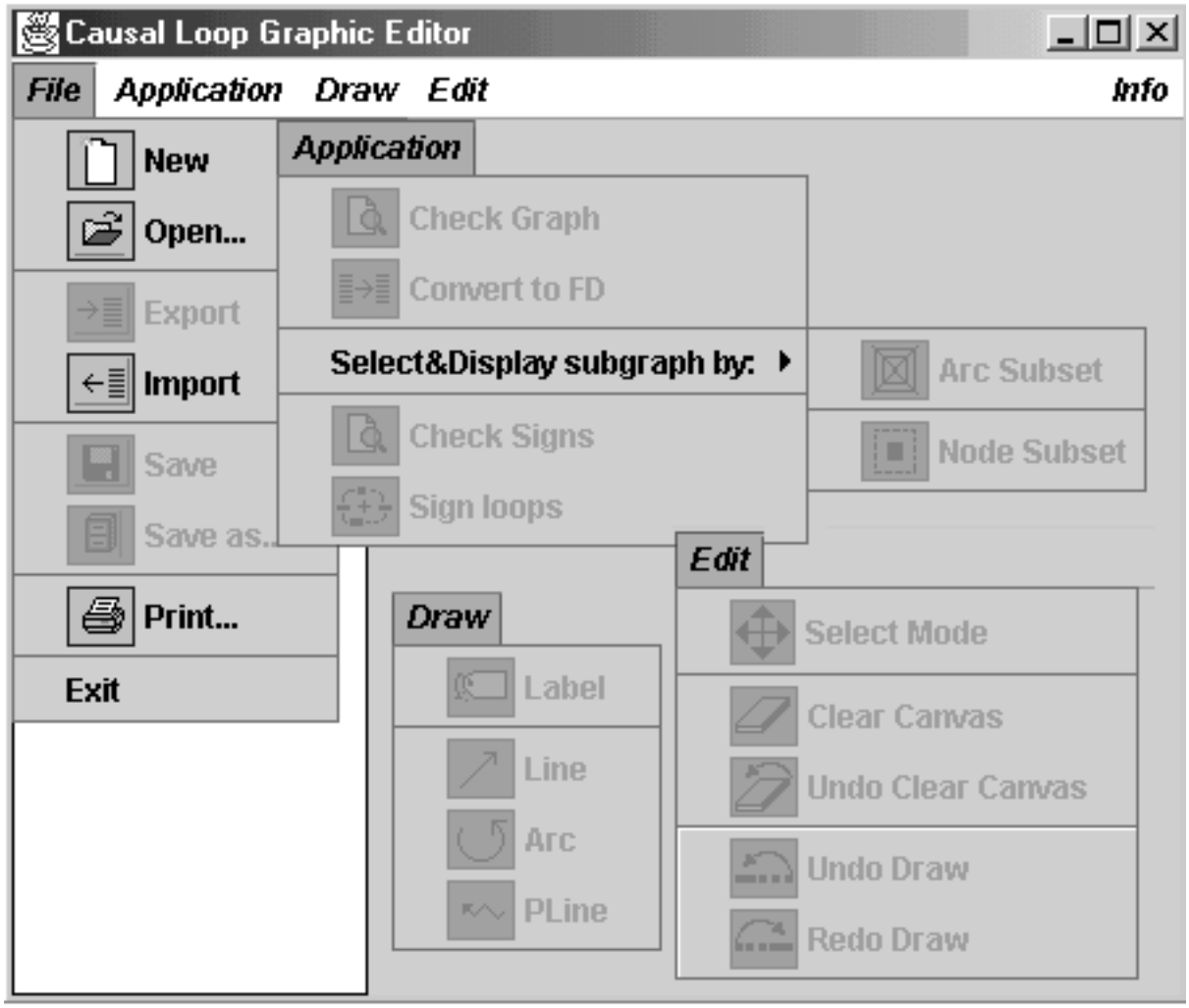

Figura 3.5: *Il "Causal Loop Graphic Editor"*

1. creare un nuovo diagramma con il comando *New*,
2. caricare un diagramma esistente con il comando *Open..* o con il comando *Import..*,
3. stampare un file contenente la descrizione di un diagramma CL con il comando *Print...*,
4. terminare il processo originato dalla esecuzione del Tool (caso di esecuzione "stand alone") oppure il thread al cui interno è in esecuzione il Tool (caso "slave") con il comando *Exit*.

Il passaggio dallo stato iniziale agli altri stati in cui può trovarsi l'editor è illustrato in dettaglio nella sezione 3.3.2.

3.3.2 I "macro stati" dell'editor

Per comprendere il funzionamento dell'editor si introduce il concetto di "macro stato". Un "macro stato" è uno stato che 'maschera' un certo numero di stati nel senso che due stati s_i e s_j appartengono allo stesso "macro stato" se sono indistinguibili in base ad un criterio dato. Ad esempio (cfr. oltre) tutte le situazioni in cui sul canvas è presente la rappresentazione di un solo nodo (ovvero una sola etichetta) sono riconducibili ad un unico macro stato (indipendentemente dal valore e dalla posizione del'etichetta) e lo stesso vale per tutte le situazioni in cui sul canvas sono rappresentati più di due nodi (ovvero sono presenti più di due etichette), indipendentemente dal numero effettivo di queste, dalle loro posizioni e dai loro valori.
I "macro stati"[3] del *causal Loop Graphic Editor* sono definiti dall'insieme dei *comandi* accessibili all'utente in ciascuno di essi. Ad ogni "macro stato" corrispondono, infatti, gli stessi comandi e più configurazioni del diagramma rappresentato sul *canvas* che sono, pertanto, indistinguibili.

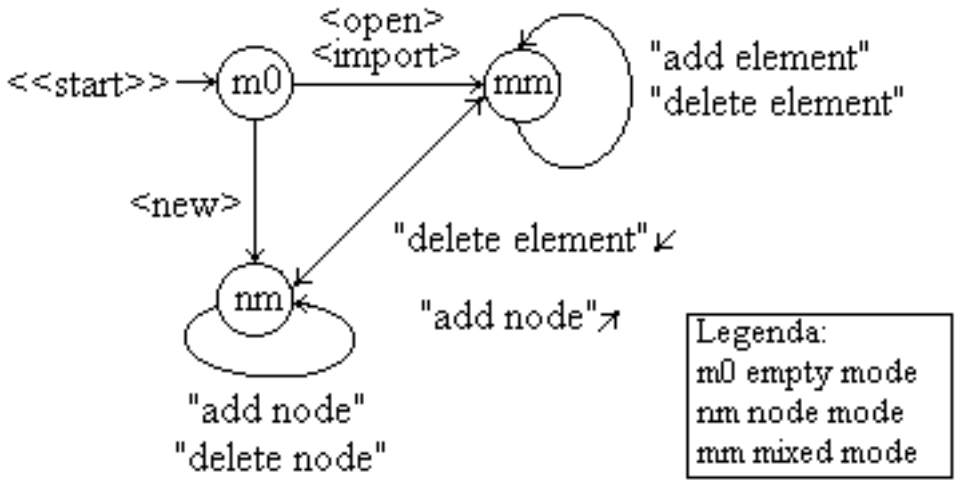

Figura 3.6: *Alcuni dei "macro stati" del Causal Loop Graphic Editor*

I "macro stati" significativi dell'editor sono illustrati in figura 3.6. In tale figura i "macro stati" sono rappresentati sotto forma di stringhe racchiuse in cerchi mentre le transizioni fra "macro stati" sono rappresentate da archi orientati con *guardia*. Ogni arco può avere una o più guardie ed ogni guardia rappresenta un comando che causa la transizione fra un "macro stato" e un altro. I comandi sono di tre tipi:

1. esterni: $<< start >>$ che manda in esecuzione il Tool;

[3]Nel seguito useremo la notazione ms_i per indicare un generico "macro stato" cui corrispondono gli stati $\{s_{i,j} \mid j \in \mathcal{N}\}$.

2. primitivi: $< new >$, $< open >$ e $< import >$ cui corrispondono voci del menù del frame principale;

3. compositi: "add node", "delete node", "add element" e "delete element" cui corrispondono azioni complesse dell'utente.

Il "macro stato" ms_0 (indicato come $m0$ o "empty mode" in figura 3.6) è quello iniziale del Tool nel quale l'interfaccia ha l'aspetto illustrato dalla figura 3.5. In tale stato né il *canvas* né i pulsanti sono presenti per cui l'utente può soltanto utilizzare i comandi abilitati (le voci in nero del menù *File* della figura 3.5) per:

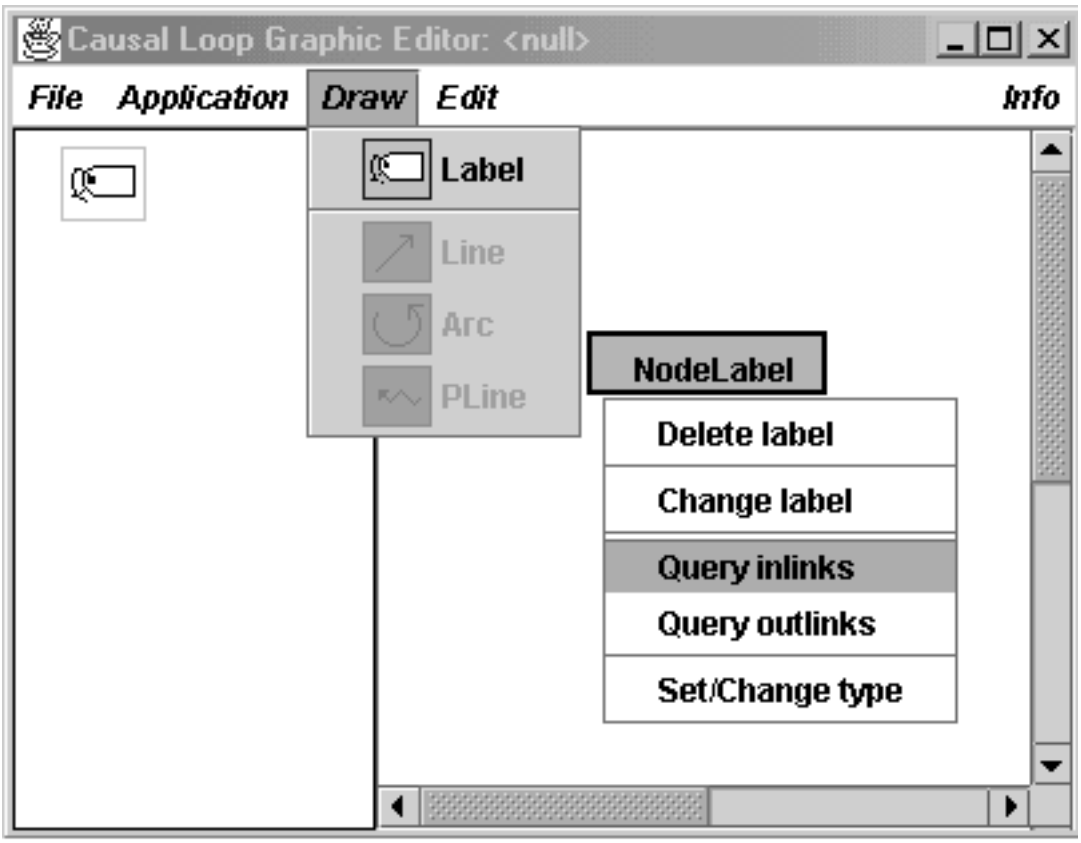

Figura 3.7: *Configurazione dell'interfaccia del Causal Loop Graphic Editor dopo una* $< new >$

1. creare un nuovo diagramma CL assegnandogli un nome (facoltativo) con il comando *New* (cui corrisponde la transazione etichettata come $< new >$ in figura 3.6),

2. caricare un diagramma CL esistente la cui struttura è contenuta in un *file di oggetti* (vedi oltre) con il comando *Open...* (cui corrisponde la transazione etichettata come $< open >$ in figura 3.6),

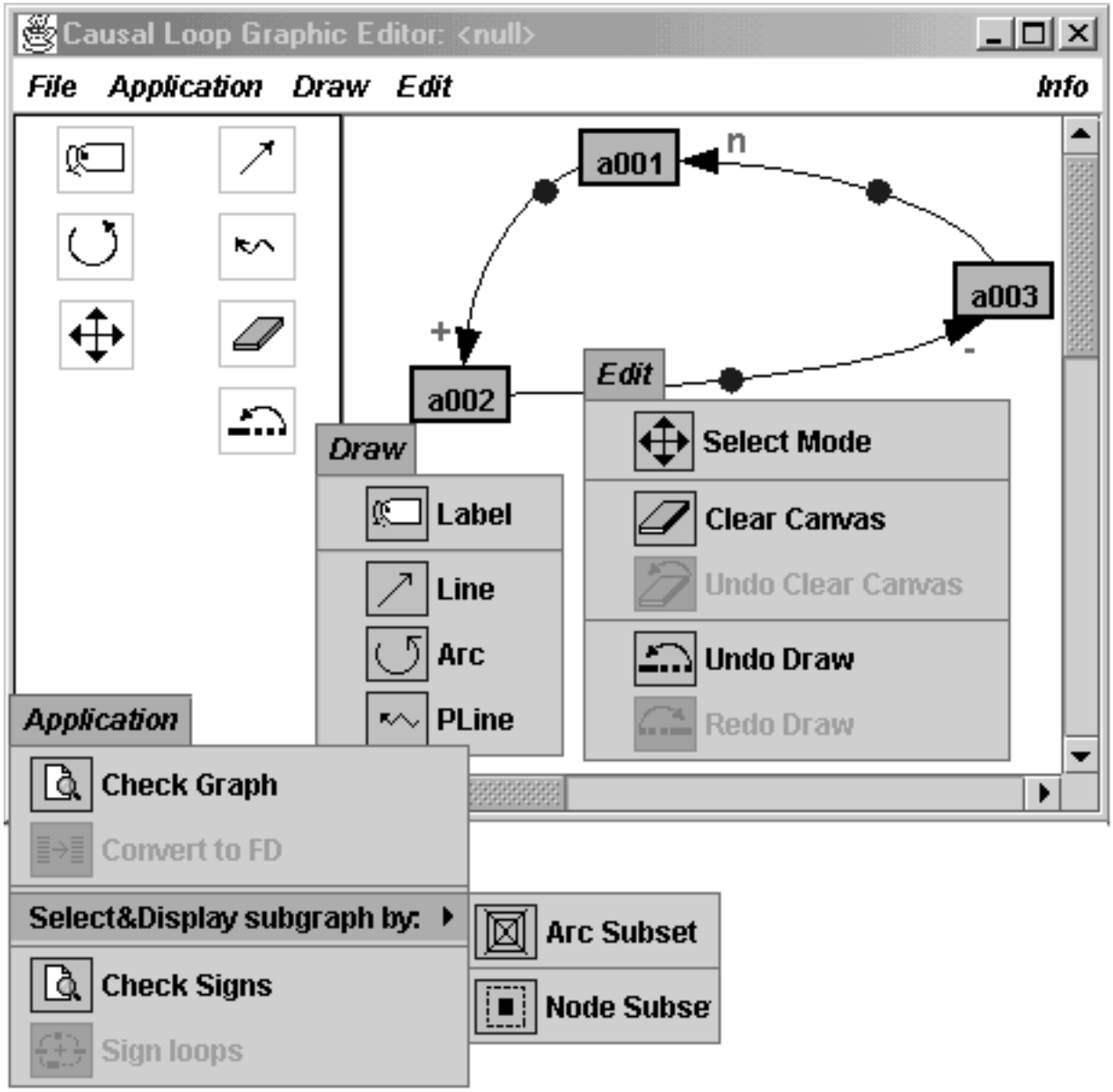

Figura 3.8: *Configurazione dell'interfaccia del Causal Loop Graphic Editor dopo una* $<$ *open* $>$ *o una* $<$ *import* $>$

3. caricare un diagramma CL esistente la cui struttura è contenuta in un *file di testo* (vedi oltre) con il comando *Import...*[4] (cui corrisponde la transazione etichettata come $<$ *import* $>$ in figura 3.6).

[4]Seguendo una prassi ormai consolidata la presenza di ... a fianco del nome di un comando preannuncia, nel caso il comando sia selezionato, l'apertura di una o più finestre di dialogo.

Il "macro stato" ms_1 (indicato come nm o "node mode" in figura 3.6) successivo ad una $< new >$ è caratterizzato dalla presenza del *canvas* e dalla abilitazione del *pulsante* per la *creazione* di etichette e dei soli comandi relativi alla *creazione*, alla *rimozione*, al *posizionamento* e alla *customizzazione* di *etichette* sul *canvas* (cfr. la figura 3.7).
La *creazione* di una *etichetta* avviene in due tempi: selezione del *pulsante* oppure del *comando Label* del menù *Draw* e individuazione (con una pressione del tasto sinistro del mouse[5]) di un punto del *canvas* in cui posizionare l'etichetta. Al momento della creazione tutte le etichette sono caratterizzate dalla stringa $NodeLabel$.
La *rimozione* può avvenire o a seguito di un doppio click con il pulsante sinistro del mouse e con il cursore posizionato sull'etichetta da rimuovere oppure selezionando il comando "*Delete label*" del menù flottante (cfr. 3.3.5) associato all'etichetta (cfr. la figura 3.7).
Il *posizionamento* di una etichetta prevede la selezione dell'etichetta e il suo trascinamento nella posizione finale con le usuali tecniche di *drag&drop*.
La *customizzazione* può essere effettuata per ciascuna etichetta utilizzando il menù flottante (cfr. 3.3.5) associato all'etichetta (cfr. la figura 3.7). Mediante tale menù è possibile modificare il valore della stringa associata all'etichetta ("*Change label*"), modificarne il tipo ("*Set/Change type*", cfr. 3.3.5) e interrogare sia la *stella uscente* sia la *stella entrante* del nodo (con i comandi *Query outlinks* e *Query inlinks* del menù flottante associato all'etichetta, cfr. la figura 3.7).
Nel "macro stato" nm l'utente può aggiungere e rimuovere etichette. Nel momento in cui il *canvas* contiene almeno due etichette l'applicazione si porta, in modo del tutto trasparente, nel "macro stato" ms_2.
Il "macro stato" ms_2 (indicato come mm o "mixed mode" in figura 3.6) successivo ad una $< open >$ o ad una $< import >$ è caratterizzato dalla presenza sul *canvas* di un diagramma CL composto da almeno due etichette e dalla abilitazione dei comandi e dei pulsanti di cui alla figura 3.8.
Nel "macro stato" ms_2, l'utente ha a disposizione *comandi* (mediante *pulsanti*, cfr. la sezione 3.3.3, *voci di menù*, cfr. la sezione 3.3.4, o *menù flottanti*, cfr. la sezione 3.3.5) per agire sia sulle *etichette* sia sugli *elementi di connessione* (cfr. le "add element" e "delete element" della figura 3.6).
Utilizzando tali comandi l'utente definisce una rappresentazione pittorica cui l'editor associa automaticamente ed in tempo reale un grafo.
In tutti i "macro stati" l'utente ha a disposizione comandi del menù file per la creazione di un nuovo diagramma (New) oppure per il caricamento di un diagramma esistente ($Open...$ o $Import...$)o la sua stampa ($Print...$) mentre altri

[5]Nel seguito useremo le dizioni *tasto sinistro* e *tasto destro* per indicare o il tasto sinistro/destro (nel caso di mouse a tre tasti) o l'unico tasto con o senza tasto modificatore (il tasto $Ctrl$) nel caso di mouse ad un solo tasto.

comandi (quali *Save*, *Save as...* e *Export* sono disponibili solo nei "macro stati" ms_1 e ms_2.
Oltre ai "macro stati" di figura 3.6, l'editor può trovarsi in altri "macro stati" di cui si parlerà nelle sezioni 3.3.3 e 3.3.4. Di tali "macro stati" sono significativi i seguenti:

1. *clearedMode*,
2. *undoDrawMode*

L'editor si porta nel "macro stato" *clearedMode* dopo che l'utente ha eseguito una cancellazione del contenuto del *canvas*, cancellazione che può essere annullata con un comando di *undo*, come vedremo nelle sezioni 3.3.3 e 3.3.4, oppure resa permanente posizionando sul *canvas* una nuova *etichetta*.
L'editor, invece, si porta nel "macro stato" *undoDrawMode* dopo che l'utente ha eseguito la rimozione dal *canvas* di un *elemento di connessione* o di una *etichetta*, rimozione che può essere annullata con un comando di *undo*, come vedremo nelle sezioni 3.3.3 e 3.3.4, oppure resa permanente posizionando sul *canvas* una nuova *etichetta* o un nuovo *elemento di connessione*.

3.3.3 I *pulsanti*

Sulla sinistra del *canvas* l'editor, in funzione del "macro stato" corrente, visualizza un *panel* contenente un certo numero di pulsanti. La figura 3.9 li presenta tutti contemporaneamente attivi, situazione che in realtà non si verifica mai.
Da sinistra a destra e dall'alto in basso si hanno i seguenti pulsanti (a ciascuno dei quali corrisponde un comando o del menù *Draw* o del menù *Edit*, cfr. la sezione 3.3.4):

1. *Draw label*
2. *Draw line*
3. *Draw arc*
4. *Draw Pline*
5. *Select mode*
6. *Clear Canvas*
7. *Undo Clear Canvas*
8. *Undo Draw*
9. *Redo Draw*

Figura 3.9: *L'insieme dei pulsanti*

I pulsanti suddetti, suddivisi in gruppi, possono essere *abilitati*, ovvero presenti sul *panel*, o *disabilitati*, ovvero assenti dal *panel*.
I gruppi di pulsanti sono i seguenti:

1. *Draw label*,
2. *Draw line*, *Draw arc*, *Draw Pline*
3. *Select mode*, *Clear Canvas*, *Undo Clear Canvas*, *Undo Draw*, *Redo Draw*
4. *Undo Clear Canvas*
5. *Redo Draw*

Nel "macro stato" ms_0 tutti i pulsanti sono *disabilitati*, nel "macro stato" ms_1 sono *abilitati* solo i pulsanti del gruppo 1.
Nel "macro stato" ms_2 sono *abilitati* i pulsanti dei gruppi 1, 2, 3 tranne quelli dei gruppi 4 e 5. L'abilitazione dei pulsanti del gruppo 5 richiede che sia stato eseguito almeno una volta il comando *Undo Draw*. L'abilitazione del gruppo 4 comporta la disabilitazione dei gruppi 2, 3 e 5.
L'uso dei pulsanti si basa su metodologie distinte. La selezione[6] di alcuni dei

[6]Per *selezione di un pulsante* si intende il *posizionamento del cursore* sul pulsante e la *pressione del tasto sinistro* del mouse.

pulsanti ha, infatti, un *effetto immediato* sul *canvas* mentre la selezione di altri si limita a *condizionare* le successive azioni dell'utente.
La tabella che segue contiene un elenco dei pulsanti e del tipo di effetto, *immediato* o *condizionante*. Mentre la selezione di un pulsante del secondo tipo ha un effetto immediato e circoscritto sul *canvas*, la selezione di un pulsante del primo tipo non ha un effetto immediatamente visibile sul *canvas* ma condiziona le successive azioni dell'utente assegnando una semantica ad alcuni degli eventi del mouse. Nel caso l'utente selezioni il pulsante *Draw label* ad ogni successivo evento del tipo *mouseClicked*[7] sul *canvas* corrisponde la creazione di una etichetta nel punto in cui l'utente genera l'evento.

Pulsante	*Effetto*
Draw label	condizionante
Draw line	condizionante
Draw arc	condizionante
Draw Pline	condizionante
Select mode	condizionante
Clear Canvas	immediato
Undo Clear Canvas	immediato
Undo Draw	immediato
Redo Draw	immediato

Tabella 3.3: *Pulsanti e relativi effetti*

Nel caso in cui l'utente selezioni uno dei pulsanti per il tracciamento di *archi* (*Draw arc*), *linee* (*Draw line*) o *spezzate* (dette *polilinee* o *pline*, *Draw Pline*) la selezione condiziona il significato dei successivi eventi del mouse.
Se l'utente seleziona il pulsante *Draw line* (considerazioni analoghe valgono anche per il pulsante *Draw arc*) per tracciare un segmento, che inizia e finisce in prossimità di due etichette distinte che non sono già connesse fra di loro da un elemento (*arco*, *linea* o *Pline*) avente lo stesso orientamento, può procedere in due modi:

1. può generare un evento *mouseClicked* in prossimità della prima etichetta e un altro evento *mouseClicked* in prossimità della seconda etichetta,

2. può generare un evento *mousePressed* in prossimità della prima etichetta seguito da un evento *mouseDragged* e da un evento *mouseReleased* in prossimità della seconda etichetta.

[7] Gli eventi associati al mouse corrispondono alla pressione e/o al rilascio di uno dei tasti (*mouseClicked*, *mousePressed*, *mouseReleased*) e al suo spostamento con o senza un pulsante premuto (*mouseDragged*, *mouseMoved*, *mouseEntered* e *mouseExited*).

Se l'utente seleziona il pulsante *Draw Pline* per il tracciamento di una *spezzata* che inizia e finisce in prossimità di due etichette distinte, che non sono già connesse fra di loro da un elemento avente lo stesso orientamento, valgono, *mutatis mutandis*, le considerazioni precedenti applicate alle coppie di punti che individuano i singoli segmenti di cui si compone la spezzata.
La selezione del pulsante *Select mode* rende neutri gli eventi *mouseClicked* singoli insieme agli eventi *mousePressed* e *mouseReleased* mentre gli eventi *mouseClicked* doppi generati su una *etichetta* o un *elemento di connessione* ne causano, come in altre condizioni, la rimozione dal *canvas* mentre l'evento *mouseDragged* generato su una etichetta ne causa lo spostamento.
La selezione di uno dei pulsanti ad effetto immediato, infine, si traduce in una modifica istantanea del contenuto del *canvas* che viene ripulito (*Clear Canvas*) o ripristinato a seguito di una ripulitaura (*Undo Clear Canvas*) e dal quale viene rimosso uno degli elementi presenti (*Undo Draw*) oppure rispristinato uno degli elementi rimossi (*Redo Draw*): la rimozione avviene a partire dagli elementi di connessione con una filosofia LIFO, ovvero "ultimo tracciato, primo rimosso" (*Undo Draw*) e "ultimo rimosso, primo ritracciato" (*Redo Draw*).

3.3.4 Le voci del menù sul *frame principale*

Le figure 3.5, 3.7 e 3.8 mostrano le voci del *menù principale* dell'editor, menù contenuto in alto sul *frame* principale[8], quello che contiene sia il *canvas* sia i *pulsanti*.
Il menù è suddiviso in *sottomenù* ciascuno dei quali contiene un certo numero di *voci* ad ognuna delle quali corrisponde un *comando*. Le *voci* possono essere *abilitate* o *disabilitate* in funzione dello *stato* dell'editor e ognuna delle voci ha un omologo o nell'insieme dei pulsanti (cfr. la sezione 3.3.3) o delle voci dei *menù flottanti* (cfr. la sezione 3.3.5).
I sottomenù presenti sono:

1. *File*,
2. *Application*,
3. *Draw*,
4. *Edit*.

Le voci del sottomenù *File* gestiscono le interazioni con il *file system* del *Sistema Operativo ospite*.
Le voci del sottomenù *Application* consentono di eseguire un certo numero di *comandi* sui diagrammi CL creati tramite l'editor.

[8] Un *frame* ([HC99]) in *Java* è una finestra *top level* ovvero non contenuta in nessun altra ma al cui interno possono essere localizzati elementi quali *pulsanti*, *menù* e *canvas*.

Le voci del sottomenù *Draw* consentono all'utente di aggiungere elementi quali *etichette* ed *elementi di connessione* ai diagrammi CL creati dall'utente mentre alle voci del sottomenù *Edit* corrispondono semplici comandi di editing.
La tabella 3.4 contiene un elenco delle voci dei sottomenù, per ciascuna delle quali è indicato il sottomenù che la contiene insieme a quali condizioni devono essere soddisfatte perchè la voce corrispondente sia abilitata e, pertanto, accessibile all'utente.
Le voci dei vari sotomenù saranno esaminate anche nel capitolo 5, nel quale saranno dati alcuni semplici esempi di utilizzo dei vari Tool. Qui ci si limita ad alcune brevi osservazioni su alcune voci che possono non avere un significato intuitivamente chiaro.

Voce	*Sottomenù*	*Abilitato*
New	File	*sempre*
Open...	File	*sempre*
Export...	File	solo se il *canvas* contiene almeno due etichette
Import...	File	*sempre*
Save	File	solo se il *canvas* ha subito modifiche
Save as...	File	*sempre*
Print...	File	*sempre*
Exit	File	*sempre*
Check Graph	Application	solo se il *canvas* contiene almeno due etichette
Convert to FD	Application	solo dopo un *Check Graph* con esito positivo
Arc Subset	Application	solo se il *canvas* contiene almeno due etichette
Node Subset	Application	solo se il *canvas* contiene almeno due etichette
Check Signs	Application	solo se il *canvas* contiene almeno due etichette
Sign Loops	Application	solo dopo un *Check Signs* con esito positivo
Label	Draw	*sempre*
Line	Draw	solo se il *canvas* contiene almeno due etichette
Arc	Draw	solo se il *canvas* contiene almeno due etichette
PLine	Draw	solo se il *canvas* contiene almeno due etichette
Select Mode	Edit	solo se il *canvas* contiene almeno due etichette
Clear Canvas	Edit	solo se il *canvas* contiene almeno due etichette
Undo Clear Canvas	Edit	solo dopo una *Clear Canvas*
Undo Draw	Edit	solo se il *canvas* contiene almeno due etichette
Redo Draw	Edit	solo dopo una *Undo Draw*

Tabella 3.4: *Le voci dei sottomenù del Causal Loop Graphic Editor*

Le voci *Open...* e *Import..* consentono il caricamento sul *canvas* di un diagramma CL contenuto in un file ed hanno come duali le voci *Save*, *Save as...* e *Export...* che consentono di salvare il contenuto del *canvas* in un file. Ciò che differenzia la

Open... dalla *Import...* è il formato dei file ai quali i due comandi danno accesso (cfr. la sezione 4.3): la prima consente di accedere a *file di oggetti* (creati usando le voci *Save* o *Save as...*) mentre la seconda consente di accedere a *file di testo* (creati con la voce *Export...*).
La voce *Check Graph* consente di eseguire un *controllo di completezza* del diagramma CL corrente. Tale controllo è condizione preliminare necessaria perché sia possibile eseguire il comando *Convert to FD*, normalmente disabilitato. Il controllo di completezza verifica che:

1. le etichette abbiano assegnato un tipo fra quelli possibili (ovvero "delay","level","rate","sink","source","constant","auxiliary"[9]),

2. gli elementi di connessione abbiano assegnato un tipo fra quelli possibili ("Information" o "Materials"),

3. vi sia coerenza fra il tipo di un nodo e il tipo degli archi da esso uscenti o in esso entranti nel senso che il tipo di un nodo determina il tipo degli archi che possono appartenere alla sua stella entrante e alla sua stella uscente.

Nel caso che il controllo abbia esito positivo viene abilitato il comando *Convert to FD* la cui esecuzione consente di ricavare da un diagramma CL il diagramma FD corrispondente. Maggiori dettagli saranno dati nella sezione 3.7.
Nel caso che il controllo dia un esito negativo, ovvero il diagramma contiene etichette e/o elementi di connessione da inizializzare, l'editor consente la visualizzazione dei sottografi individuati da tali elementi in *frame ausiliari* (cfr. la sezione 4.3.4 e il Capitolo 5).
Le voci *Arc Subset* e *Node subset* del sottomenù *Select&Display subgraph by* consentono di visualizzare, in *frame ausiliari* (cfr. la sezione 4.3.4 e il Capitolo 5), sottografi estratti dal diagramma corrente in base al *tipo* o al *segno* degli *elementi di connessione* (*Arc Subset*) oppure in base al *tipo* delle *etichette*.
La voci *Check Signs* e *Sign Loops* (inizialmente disabilitata), infine, consentono di verificare se tutti gli *elementi di connessione* del diagramma corrente hanno assegnato un segno (che può essere $+$ o $-$): in caso la verifica abbia esito positivo, viene abilitata la voce *Sign Loops* che permette di assegnare un segno ad ogni *loop* individuato nel diagramma mentre, se la verifica ha esito negativo (ovvero il diagramma contiene *elementi di connessione* con segno non specificato), l'editor consente di visualizzare il sottografo generato da tali elementi di connessione.
Ognuno dei *loop* ha associato un segno (che può essere $\oplus$ oppure $\ominus$) che può essere:

1. spostato,

[9]Una descrizione sommaria del significato di questi paramentri e di quelli di cui al punto successivo sarà data nella sezione 3.4 mentre, per un resoconto più approfondito, si rimanda al Capitolo 4.

2. rimosso,

3. selezionato.

Lo *spostamento* avviene trascinando il segno che si vuole spostare con il pulsante sinistro del mouse dalla posizione iniziale alla posizione voluta. La *rimozione* avviene generando un doppio evento *mouseClicked* sel segno che si vuole rimuovere mentre la *selezione* la si ottiene posizionando il cursore sul segno desiderato. Dopo aver selezionato un segno la generazione di un evento *mousePressed* con il pulsante destro del mouse permette di visualizzare le *etichette* e gli *elementi di connessione* del *loop* corrispondente al segno in un *frame ausiliario*. L'esecuzione di una operazione che modifica in qualche modo il diagramma corrente ha l'effetto di rimuovere tutti i segni dei *loop* individuati sul diagramma per cui la ridefinizione dei segni sui loop richiede che venga eseguito prima il comando *Check Signs* e poi, in caso di esito positivo, il comando *Sign Loops*[10] in modo da assegnare di nuovo i segni ai *loop* individauti nel diagramma modificato.

3.3.5 I menù flottanti

La figura 3.10 presenta i *menù flottanti* caratteristici dell'editor. I menù flottanti sono tre. Il primo, presentato a sinistra nella figura 3.10, consente all'utente di accedere direttamente dal *canvas* ad un sottoinsieme dei comandi accessibili tramite i *pulsanti* oppure tramite i sottomenù *Draw* ed *Edit*.
I comandi di questo menù sono già stati presentati in sezioni precedenti per cui non si ritiene di aggiungere altro rimandando al Capitolo 5 per alcuni esempi di utilizzo.
Il secondo menù, presentato nel centro della figura 3.10, consente all'utente di eseguire un certo numero di comandi, non accessibili altrimenti, su ognuna delle *etichette* del diagramma corrente.
I comandi di questo menù consentno all'utente di:

1. rimuovere l'etichetta (*Delete label*),

2. modificare la stringa associata all'etichetta (*Change label*),

3. visualizzare gli archi della *stella entrante* dell'etichetta (*Query inlink*),

4. visualizzare gli archi della *stella uscente* dell'etichetta (*Query outlink*)

5. modificare il tipo dell'etichetta (*Set/Change type*).

La modifica della stringa avviene tramite una *finestra di dialogo* mediante la quale l'utente può immettere il nuovo valore o confermare il vecchio. In modo

[10] Ad ognuna delle voci dei menù corrisponde, in genere, un comando per cui, nel seguito, salvo avviso contrario, useremo indifferentemente le due espressioni.

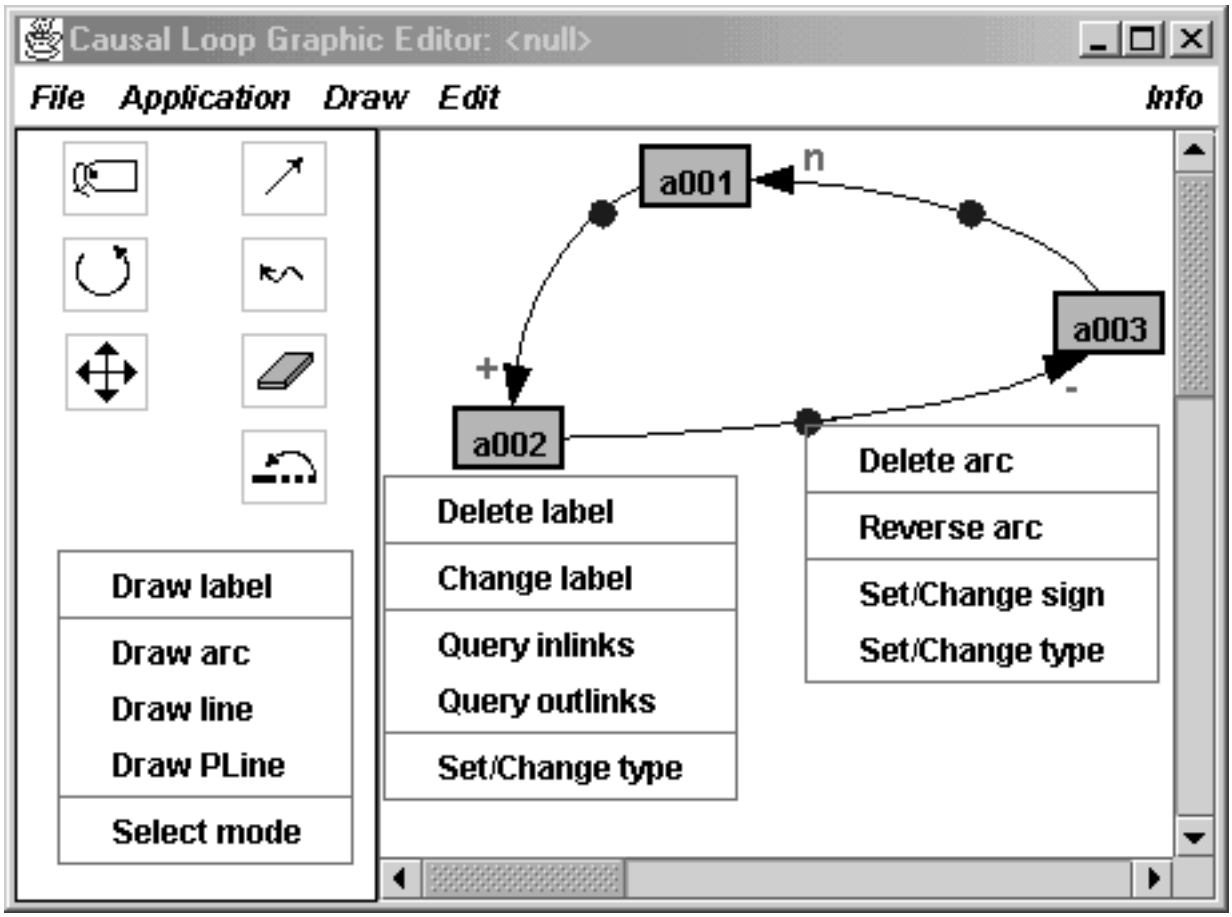

Figura 3.10: *I menù flottanti*

analogo l'utente può impostare il tipo dell'etichetta scegliendolo fra uno di quelli possibili (o confermando il vecchio tipo) mentre la visualizzazione degli elementi della *stella entrante* o della *stella uscente* avviene mediante una *finestra ausiliaria* che consente di visualizzare ricorsivamente tutte le stelle entranti/uscenti delle etichette a meno che una di queste non abbia una di tali stelle vuota.
Il terzo menù, presentato a destra nella figura 3.10, consente all'utente di eseguire un certo numero di comandi, non accessibili altrimenti, sugli *elementi di connessione* del diagramma corrente.
I comandi di questo menù consentono all'utente di:

1. rimuovere l'elemento di connessione (*Delete arc*),
2. invertire l'orientamento dell'elemento di connessione (*Reverse arc*),
3. modificare il segno dell'elemento di connessione (*Set/Change sign*)
4. modificare il tipo dell'elemento di connessione (*Set/Change type*).

L'utente può invertire l'orientamento di un *elemento di connessione* (con il comando *Reverse arc*) solo se lo stato delle connessioni del *grafo* corrispondente al diagramma corrente lo consente ovvero solo se non esiste già un arco con l'orientamento opposto a quello dell'*elemento di connessione* corrente. La modifica del *segno* e del *tipo* avviene utilizzando due distinte *finestre di dialogo* mediante le quali l'utente imposta (se il segno è non specificato) o altrimenti modifica il segno dell'*elemento di connessione* oppure imposta (se il tipo è non specificato) o altrimenti modifica il tipo dell'*elemento di connessione*.

3.4 *Flow Diagram Graphic Editor*

3.4.1 Introduzione

Il *Flow Diagram Graphic Editor*, all'interno del'ambiente *D(a)ySy Tool Box*, rappresenta il Tool che consente il tracciamento e l'editing di diagrammi FD. Per sua natura possiede molte delle caratteristiche del Tool *Causal Loop Graphic Editor* cui si riamanda (cfr. la sezione 3.3). Il *Flow Diagram Graphic Editor* si presenta all'utente con l'interfaccia illustrata in figura 3.11 (i menù *Draw* e *Edit* sono illustrati nella figura 3.12).
Il Tool consente all'utente di tracciare diagrammi FD, di associare ai nodi di tali diagrammi le grandezze necessarie alla impostazione delle relative equazioni, di eseguire operazioni di controllo della correttezza e della completezza dei diagrammi tracciati e di interagire con i Tool *Resolver* e *Display* per la risoluzione delle equazioni e la visualizzazione degli andamenti nel tempo delle soluzioni determinate.
A tale scopo il Tool mette a disposizione dell'utente (cfr. le figure 3.11, 3.12 e 3.13):

1. un "canvas" su cui tracciare i diagrammi FD,
2. un certo numero di pulsanti che compaiono sulla sinistra del "canvas" (cfr. la figura 3.13,
3. un certo numero di menù (cfr. le figure 3.11 e 3.12),
4. alcuni menù flottanti.

Il Tool, come altri Tool dell'ambiente *D(a)ySy Tool Box*, può essere mandato in esecuzione sia in modalità "stand alone", ovvero come processo autonomo, sia come *thread* del Tool *TopLevel*, in modalità *slave*.

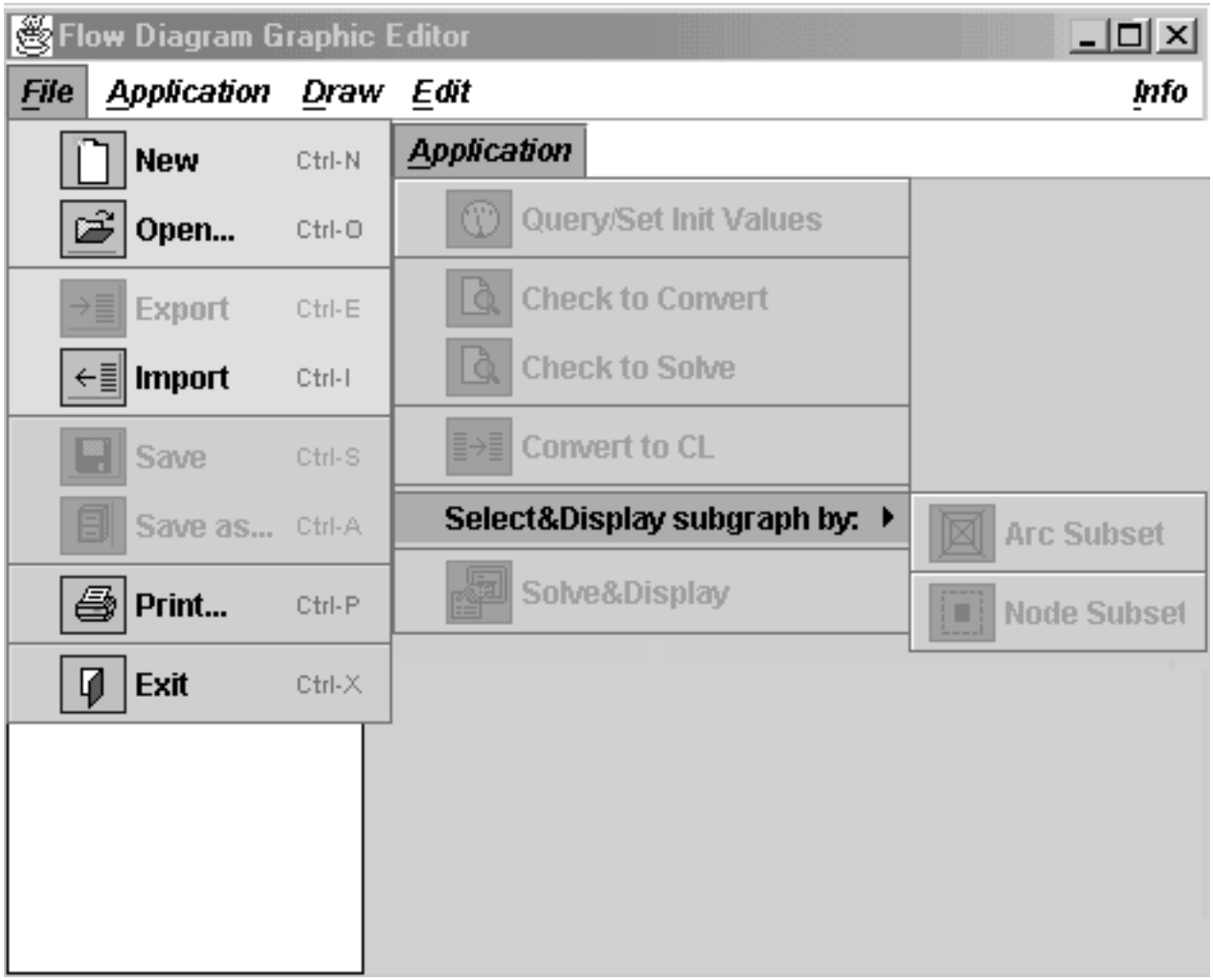

Figura 3.11: *Il "Flow Diagram Graphic Editor"*

3.4.2 Le voci del menù sul *frame principale*

Il menù sul *frame principale* è caratterizzato dalle voci *File*, *Application*, *Draw* ed *Edit* i cui contenuti sono illustrati dalle tabelle che seguono.
La tabella 3.5 illustra gli elementi della voce *File* di tale menù, indicando per ciascuno:

1. il nome dell'elemento,
2. la funzione svolta da un elemento,
3. la disponibilità.

Per ulteriori dettagli sugli elementi di questa voce si rimanda alla sezione 3.3.4.
La tabella 3.6 illustra gli elementi della voce *Application*. Le voci illustrate dalla

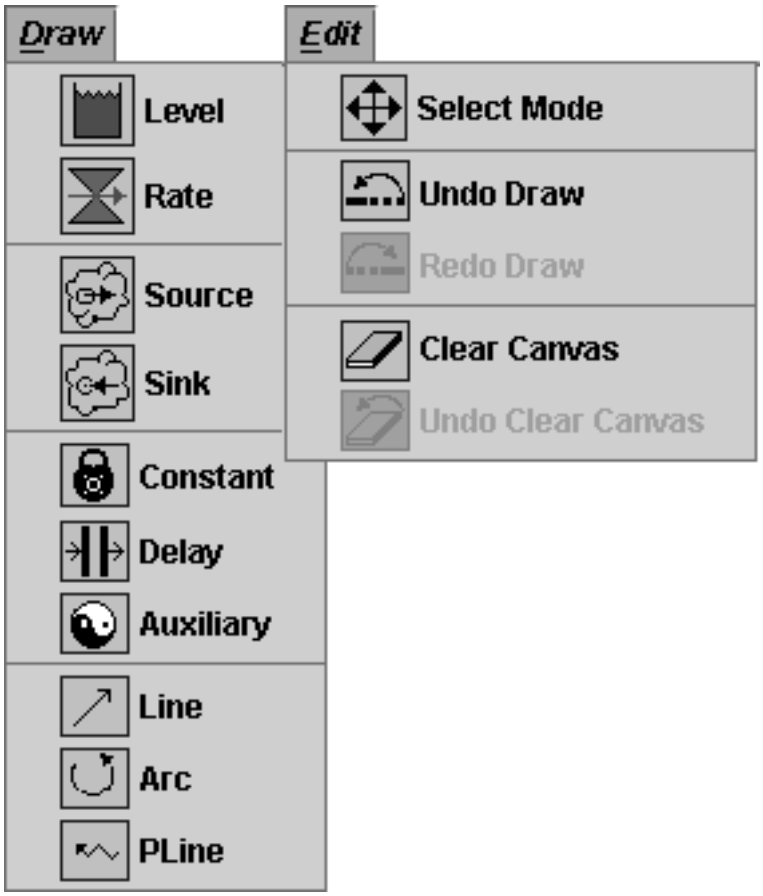

Figura 3.12: *I menù "Draw" e "Edit"*

tabella 3.6 sono suddivisibili in quattro tipologie:

1. voci per il controllo,
2. voci per l'elaborazione,
3. voci per la selezione,
4. voci per il settaggio.

Alla prima tipologia appartengono le voci *Check to Convert* e *Check to Solve*: la prima permette di verificare la congruenza di un diagramma FD per la sua conversione in diagramma CL mentre la seconda consente di verificare se i nodi di un diagramma FD sono stati inizializzati in modo corretto in modo che sia possibile risolvere le equazioni descrittive del modello rappresentato dal diagramma FD. Entrambe le voci sono accessibili solo se il canvas è significativo, ovvero se contiene almeno due nodi, ma danno luogo a vere e proprie operazioni di controllo

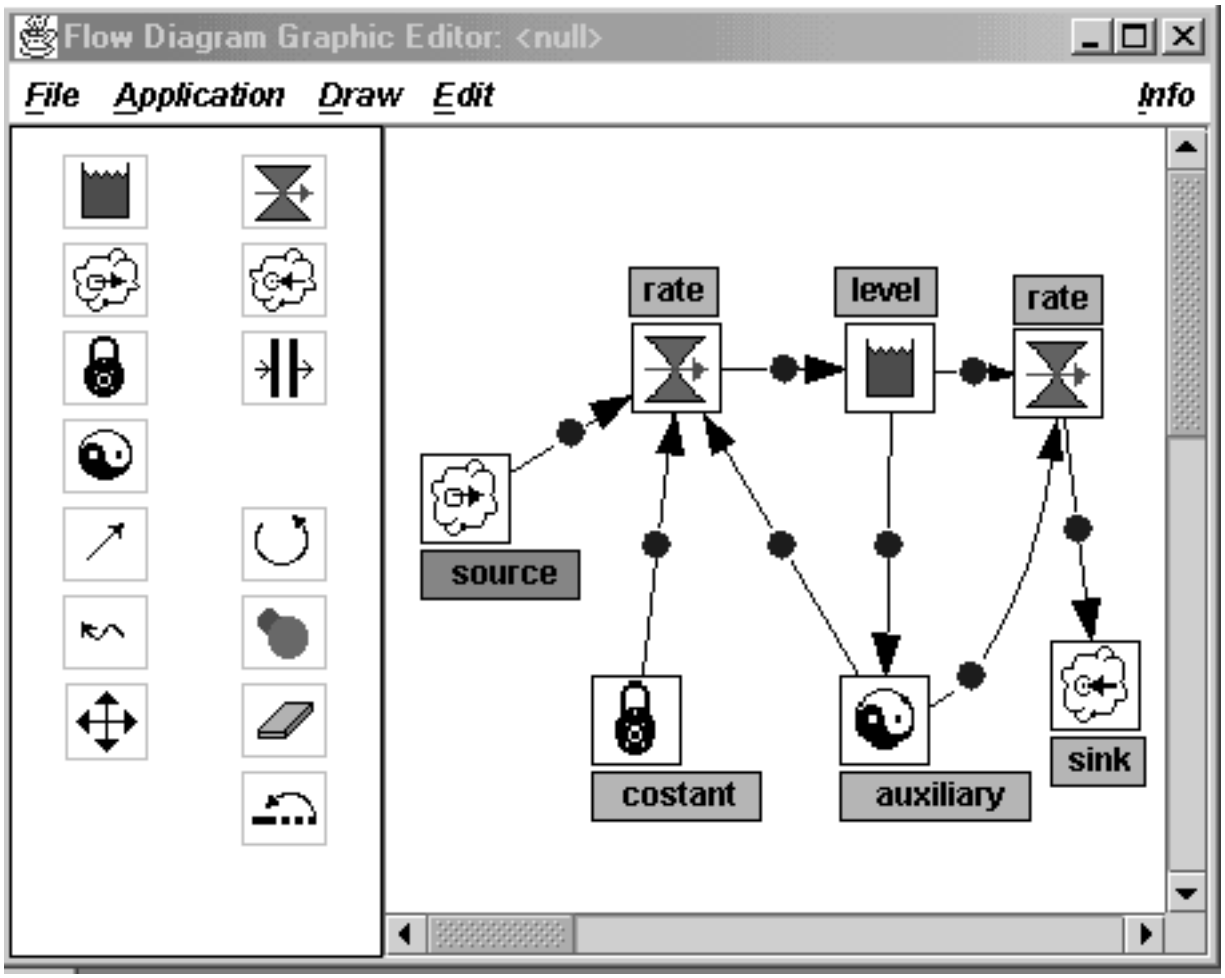

Figura 3.13: *Il "canvas" con un diagramma di esempio*

se il canvas contiene un diagramma minimo, ovvero un diagramma composto da due nodi e da un elemento di connessione. Alla seconda tipologia appartengono le voci *Convert to CL* e *Solve&Display*. La prima risulta accessibile solo se il relativo controllo di congruenza ha avuto esito positivo e consente di ottenere il diagramma CL corrispondente ad un dato diagramma FD. La descrizione del diagramma CL così ottenuta viene memorizzata, con le cautele del caso, in un file in formato oggetto con nome ed estensione corrette. Per ulteriori dettagli si veda la sezione 3.7.
La seconda risulta accessibile solo se il relativo controllo di correttezza ha avuto esito positivo e consente all'utente di richiamare in cascata e automaticamente i metodi dei moduli *Resolver* e *Display* necessari per la risoluzione delle equazioni e per la visualizzazione dei risultati. Per ulteriori dettagli si rimanda alle sezioni 3.5 e 3.6.

Elementi del menù	*Funzione*	*Disponibilità*
New	Crea nuovo canvas	sempre
Open	Importa un file oggetto	sempre
Export	Esporta contenuto canvas	solo se canvas non vuoto
Import	Import un file testo	sempre
Save	Salva contenuto canvas	solo se canvas modificato
Save as	Come sopra, in più settando nome	solo se canvas modificato
Print	Stampa file di testo	sempre
Exit	Termina l'esecuzione	sempre

Tabella 3.5: *Elementi della voce "File"*

Elementi del menù	*Funzione*	*Disponibilità*
Query/Set Init Values	Impostazione valori simulazione	dopo New, Open e Import
Check to Convert	Controllo per conversione	solo se canvas significativo
Check to Solve	Controllo per risoluzione	solo se canvas significativo
Convert to CL	Conversione	solo se controllo esito positivo
Arc Subset	Selezione sottografo	solo se canvas significativo
Node Subset	Selezione sottografo	solo se canvas significativo
Solve&Display	Risoluzione e visualizzazione	solo se controllo esito positivo

Tabella 3.6: *Elementi della voce "Application"*

Alla terza tipologia appartengono le voci *Arc Subset* e *Node Subset*, sottovoci della voce *Select&Display subgraph by:*. La prima consente di selezionare (e visualizzare in un frame a parte) il sottografo indotto dagli elementi di connessione di un certo tipo (ovvero "Information" o "Materials") mentre la seconda consente di selezionare (e visualizzare in un frame a parte) il sottografo indotto dai nodi di un uno o più tipi fra quelli previsti ("level","rate","source","sink","constant","delay","auxiliary").
Nel primo caso, si possono evidenziare in frame separati i due diversi flussi di informazioni (flusso non conservativo) e di materiali (flusso conservativo) all'interno di un diagramma FD.
Nel secondo caso,invece, si possono evidenziare le varie strutture che sono presenti in un diagramma FD quali:

1. i gruppi di elementi composti da nodi di tipo "level" e nodi di tipo "rate",
2. i gruppi di elementi composti da nodi di tipo "level" e nodi di tipo "auxiliary",
3. i gruppi di elementi composti da nodi di tipo "rate" e nodi di tipo "auxiliary",
4. i gruppi di elementi composti da nodi di tipo "rate" e nodi di tipo "constant",

e altre più complesse, semplicemente selezionando i tipi desiderati tarmite una finestra di dialogo.
All'ultima tipologia appartiene, infine, la voce *Query/Set Init Values* che consente di impostare alcuni valori significativi per l'esecuzione di una simulazione (e sui quali torneremo nella sezione 3.4.5) fra cui: istante di inizio, istante di fine, passo (ovvero valore della variabile T) e unità di misura della variabile tempo.
La tabella 3.7 illustra gli elementi della voce *Draw*. Gli elementi della voce *Draw*, che compaiono anche in un menù flottante e sotto forma di pulsanti, sono di due tipi. Quelli del primo tipo sono sempre disponibili e consentono la creazione di nodi, ovvero di coppie "icona-etichetta" (cfr. la figura 3.13). Quelli del secondo tipo sono disponibili solo se il canvas contiene almeno due nodi e consentono il tracciamento di elementi di connessione quali: segmenti orientati("Line"), archi orientati ("Arc") e spezzate orientate ("PLine").
Per quanto riguarda la creazione di elementi del primo tipo, l'utente seleziona la voce desiderata e poi selezionando un punto vuoto sul canvas (usando la tecnica *select and drop*, cfr. la sezione 2.2.8) ottiene la creazione di una istanza del tipo voluto.
La creazione degli elementi del secondo tipo avviene selezionando la voce corrispondente e poi usando una delle due tecniche di cui alla sezione 2.2.8: *point and click* e *drag and drop*.

Elementi del menù	*Funzione*	*Disponibilità*
Level	Posizionare elementi di tipo *Level*	sempre
Rate	Posizionare elementi di tipo *Rate*	sempre
Source	Posizionare elementi di tipo *Source*	sempre
Sink	Posizionare elementi di tipo *Sink*	sempre
Constant	Posizionare elementi di tipo *Constant*	sempre
Delay	Posizionare elementi di tipo *Delay*	sempre
Auxiliary	Posizionare elementi di tipo *Auxiliary*	sempre
Line	Posizionare elementi di tipo *Line*	almeno due nodi
Arc	Posizionare elementi di tipo *Arc*	almeno due nodi
PLine	Posizionare elementi di tipo *PLine*	almeno due nodi

Tabella 3.7: *Elementi della voce "Draw"*

Gli elementi della voce *Edit* coincidono con quelli della corrispondente voce del menù sul frame principale del Tool *Causal Loop graphic Editor* per cui si rimanda alla sezione 3.3.4.

3.4.3 I pulsanti

Nell'ordine da sinistra a destra e dall'alto in basso, in figura 3.13 compaiono i pulsanti:

1. Draw Level
2. Draw Rate
3. Draw Source
4. Draw Sink
5. Draw Constant
6. Draw Delay
7. Draw Auxiliary
8. Draw Line
9. Draw Arc
10. Draw PLine
11. Set Arc Type
12. Select Mode

13. Clear Canvas

14. Undo Draw

I *pulsanti* sono contenuti in un *panel* sulla sinistra del *canvas* (cfr. la figura 3.13) e consentono all'utente di eseguire:

1. i primi sette, operazioni di posizionamento di nodi con la tecnica *select and drop*,

2. i successivi tre, operazioni di tracciamento di elementi di connessione con le tecniche *point and click* e *drag and drop*,

3. gli ultimi quattro, operazioni di editing,

4. il rimanente, operazioni di settaggio del tipo degli elementi di connessione.

Quando il Tool viene mandato in esecuzione nessuno dei pulsanti è visibile sul relativo *panel*. Solo dopo che l'utente ha eseguito un comando *New* sono disponibili i pulsanti di cui al punto (1) mentre gli altri sono disponibili solo se il *canvas* contiene almeno due nodi oppure se in esso è stato caricato un diagramma FD mediante una operazione di *Open...* (diagramma precedentemente creato con una operazione di *Save*) o di *Import...* (diagramma precedentemente creato con una operazione di *Export*).
I pulsanti di editing implementano le stesse voci del menù *Edit* al quale si rimanda. Si ricorda solo che tali operazioni hanno effetto immediato sul contenuto del *canvas* e possono, di regola, essere annullate a meno che l'utente non ha successivamente utilizzato un comando di posizionamento o di tracciamento.
Due operazioni di editing i cui pulsanti non compaiono nella figura 3.13 sono le operazioni di *Undo Clear Canvas* e *Redo Draw*: la prima è disponibile solo a seguito di una operazione di *Clear Canvas* (il cui pulsante risulta quindi invisibile) mentre la seconda è accessibile solo a seguito di una operazione di *Undo Draw*.
I pulsanti *Undo Draw* e *Redo Draw* (e i corrispondenti comandi del menù *Edit*) sono soggetti, come nel caso del *Cusal Loop Graphic Editor*, alle seguenti limitazioni:

1. il comando[11] *Undo Draw* rimuove uno ad uno gli elementi di connessione, se presenti, e poi rimuove i nodi, sempre uno a d uno, nell'ordine inverso a quello in cui sono stati posizionati sul canvas,

2. il comando *Redo Draw* ripristina gli elementi di connessione in ordine inverso a quello in cui sono stati rimossi per cui prima viene ripristinato l'ultimo rimosso, poi il punultimo e così via fino al primo,

[11]Nel seguito i termini *comando* e *pulsante* saranno usati come sinonimi, salvo diversa specificazione.

3. se si rimuove un nodo si perde la possibilità di ripristinare con il comando *Redo Draw* gli elementi di connessione precedentemente rimossi,

4. se si rimuovono tutti i nodi meno uno si possono ripristinare con il comando *Redo Draw* solo i nodi e non gli elementi di connessione originariamente presenti e rimossi.

Le suddette limitazioni discendono principalmente dalla necessità di mantenere il diagramma congruente con il grafo soggiacente ed a limitazioni nella implementazione delle operazione, limitazioni destinate ad essere rimosse in future release dei Tool. Il pulsante che permette di svolgere operazioni di settaggio, infine, consente di settare il tipo degli elementi di connessione che saranno tracciati da quel momento in poi e il cui valore di default è "Materials" con cui si descrivono flussi conservativi e a cui corrispondono elementi di connessione di colore magenta.

3.4.4 Il *canvas* e i *menù flottanti*

Il *canvas* rappresenta la superfice sulla quale l'utente può posizionare i nodi (come già detto formati da coppie "icona-etichetta") e gli elementi di connessione, in modo da tracciare i diagrammi FD (cfr. la figura 3.13).
Ogni icona può essere spostata sul canvas in modo da trascinarsi dietro, nello spostamento, sia l'etichetta associata sia gli estremi degli elementi di connessione in essa incidenti. La stessa proprietà di spostabilità è posseduta dalle etichette che, tuttavia, non hanno piena autonomia rispetto all'icona associata nel senso che uno spostamento dell'icona causa, con un effetto calamita, un riposizionamento dell'etichetta nella posizione iniziale rispetto all'icona in modo che icona ed etichetta vengano spostate come fossero un elemento solo. In più uno spostamento dell'etichetta non influenza gli elementi di connessione che incidono sul nodo ovvero sull'icona, il cui centro rappresenta il punto in cui convergono tutti gli elementi incidenti.
Gli elementi di connessione, dal canto loro, non possono essere trascinati autonomamente dalle icone ma possiedono alcune proprietà che saranno esaminate a breve.
Le icone, le etichette e gli elementi di connessione sono caratterizzati da alcune proprietà manipolabili attraverso menù flottanti, alcuni dei quali sono illustrati in figura 3.14. In tale figura compare anche il menù flottante del canvas ovvero il menù che si ottiene selezionando un punto libero del canvas quando questo contiene almeno due nodi.
Il menù del canvas (cfr. il menù sulla destra della figura 3.14) è caratterizzato dai comandi del menù *Edit* che non saranno, quindi, esaminati e da un comando che consente di porre l'editor in *Select mode*, ovvero in una condizione in cui le azioni del mouse non hannno nessun significato particolare a parte quello di consentire la selezione di un elemento (singolo click) o di consentirne la cancellazione (doppio click).

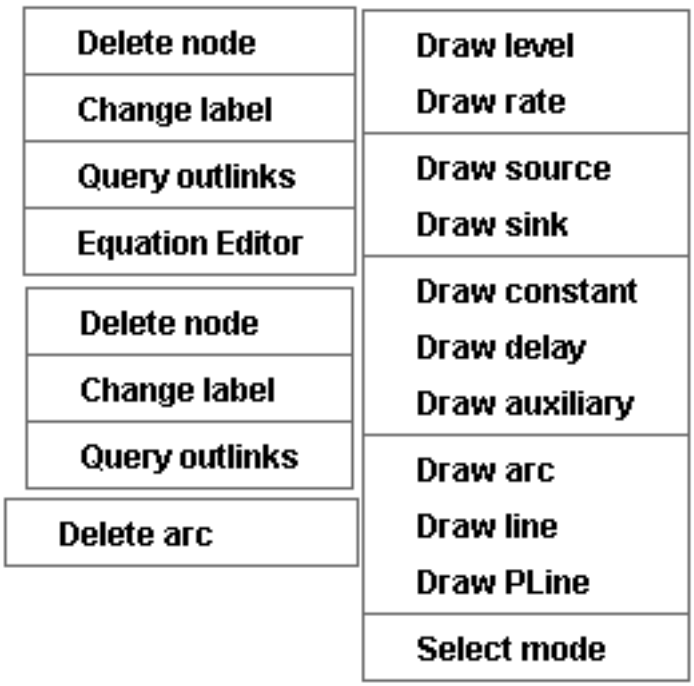

Figura 3.14: *I menù flottanti*

I nodi hanno come elementi sensibili alle azioni del mouse sia le icone sia le etichette. Posizionando il cursore su una icona o una etichetta, la forma del cursore cambia o in un "mirino" o in un cursore a forma di mano (detto *hand_cursor*). Premendo il pulsante destro del mouse su una icona si ottiene uno dei due menù a sinistra in alto della figura 3.14 a seconda che il nodo abbia associata una equazione oppure no (come accade nel caso degli elementi "source" e "sink"). Un'altra variante, non presente in figura, possiede oltre alla voce *Query outlink* anche la voce *Query inlink*. La voce *Query outlink* consente di interrogare la stella uscente di un nodo ed è presente solo se il tipo del nodo lo prevede. Similmente la voce *Query inlink* consente di interrogare la stella entrante di un nodo ed è presente solo se il tipo del nodo lo prevede. Ad esempio il menù flottante di un nodo di tipo *constant* ha solo la voce *Query outlink* mentre il menù flottante di un nodo di tipo *sink* ha solo la voce *Query inlink*. Le voci *Query outlink* e *Query inlink* consentono, mediante una finestra di dialogo, di percorrere ricorsivamente le stelle uscenti ed entranti dei vari nodi fino a nodi che hanno, rispettivamente, la stella uscente o la stella entrante vuote.
Le voci comuni ai menù associati alle icone consentono di:

1. rimuovere un nodo: *Delete node*,
2. modificare l'etichetta del nodo: *Change label*.

La rimozione di un nodo si traduce nella rimozione di tutti gli elementi di connessione in esso incidenti e nel conseguente aggiornamento della struttura del grafo soggiacente.
La voce *Change label* è disponibile anche premendo il cursore del mouse avendo selezionato una etichetta. La modifica di una etichetta è resa possibile da una finestra di dialogo mediante la quale l'utente modifica l'etichetta corrente del nodo, modifica che si ripercuote in tempo reale nella modifica della etichetta del corrispondente nodo del grafo soggiacente.
La voce *Equation Editor*, qualora presente, consente di assegnare al nodo tutti gli elementi necessari alla valutazione della equazione ad esso associata e dipendenti dal tipo del nodo. Da ciò discende il fatto che alcuni nodi hanno il menù flottante privo di tale voce. Per ulteriori dettagli vedi la sezione 3.4.5. Gli elementi di connessione, infine, sono caratterizzati da uno o più handle (sono i pallini colorati, in nero nelle figure, visibili nella figura 3.13, uno per gli archi e i segmenti, più di uno per le spezzate) mediante i quali è possibile agire su alcune proprietà di ogni elemento di connessione. Selezionando con il cursore destro del mouse un handle si ha la comparsa del menù flottante in basso a destra della figura 3.14 che consente di cancellare un elemento di connessione, operazione sempre possibile che si traduce nella modifica della rappresentazione pittorica e nell'aggiornamento del grafo.

3.4.5 L'Equation Editor

Il comando *EquationEditor* (cfr. la sezione 3.4.4) è presente nel menù flottante dei nodi di tipo "level", "rate", "constant", "auxiliary" e "delay".
Mediante tale comando è possibile associare ad ogni nodo, in funzione del suo tipo, tutti i parametri necessari per la definizione dell'equazione caratteristica del nodo stesso. Il comando utilizza alcuni parametri globali associati al singolo diagramma FD e acquisiti dal Tool al momento della esecuzione di uno dei seguenti comandi: *New...*, *Open...* e *Import...*.
I parametri globali, che vengono acquisiti tramite una finestra di dialogo (cfr. la figura 3.15) che ne visualizza i valori correnti, sono:

1. l'istante iniziale della simulazione,
2. l'istante finale della simulazione,
3. il valore della variabile *time step* T,
4. l'unità di misura della variabile tempo.

L'*istante iniziale* assume il valore 0, per il momento non modificabile, mentre

1. l'*istante finale* può assumere un valore compreso fra 10 e 100 con incrementi di 10,

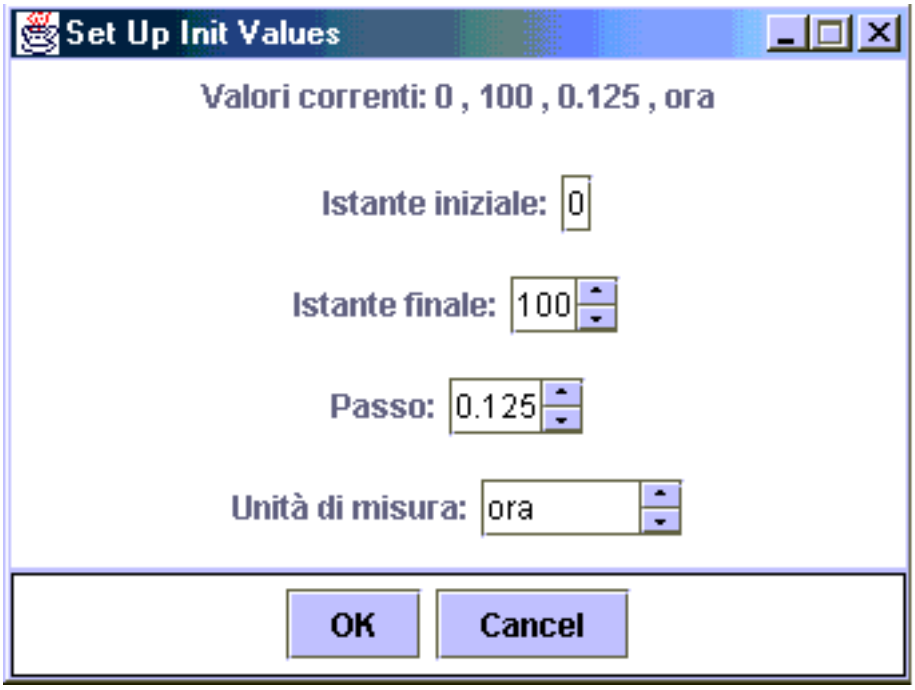

Figura 3.15: *I parametri della simulazione*

2. la variabile T può assumere uno dei valori 0.125, 0.25, 0.5 e 1.0,

3. l'unità di misura della variabile tempo può essere scelta fra le seguenti: *ora*, *minuto*, *secondo*, *Anno*, *Mese*, *Settimana*, *Giorno*.

La figura 3.16 illustra il caso dell'*Equation Editor* per un nodo di tipo "level". Nel caso di un nodo di tipo "level" isolato o con solo archi incidenti di tipo "Materials" la finestra di dialogo presenta all'utente l'elenco delle unità di misura correntemente presenti nel grafo (fra le quali è sempre presente l'unità di misura scelta per la variabile tempo) e due campi di testo.
I campi di testo permettono all'utente di impostare, nell'ordine:

1. un valore iniziale per il livello,

2. una unità di misura per il livello, scegliendola fra quelle già presenti oppure definendone una nuova.

Una eventuale nuova unità di misura viene automaticamente associata al grafo corrente ed è resa disponibile ai comandi *Equation Editor* dei nodi che vengono successivamente inizializzati.
Nel caso di un nodo di tipo "level" non isolato e con archi entranti di tipo "Information" la finestra di dialogo presenta all'utente l'elenco dei riferimenti alle

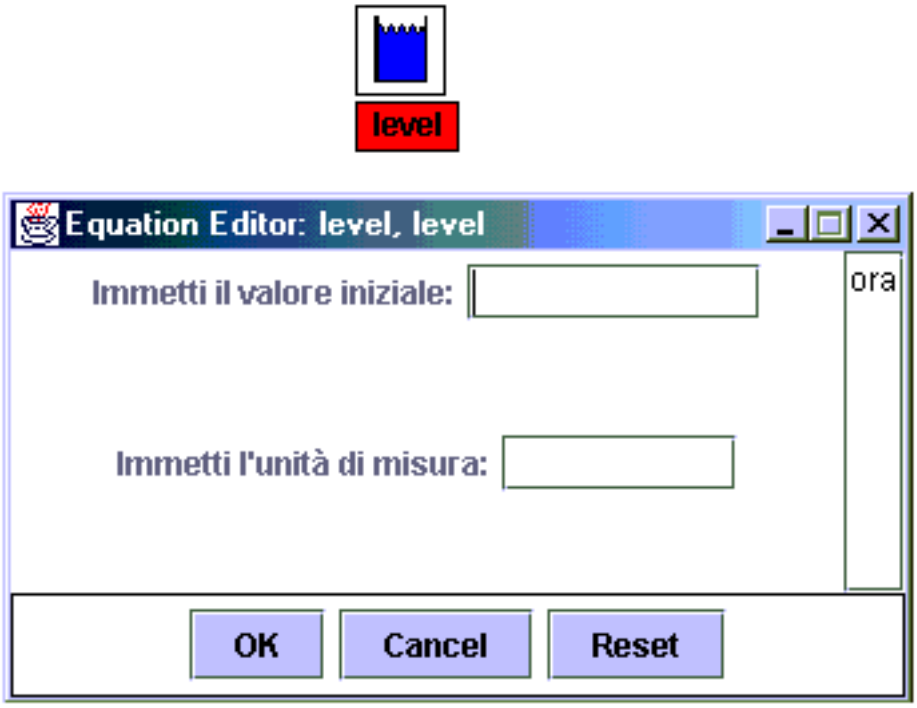

Figura 3.16: *L'Equation Editor per un nodo di tipo "level"*

variabili connesse in ingresso con archi di tipo "Information" che l'utente può comporre con operatori aritmentici.
Come risulta chiaro dall'analisi svolta nel capitolo 1, un livello è caratterizzato da una equazione avente la seguente struttura

$$level(n) = level(n-1) + T \times (\varphi_{in}(n-1, n) + \varphi_{out}(n-1, n)) \qquad (3.1)$$

in cui φ_{in} e φ_{out} sono, rispettivamente, i flussi in ingresso e in uscita. Per iniziare la valutazione di tale equazione è, pertanto, sufficiente conoscere il valore iniziale (ovvero all'istante iniziale della simulazione) della variabile *level* dal quale dipendono i valori iniziali dei flussi associati alla variabile.
In modo analogo l'*Equation Editor* per un nodo di tipo "constant" (cfr. la figura 3.17) permette all'utente di impostare, utilizzando due campi di testo:

1. un valore per la variabile,
2. una unità di misura per la variabile.

In merito alla unità di misura valgono le considerazioni fatte per il caso precedente. Nel caso dei nodi di tipo "delay", "auxiliary" e "rate" il comando *Equation Editor* permette di impostare, rispettivamente:

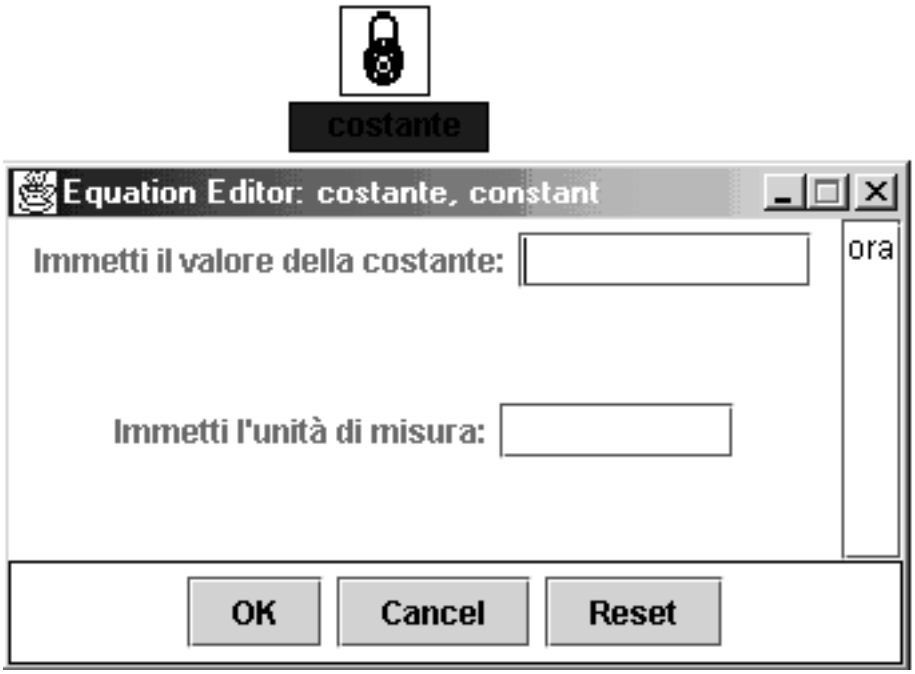

Figura 3.17: *L'Equation Editor per un nodo di tipo "constant"*

1. l'entità del ritardo (per il momento sono previsti ritardi di tipo costante sia sui flussi conservativi sia sui flussi non conservativi)
2. i valori che determinano l'andamento della variabile e la relativa unità di misura,
3. i parametri che caratterizzano il flusso e la relativa unità di misura.

In merito alle unità di misura valgono anche in questi casi le considerazioni fatte nel caso dei nodi di tipo "level". Per quanto riguarda la determinazione delle equazioni associate ad elementi di tipo "rate" o "auxiliary" si fa notare quanto segue.
Le variabili di tipo "rate" ([RAD+83]), in genere, dipendono da una variabile di tipo "level" mediante un legame di tipo "Information" ma possono dipendere anche da variabili di tipo "auxiliary" o avere un valore costante. Una variabile di tipo "rate" può infine dipendere da una funzione del tempo (ad esempio una funzione gradino unitario, una funzione rampa lineare o altre funzioni simili) che, associata ad una variabile "auxiliary" (vedi oltre), simula l'azione del mondo esterno sul sistema modellizzato. Da ciò discende la necessità di caratterizzare, per le variabili di tipo "rate", equazioni dei tipi seguenti ([RAD+83])[12]:

1. *rate=costante*

[12]Negli esempi che seguono useremo nomi di variabili coincidenti con il tipo o autoesplicativi.

2. *rate=level×fattoreDiCrescita*
3. *rate=level/vitaMedia*
4. *rate=(level-goal)/tempoDiAssestamento*
5. *rate=(goal-level)/tempoDiAssestamento*
6. *rate=K×auxiliary*

in cui "goal" è il valore desiderato della variabile "level", K rappresenta una costante di proporzionalità che ha anche lo scopo di rendere dimensionalmente corretta l'equazione e gli elementi $fattoreDiCrescita$ e $tempoDiAssestamento$ sono associati in genere a elementi di tipo "constant" (se non variano nel tempo) o "auxiliary" (se devono poter variare nel tempo). La caratterizzazione di una variabile di tipo "rate" con l'*Equation Editor* richiede, pertanto, che vengano definiti gli elementi necessari per caratterizzare una delle suddette equazioni e, cioè, che vengano individuati gli elementi in gioco e gli operatori matematici da utilizzare.
Le variabili di tipo "auxiliary" possono essere usate per rappresentare l'influsso di stimoli esterni sul modello oppure per dare maggiore espressività al modello. Nel primo caso la variabile ha una connessione di tipo "Information" in uscita verso o una variabile di tipo "rate" o un'altra variabile di tipo "auxiliary". Per la sua caratterizzazione con l'*Equation Editor* è necessario selezionare una funzione fra quelle disponibili per la quale è necessario, quindi, definire i parametri caratteristici (nel caso della funzione gradino unitario, ad esempio, è necessario stabilire sia l'ampiezza del gradino A sia l'isante in cui la funzione passa dal valore 0 al valore A).
Nel secondo caso la variabile ha connessioni in ingresso e in uscita di tipo "Information" verso altre variabili. Alcuni esempi di equazioni, in questo caso, sono i seguenti [RAD+83]):

1. *auxiliary=level-goal*
2. *aukilary=K × level*

in cui K ha il significato di una costante di proporzionalità. Di solito i legami in ingresso, di tipo "Information", derivano da nodi di tipo "level", "constant" o "auxiliary" mentre il legame in uscita è diretto verso un nodo di tipo "rate" o un altro nodo di tipo "auxiliary".
In questo caso, la caratterizzazione dell'equazione richiede che vengano scelti gli elementi e gli operatori matematici che li mettono in relazione fra di loro. Usualmente sono sufficienti gli operatori aritmetici (+, −,∗ e /) ma l'utente deve poter disporre anche almeno delle funzioni matematiche elementari quali elevamenti a potenza, estrazioni di radice, funzioni trigonometriche e così via.

3.5 *Display*

3.5.1 Introduzione

Il Tool *Display* permette all'utente di visualizzare gli andamenti nel tempo di un certo numero di grandezze che possono essere (ma non necessariamente sono) le soluzioni delle equazioni descrittive di un sistema dinamico.

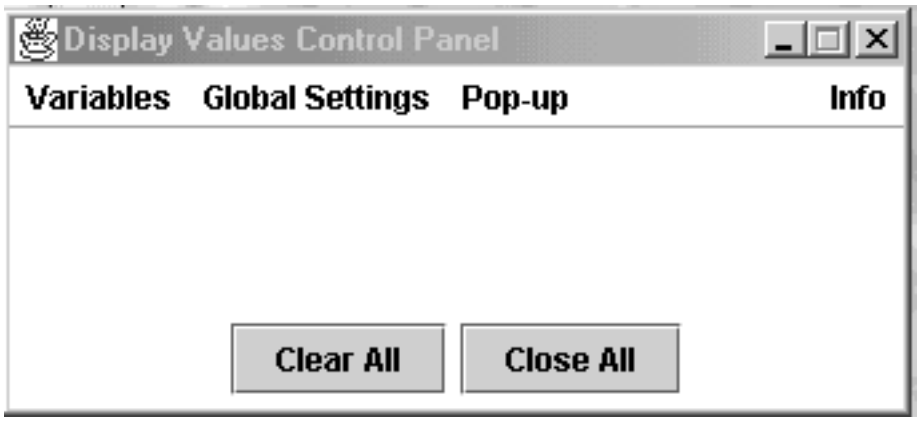

Figura 3.18: *Il frame principale del Tool Display*

Il Tool può essere utilizzato, infatti, in cooperazione con qualunque altro Tool che gli fornisca i necessari parametri di ingresso ed è questo il motivo per cui lo si presenta prima del Tool *Resolver* a sottolineare l'indipendenza dei due. Tali parametri, come sarà chiarito meglio nel seguito, sono i seguenti:

1. un certo numero di nomi di variabili con le relative unità di misura, dal momento che ciascuna variabile descrive una grandezza fisica,
2. un *array* di valori per ciascuna variabile,
3. valori ausiliari per la visualizzazione delle grandezze quali: istante iniziale, istante finale e valore del parametro T.

La presente sezione ha lo scopo di illustrare il Tool *Display* essenzialmente "lato utente" per cui contiene la descrizione del *frame* principale, dei *frame* ausiliari, delle *finestre di dialogo* e dei *comandi* dei vari menù.

3.5.2 L'interfaccia utente

Il Tool si presenta all'utente con l'interfaccia illustrata dalla figura 3.18. Il *frame* principale è caratterizzato da un menù che comprende tre sottomenù

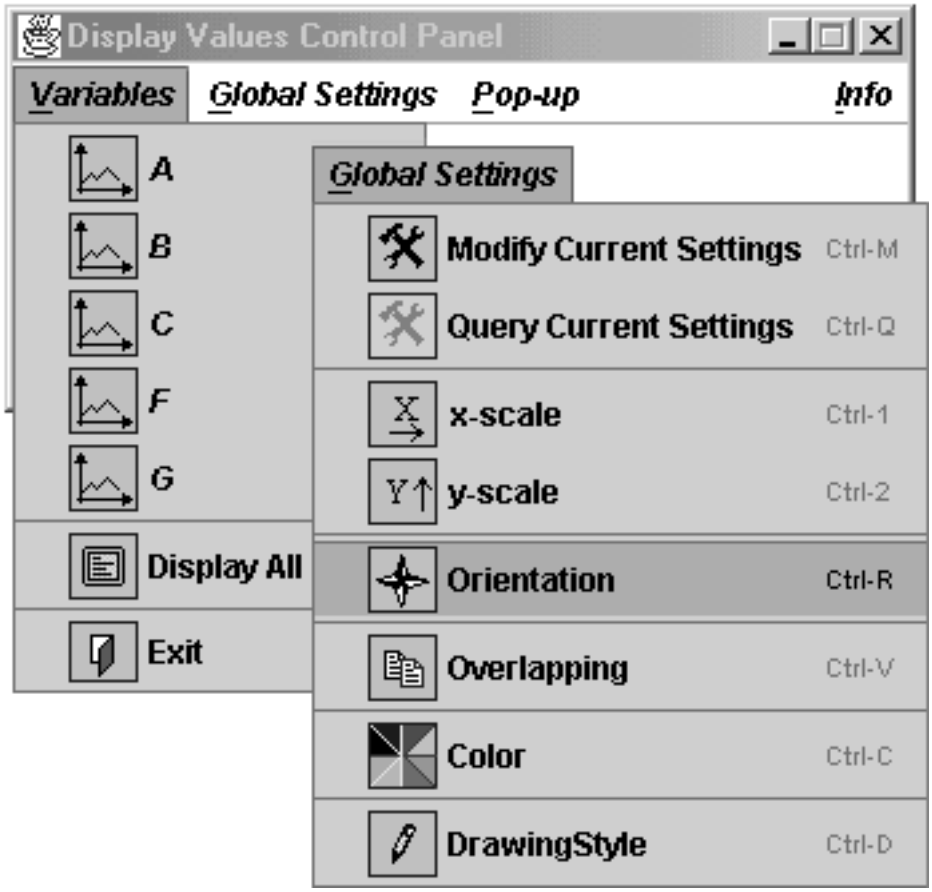

Figura 3.19: *I sottomenù del frame principale del Tool Display*

principali (*Variables*, *Global Settings* e $Pop-up$) e da due pulsanti (*Clear All* e *Close All*). Il sottomenù *Variables* contiene una parte dinamicamente determinata dall'insieme di variabili corrente (in figura 3.19 sono elencate, come esempio, la variabili "A", "B", "C", "F" e "G") e una parte statica con due voci fisse. La porzione dinamica del menù [13] consente di visualizzare gli andamenti di ciascuna variabile mentre la parte statica consente di visualizzare gli andamenti di tutte le variabili in $frame$ distinti o nello stesso $frame$ in funzione dei settaggi (comando *Display All*, cfr. oltre) oppure (comando $Exit$) di terminare il processo originato dalla esecuzione del Tool (caso di esecuzione "stand alone") oppure il thread al cui interno è in esecuzione il Tool (caso "slave").
Il menù *Global Settings* contiene un certo numero di voci per l'accesso ad un insieme di finestre ausiliarie mediante le quali l'utente può impostare i valori dei parametri "globali" del Tool (cfr. le figure 3.19 e 3.20).

[13] In tutti i casi come questo in cui non si crea ambiguità useremo indifferentemente le voci "menù" e "sottomenù" privilegiando la prima per brevità.

Il menù *Global Settings* consente all'utente di impostare:

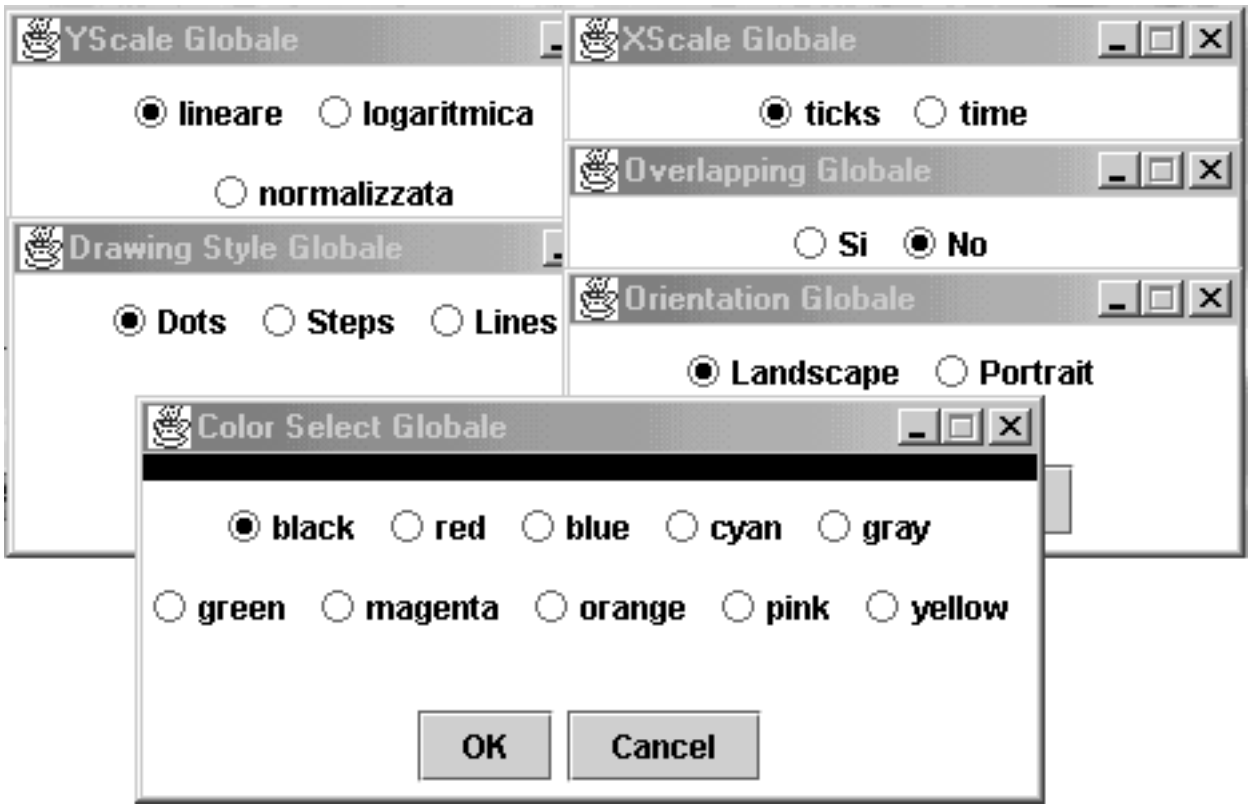

Figura 3.20: *Le finestre ausiliarie*

1. il tipo della scala dell'asse delle ascisse ($x - scale$),
2. il tipo della scala dell'asse delle ordinate ($y - scale$),
3. lo stile di tracciamento del singolo grafico($DrawingStyle$),
4. lo stile di tracciamento di un gruppo di grafici($Overlapping$),
5. l'orientamento del tracciamento ($Orientation$),
6. il colore del tracciamento ($Color$).

Ad ognuno di tali elementi corrisponde una finestra ausiliaria (cfr. la sezione 3.5.3). L'utente inoltre ha a disposizione due ulteriori finestre, una per il settaggio di tutti i parametri in una sola volta (*Modify Current Setings*) e una per la loro interrogazione (*Query Current Setings*).
L'insieme dei valori impostati dall'utente mediante le finestre di dialogo definisce lo *stato globale* del Tool, stato che ne condiziona il comportamento ma che può

essere modificato a livello del singolo frame di visualizzazione (cfr. le sezioni 3.5.3 e 3.5.4). Il menù *Pop-up*, infine, consente all'utente di portare in primo piano uno dei frame di visualizzazione (cfr. la sezione 3.5.4).

3.5.3 Le finestre ausiliarie, parametri "globali" e parametri "locali"

Al momento in cui il frame principale del Tool *Display* viene creato e visualizzato, si ha la definizione e la inizializzazione di un insieme di caratteristiche che definiscono lo *stato globale* del Tool.
Lo stato globale viene ereditato dai singoli *frame di visualizzazione* che sono, tuttavia, caratterizzati da elementi che consentono all'utente di modificare i valori di tali elementi in modo da definire uno *stato locale*, valido per il singolo frame di visualizzazione.
I valori che fanno parte dello stato globale sono già stati elencati nella parte finale della sezione 3.5.2. Per la loro modifica il Tool mette a disposizione dell'utente le seguenti finestre ausiliarie *non modali*[14](cfr. la figura 3.20).

1. XScale Globale,
2. YScale Globale,
3. Overlapping Globale,
4. Orientation globale,
5. Drawing Style Globale,
6. Color Select Globale,

   In più ci sono altre due finestre, che non verranno esaminate in dettaglio,

7. Modify Current Settings
8. Query Current Settings

che consentono, rispettivamente, la modifica di tutti i parametri mediante una sola finestra di dialogo oppure la loro interrogazione.
La finestra di dialogo *XScale Globale* permette di impostare la scala dell'asse delle X. I valori possibili sono i seguenti: *ticks* e *time*. Nel primo caso la scala sull'asse delle x viene tracciata da per i valori interi compresi fra N_{min} (di solito

[14]Con questo termine si definisce una finestra di dialogo che non obbliga l'utente ad interagire con essa e a chiuderla prima di poter proseguire ma che può rimanere aperta ed essere modificata, in questo caso trasmettendo i valori direttamente al programma associato, fino a che l'utente non decide di chiuderla.

pari a 0) e N_{max}, ovvero il numero di periodi di osservazione T durante i quali si vuole esaminare l'evoluzione delle variabili del modello. Nel secondo caso la scala dell'asse delle x viene tracciata usando consuete unità di tempo come *secondi*, *minuti* o *anni*.
La finestra di dialogo *YScale Globale* permette di impostare la scala dell'asse delle Y. I valori possibili sono:

1. lineare,
2. logaritmica,
3. normalizzata.

Nel caso di una scala *lineare* i valori delle variabili sono rappresentati così come sono, utilizzando tutto lo spazio disponibile in verticale sul *frame di visualizzazione* (cfr. oltre), nel caso della scala *logaritmica* si applica ai valori di una variabile una trasformazione logaritmica mentre nel caso della scale *normalizzata* i valori di una variabile sono mappati sull'intervallo $[0, 1]$ oppure, in valori percentuali, $[0, 100]$.
Nel caso della scala *logaritmica*, qualora una variabile assuma valori negativi (logaritmo non definito) o inferiori a 1 (logaritmo negativo) i suoi valori sono mappati sullo zero: in tal modo risulta possibile utilizzare la scala logaritmica nel caso si rappresentino gli andamenti di più variabili, alcune delle quali assumono valori negativi o minori di 1, su uno stesso riferimento cartesiano.
La finestra di dialogo *Orientation Globale* consente di fissare l'orientamento degli assi cartesiani. Si hanno due possibilità:

1. landscape: l'asse delle x è orientato, nel frame di visualizzazione, da sinistra a destra e quello delle y dal basso verso l'alto,
2. portrait: l'asse delle x è orientato, nel frame di visualizzazione, dall'alto verso il basso e quello delle y da sinistra verso destra.

La finestra di dialogo *Overlapping Globale* permette di stabilire se i frame di visualizzazione vengono creati senza che ci sia possibilità di sovrapporre i grafici (valore *no*) oppure con tale possibilità (valore *si*).
La finestra di dilaogo *Color Select Globale* consente di settare il valore del colore con il quale saranno tracciati i grafici. I colori possibili sono i seguenti: *black*, *red*, *blue*, *cyan*, *gray*, *green*, *magenta*, *orange*, *pink* e *yellow*. La selezione del colore ha effetto solo sui grafici che saranno tracciati dal momento del settaggio in poi.
La finestra di dialogo *Drawing Style Globale* consente di impostare lo stile di default di tracciamento dei grafici. Gli stili possibili sono tre:

1. dots,

2. steps,

3. lines.

Nel primo caso, in corrispondenza delle coppie di valori $(x[n], y[n])$[15] viene tracciato un marker cirolare colorato , delle dimensioni di un punto. Nel secondo caso si ha una rappresentazione stile *istogramma*. Si rappresentano, infatti, i valori $(x[n], y[n])$ e $(x[n+1], y[n])$, il segmento che li unisce e i segmenti verticali che uniscono tali punti con l'asse delle ascisse.
Nell'ultimo si usa una interpolazione lineare ovvero si rappresentano i segnemti che uniscono i punti $(x[n], y[n])$ e $(x[n+1], y[n+1])$.

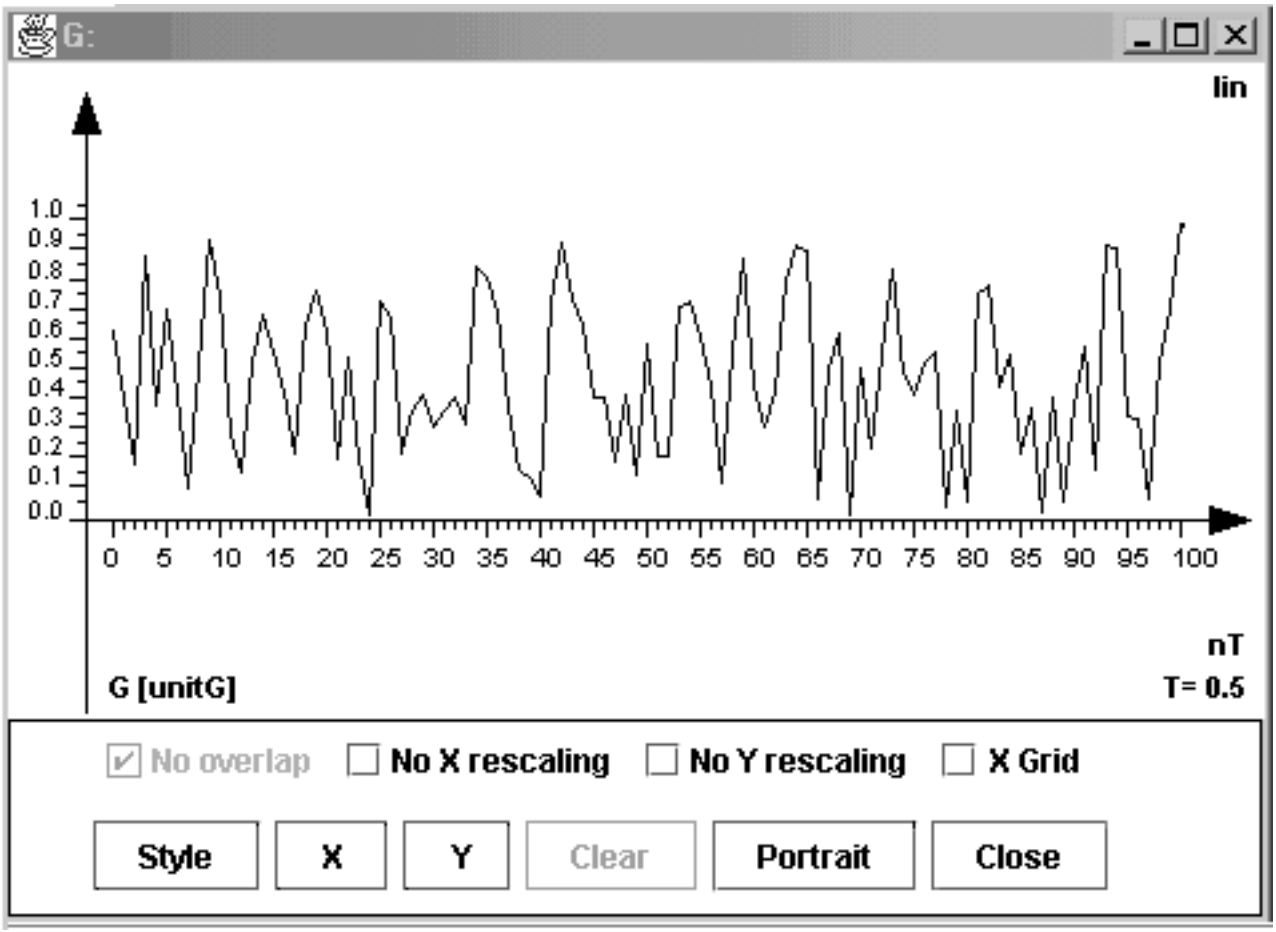

Figura 3.21: *Un esempio di frame di visualizzazione*

Mediante tali finestre di dialogo, l'utente definisce un insieme di valori globali caratteristici di tutti i frame di visualizzazione al momento della loro definizione.

[15]Per esemplificare si usa la notazione in *ticks* con n che varia su un intervallo $[0, N]$.

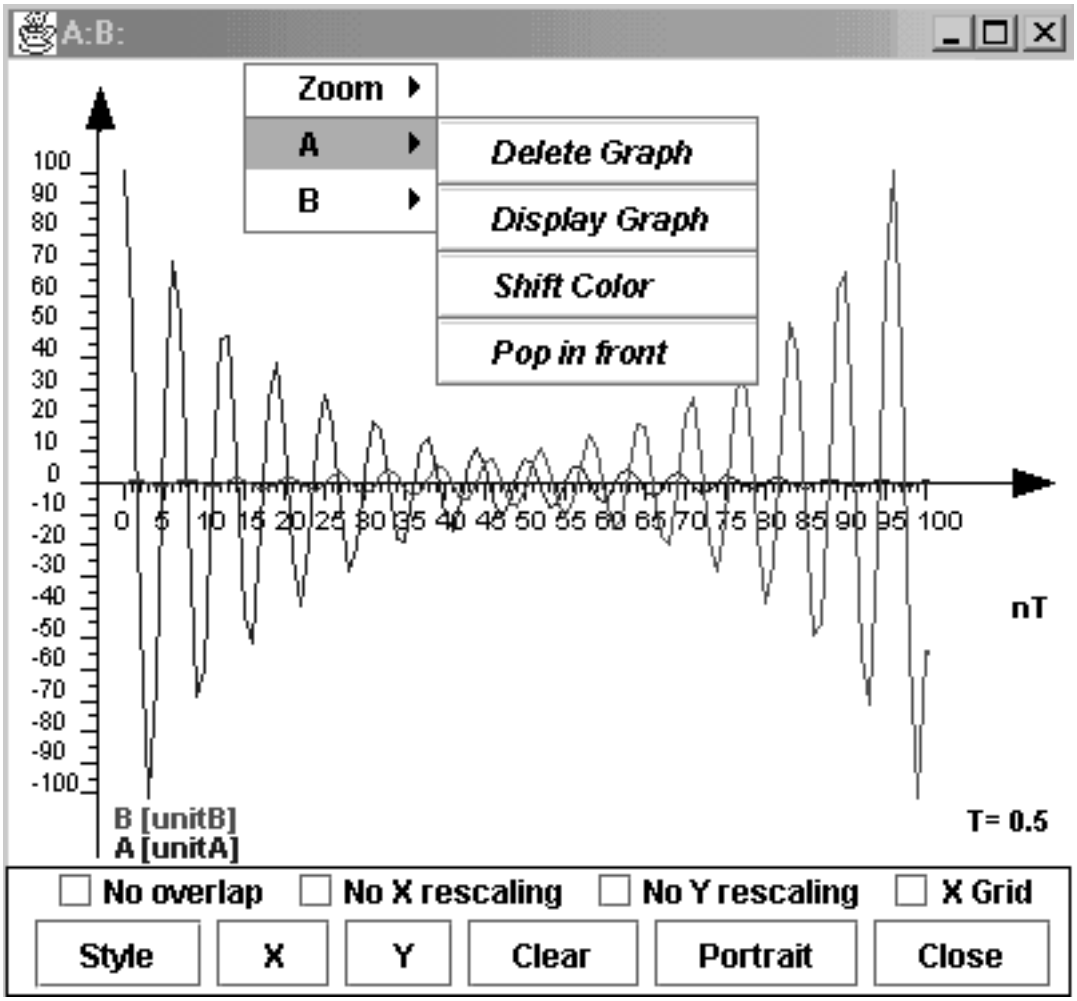

Figura 3.22: *Un frame di visualizzazione con due grafici sovrapposti*

Come risulta dalle figure 3.21 e 3.22, un frame di visualizzazione può essere customizzato mediante una serie di *pulsanti*, di *check box* e di un *menù flottante* che consentono di definire un settaggio locale e di eseguire alcune operazioni sui grafici visualizzati nel singolo frame di visualizzazione.
I pulsanti consentono di modificare lo stile di rappresentazione dei grafici visualizzati nel frame (pulsante *Style*), di modificare la scala delle x (pulsante X) e quella delle y (pulsante Y), di cancellare tutti i garfici tranne il primo tracciato (pulsante *Clear*), di modificare l'orientamento del frame da *landscape* a *portrait* (pulsante etichettato *Portait*) o viceversa (stesso pulsante etichettato *Landscape*). I check box permettono di impedire che sul frame vengano visualizzati ulteriori grafici (check box *No overlap* selezionato), che vengano modificate la scala dell'asse delle ascisse (check box *No X rescaling* selezionato e pulsante X disabilitato) opure la scala dell'asse delle ordinate (check box *No Y rescaling*

selezionato e pulsante *Y* disabilitato). Il check box *X Grid*, quando selezionato, fa si che sul frame compaia una griglia verticale di colore grigio che può agevolare la lettura dei grafici.

3.5.4 I frame di visualizzazione

I frame di visualizzazione, due esempi dei quali sono presentati nelle figure 3.21 e 3.22, consentono all'utente di visualizzare gli andamenti di un certo numero di variabili nel tempo.
Gli andamenti sono rappresentati sotto forma di grafici in un riferimento cartesiano bidimensionale e possono essere rappresentati anche sovrapposti, utilizzando colori diversi.
Di ogni variabile di cui si traccia il grafico, il frame contiene il nome e l'indicazione dell'unità di misura , che può aiutare ad identificare la variabile come un livello o come un flusso o come una variabile ausiliaria assimilabile, come comportamento, ad uno dei due tipi precedenti.
Nel caso il frame contenga un solo grafico (cfr. la figura 3.21) si ha la possibilità di modificare il colore del grafico con un menù flottante che contiene due voci:

1. Zoom in
2. Shift Color

La prima consente di evidenziare una parte del grafico, la parte centrale, in un frame indipendente dotato di pulsanti per lo scorrimento della porzione di grafico visualizzata.
La seconda consente di modificare il colore corrente del grafo con una operazione di shift ciclico sull'insieme dei colori di cui alla sezione 3.5.3: se il colore corrente del grafo, ad esempio, è *red* con una operazione di shift lo si cambia in *blue* e così via.
Nel caso che sul frame sia visualizzato più di un garfico, il menù flottante suddetto contiene, oltre alla voce *Zoom in* per ognuno dei grafici le seguenti voci:

1. Delete Graph,
2. Display Graph,
3. Shift Color,
4. Pop in front.

La prima (*Delete Graph*) consente di rimuovere selettivamente il grafico corrispondente, contrariamente al pulsante *Clear* che rimuove tutti i grafici meno il primo tracciato.
La seconda (*Display Graph*) consente di visualizzare il grafico corrispondente in

un frame indipendente utilizzando, per la visualizzazione, i valori globali del Tool. La terza (*Shift Color*) ha il significato prima detto e l'ultima (*Pop in front*) consente di alterare l'ordine di visualizzazione dei grafici portando quello selezionato in primo piano e modificando, pertanto, l'azione del pulsante *Clear*: nel caso un frame contenga più grafici, portando quello tracciato per primo in primo piano ed eseguendo una *Clear* si ottiene la cancellazione di tutti i grafici meno quello che in origine era stato tracciato per secondo.
La superfice del frame di visualizzazione, infine, può cambiare colore da bianco a tutte le sfumature del grigio in modo da consentire un miglior contrasto per la visualizzazione dei grafici e della griglia verticale (di colore grigio chiaro). Tale effetto lo si ottiene spostando il mouse con il pulsante sinistro premuto in diagonale sulla superfice del frame: se lo sfondo del frame è bianco tanto maggiore è lo spostamento tanto più scuro viene colorato lo sfondo. Se, viceversa, lo sfondo del frame è colorato la manovra ha l'effetto di riportarlo ad essere bianco.

3.6 *Equation Solver*

3.6.1 Introduzione

Il Tool *Flow Diagram Graphic Editor* (cfr. la sezione 3.4) consente all'utente

1. di impostare i parametri globali della simulazione (istante di inizio, di fine, valore dello step T e unità di misura della variabile tempo),
2. di tracciare un diagramma FD,
3. di assegnare ad ogni nodo tutti gli elementi necessari per la caratterizzazione della corrispondente equazione,
4. di eseguire sul grafo un certo numero di controlli di completezza.

Una volta che un diagramma FD sia stato tracciato, inizializzato e controllato è possibile utilizzare i metodi del Tool *Equation Solver* per ottenere le soluzioni delle equazioni associate ai nodi del diagramma FD e determinare gli andamenti nel tempo delle variabili caratteristiche del modello.
I metodi per la risoluzione delle equazioni sono accessibili in modo diretto oppure in modo indiretto. L'accesso diretto è reso possibile dalla voce *Solve&Display* del menù *Application* del Tool *Flow Diagram Graphic Editor*. L'accesso indiretto avviene attraverso un Tool ad hoc, detto *Equation Solver*, che sarà illustrato nella sezione 3.6.2.
In entrambi i casi, dato un diagramma FD cui corrisponde, in genere, un multigrafo, i passi da seguire per la risoluzione delle equazioni sono i seguenti.
Per prima cosa è necessario accedere alle informazioni che l'utente ha associato ad ogni nodo mediante l'*Equation Editor* (cfr. la sezione 3.4.5), informazioni che

rappresentano le singole *equazioni*.
Tali informazioni sono associate ai nodi del multigrafo corrispondente al diagramma FD correntemente visualizzato sul canvas oppure contenuto in un file.
Dopo aver ricavato le informmazioni caratteristiche delle equazioni è necessario ricavare le *dipendenze* fra le equazioni in modo da stabilire un ordinamento totale fra queste. L'ordinamento totale è ottenuto partendo dall'ordinamento parziale che esiste fra le equazioni scegliendo uno fra i molti ordinamenti totali possibili.
Dopo aver ottenuto le equazioni ordinate, è possibile risolverle con un procedimento iterativo di simulazione continua, a partire dai valori iniziali delle variabili "indipendenti" (in genere *livelli*, *costanti* e *variabili ausiliarie*). Tali variabili sono dette "indipendenti" in quanto caratterizzate da valori iniziali che non dipendono da altre variabili e che comunque sono ben definiti. Tipiche variabili "dipendenti" sono, invece, i $flussi$.
Il procedimento di simulazione viene fatto iterare fra gli istanti *iniziale* (T_{start}) e $finale$ (T_{end}) per incrementi della variabile tempo pari al valore della variabile T, detta *time_step* o anche *intervallo di osservazione*.
A fine simulazione si ottengono i valori assunti dalle variabili caratteristiche del modello sotto forma di tabelle di valori che possono essere visualizzati in riferimenti cartesiani bidimensionali (cfr. la sezione 3.5).

3.6.2 Il Tool *Equation Solver*

Il Tool *Equation Solver* è caratterizzato dall'interfaccia illustrata dalla figura 3.23. I comandi presenti nel menù principale del Tool hanno lo scopo di consentire l'accesso sia a file di tipo oggetto sia a file di tipo testo, sui quali possono essere eseguite operazioni di controllo prima che sia possibile risolvere le equazioni associate ai nodi di un diagramma FD e visualizzare i risultati della simulazione.

Per l'accesso ai file il Tool possiede i comandi *Open...* e *Import...*[16]. L'esecuzione di tali comandi ha come conseguenza la creazione delle strutture dati caratteristiche di un multigrafo annotato, ovvero ad ogni nodo del quale sono associate le informazioni caratteristiche della equazione relativa.
Una volta che siano note la struttura del grafo e le equazioni associate ai nodi è possibile utilizzare i comandi del menù *Application* per l'esecuzione:

1. di operazioni di controllo sul diagramma FD, comando $CheckGraph$,
2. di una simulazione del grafo, comando $SolveGraph$,
3. di visualizzazione dei risultati della simulazione, comando $DisplayGraphs$.

[16]Si trascura in questa sede il comando *Print...* che, come nel caso di altri Tool, consente la stampa del contenuto di file di tipo testo.

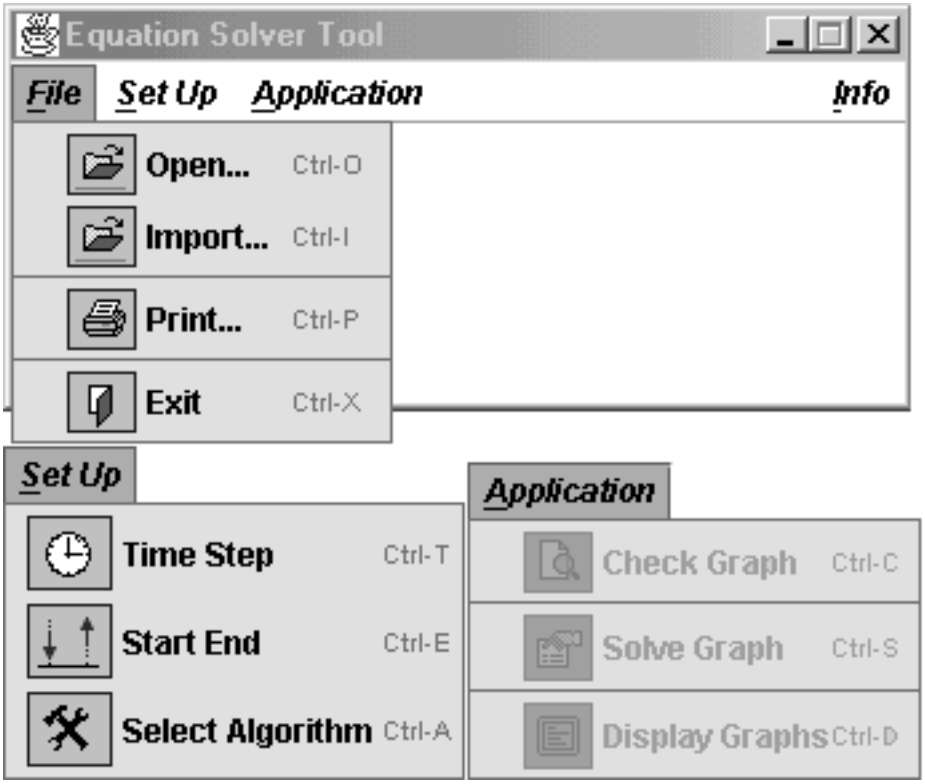

Figura 3.23: *Il Tool Equation Solver*

Il comando *CheckGraph* esegue sul diagramma corrente gli stessi controlli del comando *Check to solve* del menù *Application* del Tool *Flow Diagram Graphic Editor*.
Qualora l'esito del controllo sia positivo è possibile simulare il diagramma (comando *SolveGraph*) e, a simulazione terminata, visualizzare gli andamenti delle variabili del modello (comando *DisplayGraphs*). Il comando *SolveGraph* esegue una simulazione del diagramma FD corrente, utilizzando i valori attuali dei parametri di simulazione (gli istanti *iniziale* (T_{start}) e $finale$ (T_{end}) e la variabile T) e definisce gli andamenti nel tempo delle variabili caratteristiche del modello. Il comando *DisplayGraphs* interagisce con il Tool *Display* (cfr. la sezione 3.5) in modo da ottenere la visualizzazione di tali andamenti.
Qualora il controllo dia esito negativo sarà necessario rieditare il diagramma con il *Flow Diagram Graphic Editor* apportando le necessarie correzioni.
Dato un diagramma FD ad esso sono stati associati, mediante il Tool *Flow Diagram Graphic Editor*, i valori degli istanti di inizio e di fine di una simulazione e il valore della variabile T. Tali valori possono essere modificati utilizzando i comandi del menù *Set Up* che consentono:

1. di modificare il valore della variabile *time_step* T,

2. di modificare gli istanti di inizio e di fine della simulazione.

Il menù *Set Up*, infine, permette all'utente di scegliere l'algoritmo con cui saranno risolte le equazioni alle differenze finite associate ai nodi di un diagramma FD. Gli algoritmi che saranno resi diponibili sono i seguenti:

1. Eulero,
2. Runge-Kutta del secondo ordine,
3. Runge-Kutta del quarto ordine.

3.7 *I convertitori da CL a FD e viceversa*

3.7.1 Introduzione

I Tool di conversione (*CLtoFD Tool* e *FDtoCl Tool*) consentono all'utente di tradurre la descrizione di un diagramma CL nella descrizione di un diagramma FD e viceversa.
L'utente accede ai Tool

1. in modo diretto,
2. in modo indiretto.

L'accesso in modo diretto ai Tool di conversione è reso possibile da una voce presente nel menù *Aplication* sia del *Causal Loop Graphic Editor* (*Convert to FD*) sia del *Flow Diagram Graphic Editor* (*Convert to CL*). In entrambi i casi la conversione è possibile solo dopo che sul diagramma correntemente presente sul canvas sono state eseguite alcune operazioni di controllo che hanno avuto esito positivo. In entrambi i casi, infatti, i metodi utilizzati accedono alle strutture dati del diagramma correntemente visualizzato ma presuppongono che il diagramma stesso sia stato configurato in modo da poter essere convertito (cfr. la sezione 3.7.3).
L'accesso indiretto è possibile:

1. attraverso le voci *CL to FD* e *FD to CL* del menù *Convert* del Tool *TopLevel*,
2. attraverso un Tool dedicato detto *Converter Tool*.

Nel caso dell'accesso indiretto, l'utente utilizza metodi di conversione che accedono alle strutture dati di diagrammi contenuti in file di tipo oggetto o di tipo testo (cfr. la sezione 3.7.2). In questo caso, oltre ai metodi per la conversione, entrano in gioco metodi per l'accesso ai file e metodi per la creazione delle strutture dati che saranno utilizzate per la conversione vera e propria. Nella sezione 3.7.2

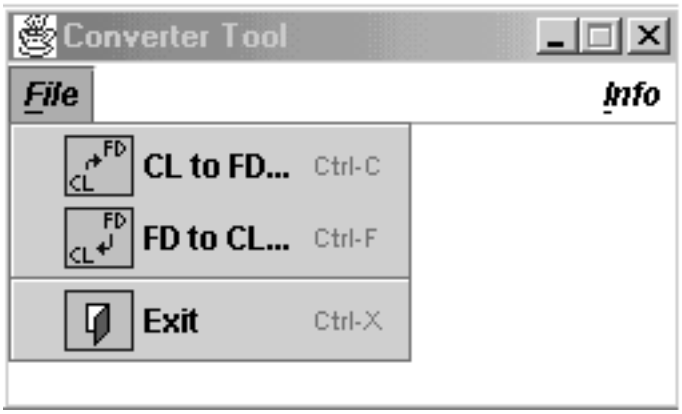

Figura 3.24: Il *Converter Tool*

viene data una descrizione "lato utente" dei convertitori come acceduti mediante il *Converter Tool*, dal momento che le operazioni di base, che sono eseguite per effettuare la conversione da un formato ad un altro, non cambiano negli altri casi (cfr. anche la sezione 3.7.3 dove si presentano le operazioni nel caso dell'accesso diretto).

3.7.2 L'interfaccia utente del *Converter Tool*

Il *Converter Tool* si presenta all'utente con l'interfaccia illustrata dalla figura 3.24. L'interfaccia è caratterizzata dal solo menù *File* che contiene le voci:

1. CL to FD,
2. FD to CL,
3. Exit.

L'ultima voce (*Exit*), come nel caso di altri Tool, consente all'utente di terminare il processo originato dalla esecuzione del Tool (caso di esecuzione "stand alone") oppure il thread al cui interno è in esecuzione il Tool (caso "slave").
Ciascuna delle altre due voci dà luogo alla comparsa di una ulteriore finestra di dialogo (cfr. la figura 3.25 in cui sono state rappresentate entrambe le finestre), finestra che si ottiene anche utilizzando la corrispondente voce del menù *Convert* del Tool *TopLevel*.
Le finestre di dialogo per l'interazione con le routine di conversione vere e proprie differiscono solo nell'header (*CLtoFD Tool* per la prima e *FDtoCL Tool* per la seconda) dal momento che entrambe contengono gli stessi tre pulsanti:

1. Convert object file

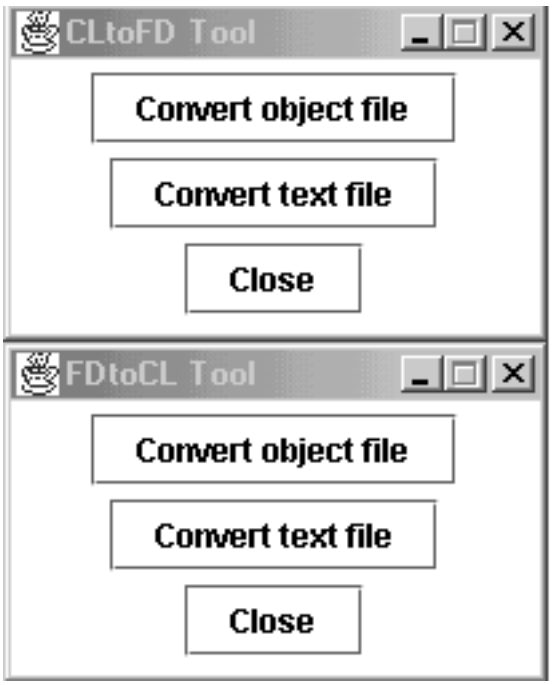

Figura 3.25: *Le interfacce dei convertitori*

2. Convert text file
3. Close

La conversione può essere eseguita, infatti, sia su file di tipo *object* (la cui estensione è *ocl* nel caso dei diagrammi CL e ofd nel caso di diagrammi FD) sia su file di tipo *text* (la cui estensione è, invece, *tcl* nel caso dei diagrammi CL e tfd nel caso di diagrammi FD). I file del primo tipo vengono creati sfruttando l'interfaccia *Serializable* di *Java* (cfr. la sezione 4.3.5) mentre quelli del secondo tipo vengono creati utilizzando stringhe di testo delimitato (cfr. sempre la sezione 4.3.5).
In entrambi i casi i processi di conversione da diagramma CL a diagramma FD e quello da diagramma FD a diagramma CL sono caratterizzati dai passi seguenti:

1. accesso al file che contiene la descrizione del diagramma da convertire,
2. esecuzione di controlli di congruenza sulla struttura del diagramma,
3. generazione della descrizione della struttura del nuovo diagramma e suo salvataggio in un nuovo file con lo stesso nome del file originario e con la opportuna estensione.

In merito ai controlli di congruenza si fa notare come la loro esecuzione sia associata a comandi espliciti nel caso dell'accesso diretto (*Check to Convert* per il

Flow Diagram Graphic Editor e *Check Graph* per il *Causal Loop Graphic Editor* mentre venga eseguita in modo trasparente nel caso dell'accesso indiretto. Nel primo caso se i controlli non hanno esito positivo non è possibile eseguire la conversione (e il corrispondente comando, *Convert to FD* o *Convert to CL*, risulta disabilitato), nell'altro caso il Tool visualizza un messaggio di errore e non esegue la conversione. In questa eventualità è necessario riaprire il diagramma con il Tool grafico opportuno ed apportare le necessarie correzioni.

3.7.3 La conversione dei diagrammi CL in diagrammi FD e viceversa

Gli elementi dei diagrammi CL vengono istanziati, mediante i comandi del *Causal Loop Graphic Editor*, come non tipizzati. Gli archi vengono creati con un segno indefinito (indicato come $< none >$ o n nei diagrammi) e con un tipo non specificato (indicato come $< unspecified >$ nei diagrammi) mentre i nodi possiedono un tipo indefinito (indicato come $< none >$ nei diagrammi).
Utilizzando elementi non tipizzati (archi e nodi) si possono creare diagrammi CL con loop di feedback positivo e negativo caratterizzati anche da elementi che definiscono le interazioni del mondo esterno con il sistema modellizzato.
La conversione di un diagramma CL nel corrispondente diagramma FD (cfr. la sezione 1.3.3) richiede necessariamente che gli elementi del diagramma siano tipizzati ovvero che

1. ad ogni nodo sia assegnato un tipo,
2. ad ogni arco sia assegnato un tipo,
3. ad ogni arco sia assegnato un segno.

I tipi possibili per i nodi sono i seguenti:
"level", "rate", "source", "sink", "delay","constant" e "auxiliary".
Mediante il tipo si individua il ruolo di ognuno dei nodi all'interno di un diagramma FD e si stabilisce il tipo e il verso degli archi in esso incidenti.
Ad esempio nodi di tipo "source" possono avere come archi uscenti solo archi di tipo "Materials" mentre nodi di tipo "constant" possono avere come archi uscenti solo archi di tipo "Information". In modo analogo nodi di tipo "sink" possono avere come archi entranti solo archi di tipo "Materials" mentre nodi di tipo "auxiliary" possono avere archi entranti e uscenti solo di tipo "Information". I nodi di tipo "level" hanno come archi entranti archi di tipo "Materials" e come archi uscenti archi di tipo "Materials", verso nodi di tipo "rate", o di tipo "Information", verso nodi di tipo "rate" o di tipo "auxiliary".
I segni possibili per gli archi sono: "+" e "-". Il primo caratterizza un arco in un diagramma CL come descrivente una relazione causa-effetto di proporzionalità

diretta mentre il secondo caratterizza un arco in un diagramma CL come descrivente una relazione causa-effetto di proporzionalità inversa. Il segno su un arco in un diagramma CL permette di determinare il verso dell'arco corrispondente nel diagramma FD, una volta che siano noti i tipi dei nodi alle sue stremità e il tipo dell'arco.
Ad esempio un segno "+" su un arco può individuare un arco di tipo "Materials" uscente da un nodo di tipo "rate" ed entrante in un nodo di tipo "level" oppure un arco di tipo "Information" uscente da un nodo di tipo "level" ed entrante in un nodo di tipo "rate". In modo simile un segno "-" su un arco può individuare un arco di tipo "Materials" uscente da un nodo di tipo "level" ed entrante in un nodo di tipo "rate" oppure un arco di tipo "Information" uscente da un nodo di tipo "level" ed entrante in un nodo di tipo "rate". Considerazioni analoghe si possono fare per gli altri tipi di nodi.
I tipi possibili per gli archi sono: "Information" e "Materials": il primo caratterizza un flusso come un flusso di tipo non conservativo mentre il secondo caratterizza un flusso come un flusso di tipo conservativo.
I flussi di tipo conservativo coinvolgono nodi dei tipi "level", "rate", "source", "sink" e "delay" e sono caratterizzati dal fatto che per ogni nodo devono essere soddisfatte le equazioni di bilancio mentre i flussi di tipo non conservativo coinvolgono anche gli altri tipi di nodi e sono caratterizzati dalla possibilità che nuova informazione venga creata in ogni punto del modello senza che debbano essere soddisfatti vincoli di bilancio.
La conversione di un diagramma CL nel corrispondente diagramma FD, pertanto, prevede:

1. la tipizzazione degli elementi di un diagramma CL,
2. l'esecuzione di operazioni di verifica su tali elementi tipizzati,
3. una modifica della rappresentazione pittorica,
4. una modifica della struttura del grafo soggiacente la rappresentazione pittorica.

Gli ultimi due passi sono, ovviamente, possibili solo se le operazioni di verifica hanno avuto esito positivo.
Le operazioni di tipizzazione devono essere eseguite singolarmente sui singoli elementi di un diagramma CL utilizzando gli opportuni comandi del Tool *Causal Loop Graphic Editor* (i comandi *Set/Change sign* e *Set/Change type* per gli archi e *Set/Change type* per i nodi).
Le operazioni di verifica sono eseguite utilizzando il comando *Check Graph* del menù *Appplication* e constano dei passi seguenti:

1. controllo della presenza del segno su tutti gli archi,

2. controllo della assegnazione di un tipo a tutti gli archi,
3. controllo della assegnazione di un tipo a tutti i nodi,
4. controllo della adeguatezza dei tipi dei nodi con i tipi degli archi in essi incidenti.

Una volta che il diagramma CL sia stato etichettato come corretto lo si può effettivamente convertire in un diagramma FD. A tale scopo la rappresentazione pittorica deve essere modificata. Per fare ciò, alle etichette sono sostituite coppie icona/etichetta dei tipi opportuni e con i valori corretti delle etichette e gli archi sono orientati e tipizzati.
Oltre ad agire sulla rappresentazione pittorica è necessario modificare la struttura del grafo ad essa soggiacente. A livello del grafo vengono aggiornati sia i nodi sia gli archi. I nodi vengono aggiornati ed estesi con l'aggiunta di campi per il tipo e per l'equazione caratteristica di ogni nodo in un diagramma FD. Per ogni arco si aggiornano sia il verso sia le informazioni relative al tipo.
La conversione di un diagramma FD nel corrispondente diagramma CL segue una strada diversa dal momento che nel passaggio:

1. vanno perdute tutte le informazioni relative al tipo dei nodi,
2. vanno perdute tutte le informazioni relative al tipo degli archi,
3. vengono aggiunte le informazioni relative al segno degli archi.

Le informazioni relative al segno degli archi sono ricavate dal tipo e dal verso di ciascun arco nel diagramma FD.
La conversione di un diagramma FD nel corrispondente diagramma CL, pertanto, prevede:

1. l'esecuzione di operazioni di verifica sugli elementi del diagramma FD,
2. una modifica della rappresentazione pittorica,
3. una modifica della struttura del grafo soggiacente la rappresentazione pittorica.

Le operazioni di verifica (comando *Check to Convert* del menù *Application* del *Flow Diagram Graphic Editor*) si limitano a controllare che gli elementi del diagramma FD formino un grafo connesso ma tale condizione non è vincolante perchè sia possibile operare la conversione. Le operazioni di modifica della rappresentazione pittorica permettono di ottenere una rappresentazione pittorica composta da etichette e da elementi di connessione non tipizzati dove gli elementi di connessione hanno ciascuno il suo segno. Le operazioni di modifica del grafo corrispondente permettono di ottenere un grafo orientato con nodi e archi caratterizzati dalle informazioni di cui alla sezione 3.3.

Capitolo 4

D(a)ySy Tool Box : la struttura interna

4.1 Introduzione

La struttura interna dell'ambiente *D(a)ySy Tool Box* viene presentata in questo Capitolo utilizzando diagrammi che illustrano le relazioni fra le classi principali e rimandando all'Appendice A per una rappresentazione più formale eseguita usando il linguaggio UML ([BSL02]).
L'ambiente *D(a)ySy Tool Box* è stato sviluppato utilzzando il linguaggio *Java*, precisamente la versione *Java2 SDK Standard Edition v 1.3.1* ([Eck98], [HC99] e [WM99]).
Per lo sviluppo sono state utilizzate esclusivamente le classi standard di tale versione in modo da definire un insieme di Tool utilizzabili su qualunque sistema di calcolo dotato della opportuna *Java Virtual Machine* ed in più sono stati adottati alcuni semplici accorgimenti in modo da consentire l'uso dei Tool sia negli ambienti operativi caratterizzati dall'uso di mouse a tre tasti sia in quelli caratterizzati dall'uso di mouse ad un solo tasto.
Dato il suo obbiettivo, la struttura del presente Capitolo rispecchia quella del Capitolo 3: le varie sezioni in cui il Capitolo è suddiviso corrispondono ad analoghe sezioni del Capitolo 3 e in ciascuna di esse viene data una descrizione delle classi utilizzate per lo sviluppo di ciascun Tool, delle relazioni fra le classi e della logica interna che presiede al funzionamento del Tool mentre per informazioni relative sia all'interfaccia utente del Tool sia al suo funzionamento "lato utente" si rimanda alla sezione corrispondente del Capitolo 3. In particolare la Tabella 3.1 e le figure 3.1 e 3.2 permettono di di ricondurre le descrizioni dei metodi propri dalle varie classi alla loro collocazione logica.

4.2 *TopLevel*

La figura 4.1 rappresenta le classi principali che fanno parte del Tool *TopLevel*[1]: la classe *Top* contiene il metodo *public static void main(String[] args)* che consente l'esecuzione del Tool in modalità stand-alone.

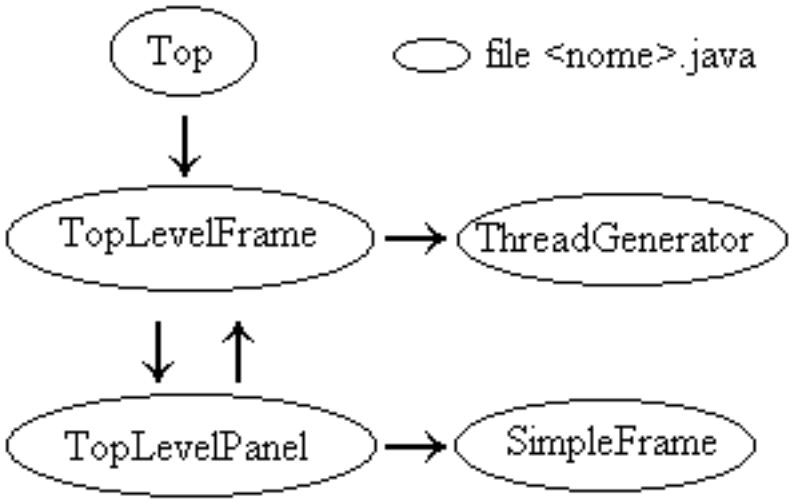

Figura 4.1: *TopLevel: la struttura interna*

La classe *Top* definisce un oggetto della classe $TopLevelFrame$, lo visualizza, ne imposta le dimensioni sullo schermo come costanti ed inizializza un contatore: come risulta dalla figura 4.1, la classe $TopLevelFrame$ può essere istanziata anche dall'interno della classe $TopLevelPanel$ (come mostrano le due frecce presenti fra tali classi) e il contatore permette di avere al più tre istanze della classe $TopLevelFrame$ presenti sullo schermo ad un dato istante.
La classe $TopLevelFrame$, a sua volta, permette di creare oggetti caratterizzati da un menù e da tre pulsanti contenuti in un $JPanel$ istanza della classe $TopLevelPanel$.
Le relazioni esistenti fra tali classi (*Top*, $TopLevelFrame$ e $TopLevelPanel$) sono rappresentate nella figura 4.1 utilizzando degli archi orientati dalla classe contenente alla classe contenuta in modo da rappresentare una relazione del tipo $has-a$: in tal modo un oggetto della classe *Top* contiene oggetti della classe $TopLevelFrame$ che a loro volta contengono oggetti delle classi $SimpleFrame$ e $TopLevelFrame$. Il menù caratteristico di un oggetto tlf del-

[1]Come è illustrato dalla figura 4.1 le classi racchiuse in ovali corrispondono a file di tipo *.java* mentre, come sarà illustrato in successive figure, se una classe A è contenuta in un file $B.java$ viene rappresentata come racchiusa in un rettangolo.

la classe $TopLevelFrame$[2] contiene voci che permettono all'utente di interagire direttamente con altri moduli dell'ambiente *D(a)ySy Tool Box* ovvero:

1. gli editor per diagrammi CL e diagrammi FD,
2. i convertitori da CL a FD e da FD a CL

ciascuno dei quali viene mandato in esecuzione in un suo proprio *thread*.
Ognuna di tali voci di menù gestisce la creazione di un oggetto tg della classe $ThreadGenerator$ cui corrisponde la attivazione di un *thread*[3] separato: all'interno di ciascun oggetto tg si ha la istanziazione di un oggetto di una classe "master" del Tool corrispondente.
L'uso dei *thread* ([Bis99]) permette di ottenere diverse computazioni disponibili simultaneamente e tutte in grado di raegire rapidamente alle richieste immesse dall'utente attraverso le rispettive interfacce grafiche.
Nel caso dei Tool, una classe "master" è una classe la cui istanziazione permette di creare un oggetto responsabile della istanziazione in cascata di tutte le classi necessarie per la esecuzione del Tool in questione: ad esempio, nel caso degli editor per diagrammi CL e FD (cfr. rispettivamente le sezioni 4.3 e 4.4) la classe "master" è responsabile della creazione del $frame$[4] che implementa (almeno in parte) l'interfaccia utente del Tool.
I pulsanti (cfr. la figura 3.3) presenti su ciascun oggetto tlf (della classe $TopLevelFrame$) e associati ad un oggetto tlp (della classe $TopLevelPanel$) consentono di istanziare nuovi oggetti tlf della classe $TopLevelFrame$ (il pulsante etichettato $Clone$) oppure oggetti sf della classe $SimpleFrame$ (il pulsante etichettato $New\ Viewer$) mentre l'ultimo pulsante (quello etichettato come $Close\ All$) svolge un'azione gestita dal listener[5] $CloseAllListener$ interno alla classe $TopLevelPanel$.
Ciascuno degli oggetti sf si occupa, a sua volta, della creazione di un $frame$ caratterizzato da una interfaccia con menù e pulsanti e da un *canvas*[6] su cui l'utente può visualizzare un diagramma CL o FD a seconda dei casi.
Le strutture dati utilizzate dalle varie classi che compongono il Tool *TopLevel*

[2]Per semplicità espositiva gli oggetti generici di una classe, salvo avviso contrario, saranno individuati da nomi composti dalle lettere minuscole che nel nome della classe compaiono in maiuscolo per cui un oggetto della classe $TopLevelFrame$ sarà individuato dal nome tlf, eventualmente seguito da un indice numerico.

[3]Un *thread* è un sottoprocesso responsabile della esecuzione di un flusso separato di istruzioni all'interno di un unico processo ([Bis99]).

[4]Un $frame$ ([HC99]) è una finestra top-level ovvero non contenuta in nessun altra finestra ed è una istanza della classe standard $JFrame$.

[5]Un listener è un metodo responsabile della esecuzione di operazioni associate tipicamente a componenti dell'interfaccia utente quali pulsanti e menù e ad azioni dell'utente quali selezione di un pulsante, spostamento del mouse, pressione di uno dei pulsanti del mouse e così via.

[6]Un *canvas* è un'area di lavoro su cui l'utente può disegnare oggetti e posizionare stringhe di testo. Tipici *canvas* sono i *panel*, oggetti istanze della classe $JPanel$.

sono limitate ad un certo numero di vettori , ovvero di istanze della classe standard $Vector$, e array di interi, $int[]$. La classe $TopLevelFrame$ utilizza un array $int[]$ $clonesId$ per tener conto del numero effettivo di istanze della classe $TopLevelFrame$ effettivamente in esecuzione e un $Vector$ $vectorOfThreads$ per la gestione dei *threads* attivi in un dato istante. La classe $TopLevelPanel$, a sua volta, utilizza un elemento della classe $Vector$ $frames$ in modo da avere, ad ogni istante, un elenco dei riferimenti agli elementi della classe $SimpleFrame$ effettivamente presenti su cui opererà il listener associato al pulsante $CloseAll$ causandone la chiusura e la scomparsa dallo schermo.

4.3 *Causal Loop Graphic Editor*

Il *Causal Loop Graphic Editor* è caratterizzato da un certo numero di classi, con metodi e strutture dati, che consentono di implementare:

1. la rappresentazione pittorica di un grafo orientato,
2. il grafo orientato,
3. la caratterizzazione del grafo,
4. i sottografi del grafo orientato e le relative rappresentazioni pittoriche,
5. le interazioni fra il grafo e la relativa rappresentazione pittorica,
6. le interazioni con il Sistema operativo ospite.

Il *Causal Loop Graphic Editor* è caratterizzato da una classe[7] $CLedit$ che si occupa di creare il $frame$ principale dell'applicazione mediante la creazione di un oggetto della classe $ClMainFrame$ e la sua visualizzazione. Il $frame$ principale dell'applicazione (cfr. la figura 3.7) gestisce le interazioni con le classi responsabili della creazione dei pulsanti ($ButtonPanel$[8]) e della creazione del *canvas* su cui l'utente posiziona *etichette* e *elementi di connessione* ($DrawPanel$): alle etichette corrispondono i *nodi* del grafo mentre agli elementi di connessione (*archi*, *linee* e *polilinee*) corrispondono gli *archi* del grafo.
La classe $DrawPanel$ istanzia un oggetto della classe $Draw$, un oggetto della classe $Graph$ e un ogggetto della classe $GraphCheck$: la classe $Draw$ implementa le strutture dati che definiscono la rappresentazione pittorica di un grafo, la

[7]Le classi sono di regola contenute in file omonimi di estensione *.java* (per cui la classe A è contenuta in $A.java$) sebbene in alcuni casi, illustrati dalle figure della sezione 4.3 e dal contenuto dell'Appendice A, una classe B possa essere contenuta in un file di nome diverso.

[8]Di norma le classi il cui nome contiene la parola $Frame$ ereditano dalla classe, standard del package $Swing$, $JFrame$ mentre quelle il cui nome contiene la parola $Panel$ ereditano dalla classe, standard del package $Swing$, $JPanel$.

classe *Graph* implementa le strutture dati che definiscono la struttura astratta grafo e, infine, la classe *GraphCheck* contiene tutti i metodi necessari per i controlli di congruenza di un grafo (vedi oltre), l'individuazione dei cicli e la definizione dei relativi segni (cfr. la sezione 1.3.3).
Il legame fra la rappresentazione pittorica (il disegno) e la rappresentazione astratta (il grafo) è rappresentato dalla classe *Draw2Graph* che traduce le operazioni eseguite dall'utente sul *canvas* e catturate dai metodi della classe *DrawPanel* nella semantica sia della rappresentazione pittorica (invocando gli opportuni metodi della classe *Draw*) sia del grafo (invocando gli opportuni metodi della classe *Graph*).

4.3.1 La rappresentazione pittorica

Gli elementi costitutivi della rappresentazione pittorica di un grafo (cfr. la figura 4.2) sono:

1. le etichette[9],
2. gli elementi di connessione con i relativi *handle*,
3. i segni sugli elementi di connessione,
4. i segni sugli anelli.

Le *etichette* sono oggetti istanze della classe *LabelPanel* e ciascuna etichetta ha associato un menù flottante, istanza della classe *LabelPanelPopUpMenu*[10] contenente le operazioni eseguibili su ciascuna etichetta.
Il fatto di usare elementi derivati per ereditarietà dalla classe *JPanel* ci consente di definire elementi attivi in grado di catturare facilmente eventi associati a pressioni e spostamenti del mouse. Ogni etichetta è caratterizzata da un certo numero di parametri quali:

1. il nome,
2. un identificativo numerico (o *stamp*),
3. il tipo.

Ogni etichetta ha, infatti, un *nome* che coincide con il nome della variabile associata al nodo del grafo corrispondente, un identificativo numerico, che ne permette l'identificazione sia all'interno della rappresentazione pittorica sia all'interno del

[9]Si ricorda che una *etichetta* è la rappresentazione pittorica di un *nodo* del grafo sottof orma di una stringa racchiusa in un rettangolo.

[10]Come regola, se gli oggetti istanze di una classe A hanno associato un menù flottante questo è ottenuto come istanza della classe $APopUpMenu$ derivata dalla classe $PopUpMenu$, estensione della classe standard di *Swing*, $JPopUpMenu$.

grafo, e un tipo. L'identificativo numerico viene utilizzato in tutte le operazioni che riguardano l'etichetta mentre il tipo, inessenziale nel caso di diagrammi CL, è indispensabile per la conversine dei diagrammi da CL a FD.
Le operazioni eseguibili sul nodo mediante il suo identificativo sono la rimozione (ovvero la rimozione del *panel* che implementa l'etichetta dalla superfice del *canvas*), lo spostamento (ovvero il trascinamento, mediante il mouse, del *panel* che implementa l'etichetta sulla superfice del *canvas*) e la customizzazione (ovvero la modifica del nome e del tipo).
Gli *elementi di connessione* possono essere *archi*, *linee* o *polilinee*, sono orientati, nel senso che hanno una freccia che ne indica il verso, e a ciascuno di essi sono associati:

1. uno (nel caso di archi e linee) o più handle (nel caso delle polilinee),
2. un segno,
3. un tipo.

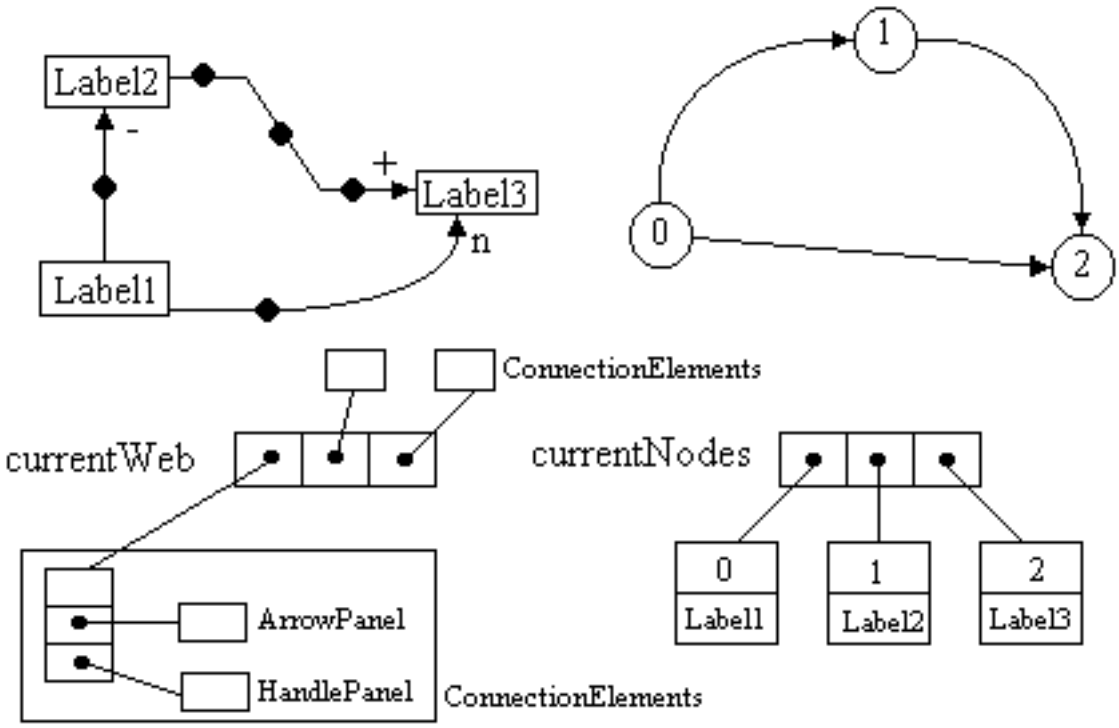

Figura 4.2: *Le strutture dati (semplificate) della "rappresentazione pittorica"*

Gli handle permettono di customizzare ogni singolo elemento di connessione, sono implementati come istanze della classe *HandlePanel* e hanno associato un

menù flottante istanza della classe $HandlePanelPopUpMenu$: il menù flottante permette di richiamare metodi della classe $Draw2Graph$ che permettono di:

1. rimuovere l'elemento di connessione
2. cambiare il verso dell'elemento di connessione, compatibilmente con i vincoli imposti dal grafo,
3. cambiare il segno dell'elemento di connessione,
4. cambiare il tipo dell'elemento di connessione, scegliendone uno fra quelli significativi per i diagrammi FD ($Information$ o $Materials$) o lasciarlo non spedcificato (il valore di default, $unspecified$).

Il segno (che può assumere uno dei valori n o $unspecified$, $+$ e $-$) è rappresentato mediante un oggetto istanza della classe $ArrowPanel$ e può essere posizionato relativamente alla freccia che individua la direzione dell'elemento di connessione mentre il tipo è codificato con un colore contenuto in un campo della classe $ConnectionElements$ (ed è reso accessibile da un insieme di metodi che ne consentono la query e il settaggio[11]).
Oltre ai segni sugli elementi di connessione, la rappresentazione pittorica prevede i segni degli anelli presenti nel grafo corrispondente (cfr. la sezione 4.3.3): gli anelli sono individuati analizzando il grafo associato alla rappresentazione pittorica con metodi della classe $GraphCheck$ per cui si rimanda alla sezione 4.3.3. Da un punto di vista pittorico gli anelli sono rappresentati con icone disegnate su elementi istanze della classe $SignPanel$ e associati alle etichette dei nodi appartenenti all'anello, etichette che possono essere evidenziate (mediante un cambiamento di colore dipendente dal segno dell'anello, "+" o "-") e visualizzate su un $frame$ ad hoc, istanza della classe $DisplaySubgraphFrame$ (cfr. la sezione 4.3.4).
La rappresentazione pittorica è implementata facendo uso di due elementi della classe $Vector$, uno contenente le etichette ($currentNodes$) e l'altro contenente gli elementi di connessione ($currentWeb$).
I due vettori[12] sono gestiti da un certo numero di metodi e hanno associati degli elementi "immagine", ovvero delle copie ottenute con metodi di clonazione, che consentono di implementare facilmente le operazioni di editing (quali la $Undo\,Draw$, la $Redo\,Draw$, la $Clear\,Canvas$ e la $Undo\,Clear\,Canvas$).
I metodi principali della classe $Draw$ sono quelli che consentono di manipolare gli elementi del vettore $currentWeb$ e gli elementi del vettore $currentNodes$.
I metodi del primo gruppo consentono:

[11] I campi privati delle classi sono di solito acceduti mediante metodi di interrogazione e di settaggio: i primi hanno il suffisso *get* e restituiscono un tipo coincidente con quello del campo che gestiscono mentre i secondi hanno il suffisso *set*, non restituiscono nulla e hanno un paramentro il cui tipo coincide con quello del campo che gestiscono. Definito il campo a di tipo T, i metodi suddetti saranno rispettivamente $getA()$ e $setA(T\,b)$.

[12] Nel seguito useremo il termine vettore per individuare un elemento classe $Vector$ per cui la dizione "il vettore a" deve essere letta come "l'oggetto a istanza della classe $Vector$".

1. di aggiungere una etichetta istanza della classe *LabelPanel* alla rappresentazione pittorica,
2. di rimuovere una etichetta sulla base del valore dello *stamp*,
3. di individuare l'etichetta prossima alla posizione corrente del cursore in modo da consentire il settaggio della forma del cursore oppure l'inizio del tracciamento di un elemento di connessione,
4. di accedere ai singoli elementi del vettore $currentWeb$ e al vettore nella sua interezza.

I metodi del secondo gruppo sono relativi agli elementi del vettore che contiene gli elementi di connessione, istanze della classe $ConnectionElements$, e consentono:

1. di aggiungere un elemento di connessione del tipo specificato dall'utente, dopo averne verificata la legalità,
2. di rimuovere un elemento di connessione e i suoi elementi accessori (handle e segno),
3. di accedere ai singoli elementi del vettore $currentNodes$ e al vettore nella sua interezza,
4. di ritracciare gli elementi di connessione in modo conforme alla nuova posizione delle etichette poste alle sue estremità.

I metodi della classe $Draw$, ovviamente, interagiscono direttamente con gli opportuni metodi delle classi $ConnectionElements$, $HandlePanel$, $LabelPanel$ e indirettamente, attraverso metodi della classe $ConnectionElements$, con metodi della classe $ArrowPanel$.

4.3.2 Il "canvas" e le modalità di tracciamento

Il *canvas* usato dall'utente per creare e/o modifiacre la rappresentazione pittorica di un grafo è un oggetto istanza della classe $DrawPanel$, classe in grado di interagire sia con la rappresentazione pittorica (istanza della classe $Draw$) sia con il grafo (istanza della classe $Graph$) sia con la classe di "mediazione" fra la rappresentazione pittorica e il grafo (la classe $Draw2Graph$) e caratterizzata da un certo numero di metodi che catturano le azioni dell'utente eseguite mediante il mouse.
A tale scopo i metodi della classe implementano, in cooperazione con metodi della classe $Draw$, due filosofie di tracciamento che diremo:

1. *dragging*[13],
2. *clicking*.

Mentre le etichette sono posizionate sul *canvas* semplicemente selezionando l'opportuna operazione (cfr. la sezione 3.3) e generando un evento *mouseClicked* (cfr. la nota 13) nel punto in cui le si vuole posizionare, il tracciamento degli elementi di connessione può avvenire in due modi e, in più, richiede una interazione con la classe responsabile della implementazione del grafo (cfr. la sezione 4.3.3) perchè è necessario controllare la liceità o meno di un collegamento, data una coppia di etichette.
Per quanto riguarda il tracciamento, *archi* e *linee* possono essere tracciati, in modalità *clicking*, generando due eventi *mouseClicked* in prossimità di due etichette distinte non connesse già fra di loro oppure, in modalità *dragging*, generando un evento *mousePressed* in prossimità di una etichetta e un evento *mouseReleased* in prosimità di un'altra etichetta.
Le *polilinee* sono tracciate per segmenti in modo simile alle *linee*. I vincoli ovvi per il loro tracciamento sono che i segmenti siano fra loro contigui (abbiano un punto in comune) e che il primo si origini in prossimità di una etichetta distinta da quella in prossimità della quale termina l'ultimo.
A tracciamento eseguito, su ciascun elemento di connessione vengono posizionati:

1. la freccia, che ne indica il verso, posizionata in prossimità della seconda etichetta,
2. il segno, posizionato in un'intorno della freccia appartenente ad un oggetto istanza della classe *ArrowPanel*,
3. un *handle*, per *archi* e *linee*, o un insieme di *handle*, uno per ciascun segmento, nel caso delle *polilinee*.

Il *canvas*, istanza della classe *DrawPanel*, rappresenta, pertanto, la superfice sulla quale l'utente può posizionare gli elementi della rappresentazione pittorica che possono essere suddivisi in:

1. *elementi attivi*,
2. *elementi passivi*.

[13]Per *dragging* si intende lo spostamento del cursore sul video con il pulsante principale del mouse, il sinistro per mouse a tre tasti, premuto mentre per *clicking* si intende la successione di presione e rilascio di tale pulsante senza movimento apprezzabile del cursore. Il *dragging* genera una successione di eventi *mouseDragged* mentre il *clicking* genera un singolo evento *mouseClicked*. Sia l'evento *mouseDragged* sia l'evento *mouseClicked* sono caratterizzati dagli eventi *mousePressed* e *mouseReleased* che possono essere catturati e gestiti, in modo analogo agli eventi *mouseDragged* e *mouseClicked*, da un opportuno *listener*.

Gli *elementi passivi* sono gli elementi grafici rappresentativi di *archi*, *linee* e *poliline*.
Gli *elementi attivi* sono elementi in grado di reagire ad eventi del mouse e ad essi sono associati dei menù flottanti. Tipici elementi attivi cui è associato un menù flottante sono le *etichette*, gli *handle* e i *segni* sugli *elementi di connessione*. Elementi attivi cui non è associato un menù flottante sono i *segni degli anelli* (cfr. la sezione 4.3.4). Il *canvas* stesso appartiene a questa categoria e ad esso è associato un menù flottante che consente di accedere in modo rapido alle operazioni per la creazione degli elementi della rappresentazione pittorica (*etichette* ed *elementi di connessione*).
Le *etichette* e i *segni degli anelli* sono gli unici elementi attivi che possono essere trascinati sul *canvas* con operazioni di *dragging* eseguite con il cursore posizionato sull'elemento da spostare. Lo spostamento di una *etichetta* comporta lo spostamento degli elementi attivi e passivi ad essa associati ed, in particolare, il ritracciamento degli *elementi di connessione* incidenti nell'etichetta.

4.3.3 La struttura astratta *"grafo"*

Alla rappresentazione pittorica, che l'utente crea e/o modifica sul *canvas*, corrisponde in tempo reale un *grafo* caratterizzato da un certo numero di *nodi* e da un certo numero di *archi* in corrispondenza biunivoca, rispettivamente, con le *etichette* e gli *elementi di connessione* della rappresentazione pittorica (cfr. la sezione 4.3.1).

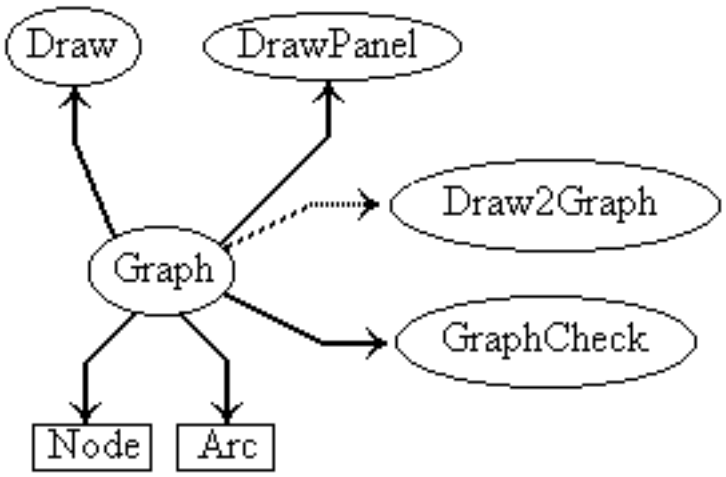

Figura 4.3: *La classe "Graph" e le classi "ausiliarie"*

I *grafi* sono rappresentati da oggetti istanze della classe *Graph* che si appoggia ad un certo numero di classi ausliarie quali (cfr. la figura 4.3):

1. la classe *Arc*, che implementa gli archi del grafo, corrispondente alla classe *ConnectionElements* che implementa gli *elementi di connessione*,
2. la classe *Node*, che implementa i nodi del grafo, corrispondente alla classe *LabelPanel* che implementa le *etichette*,
3. la classe *GraphCheck*, che contiene i metodi per l'esecuzione di controlli sul grafo e l'assegnazione a ciascun anello presente nel grafo dell'opportuno segno,
4. la classe *Draw2Graph* che possiede metodi che, in tempo reale, determinano gli anelli presenti nel grafo corrente (la connessione a tratto con la classe *Graph* in figura 4.3 sta ad indicare una relazione indiretta e non di tipo $has-a$).

Le strutture dati che implementano un $grafo$ (cfr. la figura 4.4) sono i due vettori *arcs* e *nodes* acceduti mediante un certo numero di metodi, alcuni dei quali saranno descritti a breve. Il primo di tali vettori contiene gli *archi* del grafo corrente mentre l'altro ne contiene i *nodi*. La classe *Vector* fornisce i metodi di base per l'aggiunta ai/la rimozione degli elementi dai vettori e la gestione dell'allocazione di memoria.

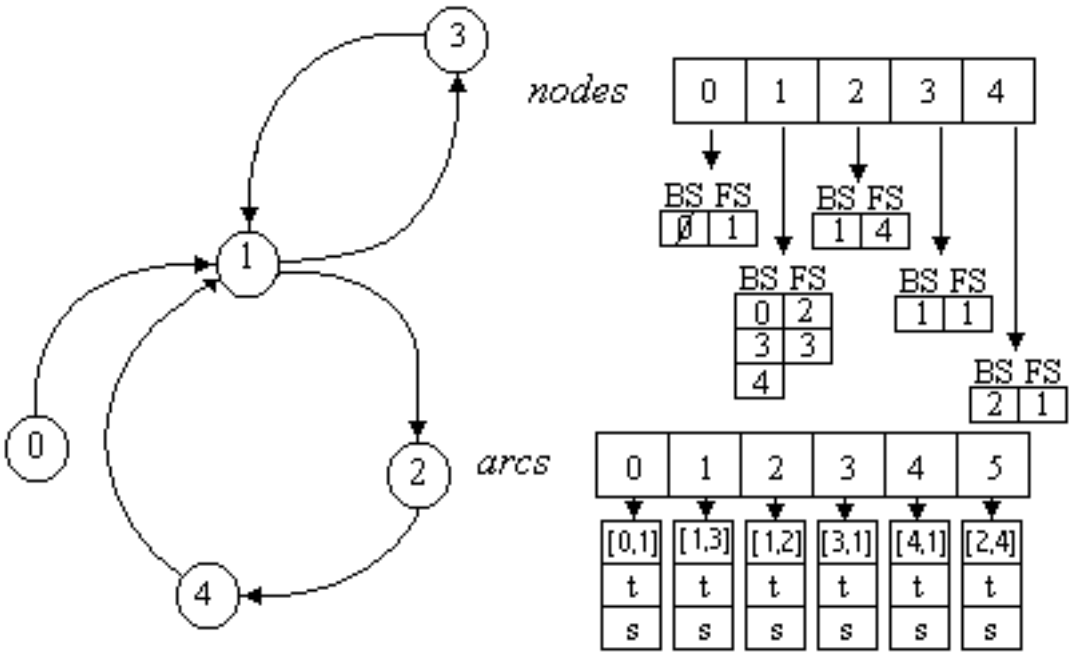

Figura 4.4: *Le strutture dati (visione semplificata) associate ad un "grafo"*

Gli *archi* del $grafo$, contenuti nel vettore *arcs*, sono caratterizzati dalle seguenti strutture dati:

1. un elemento della classe *Point* (standard di *Java*) che contiene una coppia di interi che sono gli identificativi numerici (*stamp*) dei nodi agli estremi dell'arco (il campo contenente i valori racchiusi fra [] della figura 4.4),

2. un elemento della classe *String* che contiene il *segno* dell'arco (il campo s della figura 4.4),

3. un elemento della classe *String* che contiene il *tipo* dell'arco (il campo t della figura 4.4).

Per ogni arco il segno e il tipo coincidono con quelli del corrispondente elemento di connessione.
I *nodi* del $grafo$, contenuti nel vettore *nodes*, sono a loro volta caratterizzati dalle seguenti strutture dati:

1. un *intero* che rappresenta lo *stamp* del nodo,

2. un elemento della classe *String* che contiene la *variabile* associata al nodo e coincidente con il *nome* dell'etichetta corrispondente,

3. un *intero* che contiene il *tipo* del nodo, coincidente con il *tipo* dell'etichetta corrispondente,

4. un elemento della classe $Vector$ ($forwardStar$) che rappresenta la *stella uscente* del nodo (indicata come FS nella figura 4.4),

5. un elemento della classe $Vector$ ($backwardStar$) che rappresenta la *stella entrante* del nodo (indicata come BS nella figura 4.4).

I valori che può assumere il campo *tipo* di un nodo verranno esaminati più in dettaglio nella sezione 4.4, qui ci si limita a segnalare come siano le codifiche su un range di interi delle stringhe "level", "rate", "source","sink", "delay", "constant", "auxiliary" e "unspecified".
I vettori $forwardStar$ e $backwardStar$ contengono i valori dei campi *stamp* dei nodi che fanno parte, rispettivamente, della *stella uscente* e della *stella entrante* del nodo corrente, memorizzati come istanze della classe *wrapper* $Integer$.
La classe *Graph* è caratterizzata da un certo numero di metodi che possono essere suddivisi in:

1. metodi per l'accesso al grafo,

2. metodi per la gestione dei nodi,

3. metodi per la gestione degli archi.

I metodi del gruppo 1 permettono di interrogare la struttura dati $grafo$ per conoscere:

1. il numero di nodi e di archi del grafo,
2. i vettori *arcs* e *nodes*,
3. l'arco che ha una certa posizione nel vettore *arcs*,
4. l'arco che ha per estremi due nodi di cui sono noti gli *stamp*,
5. il nodo che ha un certo valore dello *stamp*,
6. i nodi che fanno parte della *stella uscente* di un nodo di cui è noto il valore dello *stamp*,
7. i nodi che fanno parte della *stella entrante* di un nodo di cui è noto il valore dello *stamp*.

I metodi dei gruppi 2 e 3 permettono, d'altro lato, di costruire incrementalmente il grafo per aggiunta e rimozione di nodi e di archi, oltre a permettere di eseguire altre operazioni sui nodi e gli archi, molte di tipo *private* e pertanto inaccessibili a metodi esterni alla classe.
Utilizzando i metodi *getNodes*() e *getArcs*() della classe *Graph* è, infine, possibile ottenere i vettori *arcs* e *nodes* del grafo corrente mediante i quali, con operazioni di *cast*, rispettivmente, ad *Arc* e a *Node* si hanno a disposizione tutti i metodi delle classi *Arc* e *Node*.
I metodi della classe *Arc* permettono il settaggio e l'interrogazione delle strutture dati di un *arco* ovvero gli identificativi dei nodi alle estremità dell'arco, il segno dell'arco e il suo tipo. I metodi della classe *Node*, invece, sono più numerosi dato che ogni *nodo* è caratterizzato da strutture dati di complessità maggiore.
Tali metodi consentono:

1. di accedere, con metodi di tipo *get* e *set*, allo *stamp* di un nodo,
2. di accedere, con metodi di tipo *get* e *set*, alla *etichetta* di un nodo,
3. di accedere, con metodi di tipo *get* e *set*, al *tipo* di un nodo,
4. di manipolare il vettore *forwardStar* di un nodo,
5. di manipolare il vettore *backwardStar* di un nodo.

I metodi per la manipolazione dei vettori *forwardStar* e *backwardStar* permettono l'aggiunta e la rimozione di nodi e la verifica della presenza o meno di un nodo in tali vettori. Li si utilizza in corrispondenza delle operazioni di aggiunta e di rimozione di elementi di connessione dalla rappresentazione pittorica che si traducono, rispettivamente, nella aggiunta e nella rimozione di nodi dal grafo corrispondente.

4.3.4 I *frame* ausiliari e il *check* di un grafo

Il *Causal Loop Graphic Editor* fa uso di un *frame principale* al cui interno è posizionato il *canvas* utilizzato per la rappresentazione pittorica di diagrammi CL (cfr. le sezioni 4.3.1 e 4.3.2).
Oltre al *frame principale* istanza della classe $ClMainFrame$, il *Causal Loop Graphic Editor* mette a disposizione dell'utente i *frame ausiliari*, istanze delle classi $DisplaySubgraphFrame$ e $DisplayLoopFrame$, per la visualizzazione delle rappresentazioni pittoriche di sottografi del grafo corrente.
I *frame ausiliari* possono essere creati su iniziativa dell'editor (e previa conferma da parte dell'utente) oppure dietro richiesta dell'utente (cfr. la sezione 3.3).
I $frame$ del primo tipo visualizzano situazioni anomale che si verificano in seguito alla esecuzione:

1. di un metodo per il controllo dei segni degli archi di un grafo,

2. di un metodo per il controllo dello stato di nodi e archi di un grafo.

Nel caso in cui l'utente voglia assegnare il segno agli anelli presenti nel grafo corrente (cfr. la sezione 3.3) è neccessario che esegua preventivamente un *check* dei segni dei singoli archi del grafo: se il metodo $signChecker(currGraph)$ della classe $GraphCheck$ trova che il grafo corrente (l'oggetto $currGraph$ istanza della classe $Graph$) contiene archi il cui segno ha valore "unspecified" può (se l'utente lo richiede) visualizzare il sottografo indotto da tali archi in un $frame$ istanza della classe $DisplaySubgraphFrame$.
In modo analogo, se il metodo $graphChecker(currGraph)$ che esegue il controllo di nodi e archi di un grafo, controllo necessario prima di poter convertire un diagramma CL nel corrispondente diagramma FD (cfr. la sezione 3.3), individua archi e/o nodi che non hanno assegnato un tipo può (se l'utente lo richiede) visualizzare i sottografi indotti sia dagli archi sia dai nodi in due $frame$ distinti, istanze della classe $DisplaySubgraphFrame$.
L'utente, d'altro lato, può voler evidenziare dei sottografi del grafo correntemente visualizzato sul $frame$ principale.
A tale scopo la classe $GraphCheck$ possiede dei metodi che, in cooperazione con metodi della classe $DisplaySubgraphFrame$, permettono la selezione di un sottografo del grafo corrente e la sua visualizzazione in un $frame$ separato, istanza della classe $DisplaySubgraphFrame$. Per visualizzare le rappresentazioni pittoriche dei sottografi, la classe $DisplaySubgraphFrame$ utilizza un *canvas*, istanza della classe $DrawSubgraphPanel$, sul quale visualizza oggetti particolari ovvero:

1. elementi di connessione senza *handle* e senza segno,

2. etichette, istanze della classe $CloneLabelPanel$, in grado di reagire solo ad eventi del tipo *mouseEntered* e *mouseExited*[14].

[14] L'evento *mouseEntered* viene prodotto ogni volta che il cursore viene posizionato sopra un

L'uso di tali oggetti semplificati si è reso necessario per poter creare rappresentazioni immodificabili del sottografo e perfettamente congruenti con quella del grafo completo.
I metodi *arcSubsetDisplay* e *nodeSubsetDisplay* della classe *GraphCheck* (in cooperazione con il metodo *filterData* della classe *DisplaySubgraphFrame*) permettono la selezione, rispettivamente, di:

1. archi, sulla base del *tipo* o del *segno*,

2. nodi, sulla base del *tipo*.

Archi e nodi così selezionati individuano i sottografi che vengono visualizzati su *frame* separati.
L'utente, infine, può visualizzare un sottografo in un *frame* separato sfruttando un metodo della classe *DrawPanel* (*displayLoopInFrame*) acceduto tramite la classe *SignPanel*.
Tale metodo consente la visualizzazione del sottografo composto dai nodi e dagli archi contenuti in uno degli anelli presenti nel grafo corrente (cfr. la sezione 3.3). In questo caso il *frame* utilizzato è una istanza della classe *DisplayLoopFrame* che usa un *canvas* istanza della classe *DrawLoopPanel* per visualizzare gli oggetti semplificati che formano la rappresentazione pittorica del sottografo.

4.3.5 L'interazione con il Sistema Operativo ospite

Come osservato nella sezione 2.2.7, un editor grafico deve dare all'utente la possibilità di salvare il lavoro fatto in strutture dati persistenti. La soluzione adottata nel caso dell'ambiente *D(a)ySy Tool Box* è stata quella di fare uso di file la cui gestione viene fatta interagendo con il *file system* del Sistema Operativo ospite (cfr. la sezione 2.2.8).
Il Tool *Causal Loop Graphic Editor* usa a tale scopo un oggetto istanza della classe *Interactor* che definisce ed implementa tutti i metodi necessari a tale interazione.
I metodi della classe *Interactor* sono acceduti direttamente dai listener del menù *File* della classe *ClMainFrame* (cfr. la sezione 3.3).
La classe *ClMainFrame* istanzia allo scopo un oggetto della classe *Interactor* ($soInteract = newInteractor(containerMf)$) cui passa l'handle di un oggetto della classe *ClMainFrame* stessa ($containerMf$) e che poi usa per accedere ai metodi della classe.
I metodi della classe *Interactor* consentono l'interazione con file di due tipi ([HC99]):

1. *stream di oggetti*,

elemento in grado di reagire ad eventi del mouse mentre l'evento *mouseExited* viene prodotto ogni volta che il cursore esce da tale elemento.

2. *stream di stringhe di testo delimitato.*

I file del primo tipo contengono oggetti di cui viene fatto l'*upcast* a oggetti istanze della classe univarsale *Object* (e saranno detti *file di oggetti*) mentre i file del secondo tipo contengono oggetti istanze della classe *String* (e saranno detti *file di testo*).
I file di oggetti sono acceduti:

1. in scrittura mediante uno *stream* di tipo $ObjectOutputStream$ che, in combinazione con uno *stream* di tipo $FileOutputStream$, consente di accedere in scrittura ad uno specifico $file$,

2. in lettura mediante uno *stream* di tipo $ObjectInputStream$ che, in combinazione con uno *stream* di tipo $FileIntputStream$, consente di accedere in lettura ad uno specifico $file$.

La scrittura e la lettura effettiva degli oggetti avvengono, rispettivamente, mediante i metodi $writeObject(Object\ obj)$ e $readObject$. Perchè il metodo $writeObject$ sia applicabile agli oggetti di una generica classe A questa deve implementare l'interfaccia *Serializable* ([HC99]).
I file di testo contengono linee di testo di lunghezza variabile. La scrittura dei dati su un file è resa possibile dalla definizione di un oggetto di tipo $PrintWriter$ mentre i dati sono effettivamente scritti nel file una linea alla volta utilizzando il metodo $outPw.println(dataOut)$, in cui $outPw$ è un oggetto di tipo $PrintWriter$ e $dataOut$ è un oggetto di tipo *String* che contiene i dati dalla linea corrente di testo.
Ogni linea di testo contiene più campi separati da un carattere separatore, nel nostro caso si è usato il carattere "|", e le linee possono avere lunghezza variabile sia in numero di campi sia in numero di caratteri.
La lettura dei dati avviene una linea di testo alla volta utilizzando il metodo $readLine$ applicato ad un oggetto della classe $BufferedReader$. Una volta letta una linea di testo in una variabile locale (ad esempio $line$), la si può spezzare nei suoi campi costituenti (detti $token$) utilizzando un oggetto della classe $StringTokenizer$ che consente di spezzare ognuna delle linee di testo in sottostringhe usando il separatore utilizzato nella creazione del file e i metodi $nextToken$ e $hasMoreToken$, il cui signifiacto è facilmente intuibile.
La classe $Interactor$, pertanto, possiede metodi per gestire file dei due tipi suddetti. Per l'accesso ai file di oggetti, la classe contiene i metodi $newFile$, $openFile$, $saveFile$ e $saveAsFile$ cui corrispondono i relativi comandi del menù $File$ (cfr. la sezione 3.3) oltre ad un certo numero di metodi "interni" fra cui si segnalano i metodi $loadFileOnOpen$ e $downloadFileOnSave$ che eseguono effettivamente la lettura (il primo) e la scrittura (il secondo) dei dati.
Per l'accesso ai file di testo, la classe contiene i metodi $exportFile$ e $importFile$ cui corrispondono i relativi comandi del menù $File$ (cfr. la sezione 3.3)

oltre ad un certo numero di metodi "interni" fra cui si segnalano i metodi *loadFileOnImport* e *downloadFileOnExport* che eseguono effettivamente la lettura (il primo) e la scrittura (il secondo) dei dati.
I metodi "esterni", cui corrispondono voci del menù *File*, ottengono informazioni dall'utente (quali il nome del file su cui l'utente vuole lavorare e la eventuale directory in cui è contenuto) facendo uso di *finestre di dialogo* istanze della classe *JOptionPane* come *showInputDialog* o della classe *JFileChooser* quali *showOpenDialog* e *showSaveDialog*.
I dati che i metodi *downloadFileOnSave* e *downloadFileOnExport* salvano sui file, rispettivamente, come *stream* di oggetti e come *stream di stringhe di testo delimitato* sono contenuti in oggetti istanze delle classi *Draw* e *Graph* ottenuti utilizzando metodi di tali classi.
Nel caso di file di oggetti il *download* avviene semplicemente utilizzando ripetutamente il metodo *writeObject*. Nel caso dei file di testo è necessario, prima di effettuare il download su file, eseguire una elaborazione dei dati per operare una conversione da oggetti a stringhe di testo e per inserire nei file un certo numero di stringhe di testo di contenuto fisso che suddividono ciascun file in sezioni e ne aumentano la leggibilità. Mentre i file di oggetti sono accessibili solo dall'interno dell'editor, i file di testo sono, infatti, accessibili e leggibili da un qualunque editor di testo e contengono informazioni in merito alla struttura sia della rappresentazione pittorica sia del grafo corrispondente.
Per quanto riguarda il *load* dei dati da file, di nuovo è necessario distinguere il caso in cui i dati sono contenuti in file di oggetti da quello in cui sono contenuti in file di testo. Nel primo caso i dati sono letti utilizzando ripetutamente il metodo *readObject* ed eseguendo il *cast* esplicito al tipo corretto di ciascun oggetto letto.
Ad esempio, se in scrittura si applica il metodo *writeObject* prima ad un oggetto $a1$ della classe A e poi ad uno $b1$ della classe B ovvero si ha:

```
out.writeObject(a1);
out.writeObject(b1);
```

(dove *out* è un oggetto della classe *ObjectOutputStream*) in lettura si deve usare la successione di chiamate del metodo *readObject* seguente:

```
A a1=(A)in.readObject();
B b1=(B)in.readObject();
```

(dove *in* è un oggetto della classe *ObjectInputStream*).
Nel caso, invece, che i dati sono contenuti in un file di testo devono essere eseguite le operazioni duali di quelle che sono state eseguite durante il *download*: utilizzando il metodo *readLine* si leggono le linee di testo del file, si scartano le linee ridondanti a contenuto fisso, si separano le linee significative in token e

utilizzando i token (eseguendo ogni volta che è necessario le conversioni da *String* ad *int*) si ricostruiscono i dati caratteristici di un grafo e della corrispondente rappresentazione pittorica.
I dati così ricostruiti devono essere passati ai metodi della classe *DrawPanel* affinchè possano essere rappresentati sul *canvas* (la rappresentazione pittorica) ed essere modificabili dall'utente (il grafo e la rappresentazione pittorica).

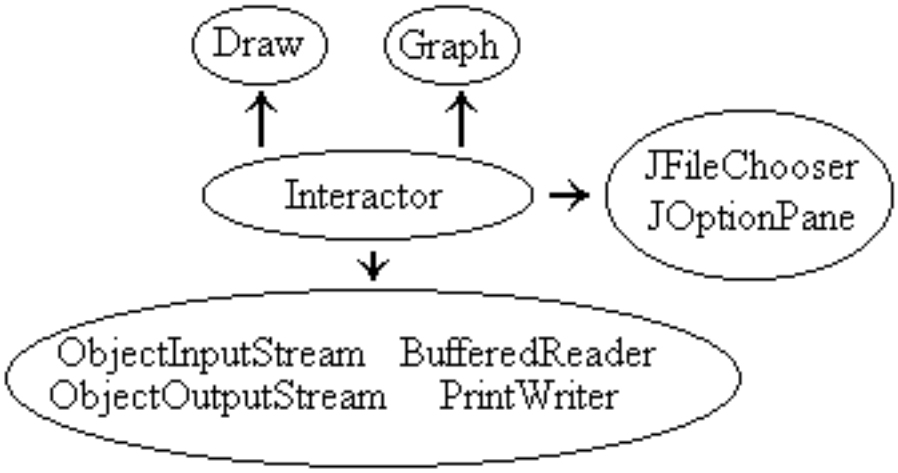

Figura 4.5: *Le interazioni della classe "Interactor"*

La figura 4.5 illustra schematicamente le relazioni esistenti fra la classe *Interactor* e le altre classi presentate nei paragrafi precedenti. Le classi standard di *Java* sono state raggruppate, per semplicità, separando quelle responsabili della craezione delle finestre di dialogo da quelle responsabili delle interazioni con i file.

4.4 *Flow Diagram Graphic Editor*

Il *Flow Diagram Graphic Editor*, la cui struttura esterna è stata presentata nella sezione 3.4, ha molte caratteristiche in comune con il *Causal Loop Graphic Editor* per cui nelle sezioni che seguono si cercherà di evidenziare sopratutto le differenze esistenti fra le strutture interne dei due Tool.
Gli scopi principali del Tool *Flow Diagram Graphic Editor* sono i seguenti:

1. consentire all'utente di creare rappresentazioni pittoriche di diagrammi FD,
2. creare e mantenere in tempo reale un multigrafo corrispondente ad una rappresentazione pittorica,

3. mettere a disposizione dell'utente operazioni per caratterizzare sia la rappresentazione pittorica sia il multigrafo sottostante,
4. fornire comandi per l'analisi dei diagrammi FD creati dall'utente,
5. consentire all'utente di associare agli elementi di un diagramma le equazioni caratteristiche,
6. permettere all'utente di evidenziare porzioni di un diagramma FD.

Molte di tali operazioni hanno equivalenti nel caso dei diagrammi CL per cui in quanto segue vi si farà solamente un rapido cenno.

4.4.1 La rappresentazione pittorica e il multigrafo soggiacente

Nel caso dei diagrammi FD la rappresentazione pittorica si basa su elementi di connessione identici come tipologie (archi, linee e polilinee) a quelli esaminati nella sezione 4.3.1 mentre i nodi[15] sono caratterizzati da elementi più complessi dal momento che ad ogni nodo corrisponde una coppia composta da:

1. una icona con associata una immagine che rispecchia il tipo dell'elemento,
2. una etichetta che contiene il nome dell'elemento.

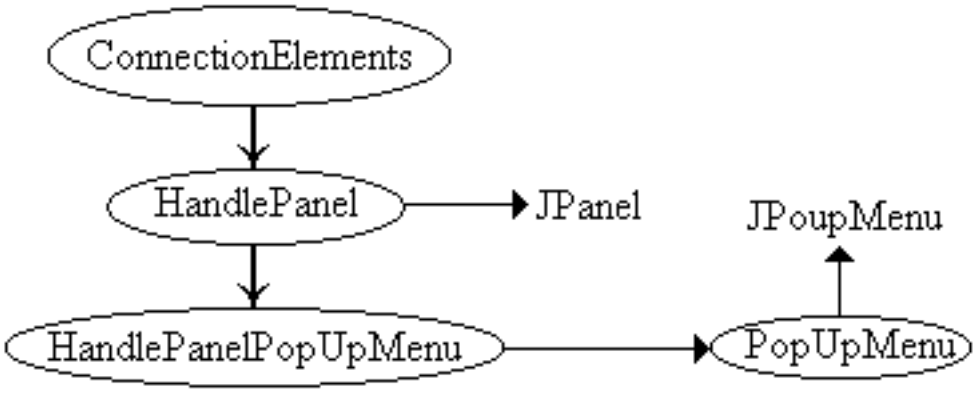

Figura 4.6: *Le classi per gli elementi di connessione*

Gli elementi di connessione sono istanze della classe *ConnectionElements* prive di *Panel* del segno ma caratterizzate da handle (istanze della classe

[15]Nel seguito, per evitare ambiguità, useremo il termine "icona" per la rappresentazione pittorica e "nodo" per il multigrafo. Si fa notare come ad una icona sia sempre associata una etichetta.

HandlePanel) a ciascuno dei quali è associato un menù istanza della classe *HandlePanelPopUpMenu*). La figura 4.6 illustra le classi utilizzate per l'implementazione degli elementi di connessione. In tale figura, come nelle altre della sezione 4.4, una relazione di ereditarietà viene rappresentata con un segmento che termina con un triangolo di colore nero ed orientato dalla classe alla superclasse mentre le classi standard di *Java* sono rappresentate con il solo nome.
I menù flottanti (ottenuti come istanze di classi quali *HandlePanelPopUpMenu*, *LabelPanelPopUpMenu* e *DrawPanelPopUpMenu*) ereditano dalla classe *PopUpMenu* che a sua volta eredita dalla classe standard *Java JPopupMenu* mentre le classi il cui nome contiene la parola *Panel* ereditano dalla classe standard *Java JPanel*.
Gli elementi di connessione (cfr. anche le sezioni 4.3.1 e 4.3.3) sono caratterizzati da una struttura dati che contiene gli elementi sia per una loro caratterizzazione geometrica sia per una loro caratterizzazione funzionale. Al primo tipo apparengono:

1. un vettore di punti per il tracciamento sul *canvas* degli elementi di connessione,
2. un vettore di handle,
3. una fraccia, che ne determina l'orientamento,
4. una coppia di elementi, sorgente e destinazione dell'elemento di connessione, istanze della classe *ComplexShapePanel*, di cui si dirà a breve.

Degli elementi sorgente e destinazione rivestono un ruolo particolare i "baricentri", che rappresentano i punti in cui convergono tutti gli elementi di connessione che si originano da una icona o che incidono in una icona.
Al secondo tipo appartengono informazioni relative al *colore* e al *tipo* di un elemento di connessione. Il colore rappresenta essenzialmente un ausilio visivo che consente di individuare gli elementi di connessione come caratterizzati da un tipo mentre il secondo svolge un ruolo nella definizione dei flussi conservativi (di tipo "Materials", colore *magenta*) e non conservativi (di tipo "Information", colore *blue*) e delle relazioni fra i vari tipi delle icone.
La figura 4.7 illustra le classi utilizzate per implementare le icone, compresi i menù flottanti per la loro customizzazione. Ad ogni icona corrisponde una immagine, memorizzata in un file *.gif*, che ne rappresenta il tipo. I tipi possibili, ciascuno individuato da una immagine diversa, sono i seguenti: "level","rate","source","sink","delay","auxiliary" e "constant".
Le strutture dati associate ad ogni icona sono gli elementi necessari per il suo tracciamento (posizione e dimenzioni) e i riferimenti all'etichetta associata la quale è caratterizzata, in più, dalle strutture dati per il tracciamento e la modifica della stringa che ne rappresenta il valore. Gli altri comandi contenuti nel menù flottante associato a ciascuna icona (e alla corrispondente etichetta) sono implementati

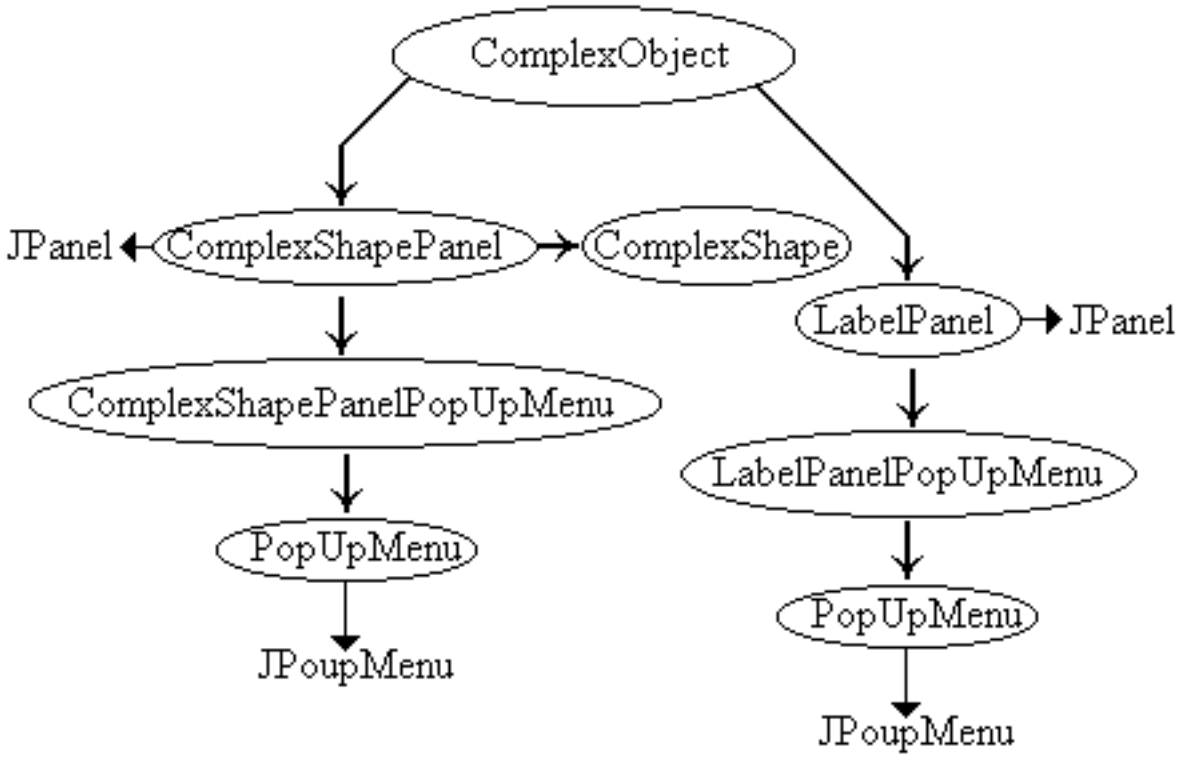

Figura 4.7: *Le classi per le icone (e le etichette)*

direttamente dalla struttura $grafo$ soggiacente.
Come nel caso del *Causal Loop Graphic Editor* (cfr. la sezione 4.3.3), alla rappresentazione pittorica tracciata dall'utente sul *canvas* il Tool fa corrispondere (e tiene aggiornata in tempo reale) una struttura astratta che, nel caso del *Flow Diagram Graphic Editor* è un multigrafo. In questo caso, infatti, fra due nodi possono essere presenti fino a due archi equiorientati e di tipo diverso (è tipicamente il caso di un nodo di tipo "level" collegato ad un nodo di tipo "rate" da due archi, uno di tipo "Information" e uno di tipo "Materials").
La struttura astratta multigrafo è implemetata utilizzando le classi illustrate nella figura 4.8. La classe *Graph* rappresenta un multigrafo come caratterizzato da un vettore di nodi, da un vettore di archi e dalle strutture dati per rappresentare i valori utilizzati nella simulazione e le unità di misura associate ai nodi del multigrafo. La classe possiede i metodi necessari per l'aggiunta e la rimozione di nodi e archi oltre a metodi per l'esame del multigrafo e per l'accesso sia ai valori di simulazione sia alle unità di misura.
Per la gestione di nodi ed archi la classe *Graph* fa uso delle seguenti classi:

1. *Arc*,

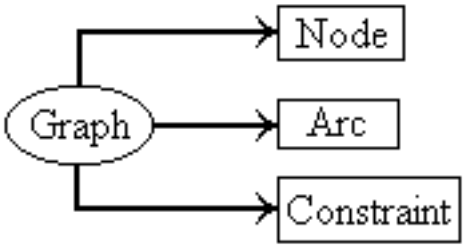

Figura 4.8: *Le classi per il multigrafo*

2. *Node*,
3. *Constraint*.

La classe *Arc* rapresenta i singoli archi come individuati da coppie di interi (che rappresentano gli identificativi dei nodi estremi dell'arco) e da una stringa che rappresenta il tipo dell'arco. La classe possiede tutti i metodi accessori per il settaggio e l'interrogazione dei campi dati.
La classe *Node* implementa i nodi del multigrafo come caratterizzati da:

1. un identificativo numerico,
2. una etichetta,
3. una equazione con il relativo tipo,
4. un vettore che rappresenta la stella entrante del nodo,
5. un vettore che rappresenta la stella uscente del nodo,
6. la stella uscente ed entrante del nodo rappresentate come stringhe di testo delimitato.

Le stelle di un nodo sono vettori di interi che rappresentano gli identificativi dei nodi in esse contenuti. Allo scopo di poter rappresentare tali stelle nella equazione associata ad un nodo (implementata come una stringa) la classe possiede dei metodi che di ciascuna stella danno una rappresentazione sotto forma di una stringa composta da campi separati da un carattere speciale detto delimitatore (il carattere ","). La classe *Node* possiede tutti i metodi necessari per la manipolazione delle strutture dati suddette.
La classe *Constraint*, infine, definisce le strutture dati e un metodo necessari per verificare se un arco di un dato tipo può essere tracciato fra due nodi di cui sono noti i tipi. Il controllo è di tipo statico, basato sui tipi di un arco e di una coppia di nodi in cui l'arco incide. Per eseguire il controllo di ammissibilità la classe usa

un array tridimensionale (tipo del nodo sorgente, tipo del nodo estinazione e tipo dell'arco) inizializzato dal costruttore e i cui elementi valgono 1 se l'arco di quel tipo fra nodi di quei tipi è ammissibile o 0 altrimenti.
Il controllo della legalità di un arco che l'utente vorrebbe aggiungere alla rappresentazione pittorica e, pertanto, al multigrafo viene eseguito su più livelli. La classe *Graph* possiede, infatti, alcuni metodi che consentono di vericare se un arco di un certo tipo è già presente o meno fra due nodi. Se l'arco è già presente ne viene impedito il tracciamento altrimenti il metodo *isLegal* della classe *Graph* utilizza il metodo di controllo della classe *Constraint* per verificarne l'ammissibilità. Nel caso di archi di tipo "Materials" viene controllato anche che il flusso che questi individuano sia di tipo conservativo.

4.4.2 Il *frame* principale: *Draw* e *Graph*

Il Tool *Flow Diagram Graphic Editor* è ottenuto come istanza della classe *FdMainFrame* in due modi:

1. utilizzando la classe *FdEdit*, nella modalità stand alone,
2. utilizzando un metodo interno al Tool *TopLevel*, nella modalità slave.

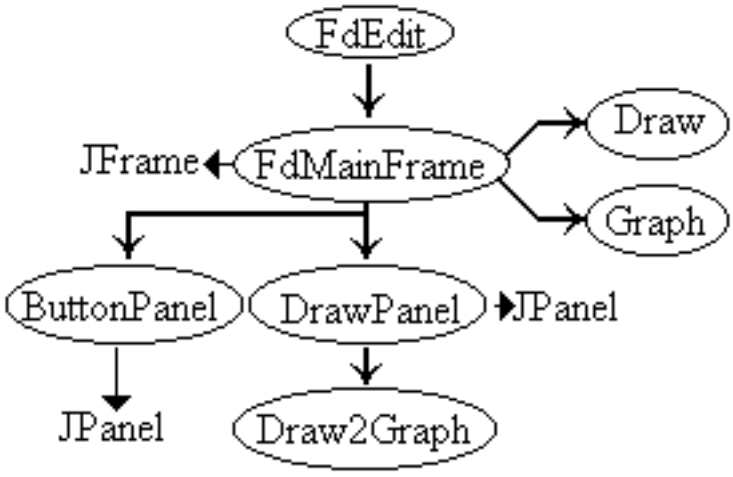

Figura 4.9: *Il frame principale e alcune delle classi collegate*

La classe *FdMainFrame* è responsabile della creazione e della gestione, mediante gli opportuni *listener*, delle voci del menù principale del Tool ed inoltre si occupa di creare e rendere disponibili all'utente:

1. il *panel*, istanza della classe *ButtonPanel*, con i pulsanti che implementano un sottoinsieme dei comandi del Tool,

2. il *canvas* sul quale l'utente può tracciare e modificare i diagrammi FD.

Oltre a tali compiti la classe $FdMainFrame$ si occupa di gestire la visibilità dei comandi contenuti nel menù principale e dei pulsanti in funzione dello stato del Tool.
La classe $FdMainFrame$ istanzia, inoltre, un oggetto della classe $Draw$ ed un oggetto della classe $Graph$ i cui riferimenti sono passati ad una istanza della classe $DrawPanel$ che si occupa di intercettare le azioni dell'utente e di tradurle in comandi significativi per il disegno (e pertanto gestiti dalla classe $Draw$) o per il grafo (e pertanto gestiti dalla classe $Graph$).
Scopo della classe $DrawPanel$ è, infatti, quello di catturare gli eventi del mouse prodotti dall'utente e di tradurli in azioni di disegno di oggetti, a meno che questi non siano prodotti in corrispondenza di icone, etichette o handle nel qual caso sono gestiti direttamente dai metodi delle classi di cui alla sezione 4.4.1.
Le azioni di disegno comprendono:

1. la istanziazione di coppie icona/etichetta,

2. il tracciamento degli elementi di connessione.

Il tracciamento degli elementi di connessione (*archi*, *linee* e *polilinee*) avviene secondo le due modalità (*dragging* e *clicking*) già descritte nella sezione 4.3.2 mentre la istanziazione di una coppia icona/etichetta richiede la selezione dell'icona opportuna in funzione del tipo dell'elemento che l'utente intende creare. In entrambi i casi, utilizzando metodi della classe di "mediazione" $Draw2Graph$, tali azioni si traducono in azioni di aggiornamento della rappresentazione pittorica (ovvero in chiamate a metodi della classe $Draw$ e, indirettamente, a metodi della classe $ConnectionElements$ per gli elementi di connessione) e del multigrafo (ovvero in chiamate a metodi della classe $Graph$).
La classe $DrawPanel$, infine, qualora l'utente selezioni un punto libero del *canvas*, crea una istanza della classe $DrawPanelPopUpMenu$ in modo da presentare all'utente il menù del *canvas* contenente i comandi presenti nella voce $Draw$ del menù principale.

4.4.3 I *frame* ausiliari

Come nel caso del *Causal Loop Graphic Editor* (cfr. la sezione 4.3.4) anche il *Flow Diagram Graphic Editor* fa uso di *frame ausiliari* per evidenziare porzioni del diagramma FD corrente oppure per consentire la visualizzazione della stella entrante e/o della stella uscente di un dato nodo.
Il Tool può evidenziare porzioni di un diagramma o su richiesta dell'utente oppure a seguito della esecuzione di operazioni di controllo su un diagramma che non hanno avuto esito positivo. L'utente può, infatti, eseguire:

1. una selezione degli archi o dei nodi sulla base del tipo, in modo che il Tool visualizzi su un *frame* ausiliario la rappresentazione pittorica del sottografo così individuato,

2. una operazione di controllo prima di eseguire la simulazione del diagrama, in modo che il Tool visualizzi le icone sulle quali il controllo ha avuto esito negativo (cfr. le sezioni 4.4.4 e 4.4.5).

In entrambi i casi, allo scopo viene usato un *frame* istanza della classe *DisplaySubfraphFrame* al cui interno sono visualizzati gli elementi di connessione (istanze della classe *ConnectionElements*) e le coppie icona/etichetta (istanze delle classi *CloneComplexObjec*, *CloneComplexShapePanel* e *CloneLabelPanel*). L'uso delle classi "cloni" permette di ottenere copie degli oggetti delle classi originarie prive di funzionalità inessenziali in tale contesto. Ad esempio gli oggetti clonati con tali classi non possono essere rimossi, spostati o modificati se non agendo sul loro originale.
Per la selezione degli archi in base al tipo viene fatto uso di una finestra di dialogo istanza della classe, standard *Java*, *JOptionPane* mentre per la selezione delle icone in base al tipo si fa uso di una finestra di dialogo, istanza della classe *NodeFilterFrame*.
Per quanto riguarda la visualizzazione della stella entrante e/o della stella uscente di un dato nodo, il Tool fa uso di istanze della classe *StarDisplay* (che eredita dalla classe, standard *Java*, *JFrame*) per visualizzare le informazioni relative ai nodi contenuti nella stella entrante e/o nella stella uscente. La classe *StarDisplay* fa uso di istanze della classe *NodeId* per memorizzare sia la stringa sia l'identificativo di ciascuno dei nodi del multigrafo.
Il Tool fa, inoltre, uso di *frame* ausiliari sia per consentire all'utente di impostare le equazioni dei nodi sia per permettergli di impostare o di modificare i parametri da utilizzare per la simulazione del diagramma FD (cfr. la sezione 4.4.4 e la figura 4.10).

4.4.4 Le strutture dati per la simulazione

Nel caso dei diagrammi FD, oltre a definire una rappresentazione pitorica cui corrisponde un multigrafo, l'utente deve poter definire le equazioni caratteristiche dei nodi in modo da poter simulare il modello rappresentato da un diagramma ed ottenere gli andamenti nel tempo delle sue variabili caratteristiche.
Le variabili sono rappresentate da stringhe associate sia alle coppie icona/etichetta (nella rappresentazione pittorica) sia ai nodi (nel multigrafo) mentre per la descrizione delle equazioni il Tool utilizza un certo numero di *frame* ausiliari (le cui classi illustrate nella figura 4.10) e alcune strutture dati.
Le strutture dati sono state aggiunte alla classe *Graph*, responsabile della descrizione del multigrafo, e consentono di associare ad un multigrafo le informazioni necessarie e sufficienti perchè il Tool *Equation Solver* (cfr. la sezione 4.6) sia in

grado di ricavare le equazioni e risolverle.

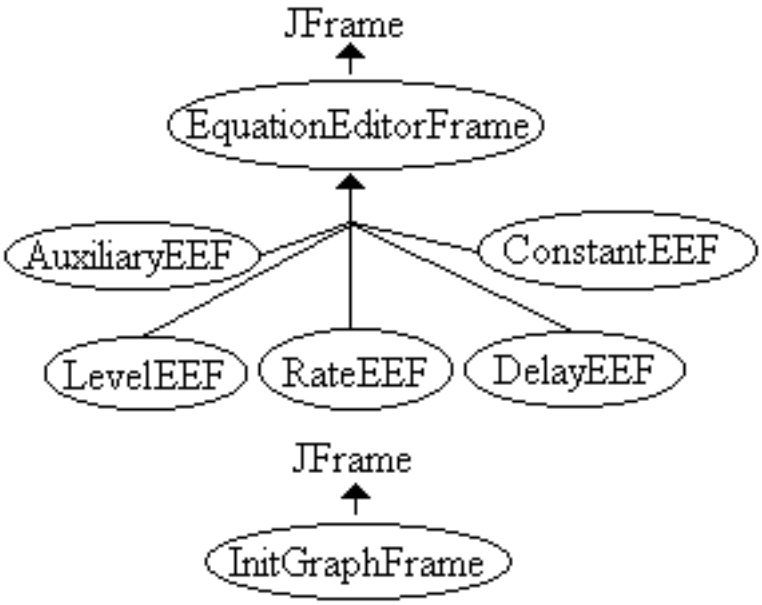

Figura 4.10: *I frame ausiliari per la simulazione*

La soluzione scelta, in questo come in altri casi, è stata, infatti, quella di suddividere le responsabilità per lo svolgimento di un compito complesso fra vari Tool in modo che ciascuno di essi sia in grado di eseguire i compiti affidatigli producendo le informazioni necessarie perchè altri Tool possano svolgere i propri.
Un altro esempio è rappresentato dalla interazione fra il Tool *Equation Solver* e il Tool *Display*: in questo caso il primo svolge il compito di determinare, risolvendo le equazioni associate ai nodi, le grandezze che il secondo si occupa di visualizzare, su richiesta dell'utente, in riferimenti cartesiani bidimensionali.
Nel caso del Tool *Flow Diagram Graphic Editor* la classe *Graph* contiene le strutture dati ed i metodi necessari per memorizzare e gestire le informazioni relative a:

1. i parametri necessari per l'esecuzione della simulazione,
2. le unità di misura associate alle variabili presenti in un diagramma,
3. le equazioni associate ai singoli nodi.

I valori dei parametri necessari per l'esecuzione della simulazione (ovvero l'istante iniziale, l'istante finale, il valore della variabile *time step*, T, e l'unità di misura della variabile tempo) possono essere impostati in qualunque momento utilizzando un $frame$ ausiliario istanza della classe $InitGraphFrame$ (cfr. la figura

4.10[16]).
Al momento della sua istanziazione, ogni elemento della classe *Graph* è caratterizzato da un insieme di valori di default di tali parametri. Tali valori sono memorizzati in una variabile di tipo *String* della classe *Graph* sotto forma si sottostrighe separate da un carattere speciale (il carattere "|") e sono acceduti mediante un elemento della classe *StringTokenizer*, inserito in metodi che ne consentono l'interrogazione e l'aggiornamento.
Le unità di misura, fra le quali è sempre presente quella di default per la variabile tempo, sono immesse dall'utente mediante altri *frame* ausiliari (istanze delle classi illustrate nella figura 4.10) e sono memorizzate in un vettore della classe *Graph* in modo da funzionare come parametri globali per il diagramma su cui l'utente sta lavorando. La struttura dati utilizzata permette all'utente di inserire le unità di misura che ritiene necessarie e che, in seguito, risultano disponibili per essere usate al momento della definizione delle equazioni associate ad altri nodi.
Per l'acquisizione dei parametri necessari alla definizione delle equazioni, la cui struttura dipende dal tipo di nodo cui sono associate, è stata implementata la classe *EquationEditorFrame* dalla quale sono state ricavate, per ereditarietà, le altre classi della figura 4.10, una per ciascun tipo di nodo a cui sia possibile associare una equazione[17]. Risulta pertanto ovvio il motivo per cui in figura 4.10 non compaiano le classi associate ai tipi "sink" e "source".
Ad ognuna delle classi derivate dalla classe *EquationEditorFrame* corrisponde un *frame* ausiliario di struttura diversa mediante il quale l'utente può impostare le grandezze necessarie per caratterizzare l'equazione associata ad un nodo di quel tipo.
Le grandezze immesse dall'utente sono memorizzate in una variabile di tipo *String* (di nome *nodeEquation*) della classe *Node* utilizzata dalla classe *graph* per implementare i nodi del multigrafo. All'interno della variabile *nodeEquation* (separati dal carattere "|") sono memorizzati nell'ordine:

1. l'etichetta del nodo,
2. l'indice del nodo,
3. il tipo del nodo,
4. la stella entrante del nodo, codificata come elemento di tipo *String*,
5. la stella uscente del nodo, codificata come elemento di tipo *String*,
6. i campi aggiunti dalle istanze delle classi dei *frame* ausiliari, la cui struttura dipende dal tipo del nodo corrispondente.

[16] In figura 4.10 si è fatto uso, per motivi di spazio, della sigla *EEF* al posto della dizione estesa di *EquationEditorFrame*.
[17] Ciascuna delle classi derivate della figura è associata al tipo corrispondente per cui, ad esempio, al tipo "level" corrisponde la classe *LevelEEF*.

La stella uscente di un nodo permette di ricavare quali sono i nodi le cui equazioni sono influenzati dalla equazione del nodo corrente mentre la stella entrante permette di sapere quali sono i nodi le cui equazioni influenzano quella del nodo corrente.
Le classi che creano i *frame* ausiliari con cui l'utente imposta le equazioni dei singoli nodi sono caratterizzate da metodi che consentono di inserire (o di aggiornare, a seconda dei casi) in coda alla variabile *nodeEquation* le informazioni caratteristiche di ciascuna equazione.
Ad esempio, ad un nodo di tipo "constant" corrisponde un *frame* ausiliario della classe *ConstantEEF* mediante il quale l'utente può impostare (o aggiornare) il valore costante associato all'icona (che viene inserito o sostituito in coda della variabile *nodeEquation* del nodo corrispondente) e la sua unità di misura (che, se non è già presente, viene inserita fra le altre unità di misura associate al grafo e resa disponibile ai *frame* ausiliari di altre variabili o anche della stessa).
Considerazioni analoghe valgono per i nodi degli altri tipi per i quali la porzione della variabile *nodeEquation* che contiene la descrizione dell'equazione del nodo può avere una struttura più complessa ma tale strutura viene in ogni caso codificata come una sottostringa di una variabile di tipo *String*.

4.4.5 Controlli e persistenza

Il Tool *Flow Diagram Graphic Editor* permette all'utente di:

1. eseguire il controllo della struttura di un diagramma FD prima di richiederne la conversione nel corrispondente diagramma CL,

2. eseguire il controllo della completezza di un diagramma FD prima che questo possa essere simulato e gli andamenti nel tempo delle sue variabili caratteristiche visualizzati,

3. interagire con il *file system* del Sistema Operativo ospite.

Per quanto riguarda le interazioni con il *file system* del Sistema Operativo ospite, implementate mediante metodi della classe *Interactor*, si rimanda alla sezione 4.3.5 dal momento che i comandi per i due Tool sono coincidenti e lo stesso dicasi per la loro implementazione, fatte salve alcune piccole differenze dovute alla presenza delle eqauzioni e delle altre strutture dati per la simulazione.
Il controllo della struttura ed il controllo di completezza (che diremo *controlli asincroni*) sono richiesti esplicitamente dall'utente mediante due comandi della voce *Application* del menù principale presente sul *frame* principale del Tool. La loro gestione è demandata a due metodi della classe *GraphCheck* di cui diremo a breve. Il primo rappresenta il passo preliminare per poter convertire un diagramma FD nel corrispondente diagramma CL e, in caso di esito negativo, non impedisce la conversione. Il secondo deve avere, invece, esito positivo perchè sia

possibile passare alla simulazione di un diagramma FD.
Oltre ai controlli asincroni il Tool esegue su un diagramma FD un certo numero di controlli detti *sincroni*. Tali controlli vengono eseguiti in tempo reale tutte le volte che l'utente modifica un multigrafo agendo sulla rappresentazione pittorica. I controlli sincroni sono stati già descritti nella sezione 4.4.1, cui si rimanda, e riguardano la liceità o meno di un arco di un certo tipo dati i tipi dei nodi in cui esso incide. I controlli asincroni mirano, rispettivamente, a verificare:

1. se ogni nodo è correttamente connesso agli altri nodi del multigrafo,
2. se ad ogni nodo è stata assegnata una equazione e se ogni nodo è correttamente connesso agli altri nodi del multigrafo.

Il controllo della struttura (1) utilizza il metodo *graphCheckToConvert* della classe *GraphCheck* per verificare se le dimensioni della stella entrante e della stella uscente di un nodo sono congruenti con il tipo del nodo. Ad esempio, per i nodi di tipo "sink" se la stella entrante ha dimensione pari a 0 il nodo risulta isolato. In modo simile si ragiona per la stella uscente per nodi di tipo "source", "constant" o "auxiliary" facendo riferimento, in questi casi, alla dimensione della stella uscente dei singoli nodi. Per nodi di tipo "level", "rate" o "delay", infine, il controllo viene eseguito sia sulla stella entrante sia sulla stella uscente. Un nodo di uno di tali tipi è etichettato come isolato solo se entrambe le stelle hanno dimensione pari a 0.
Il controllo della completezza (2) utilizza il metodo *graphCheckToSolve* della classe *GraphCheck* per eseguire due controlli: un controllo di connettività, come nel caso precedente, e un controllo di *soundness*.
Al momento della istanziazione di ciascun nodo di un multigrafo tutti gli attributi vengono inizializzati. Fra gli attributi esiste una variabile di tipo *boolean*, *sound*, che viene inizializzata a $false$ (a $true$ per i nodi di tipo "source" e "sink" che non hanno associata una equazione) e viene posta a $true$ solo dopo che ad un nodo è stata associata una equazione con il comando *EquationEditor*.
Il controllo di *soundness* mira a verificare quali nodi hanno avuta assegnata una equazione e quali no e si basa sul presupposto che i controlli relativi alla correttezza delle singole equazioni vengono eseguiti dai $frame$ ausiliari per la loro definizione.
Se il controllo di *soundness* ha esito positivo allora il multigrafo è pronto ad essere simulato mentre se ha esito negativo il metodo, su richiesta dell'utente, può usare un $frame$ ausiliario per visualizzare i nodi cui deve essere ancora assegnata una equazione.
Il controllo di connettività, invece, si limita a controllare se ciascun nodo è correttamente collegato agli altri ma un suo esito negativo non impedisce l'esecuzione della simulazione. Eventuali nodi scollegati si limiteranno a dare un contributo parziale o nullo alla simulazione.

4.5 *Display*

Il Tool *Display* permette all'utente di visualizzare un certo numero di grandezze variabili nel tempo. Tale Tool è caratterizzato da un *frame* principale, che contiene il menù principale con i comandi per la visualizzazione delle singole grandezze e per la customizzazione del Tool, e da un certo numero di *frame* di visualizzazione (cfr. la sezione 4.5.1). Oltre che da tali *frame* il Tool è caratterizzato da un certo numero di *frame* ausiliari mediante i quali l'utente può impostare i valori globali del Tool (cfr. la sezione 4.5.2).

4.5.1 Il *frame* principale ed i *frame di visualizzazione*

Il Tool (cfr. le figure 4.11 e 4.12) è ottenuto come istanza della classe *DisplayFrame*. La classe *DisplayFrame* svolge le seguenti funzioni:

1. definisce le voci del menù principale con i relativi *listener*,
2. definisce una istanza della classe *GlobalSettings*,
3. definisce una istanza della classe *TopLevelPanel*,
4. crea tante istanze della classe *GraphDisplayFrame* quante risultano necessarie.

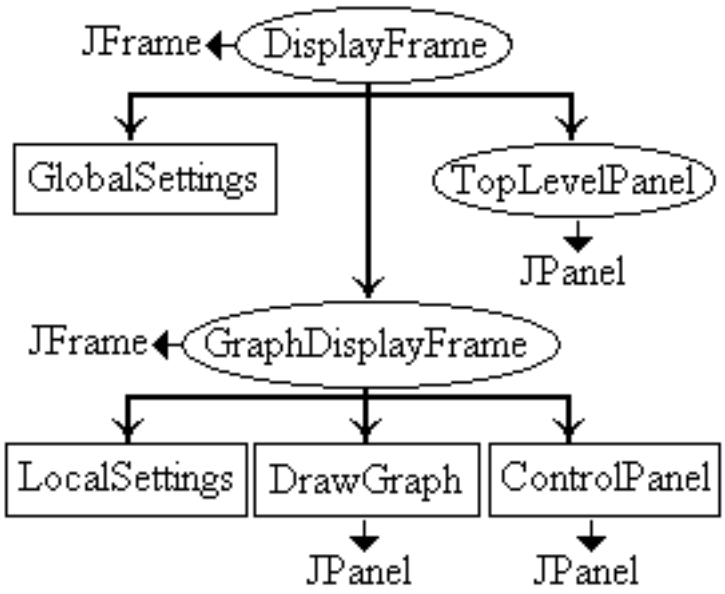

Figura 4.11: *Il frame principale e i frame di visualizzazione (1)*

Le voci del menù principale sono definite mediante le classi, standard *Java*, $JMenu$ e $JMenuItem$ e a ciascuna di esse è associato un *listener* che, a seconda dei casi, può causare la creazione di un $frame$ ausiliario (cfr. la sezione 4.5.2) oppure di un $frame$ di visualizzazione istanza della classe $GraphDisplayFrame$.

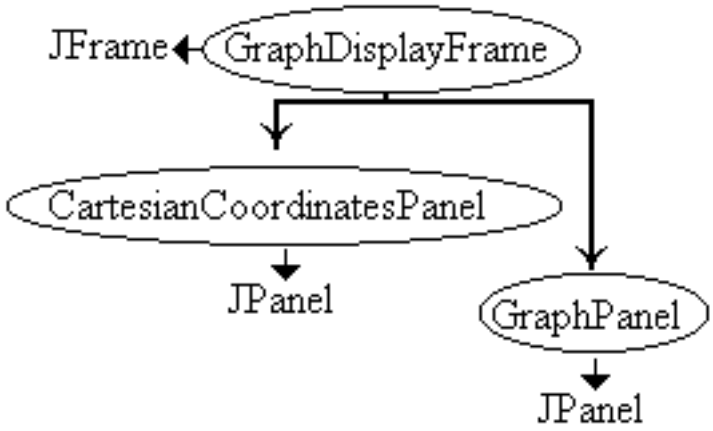

Figura 4.12: *Il frame principale e i frame di visualizzazione (2)*

La classe *GlobalSettings* permette di definire un oggetto *gs* che rappresenta lo *stato globale* del Tool. L'oggetto *gs* è caratterizzato da un certo numero di attributi con i relativi metodi accessori. Gli attributi corrispondono a caratteristiche di visualizzazione dei grafici quali: colore, stile di tracciamento, orientamento, sovrapponibilità dei grafici, tipo di scala per l'asse delle ascisse e tipo di scala dell'asse delle ordinate. A ciascun attributo corrisponde un $frame$ ausiliario (cfr. la sezione 4.5.2).
La classe $TopLevelPanel$ crea sul $frame$ principale un $JPanel$ contenente i pulsanti *Clear All* e *Close All* i cui *listener* agiscono su tutte le istanze della classe $GraphDisplayFrame$ correntemente visualizzate e i cui riferimenti sono memorizzati in un vettore locale alla classe $DisplayFrame$.
Le istanze della classe $GraphDisplayFrame$ sono utilizzate per visualizzare uno o più grafici su un unico riferimento cartesiano bidimensionale.
A tale scopo la classe $GraphDisplayFrame$ fa uso delle classi seguenti (cfr. le figure 4.11 e 4.12):

1. $LocalSettings$,
2. $ControlPanel$,
3. $CartesianCoordinatesPanel$,
4. $DrawGraph$,

5. *GraphPanel*.

La visualizzazione del primo grafico all'interno di un oggetto *gdf*, istanza della classe *GraphDisplayFrame*, avviene, sulla base dei valori dello stato globale del Tool, in un riferimento cartesiano bidimensionale creato come istanza della classe *CartesianCoordinatesPanel*. Il riferimento cartesiano viene tracciato su un *panel dg*, istanza della classe *DrawGraph*. Tale *panel* viene creato opaco e in grado di catturare gli eventi del mouse. Ad esso è infatti associato un menù flottante (istanza della classe *DrawGraphPopUpMenu*) che consente l'esecuzione di operazioni sui grafici visualizzati in un *frame gdf*.
La classe *CartesianCoordinatesPanel* si occupa di:

1. tracciare gli assi delle ascisse e delle ordinate,
2. tracciare i riferimenti e i valori numerici sull'asse delle ascisse,
3. nel caso la scala di visualizzazione sia di tipo normalizzato, tracciare i riferimenti e i valori numerici sull'asse delle ordinate,
4. tracciare informazioni ausiliarie.

Nel caso che la scala dell'asse delle ordinale sia di tipo *lineare* o *logaritmica* il compito di tracciare i riferimenti e i valori sull'asse delle ordinate spetta alla classe *GraphPanel* che crea il *panel* sul quale viene tracciato il grafico della variabile.
Le informazioni ausiliarie sono:

1. il tipo di scala sull'asse delle ascisse,
2. il tipo di scala sull'asse delle ordinate,
3. il nome di ciascuna variabile e la relativa unità di misura,
4. il valore della varibaile T, nel caso la scala sull'asse delle ascisse sia in *ticks*.

Oltre a creare un *panel* in grado di contenere un riferimento cartesiano e sul quale saranno sovrapposti i vari grafici, la classe *GraphDisplayFrame* crea anche una istanza della classe *ControlPanel*.
La classe *ControlPanel* ha il compito di creare un *panel* caratterizzato da un insieme di elementi di controllo (*check boxes* e *pulsanti* con i relativi *listener*) che consentono di modificare lo stato globale del Tool in modo da definire uno stato locale al singolo *frame* di visualizzazione (istanza della classe *LocalSettings*).
I grafici successivi vengono tracciati, pertanto, in un modo che risulta determinato sia dai valori dello stato globale *gs* sia dai valori dello stato locale *ls*.
Lo stato locale permette di modificare il valore di parametri quali: lo stile di tracciamento, la sovrapponibilità o meno di altri grafici sul *frame* di visualizzazione, il tipo di scala sull'asse delle ascisse, il tipo di scala sull'asse delle ordinate

e l'orientamento dei grafici sul $frame$ di visualizzazione. La relazione esistente fra lo stato globale Tool e lo stao locale del singolo $frame$ di visualizzazione è la seguente: lo stato globale gs influenza la modalità di tracciamento dei grafici su nuovi $frame$ di visualizzazione, lo stato locale determina le caratteristiche dei grafici già tracciati oppure il tracciamento di altri grafici su un $frame$ di visualizzazione esistente.
Se lo stato globale impedisce la sovrapposizione dei grafici in uno stesso $frame$, i grafici successivi sono tracciati in $frame$ di visualizzazione distinti, uno per ciascun $frame$. Se lo stato globale consente la sovrapposizione di più grafici su uno stesso $frame$, i grafici vengono tracciati sovrapposti, a meno che lo stato locale di un singolo $frame$ non dichiari il $frame$ stesso come non sovrapponibile, in modo da impedire il tracciamento dei grafici successivi su quelli tracciati fino a quel momento. Lo stato locale, infatti, può essere modificato in qualunque momento e, quindi, anche dopo che su un $frame$ *gdf* è stato visualizzato più di un grafico.
I singoli grafici non sono, tuttavia, tracciati su *panel* *dg* istanze della classe $DrawGraph$ ma sono tracciati su *panel* istanze della classe $GraphPanel$. Tali *panel* vengono creati come trasparenti (ovvero senza un colore dello sfondo) in modo da poter essere sovrapposti gli uni agli altri come se fosero dei lucidi. In questo modo più grafici possono essere sovrapposti uno all'altro senza che un tracciamento oscuri gli altri. Il tracciamento degli assi cartesiani e dei vari grafici avviene, infatti, utilizzando uno *stack* di elementi di cui si fa l'*upcast* a $JPanel$. Il primo elemento dello *stack* è una istanza della classe $CartesianCoordinatesPanel$ mentre i successivi sono istanze della classe $GraphPanel$ su cui sono tracciati i singoli grafici.
La disponibilità di tale *stack* permette di definire un insieme di metodi per implementare:

1. le operazioni del menù flottante (istanza di $DrawGraphPopUpMenu$), che agiscono sui singoli grafici,

2. le operazioni del pannello di controllo locale (istanza di $ControlPanel$), che agiscono su tutti i grafici visualizzati su uno stesso $frame$ di visualizzazione,

3. le operazioni del pannello di controllo globale (istanza di $TopLevelPanel$), che agiscono su tutti i $frame$ di visualizzazione.

Alcune delle operazioni del menù flottante (cancellazione di un grafo, visualizzazione di un grafo, cambiamento dell'ordine di visualizzazione dei grafi) fanno riferimento a metodi della classe $GraphDisplayFrame$ che, in pratica, gestisce anche le azioni catturate dai *listener* della classe $ControlPanel$ mentre i metodi della classe $TopLevelPanel$ fanno riferimento a metodi "centralizzati" ovvero propri della classe $DisplayFrame$.

4.5.2 I *frame* ausiliari

La figura 4.13 presenta le classi utilizzate per la definizione dei *frame ausiliari* del Tool *Display*. Lo scopo dei *frame ausiliari* è quello di consentire il settaggio dei valori che definiscono lo stato globale del Tool.

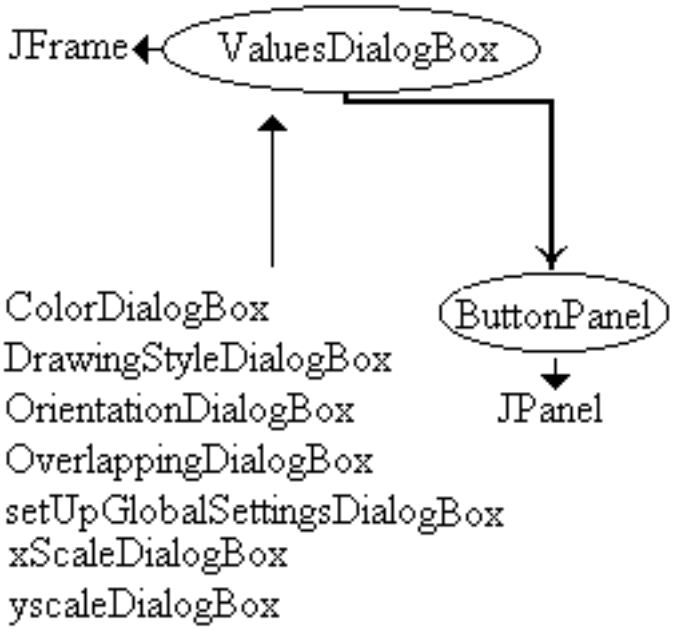

Figura 4.13: *Le classi per i frame ausiliari*

Al momento della istanziazione di un oggetto della classe $DispalyFrame$ viene creato un oggetto gs, istanza della classe $GlobalSettings$. Tale oggetto viene inizializzato con i valori di default dei parametri che caratterizzano lo stato globale del Tool. Al momento della creazione di un generico $frame$ di visualizzazione gdf (istanza della classe $GraphDisplayFrame$) viene creato (dalla classe $ControlPanel$) un oggetto ls della classe $LocalSettings$. Tale oggetto è inizializzato con gli stessi valori dell'oggetto gs.
L'oggetto ls, associato ad un oggetto gdf, può essere modificato con gli elementi della classe $ControlPanel$ mentre l'oggetto gs può essere modificato usando i *frame ausiliari* ottenibili mediante i *listener* associati agli elementi della voce *Global Settings* del menù principale del Tool, creato dalla classe $DisplayFrame$. I $frame$ ausiliari consentono il settaggio e l'interrogazione di tutti gli elementi caratteristici dello stato globale del Tool. Li si può pensare suddivisi in tre categorie:

1. $frame$ per il settaggio globale,
2. $frame$ per l'interrogazione globale,

3. *frame* per il settaggio puntuale.

Alla prima categoria appartengono i *frame* della classe *setUpGlobalSettingsDialogBox*. I *frame* della seconda categoria sono ottenuti dalla stessa classe settando un parametro nel costruttore che definisce i parametri com "read only", in modo da consentirne l'interrogazione ma non il settaggio.
Alla terza categoria (*frame* puntuali, cfr. la figura 4.13) appartengono i *frame* delle classi:

1. *ColorDialogBox*,
2. *DrawingStyleDialogBox*,
3. *OrientationDialogBox*,
4. *OverlappingDialogBox*,
5. *xScaleDialogBox*,
6. *yScaleDialogBox*,

I *frame* di tali classi permettono di settare un parametro alla volta e perciò sono utilizzabili per il settaggio puntuale dei parametri. I *frame* ausiliari sono gestiti dalla classe *DisplayFrame* in modo che se è presente un *frame* globale non possono essere visualizzati *frame* puntuali e viceversa. In più, i *frame* puntuali possono essere aperti in copia unica in modo da evitare settaggi conflittuali dei singoli parametri: i *frame* ausiliari sono implementati, infatti, come finestre di dialogo non modali per cui possono essere tenuti aperti senza che questo impedisca all'utente di interagire con gli altri *frame* del Tool, come avverrebbe se i *frame* ausiliari fossero stati implementati come finestre di dialogo modali. Indipendentemente dalla categoria alla quale appartengono, i *frame* ausiliari (cfr. la figura 4.13) derivano per ereditarietà dalla classe *ValuesDialogBox* che, a sua volta, eredita dalla classe, standard *Java*, *JFrame*.
La classe *ValuesDialogBox* ha il compito di settare le dimensioni del *frame* in funzione della classe derivata e di istanziare e posizionare un oggetto della classse *ButtonPanel*. La classe *ButtonPanel* (che eredita da *JPanel*) definisce un *panel* con due pulsanti, etichettati "OK" e "Cancel", dei quali fornisce i *listener*.
Il listener del pulsante "Cancel" della classe *ButtonPanel* gestisce localmente il ripristino del valore precedente. Entrambi i listener fanno riferimento ad un metodo opportuno di ciascuna classe derivata. Le classi derivate dalla *ValuesDialogBox*, infatti, ridefiniscono i metodi *closeFrameOnOK*() e *closeFrameOnCancel*() della classe *ValuesDialogBox* in modo che entrambi svolgano le opportune operazioni di chiusura e di inizializzazione.
Per il settaggio dei valori, le classi derivate definiscono ciascuna un proprio

JPanel contenente un numero variabile di *pulsanti di scelta*. Il numero dei *pulsanti di scelta* (istanze di *JradioButton*) dipende dalla classe derivata. Ad esempio, nel caso di classi come *xScaleDialogBox*, *OrientationDialogBox* o *OverlappingDialogBox* si hanno due pulsanti di scelta, nel caso di classi come *yScaleDialogBox* e *DrawingStyleDialogBox* si hanno tre pulsanti di scelta e, infine, nel caso della classe *ColorDialogBox* si hanno dieci pulsanti di scelta.
I pulsanti di scelta sono inizializzati con il valore corrente del parametro corrispondente nello stato globale. I pulsanti di scelta, inoltre, hanno associato un *listener*: il *listener* intercetta l'azione di selezione dei vari pulsanti ed aggiorna il valore del parametro corrispondente nello stato globale. Tale gestione permette di tenere aperte le singole finestre di dialogo sino a che l'utente non decide di chiuderle con uno dei due pulsanti a disposizione: "OK" o "Cancel".
Nel primo caso l'ultimo valore impostato diventa il valore corrente (nello stato globale) del parametro corrispondente alla finestra di dialogo mentre, nel secondo caso, viene ripristinato il valore che il parametro aveva al momento della istanziazione della finestra di dialogo. Tutte le selezioni effettute dal momento della apertura di una finestra di dialogo al momento della sua chiusura sono state registrate nello stato globale del Tool e possono aver influenzato la creazione dei *frame* di visulizzazione.

4.6 Il Tool *Equation Solver* ed *i convertitori da CL a FD e viceversa*

4.6.1 Introduzione

Dei Tool descritti nella presente sezione sono state, attualmente, implementate solo le interfacce utente per cui di questi due Tool ne verrà data solo una descrizione sommaria.
Il Tool *Equation Solver* è caratterizzato da una interfaccia utente perchè è stato pensato per poter lavorare sia su diagrammi Fd la cui descrizione è contenuta in file, di formato oggetto o di formato testo, sia su un diagramma correntemente visualizzato sul *canvas* del *Flow Diagram Graphic Editor*. Nel primo caso il Tool viene acceduto tramite la sua interfaccia utente, mediante la quale è possibile utilizzare comandi per l'accesso ai file, per il settaggio dei parametri per la simulazione, per il controllo e la simulazione di un diagramma FD e la visualizzazione dei risultati così ottenuti. Nel secondo caso l'accesso al Tool avviene direttamente tramite i metodi per la simulazione di un diagramma FD, come visto nella descrizione del Tool *Flow Diagram Graphic Editor* fatta nelle sezioni 3.4 e 4.4.
Un discorso analogo vale per *i convertitori da CL a FD e fa FD a CL* che possono essere acceduti in tre modi distinti: tramite il Tool *TopLevel*, tramite le interfacce dei Tool *Causal Loop Graphic Editor* (il convertitore da diagrammi CL a diagrammi FD) e *Flow Diagram Graphic Editor* (il convertitore da diagrammi

FD a diagrammi CL) oppure mediante una interfaccia ad hoc (detta *Converter Tool*) in modo da permettere all'utente di agire sulla descrizione di diagrammi contenuti in file di tipo testo oppure di tipo oggetto indipendentemente da ogni altro Tool.

4.6.2 *Equation Solver*

Il Tool *Equation Solver* è stato pensato per operare sia in cooperazione con il *Flow Diagram Graphic Editor* sia come Tool autonomo per l'accesso a file contenenti diagrammi FD. Il Tool è caratterizzato da un certo numero di classi che:

1. ne definiscono l'interfaccia,
2. consentono l'interazione con il *file system* del Sistema Operativo ospite,
3. consentono la esecuzione di operazioni di controllo sulla correttezza delle equazioni associate ai nodi di un multigrafo,
4. consentono di eseguire la simulazione di un diagramma di cui si sia controllata la correttezza.

Il Tool, come illustra la figura 4.14, oltre che con il Tool *Flow Diagram Graphic Editor*, interagisce anche con il Tool *Display* del quale accede la classe $DisplayFrame$ passandole i valori da visualizzare.

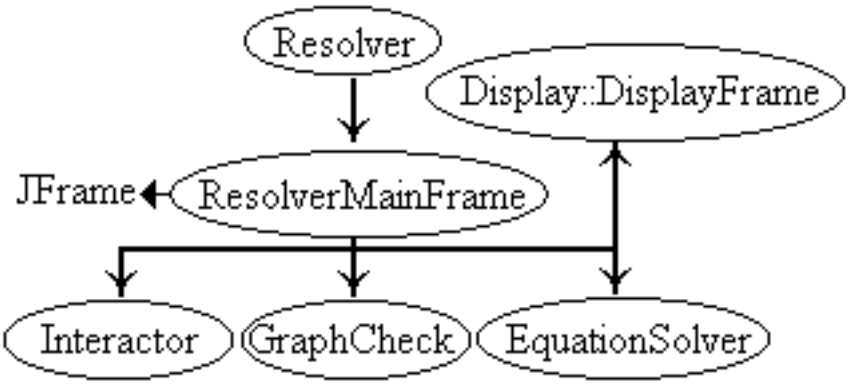

Figura 4.14: *Le classi del Tool Equation Solver*

L'interfaccia è realizzata con una classe $ResolverMainFrame$, che eredita dalla classe $JFrame$, al cui interno sono definiti i *listener* delle voci del menù principale. I *listener* gestiscono l'apertura di $frame$ ausiliari per:

1. l'interazione con il *file system* del Sistema Operativo ospite,

2. il settaggio dei parametri necessari per l'esecuzione di una simulazione,

3. l'esecuzione della simulazione.

L'interazione con il *file system* del Sistema Operativo ospite è realizzata mediante metodi della classe $Interactor$ che consentono l'accesso a file contenenti la descrizione di diagrammi FD.
Il settaggio dei parametri necessari per l'esecuzione di una simulazione prevede l'utilizzo di un certo numero di $frame$ ausiliari per:

1. il settaggio del valore della variabile *time step*, T,

2. il settaggio degli istanti iniziale (T_0) e finale (T_N) della simulazione,

3. la scelta dell'algoritmo di risoluzione delle equazioni alle differenze finite fra quelli disponibili, Eulero e Runge-Kutta del secondo e del quarto ordine.

I valori dei parametri numerici (T, T_0, T_N) sono contenuti, come valori predefiniti per il *Resolver*, nelle strutture dati del multigrafo (cfr. la sezione 4.4.4) mentre la selezione dell'algoritmo avviene a questo livello perchè è solo al momento dell'esecuzione della simulazione che il tipo dell'algoritmo da usare acquisisce pienamente significato.
L'esecuzione della simulazione di un diagramma FD prevede che:

1. le strutture dati di un diagramma FD siano caricate in memoria,

2. per ogni nodo del multigrafo sia ricavata la corrispondente eqauazione,

3. sia verificata la correttezza di tutte le equazioni.

Se il controllo (eseguito con metodi della classe $GraphCheck$) ha esito positivo allora è possibile eseguire la simulazione (usando i metodi della classe $EquationSolver$) e visualizzare gli andamenti delle variabili del diagramma (mediante una interazione diretta con la classe $DiaplayFrame$ del package $Display$).
Se, invece, il controllo ha avuto esito negativo è necessario riaprire il diagramma con il Tool *Flow Diagram Graphic Editor* e correggere le equazioni incomplete e/o inesatte. A tale scopo, come ausilio per il processo di revisione, i metodi di controllo della classe $GraphCheck$ generano un *report* in formato testo con l'elenco delle equazioni che è necessario correggere.
La classe $EquationSolver$ svolge il duplice ruolo di motore di risoluzione delle equazioni associate ai singoli nodi e di $parser$ di tali equazioni.
Il motore di risoluzione ordina le equazioni in base al tipo ed alle relazioni di precedenza e passa le equazioni così ordinate, una alla volta, al parser.
L'ordinamento prevede che, sulla base dei valori iniziali delle variabili di tipo "level" e delle variabili esogene (che possono essere sia di tipo "constant" sia di tipo "auxiliary") vengano calcolati i valori iniziali sia delle variabili di tipo "rate" sia

delle variabili di tipo "auxiliary" non esogene.
Una volta avuti i valori iniziali di tutte le variabili è possibile iniziare il procedimento iterativo di risoluzione calcolando, ad ogni passo, prima i valori delle variabili di tipo "level" e delle variabili di tipo "auxiliary" e poi i valori delle variabili "rate". Il parser interpreta ciascuna equazione nell'ordine stabilito dal motore di risoluzione e traduce le informazioni in essa contenute nelle chiamate degli opportuni metodi matematici. Il parser, in pratica, risolve le singole equazioni e memorizza il risultato in un'array associato ad ogni variabile.
Il procedimento di risoluzione, di tipo iterativo, viene ripetuto dal'istante iniziale della simulazione T_0 all'istante finale T_N per un dato valore del *time step*, T.
Il risultato della simulazione è rappresentato da un N array, se N sono i nodi del multigrafo, che possono essere passati (insieme ai nomi delle variabil, alle relative unità di misura ed ai valori T, T_0, T_N) al Tool *Display* per la loro visualizzazione.

4.6.3 *I convertitori da CL a FD e viceversa*

I Tool di conversione dei diagrammi CL in diagrammi FD e viceversa sono accessibili in vari modi (cfr. la sezione 4.6.1). Nella presente sezione viene descritta la struttura interna del *Converter Tool* che rappresenta un Tool autonomo mediante il quale è possibile accedere ai vari convertitori in modo autonomo rispetto agli altri Tool.

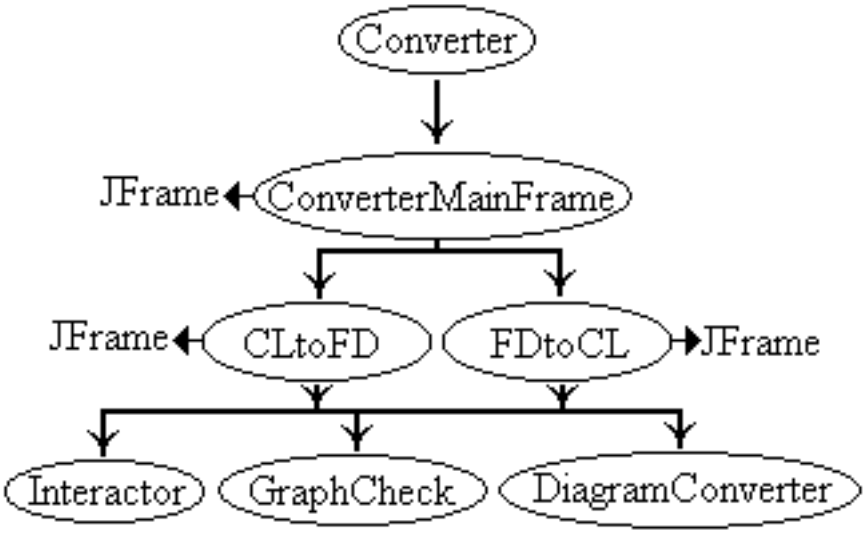

Figura 4.15: *Le classi per la conversione*

L'interfaccia del Tool è ottenuta (tramite la classe *Converter*) come istanza della classe *ConvertermainFrame*. L'interfaccia ha una struttura molto semplice in cui compaiono un menù con due voci a ciascuna delle quali è associato un

listener. I due *listener* gestiscono, rispettivamente, l'istanziazione in copia unica di un oggetto della classe $CLtoFD$ e di un oggetto della classe $FDtoCL$.
Entrambi gli oggetti fanno uso dei metodi della classe $Interactor$ per accedere al file contenente la descrizione in formato testo o in formato oggetto del diagramma da convertire (cfr. la sezione 3.7).
Una volta acceduta la descrizione di un diagramma il *Converter Tool* esegue, in modo trasparente ed automatico, una serie di controlli. Se i controlli hanno esito positivo, il *Converter Tool* provvede alla conversione del diagramma dal formato sorgente al formato destinazione, utilizzando i metodi della classe $DiagramConverter$.
Se i controlli hanno esito negativo il *Converter Tool* visualizza un messaggio d'errore e produce un *report* in formato testo con tutti gli errori riscontrati durante l'analisi del diagramma da convertire. L'unico modo per ovviare agli errori è quello di riaprire il diagramma con l'opportuno Tool grafico (il *Causal Loop Graphic Editor* per i diagrammi CL o il *Flow Diagram Graphic Editor* per i diagrammi FD) ed apportarvi le necessarie correzioni (cfr. le sezioni 4.3.4 e 4.4.5).
I controlli eseguiti dal *Converter Tool* sono realizzati con metodi della classe $GraphCheck$.
Nel caso di diagrammi CL da convertire in diagrammi FD viene controllato che:

1. alle etichette del diagramma sia stato assegnato un tipo,
2. agli elementi di connessione del diagramma siano stati assegnati tipo e segno,
3. che vi sia congruenza fra il tipo e l'orientamento degli elementi di connessione e il tipo delle etichette in cui questi incidono.

Nel caso di diagrammi FD da convertire in diagrammi CL vengono eseguiti controlli più semplici che mirano solo a verificare che tutti i nodi di un diagramma siano connessi. In più i metodi di controllo cercano di ricavare i segni sugli elementi di connessione nel diagramma CL in base alle informazioni presenti nel diagramma FD, quali tipo ed orientamento di un elemento di connessione ed equazioni dei nodi alle sue estramità.
La conversione dei diagrammi coinvolge sia la loro rappresentazione pittorica sia la struttura astratta (grafo o multigrafo) corrispondente ed è effettuata dai metodi della classe $DiagramConverter$. Dato un diagramma da convertire, i metodi di conversione lavorano in due tempi:

1. lavorano sulla struttura astratta,
2. lavorano sulla rappresentazione pittorica.

Lavorando sulla struttura astratta i metodi di conversione trasformano (nel caso della conversione da CL a FD) un grafo in un multigrafo oppure (nel caso della

conversione da FD a CL) un multigrafo in un grafo.
In entrambi i casi per passare poi alla nuova rappresentazione pittorica è sufficiente riportare dalla vecchia rappresentazione alla nuova le coordinate dei punti di incidenza degli elementi di connessione e orientare coerentemente con la nuova rappresentazione gli elementi di connessione.
Nel caso il passaggio sia da un diagramma CL ad uno Fd nei punti di incidenza verranno disegnate le opportune coppie icona/etichetta mentre se il passaggio è da un diagramma Fd ad uno CL nei punti di incidenza verrà rappresentata la sola etichetta. Per ulteriori dettagli in merito alle problematiche relative alla conversione delle rappresenatzioni pittoriche e delle strutture astratte si rimanda alla sezione 6.3.

Capitolo 5

$D(a)ySy\,ToolBox$: note di utilizzo

5.1 Introduzione

A questo punto della trattazione, dopo aver descritto la struttura astratta (cfr. il Capitolo 3) e la struttura interna (cfr. il Capitolo 4) dei vari Tool che costituiscono l'ambiente di simulazione $D(a)ySy\,Tool\ Box$, si ritiene opportuna la descrizione, per i Tool in più avanzata fase di implementazione (cfr. la sezione 6.2), di alcuni *casi d'uso* ([BSL02]).
Scopo del presente Capitolo, quindi, non è tanto quello di dare un *manuale d'uso* di ciascun Tool quanto quello di presentare alcuni *scenari* di uso dei singoli Tool in modo da consentirne un più agevole utilizzo.
Per la presentazione dei singoli scenari verrà fatto uso, essenzialmente, della notazione UML per i *casi d'uso* ([BSL02], cfr. l'Appendice A). Una breve introduzione a tale notazione viene data nella sezione A.1.2. I diagrammi che verranno usati per descrivere i singoli Tool sono stati tracciati utilizzando il programma *Poseidon* (cfr. la sezione A.1.4).
Alcuni dei Tool che compongono l'ambiente di simulazione $D(a)ySyToolBox$ sono stati progettati per essere acceduti sia direttamente sia attraverso un Tool di coordinamento, detto *TopLevel* (cfr. la sezione 5.2) mentre altri sono accessibili solo autonomamente o in modo indiretto tramite comandi di altri Tool.
Alla prima categoria appartengono sia il *Causal Loop Graphic Editor* sia il *Flow Diagram Grafic Editor* sia i convertitori $CLtoFD$ e $FDtoCL$. Alla seconda categoria, invece, appartengono i Tool $ConverterTool$, $EquationSolver$ e $Display$. Il primo di tali Tool permette di accedere ai Tool di conversione in modo autonomo da ogni altro Tool così da consentire all'utente di lavorare su diagrammi CL o FD contenuti in file. Il secondo, acceduto anche indirettamente mediante un comando del Tool *Flow Diagram Grafic Editor*, permette l'esecuzione di simulazioni su diagrammi FD preventivamente inizializzati ma sui quali l'utente non voglia eseguire operazioni di editing e dei quali ritenga superflua la visualizzazione.
Il terzo, infine, accessibile solo indirettamente tramite il *Flow Diagram Grafic*

Editor, permette la visualizzazione delle grandezze risultanti da una simulazione di un diagramma FD.

5.2 *TopLevel*

Il Tool *TopLevel* è caratterizzato da una interfaccia che può essere istanziata (ovvero clonata con il comando *Clone*) fino a tre volte in modo da produrre tre copie del Tool fra loro indipendenti. Ciasuna copia possiede gli stessi comandi delle altre e può essere chiusa indipendentemente dalle altre.

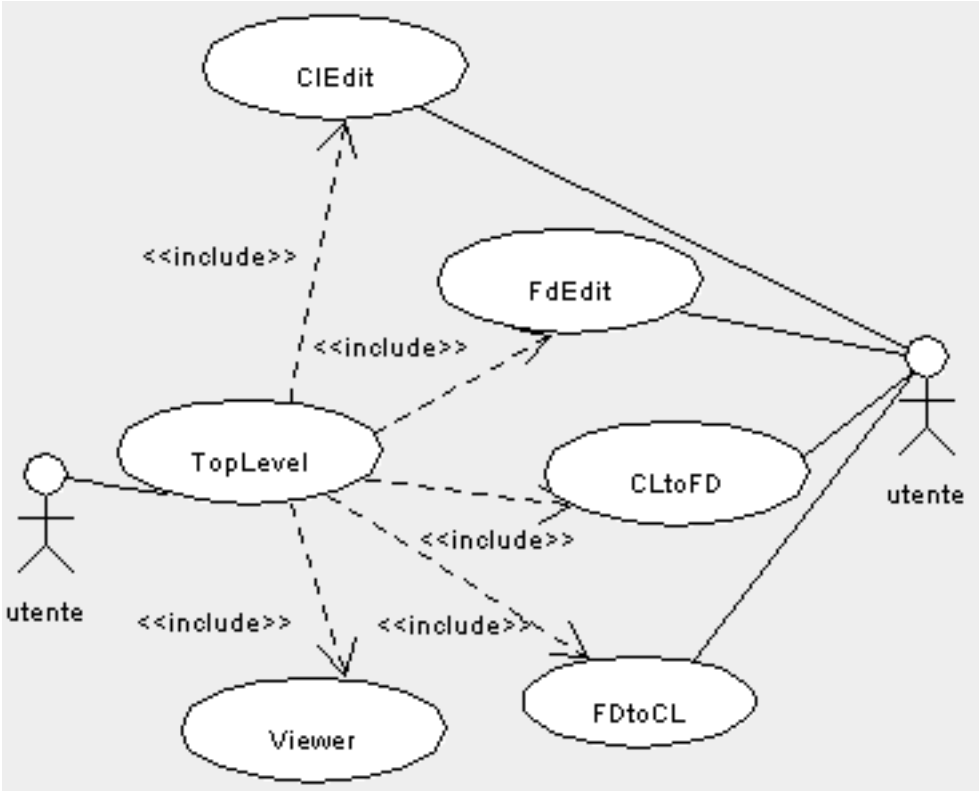

Figura 5.1: *TopLevel: alcuni dei casi d'uso*

Il Tool (cfr. la figura 5.1) è stato progettato per svolgere le seguenti funzioni:

1. consentire l'accesso ad un numero qualunque di istanze del *Causal Loop Graphic Editor*, *ClEdit*, ciascuna in un thread separato,

2. consentire l'accesso ad un numero qualunque di istanze del *Flow Diagram Graphic Editor*, *FdEDit*, ciascuna in un thread separato,

3. consentire l'accesso ad un numero qualunque di copie del Tool di conversione da diagrammi CL a diagrammi FD ($CLtoFD$) contenuti in file,

4. consente l'accesso ad un numero qualunque di copie del Tool di conversione da diagrammi FD a diagrammi CL ($FDtoCL$) contenuti in file,

5. consentire la creazione di un certo numero di finestre di visualizzazione.

Le finestre di visualizzazione (dette *viewer*, cfr. le sezioni 3.2 e 4.2) sono attualmente in fase di implementazione. Correntemente è possibile istanziarne sino a cinque per ciascuna istanza dell'interfaccia per un totale di al più quindici *viewer* contemporaneamente aperti sullo schermo. I singoli *viewer* possono essere resi attivi utilizzando la corrispondente voce del menù $Pop-up$ presente sull'interfaccia mediante la quale il *viewer* è stato creato.
Secondo le specifiche del progetto (cfr. le sezioni 3.2 e 4.2) l'utente potrà utilizzare i singoli *viewer* per:

1. visualizzare diagrammi CL o FD contenuti in file,

2. eseguire sui diagrammi visualizzati alcune operazioni dipendenti dal tipo di diagramma.

Nel caso dei diagrammi CL la sola operazione prevista è una operazione di conversione del diagramma visualizzato nel corrispondente diagramma FD. La conversione viene effettuata utilizzando metodi del Tool $CLtoFD$ (interno al Tool *Converter Tool*) sulle strutture dati del diagramma visualizzato.
Si ricorda, infatti, che la conversione diretta mediante il Tool $CLtoFD$ è, invece, una conversione "al buio" ovvero da file a file senza che l'utente possa visualizzare il digramma che sta per essere convertito.
Nel caso dei diagrammi FD sono previste:

1. l'operazione di conversione dal diagramma FD al corrispondente diagramma CL, per la quale valgono considerazioni analoghe alle precedenti,

2. l'operazione di simulazione del diagramma ed eventualmente di visualizzazione dei risultati della simulazione.

La simulazione del diagramma presuppone che il diagramma sia stato correttamente inizializzato e fa uso di metodi del Tool *Equation Solver* mentre la visualizzazione dei risultati della simulazione richiede una interazione con il Tool *Display*.

5.3 *Causal Loop Graphic Editor*

Il *Causal Loop Graphic Editor* (al quale, nelle figure e nel testo di questo Capitolo, si farà riferimento con il nome di *ClEdit*) è il Tool che consente la creazione e la manipolazione di diagrammi CL.
Come risulta dalla figura 5.2, il Tool è accessibile sia in modo autonomo sia atraverso il Tool *TopLEvel*. Le principali funzionalità del Tool (illustrate nelle figure 5.2, 5.3 e 5.4) comprendono operazioni per:

1. la gestione delle rappresentazioni pittoriche e dei grafi, cfr. la figura 5.3,
2. la gestione delle interazioni con il Sistema Operativo ospite, cfr. la figura 5.4,
3. la conversione di formato.

Le operazioni che consentono la gestione dei grafi e delle corrispondenti rappresentazioni pittoriche, raggruppate sotto la dizione generica di $GestioneGrafi$ nelle figure 5.2 e 5.3, comprendono operazioni:

1. di tracciamento ($Draw$) di etichette e di elementi di connessione,
2. di editing ($Edit$) di etichette e di elementi di connessione,
3. di selezione di sottografi ($Sottografi$),
4. di definizione dei segni sugli anelli presenti nei diagrammi.

Le operazioni che consentono la gestione delle interazioni con il Sistema Operativo ospite, raggruppate sotto la dizione generica di $GestioneFile$ nelle figure 5.2 e 5.4, comprendono operazioni:

1. per la creazione di un nuovo diagramma che potrà essere memorizzato in un nuovo file,
2. per l'accesso a diagrammi la cui descrizione è contenuta in un file esistente,
3. per la stampa della descrizione di un diagramma, descrizione contenuta in un file in formato testo.

Le operazioni di conversione di formato (che consentono di trasformare un diagramma CL nel corrispondente diagramma FD) includono operazioni per la verifica della corretta tipizzazione di tutti gli elementi di un diagramma CL. Tali operazioni non saranno descritte in ulteriore dettaglio in questa sede in quanto ancora in fase di implementazione.

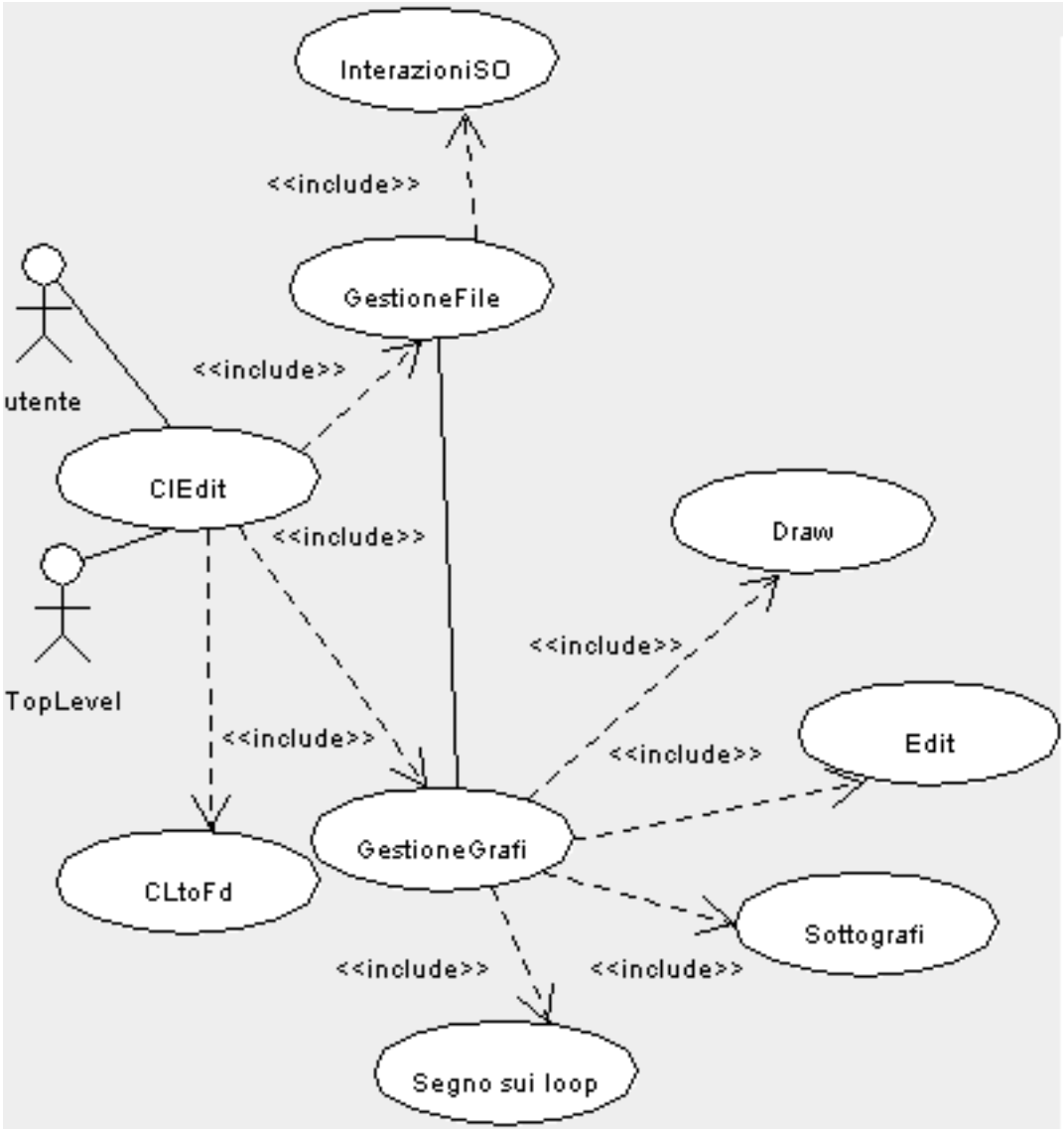

Figura 5.2: *ClEdit: alcuni dei casi d'uso*

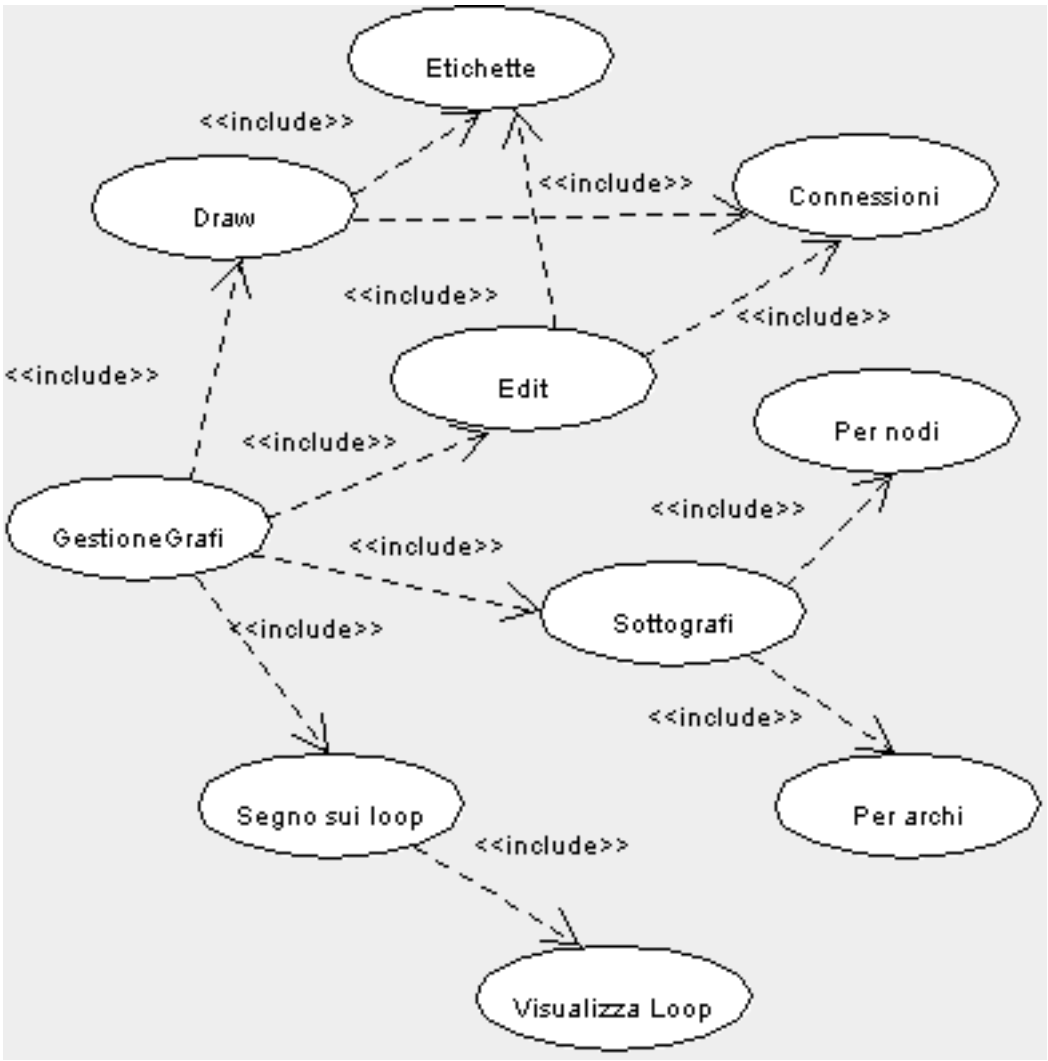

Figura 5.3: *ClEdit: gestione dei grafi*

La procedura di utilizzo del Tool *ClEdit* prevede che l'utente:

1. istanzi un nuovo diagramma CL con il comando *New* oppure acceda ad un diagramma la cui descrizione è memorizzata in un file in formato testo (*Import*) o in formato oggetto (*Open*),
2. compia una serie di operazioni sul diagramma,
3. se lo desidera, salvi il diagramma modificato in un file in formato testo (*Export*) o in formato oggetto, con lo stesso nome (*Save*) o con un nome diverso (*Save as*).

Il comando *New* mette a disposizione dell'utente un *canvas* vuoto sul quale l'utente può tracciare un diagramma CL. Al diagramma così creato non corrisponde

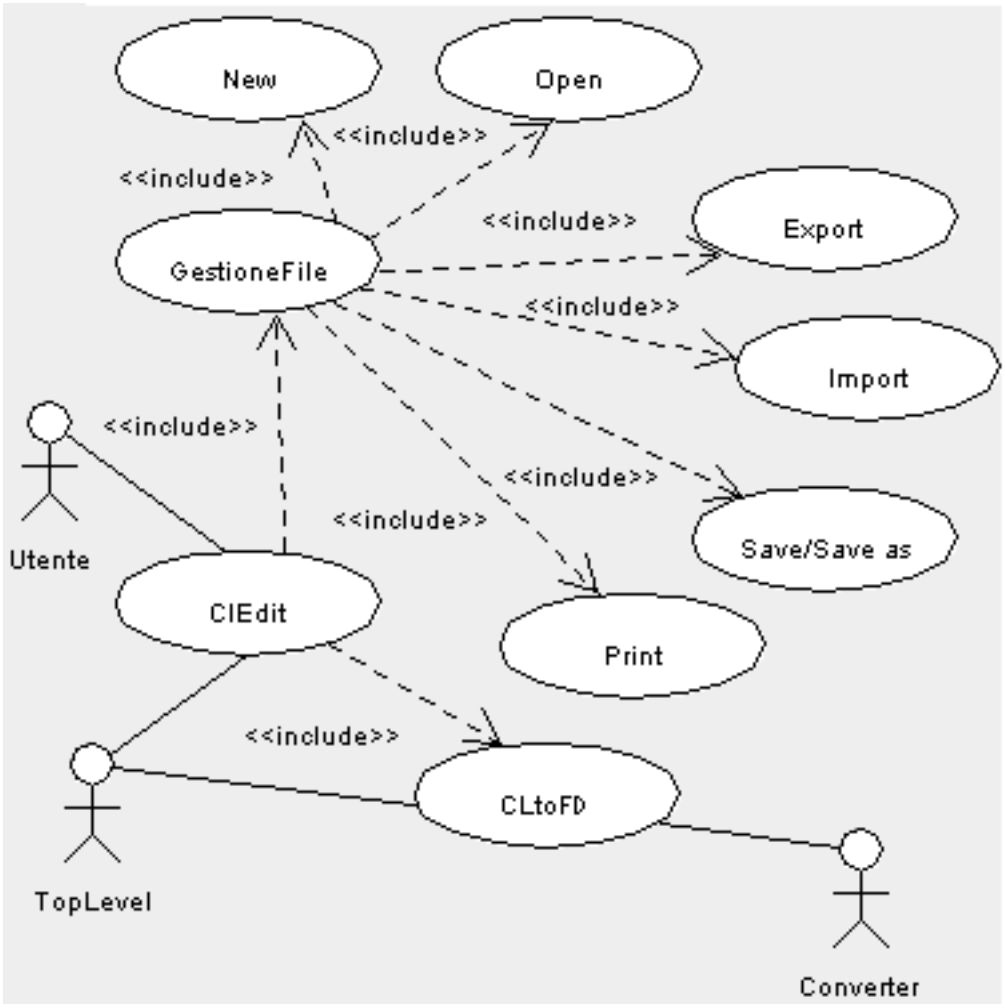

Figura 5.4: *ClEdit: gestione dei file*

un file per cui, per il salvataggio in modo permanente del diagramma, sono disponibili sia il formato testo (*Export*) sia il formato oggetto (*Save* o *Save as*). Nel caso, invece, l'utente acceda a diagrammi contenuti in file, il Tool consente una agevole conversione di formato. È sempre possibile, infatti, aprire un file con il comando *Open* e salvarlo con il comando *Export*, in modo da ottenere una conversione da formato oggetto a formato testo, oppure aprire un file con il comando *Import* e salvarlo con il comando *Save as*, in modo da ottenere una conversione da formato testo a formato oggetto.
Il salvataggio di una rappresentazione pittorica in un file in formato testo (con il comando *Export*) si traduce nella creazione di un file contenente sia i dati relativi alla rappresentazione pittorica sia i dati relativi al grafo corrispondente in un formato editabile con strumenti tradizionali e leggibile.

Un esempio è riportato qui di seguito:

```
####################################
prova1.tcl
##
Info section:draw
Number of Labels|2
Number of ConnectionElements|1
Info section:graph
Number of Nodes|2
Number of Arcs|1
^^
##
Draw section
109,64|64,52|etichetta0|0
386,216|341,204|etichetta1|1
0,1|109,64|386,216|1
^^
##
Graph section
0|etichetta0|0|1||false
1|etichetta1|0||0|false
0,1|+|Materials
^^
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Il file è suddiviso in un certo numero di sezioni. La prima contiene informazioni di tipo generale relative alla rappresentazione pittorica ($Infosection : draw$) e al grafo ($Infosection : graph$) quali: numero di etichette e di elementi di connessione per il primo e numero di nodi e di archi per il secondo.
Le due sezioni successive contengono i dati necessari per il tracciamento della rappresentazione pittorica (*Draw section*) e per la definizione del grafo corrispondente (*Graph section*).
Nella prima sezione, per ogni etichetta sono memorizzati due punti necessari per il tracciamento, il valore della stringa ad essa associata e un identificativo mentre, per ogni elemento di connessione, sono memeorizzati gli identificativi delle etichette agli estremi dell'elemento, i punti estremi dell'elemento e il tipo (1 individua una linea, 2 un arco e 3 una polilinea).
Nella seconda sezione per ogni nodo sono memorizzati, nell'ordine:

1. l'identificativo del nodo,
2. la stringa associata al nodo,
3. il tipo del nodo (0 sta per "$< unspecified >$"),

4. gli identificativi dei nodi della stella uscente o una stringa vuota se la stella non contiene nodi,

5. gli identificativi dei nodi della stella entrante o una stringa vuota se la stella non contiene nodi,

6. una indicazione se il nodo è stato tipizzato ($true$) o no ($false$).

Per ciascun arco del grafo, infine, sono memorizzati: gli identificativi dei nodi alle estremità dell'arco, il segno e il tipo dell'arco.
Il comando che gestisce la stampa dei file ($Print$) al momento consente la sola stampa di file in formato testo.
L'utente crea o modifica una rappresentazione pittorica (e il Tool tiene aggiornata in tempo reale la struttura del grafo corrispondente) mediante un certo numero di comandi (cfr. la figura 5.3). I comandi di cui l'utente dispone consentono il tracciamento ($Draw$) o la modifica ($Edit$) di etichette ($Etichette$) e di elementi di connessione ($Connessioni$). Le operazioni di modifica o editing, accessibili mediante menù flottanti, permettono:

1. la modifica della stringa associata ad una etichetta,

2. l'attribuzione di un tipo ad una etichetta,

3. la rimozione di una etichetta e di tutti gli elementi di connessione in essa incidenti,

4. la rimozione di un elemento di connessione,

5. la modifica dell'orientamento di un elemento di connessione,

6. l'attribuzione di un segno ad un elemento di connessione,

7. l'attribuzione di un tipo ad un elemento di connessione.

Oltre a tali operazioni, il Tool mette a disposizione dell'utente un certo numero di operazioni che consentono di visualizzare porzioni del diagramma CL corrente (cfr. la voce *Sottografi* in figura 5.3), di assegnare segni agli anelli presenti nel diagramma (cfr. la voce *Segno sui loop* in figura 5.3) e di visualizzare i singoli anelli (cfr. la voce *Visualizza loop* in figura 5.3).
La visualizzazione di porzioni del diagramma correntemente visualizzato sul *canvas* prevede la selezione di un sottoinsieme dei nodi mediante la specificazione di uno o più tipi (cfr. la voce *Per nodi* in figura 5.3) oppure la selezione di un sottoinsieme degli archi (cfr. la voce *Per archi* in figura 5.3): la selezione degli archi può essere fatta sulla base del tipo (che può assumere uno dei valori "$< unspecified >$", "Information" o "Materials") o del segno (che può assumere uno dei valori "n" o "non assegnato", "+ o "-). Una volta individuata la porzione

del diagramma voluta questa viene visualizzata in un frame ausiliario.
L'assegnamento dei segni agli anelli presenti in un diagramma CL presuppone che a tutti gli archi del diagramma sia stato assegnato un segno ("+ o "-) e si traduce nella visualizzazione sul diagramma di due simboli speciali ($\oplus$ per gli anelli con $feedback$ positivo e $\ominus$ per gli anelli con $feedback$ negativo) in corrispondenza del centro geometrico di ciascun anello. La selezione di uno di tali simboli con il pulsante destro del mouse si traduce nella visualizzazione in un frame ausiliario delle etichette e degli elementi di connessione che fanno parte dell'anello corrispondente.

5.4 *Flow Diagram Graphic Editor*

Il *Flow Diagram Graphic Editor* (al quale, nelle figure e nel testo di questo Capitolo, si farà riferimento con il nome di $FdEdit$) è il Tool che consente la creazione e la manipolazione di diagrammi FD. Come risulta dalla figura 5.5, il Tool è accessibile sia in modo autonomo sia atraverso il Tool $TopLEvel$. Le principali funzionalità del Tool (illustrate nelle figure 5.5, 5.6, 5.7 e 5.8) comprendono operazioni per:

1. la gestione delle equazioni associate ai singoli nodi del multigrafo, cfr. la figura 5.6,
2. la gestione delle rappresentazioni pittoriche e dei grafi, cfr. la figura 5.7,
3. la gestione delle interazioni con il Sistema Operativo ospite, cfr. la figura 5.8,
4. la conversione di formato.

Le operazioni che consentono la gestione dei grafi e delle corrispondenti rappresentazioni pittoriche, raggruppate sotto la dizione generica di $GestioneGrafi$ nelle figure 5.5 e 5.7, comprendono operazioni:

1. di tracciamento ($Draw$) di nodi e di elementi di connessione,
2. di editing ($Edit$) di nodi e di elementi di connessione,
3. di selezione di sottografi ($Sottografi$).

Nel caso dei diagrammi FD alcune delle differenze sostanziali rispetto ai diagrammi CL si traducono nel fatto che i nodi della rappresentazione pittorica sono composti da coppie icona/etichetta e che gli elementi di connessione hanno un tipo ($Information$ o $Materials$) che ne condiziona le possibilità di tracciamento al tipo dei nodi alle sue estremità. Le icone caratterizzano il tipo dei diversi nodi (ad ogni tipo corrisponde una icona diversa) mentre le etichette rappresentano i

valori delle variabili associate ai singoli nodi, una per nodo.
Date queste premesse, le operazioni di editing di un nodo consentono soltanto la rimozione del nodo (e di tutti gli elementi di connessione in esso incidenti) e la modifica del nome della variabile associata al nodo mentre le operazioni di editing di un elemento di connessione si riducono alla sola operazione di cancellazione.

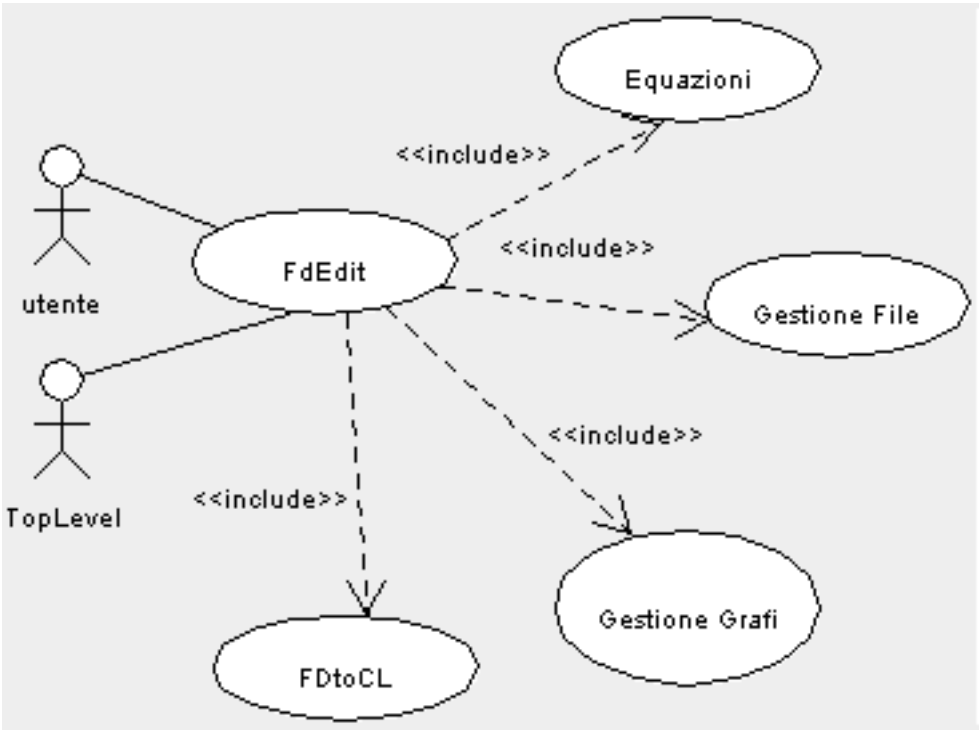

Figura 5.5: *FdEdit: alcuni dei casi d'uso*

Le operazioni di selezione dei sottografi, infine, sono analoghe a quelle esaminate nel caso dei diagrammi CL per cui, per una loro descrizione, si rimanda alla sezione 5.3. In questa sede ci si limita a far notare come la selezione di una porzione di un diagramma in base agli elementi di connessione possa sfruttare solo il tipo di questi dal momento che, in un diagramma FD, agli elementi di connessione non sono assegnati i segni. Operando una selezione con questo criterio è possibile evidenziare, rispettivamente, i flussi di informazione e i flussi di materiali presenti nel modello.
Le operazioni che consentono la gestione delle interazioni con il Sistema Operativo ospite, raggruppate sotto la dizione generica di *GestioneFile* nelle figure 5.5 e 5.8, comprendono operazioni:

1. per la creazione di un nuovo diagramma che potrà essere memorizzato in un nuovo file,

2. per l'accesso a diagrammi la cui descrizione è contenuta in un file esistente,

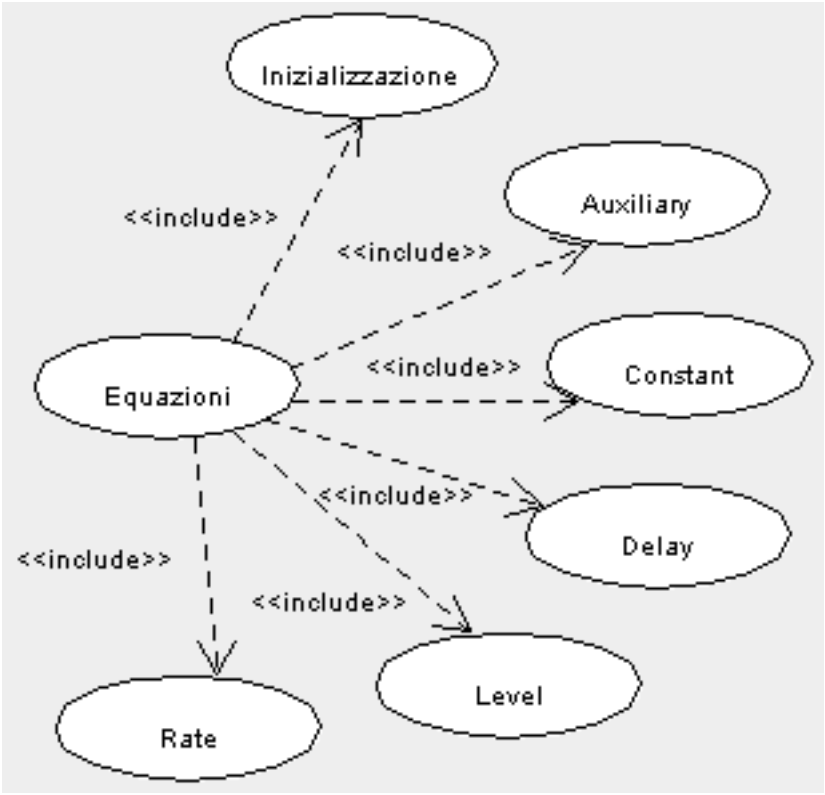

Figura 5.6: *FdEdit: la gestione delle equazioni*

3. per la stampa della descrizione di un diagramma, descrizione contenuta in un file in formato testo.

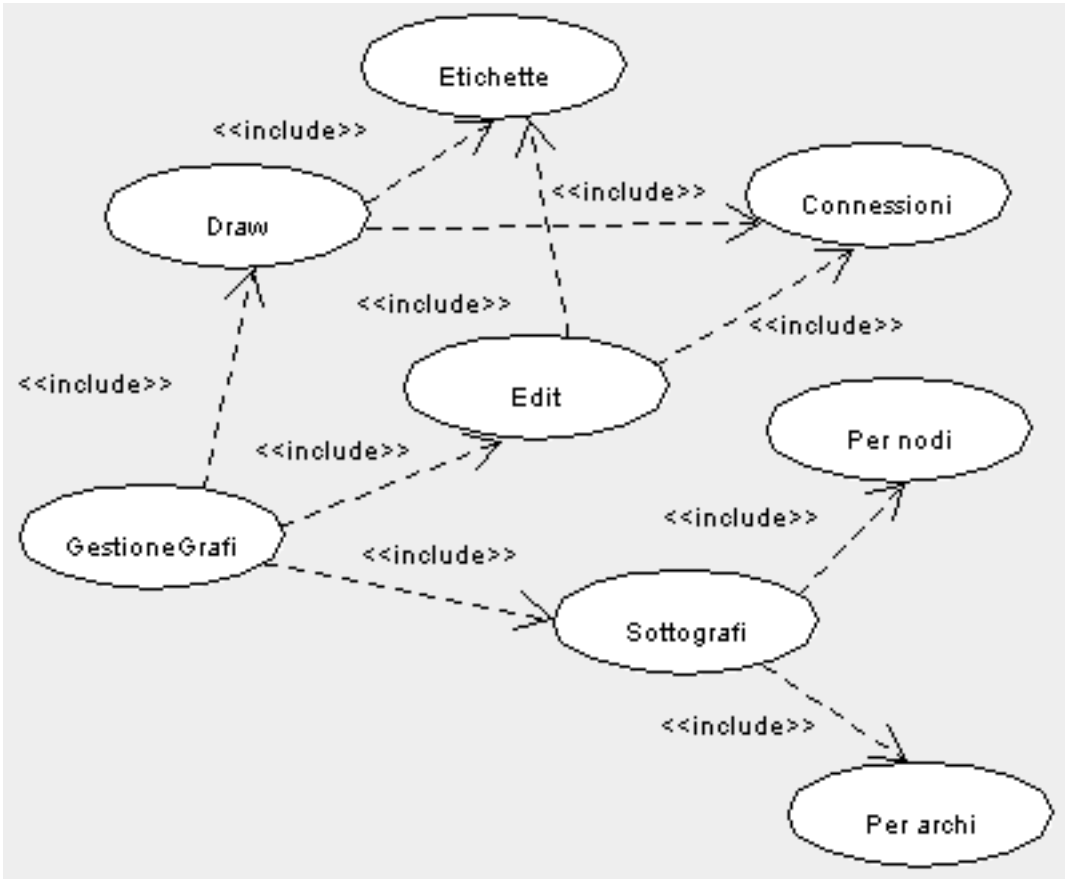

Figura 5.7: *FdEdit: gestione dei grafi*

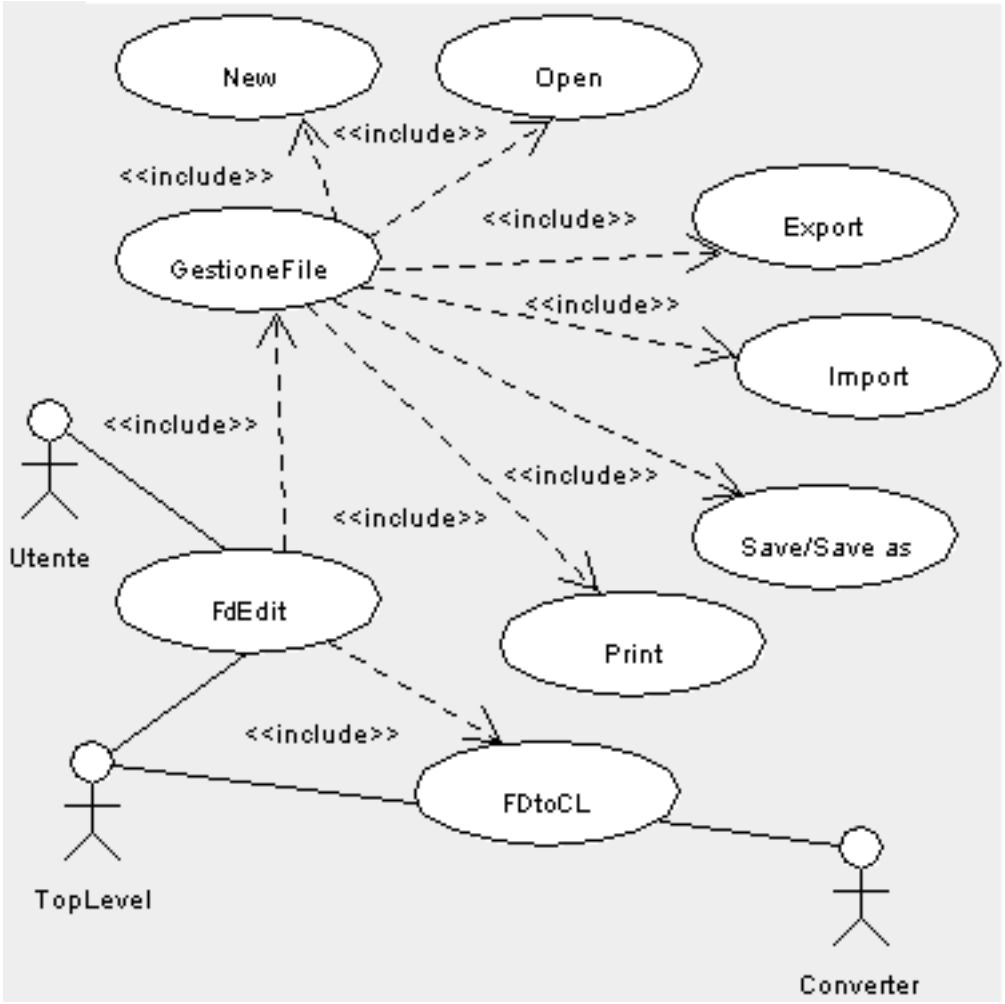

Figura 5.8: *FdEdit: gestione dei file*

Le operazioni di *gestione file* hanno una semantica analoga a quella esaminata nella sezione 3.3 alla quale si rimanda. In questa sede ci si limita a far notare come la struttura dei file in formato testo sia praticamente identica a quella vista nel caso dei diagrammi CL (nonostante una apparente diversità di rappresentazione) dal momento che il tipo di ciascun nodo, codificato nei campi opportuni delle righe della sezione *Graph section*, consente di determinare in modo univoco l'icona da utilizzare nella corrispondente rappresentazione pittorica.
Le operazioni di conversione di formato (che consentono di trasformare un diagramma FD nel corrispondente diagramma CD) non saranno descritte in ulteriore dettaglio in questa sede in quanto ancora in fase di implementazione.
Nel caso del Tool *FdEdit* l'utente ha a disposizione un certo numero di comandi per la gestione delle equazioni associate ai singoli nodi di un diagramma. Tali comandi (cfr. la figura 5.6) permettono sia l'impostazione delle equazioni mediante finestre ausiliarie (*Auxiliary*, *Constant*, *Delay*, *Level* e *Rate*) che dipendono dal tipo del nodo sia l'impostazione e/o la modifica dei valori di un certo numero di parametri globali del diagramma in corso di definizione (*Inizializzazione*).
Questa parte del Tool che comprende i metodi di controllo delle equazioni e di interfaccia con il *Resolver* è, al momento, in fase di implementazione.

5.5 Gli altri Tool

Gli altri Tool che compongono l'ambiente di simulazione $D(a)ySy\,Tool\ Box$ (ovvero il Tool *Display*, il Tool *Equation Solver* (detto anche *Resolver*, vedi oltre) e il *Converter Tool*) non sono stati, al momento, completamente implementati (cfr. la sezione 6.2). In questa sede, pertanto, ci si limita a descrivere le possibili interazioni fra questi Tool e i Tool descritti nelle sezioni 5.2, 5.3 e 5.4 e, al contempo, a chiarire alcuni aspetti della struttura di Tool quali *Equation Solver* e *Converter Tool*.
Il Tool *Display* è stato progettato in modo da interagire sia con il Tool *FdEdit* sia con il Tool *Resolver* (cfr. la figura 5.9) e non può essere eseguito in modo autonomo oppure attraverso il Tool *TopLevel* come accade per altri Tool dell'ambiente.

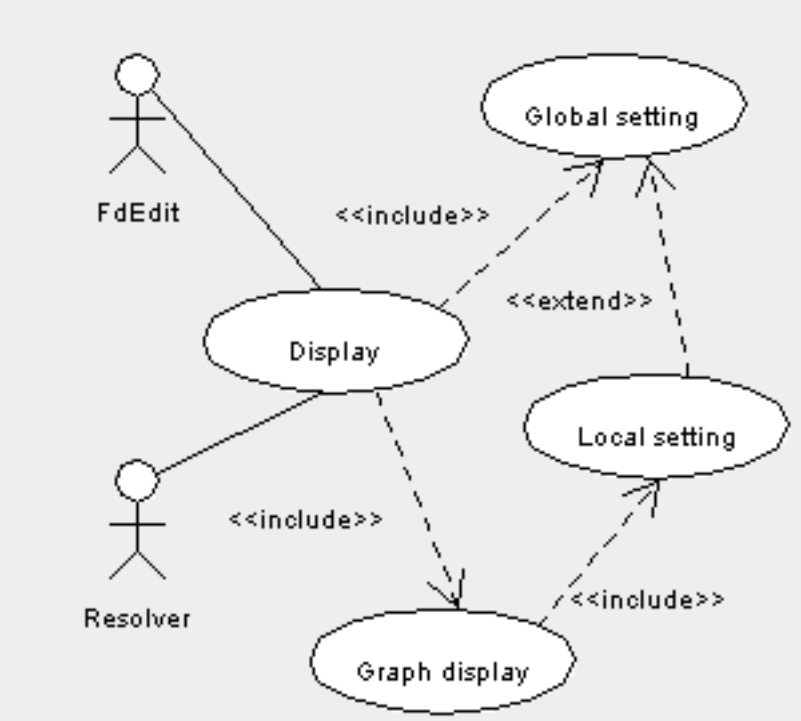

Figura 5.9: *Display: le interazioni con gli altri Tool e i principali casi d'uso*

Le operazioni accessibili mediante tale Tool sono state già illustrate nei Capitoli 3 e 4, ai quali si rimanda. In questa sede ci si limita a far notare come il Tool *Display* definisca operazioni per la visualizzazione di grafici (*Graph display*) e per la definizione di settaggi globali (*Global setting*) e locali (*Local setting*) e le renda accessibili mediante un frame principale (*Display*) ai Tool con i quali interagisce. Sia il Tool *Resolver* sia il Tool *FdEdit*, da parte loro, sono caratterizzati da metodi che determinano, a partire da un diagramma FD e dalle equazioni associate

ai suoi nodi, un insieme di valori che il Tool *Display* si occupa di visualizzare.

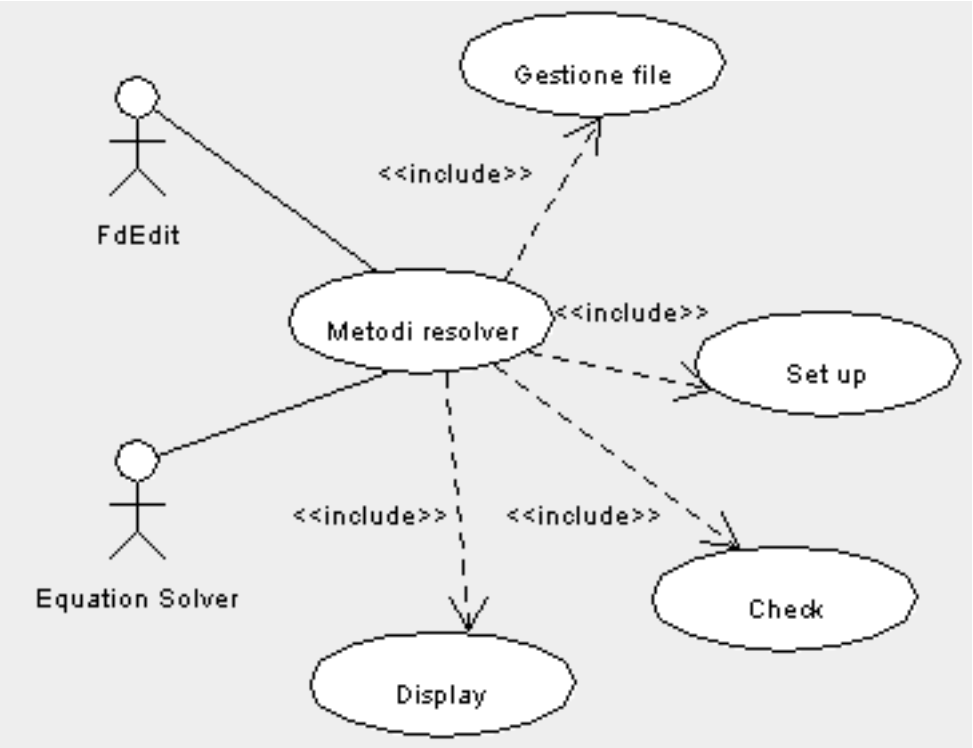

Figura 5.10: *Resolver: le interazioni con gli altri Tool e i principali casi d'uso*

Il Tool *Resolver* (cfr. la figura 5.10) è stato progettato in modo da essere accessibile sia attraverso il Tool *FdEdit* sia attraverso una interfaccia ad hoc che costituisce il Tool *Equation Solver*.
L'accesso attraverso il Tool *FdEdit* fa in modo che i metodi del *Resolver* siano applicati direttamente al diagramma FD correntemente visualizzato. L'utente, tuttavia, ha la possibilità di ridefinire i parametri che devono essere utilizzati per la simulazione, fra cui: l'istante di inizio, l'istante di fine, il valore del *time step* e l'algoritmo che deve essere utilizzato per la risoluzione delle equazioni alle differenze finite presenti nel modello.
L'accesso attraverso l'interfaccia dell'*Equation Solver* mette a disposizione dell'utente un Tool autocontenuto per la simulazione di diagrammi FD creati in precedenza, inizializzati e memorizzati in file. L'interfaccia è, infatti, caratterizzata da comandi che consentono (cfr. la figura 5.10):

1. l'accesso ai file contenenti le descrizioni dei diagrammi FD (*Gestione file*),
2. la verifica dei diagrammi contenuti nei file (*Check*),
3. il settaggio dei parametri per la simulazione (*Set up*),

4. l'esecuzione della simulazione (*Metodi resolver*),

5. l'interazione con il Tool *Display* per la visualizzazione del risultato di una simulazione (*Display*).

I Tool utilizzabili per la conversione di formato (da CL a FD o da FD a CL, cfr. le figure 5.1, 5.2, 5.4, 5.5 e 5.8) sono accessibili sia attraverso i Tool quali *TopLevel*, *ClEdit* e *FdEDit* sia mediante un Tool ad hoc detto *Converter Tool*. Il *Converter Tool* è caratterizzato da una interfaccia che permette l'accesso ai metodi per la conversione di formato in modo autonomo rispetto agli altri Tool e si presta alla esecuzione di una conversione di tipo *batch* di gruppi di file.

Capitolo 6

Conclusioni

6.1 Introduzione

L'ambiente di simulazione *D(a)ySy Tool Box* descritto nella presente Tesi è un ambiente caratterizzato da un certo numero di Tool interagenti il cui scopo è quello di consentire la rappresentazione di *modelli* di *sistemi dinamici*, la loro *simulazione* e la *visualizzazione* dei risultati delle simulazioni.
I Tool che compongono l'ambiente possono essere suddivisi nelle seguenti categorie:

1. Tool di coordinamento,
2. Tool di editing,
3. Tool di elaborazione,
4. Tool di rappresentazione.

Alla prima categoria appartiene il Tool *TopLevel*, alla seconda categoria appartengono i Tool *Causal Loop Graphic Editor* e *Flow Diagram Graphic Editor*, alla terza categoria appartengono i Tool *Equation Solver* e *Converter* mentre all'ultima appartiene il Tool *Display*.
I Tool di editing, oltre mettere a disposizione dell'utente un certo numero di operazioni per l'editing di diagrammi CL e FD, modelli di sistemi dinamici, possiedono anche alcuni strumenti per l'analisi dei diagrammi. Tali stumenti consentono l'accesso a sottodiagrammi variamente individuati (cfr. le sezioni 3.3 e 3.4).
L'ambiente, tuttavia, è stato implementato solo parzialmente. Alcuni dei Tool sono stati progettati ma la loro implementazione è ancora da terminare (cfr. la sezione 6.2) mentre anche per i Tool la cui implementazione è stata portata praticamente a termine sono state concepite aggiunte ed estensioni che saranno presentate nella sezione 6.3.

6.2 Lo stato dei singoli Tool

Come risulta anche dai Capitoli 3 e 4, la fase di progettazione dettagliata è stata portata a termine per tutti i Tool che compongono l'ambiente di simulazione *D(a)ySy Tool Box*.
Per ciascuno dei Tool che può essere eseguito sia in modo autonomo sia attraverso o il Tool *TopLevel* o un qualche altro Tool si è provveduto alla implementazione dell'interfaccia utente e della struttura interna mediante la definizione e la implementazione delle classi principali, degli eventi e dei relativi metodi *listener*.
Per alcuni dei Tool la fase di implementazione è andata oltre la implementazione dell'interfaccia e della struttura interna e si è spinta fino alla implementazione della maggior parte dei metodi in modo da rendere effettivi gran parte dei comandi caratteristici di ciascun Tool.
I Tool in questione sono i seguenti:

1. *TopLevel*,
2. *Causal Loop Graphic Editor*,
3. *Flow Diagram Graphic Editor*,
4. *Display*.

6.2.1 *TopLevel*

Il Tool *TopLevel* è stato pienamente implementato nella sua veste di applicazione *master* di coordinamento della esecuzione dei Tool di editing e di conversione. Utilizzando il Tool *TopLevel* è, pertanto, possibile controllare l'esecuzione di più istanze di tali Tool.
Rimane da terminare, invece, la implementazione delle funzionalità di *viewer* del Tool. Al momento è possibile istanziare i frame di visualizzazione dei diagrammi ma i relativi menù non sono operativi.

6.2.2 *Causal Loop Graphic Editor*

Per quanto riguarda il Tool *Causal Loop Graphic Editor*, la sua implementazione è stata praticamente portata a termine dal momento che tutti i comandi sono pienamente operativi, tranne il comando per la conversione di un diagramma CL nel corripondente diagramma FD (*Convert to FD*).
Utilizzando il Tool è, pertanto, possibile tracciare ed editare diagrammi CL che possono essere salvati su file nei due formati previsti, testo ed oggetto. È inoltre possibile selezionare porzioni dei diagrammi da visualizzare in frame separati e individuare, assegnando loro un segno, gli anelli presenti in un diagramma CL, anelli che possono essere facilmente visualizzati in frame separati.

6.2.3 *Flow Diagram Graphic Editor*

Per quanto riguarda il Tool *Flow Diagram Graphic Editor*, la sua implementazione è stata portata al punto in cui buona parte dei comandi sono pienamente operativi.
Fanno eccezione i comandi del menù *Application* (*Check to Convert*, *Check to Solve*, *Convert to CL* e *Solve&Display*) e le finestre di dialogo per l'impostazione delle equazioni dei singoli nodi la cui implementazione è, al momento, ancora in corso.
Utilizzando il Tool è, pertanto, possibile tracciare ed editare diagrammi FD che possono essere salvati su file nei due formati previsti, testo ed oggetto. È inoltre possibile selezionare porzioni dei diagrammi da visualizzare in frame separati.

6.2.4 *Display*

Il Tool *Display*, accessibile solo tramite il Tool *Flow Diagram Graphic Editor* ed utilizzabile per la visualizzazione dei risultati delle simulazioni, è stato implementato quasi completamente. Rimangono da implementare le operazioni di *zoom* sui grafici visualizzati e devono essere messe a punto le modalità di visualizzazione sia dei grafici singoli sia di più grafici su uno stesso frame. La effettiva utilizzabilità del Tool dipende dalla implementazione dei metodi per la simulazione dei diagrammi FD che devono produrre i valori che il Tool *Display* si limita a visualizzare.

6.3 Problemi aperti ed estensioni

I problemi aperti che devono essere affrontati e risolti perchè l'ambiente proposto sia completo e pienamente operativo coprono, sostanzialmente, le seguenti aree:

1. conversione di formato,
2. risoluzione di equazioni,

mentre le estensioni riguardano sia i singoli Tool sia l'ambiente *D(a)ySy Tool Box* nel suo complesso.

6.3.1 Problemi aperti

La *conversione di formato* coinvolge sia la *rappresentazione pittorica* sia la struttura astratta ad essa associata, che è un grafo nel caso di diagrammi CL e un multigrafo nel caso di diagrammi FD.
La conversione avviene in due passi:

1. conversione della struttura astratta,

2. conversione della rappresentazione pittorica.

La conversione della struttura astratta richiede sostanzialmente solo un riorientamento di alcuni degli archi della struttura di partenza in modo che questi, nella struttura di destinazione, descrivano le relazioni ad essi associate con la corretta semantica.
Nel caso di diagrammi CL, ad esempio, i grafi corrispondenti ai singoli diagrammi sono caratterizzati da nodi connessi da archi orientati in modo da definire relazioni di causa effetto. Tali relazioni di causa effetto sono composte fra di loro a formare cicli nei nodi dei quali possono incidere elementi estranei ai cicli stessi e che rappresentano variabili esogene. La trasformazione di un grafo così strutturato in un multigrafo cui corrisponde un diagramma FD richiede che ad alcuni degli archi sia invertito l'orientamento in modo da rispecchiare l'andamento dei flussi ad essi associati nel corrispondente diagramma FD.
Ad esempio nel caso di un anello semplice di due elementi A e B in cui l'arco (A, B) abbia segno $+$ e l'arco (B, A) abbia segno $-$ (ovvero un anello con feedback negativo) si hanno i passi seguenti:

1. tipizzazione degli elementi,

2. conversione della rappresentazione astratta dalla forma CL alla forma FD.

La tipizzazione, ad esempio, prevede che il nodo A sia caratterizzato come di tipo "level", il nodo B come di tipo "rate", l'arco (A, B) come di tipo "Information" e l'arco (B, A) come di tipo "Materials" in modo che la conversione della struttura astratta si traduca nella inversione dell'orientamento dell'arco (B, A) in modo che, nel diagramma FD, compaia un arco di tipo "Materials" uscente da A e incidente in B come l'arco di tipo "Information" (A, B).
In modo analogo si procede per la conversione della struttura astratta di un diagramma FD nella corrispondente struttura astratta dell'associato diagramma CL.
Nella conversione della struttura astratta di un diagramma FD nel corrispondente struttura astratta di un diagramma CL è possibile ricavare il segno sugli archi sfruttando informazioni quali:

1. il tipo dell'arco,

2. il verso dell'arco,

3. l'equazione associata al nodo da cui l'arco esce.

Dal punto di vista delle informazioni associate ai nodi ed agli archi di un diagramma FD, la conversione della struttura astratta di un diagramma FD nella struttura astratta dell'associato diagramma CL può avvenire in due modi:

1. in modo conservativo,

2. in modo minimale.

Nel primo caso le informazioni relative al *tipo* di *nodi* ed *archi* sono mantenute, sebbene non abbiano molto senso nel dominio dei diagrammi CL, e sono assegnate ai campi corrispondenti degli elementi della struttura dati che implementa un diagramma CL, strutture predisposte per la conversione di diagrammi CL in diagrammi FD. Nel secondo caso, invece, tali informazioni sono rimosse in modo da creare una struttura astratta non tipizzata, in cui le strutture dati suddette hanno valori "non specificati".
Dopo avere effettuato la conversione della rappresentazione astratta risulta necessario effettuare la *conversione della rappresentazione pittorica.*
La conversione della rappresentazione pittorica coinvolge[1]:

1. la rappresentazione dei nodi e degli archi,

2. il posizionamento dei nodi e il tracciamento degli archi.

Nel passaggio da un diagramma CL al corrispondente diagramma FD, infatti, a ciascuna *etichetta* del primo deve essere sostituita una coppia *icona/etichetta* in cui:

1. l'icona contiene una immagine che riflette il tipo del nodo,

2. l'etichetta (nome del nodo cui corrisponde una variabile del modello) ha lo stesso valore che aveva nel diagramma CL.

Nel passaggio da un diagramma FD al corrispondente diagramma CL, viceversa, a ciascuna coppia *icona/etichetta* viene fatta corrispondere la sola *etichetta* con o senza conservazione delle informazioni di tipo, in funzione del tipo di conversione scelto.
Per quanto riguarda gli archi, in linea di principio ciò che cambia è solo l'orientamento di alcuni degli archi mentre le informazioni di tipo (nel passaggio da FD a CL) possono essere conservate o meno, anche in questo caso in funzione del tipo di conversione scelto, oppure (nel passaggio da CL a FD) si traducono nell'uso di un colore per rappresentare il tipo di flusso sull'arco.
Per ottenere una rappresentazione pittorica, oltre a ricavare le necessarie informazioni relative ai nodi ed agli archi, è necessario stabilire regole per:

1. il posizionamento dei nodi,

2. il tracciamento degli archi

[1]Nel seguito useremo i termini generici *nodo* ed *arco* anche per gli elementi della rappresentazione pittorica in modo da uniformare il trattamento dei due tipi di diagrammi, CL e FD.

in modo da ottenere rappresentazioni pittoriche leggibili e con le caratteristiche tipiche delle rappresentazioni di un certo tipo di diagramma (CL o FD a seconda dei casi).
Le strategie che si pensa di utilizzare nei Tool di conversione in corso di implementazione prevedono il *posizionamento dei nodi* e il *tracciamento degli archi* in modo da produrre diagrammi di struttura accettabile semanticamente corretti su cui l'utente può apportare i desiderati aggiustamenti di tipo estetico, utilizzando il Tool grafico opportuno.
Nel caso della conversione della rappresentazione pittorica di un diagramma FD nella rappresentazione pittorica del corrispondente diagramma CL, ad esempio, la strategia che verrà implementata nei metodi del Tool $FDtoCL$ prevede che:

1. le coppie icona/etichetta siano sostituite sic et simpliciter da etichette posizionate nel punto di centro della icona corrispondente (in cui convergono le rappresentazioni degli archi incidenti nel nodo),

2. gli archi il cui orientamento non cambia vengano tracciati invariati mentre quelli il cui orientamento viene invertito siano rappresentati sotto forma di archi di circonferenza.

Nel caso, invece, della conversione della rappresentazione pittorica di un diagramma CL nella rappresentazione pittorica del corrispondente diagramma FD la strategia che verrà implementata nei metodi del Tool $CLtoFD$ prevede che:

1. vengano inizialmente posizionate le coppie icona/etichetta corrispondenti a nodi di tipo "source", "sink", "level" o "rate" con il punto di centro delle icone nel punto di centro della corrispondente etichetta,

2. vengano posizionate le coppie icona/etichetta corrispondenti a nodi di tipo "delay" su connessioni di tipo "Materials", con il punto di centro delle icone nel punto di centro della corrispondente etichetta,

3. vengano tracciati gli archi di tipo "Material" sotto forma di segmenti con il corretto orientamento,

4. vengano posizionate le coppie icona/etichetta corrispondenti a nodi di tipo "constant" o "auxiliary" in prossimità dei nodi con cui interagiscono e, comunque, in modo da mentenere le catene di influenza fra nodi di tipo "constant" e nodi di tipo "auxiliary",

5. vengano posizionate le coppie icona/etichetta corrispondenti a nodi di tipo "delay" su connessioni di tipo "Information" in genere nel punto di mezzo del segmento che unisce i nodi estremi del cammino su cui è inserito il ritardo,

6. vengano tracciati gli archi di tipo "Information" sotto forma di archi di circonferenza con il corretto orientamento.

I problemi aperti relativi alla *risoluzione delle equazioni* sono legati:

1. alla implementazione, ancora in corso, di parti del Tool *Flow Diagram Equation Editor*,

2. alla implementazione, ancora in corso, del Tool *Equation Solver*.

Le porzioni del Tool *Flow Diagram Equation Editor* in corso di implementazione sono relative alla impostazione dei parametri per le equazioni dei nodi di tipo "level", "rate" e "auxiliary" ed ai metodi per la loro verifica.
I metodi di verifica previsti eseguono controlli relativi:

1. al fatto che ad ogni nodo sia stata o meno associata una equazione,

2. al fatto che tale equazione abbia tutti i suoi parametri inizializzati e prenda in esame tutti i contributi in ingresso al nodo (se presenti),

3. alla correttezza dimensionale sia di ciascuna equazione sia di tutte le equazioni associate ai nodi di un diagramma.

Nel caso dei diagrammi FD, infatti, le variabili relative ai nodi sono caratterizzate oltre che da una equazione anche da delle unità di misura che costituiscono delle vere e proprie equazioni dimensionali di cui è necessario valutare la correttezza.
Le equazioni nel loro complesso, inoltre, devono costituire un insieme coerente nel senso che:

1. tutte le equazioni devono usare la stessa unità di misura per la variabile tempo,

2. tutte le equazioni devono usare unità di misura dimensionalmente adeguate al tipo di variabile,

3. tutte le equazioni devono usare unità di misura adeguate alle relazioni in cui è coinvolto il nodo cui una equazione è associata.

In merito al punto (2) ci si limita a far notare che, ad esempio, una variabile di tipo "level" deve essere caratterizzata da una unità di misura non dipendente dalla variabile tempo mentre una variabile di tipo "rate" deve essere caratterizzata da una unità di misura in cui compare la variabile tempo a denominatore dell'espressione dimensionale.
In merito al punto (3) ci si limita a far notare che, ad esempio, una variabile di tipo "level" che ha una unità di misura del tipo $[litri]$ deve essere in relazione con variabili di tipo "rate" con unità di misura del tipo $[litri]/[ora]$ o analoghe.

La porzione del Tool *Equation Solver* attualmente in corso di progettazione e implementazione è quella relativa al motore di risoluzione delle equazioni associate ai nodi di un diagramma FD. Il motore di risoluzione in corso di progettazione e implementazione dovrà svolgere compiti quali:

1. ordinamento delle equazioni,
2. parsing di ciascuna equazione,
3. risoluzione di ciascuna equazione sulla base dei valori delle eqauzioni ad essa associate e che la precedono nell'ordinamento,
4. determinazione dei valori delle variabili ad ogni passo della simulazione.

6.3.2 Estensioni

Le estensioni relative all'ambiente *D(a)ySy Tool Box* nel suo complesso riguardano la definizione e la implementazione di un *sistema di help in linea* accessibile dall'utente sia attraverso i singoli Tool sia autonomamente mediante un browser. Per lo sviluppo di un tale sistema di help si pensa di far ricorso ad applet Java con eventuale supporto di script JavaScript.
Altre estensioni previste riguardano la trasformazione dei Tool *Causal Loop Graphic Editor* e *Flow Diagram Graphic Editor* in Applet in modo da essere accessibili tramite browser. In merito ai singoli Tool, le estensioni maggiori riguardano i Tool *Causal Loop Graphic Editor* e *Flow Diagram Graphic Editor* per i quali si prevede di:

1. usare il linguaggio XML per le operazioni di *Export* di file in formato testo in modo da ottenere file con un formato di tipo più generale ed "application independent",
2. inserire alcune operazioni di editing classiche attualmente mancanti (quali *cut*, *copy*, *paste*),
3. modificare gli algoritmi di tracciamento degli elementi di connessione in modo da usare le curve di Bezier per il tracciamento degli archi e migliorare il tracciamento delle spezzate,
4. introdurre ulteriori strumenti per l'analisi dei diagrammi che consentano di evidenziare, data una variabile, quali sono le variabili che la influenzano o che sono influenzate da essa e quali sono i anello in cui è inserita la variabile.

Appendice A

Descrizione del software mediante il linguaggio UML

A.1 Introduzione

La presente Appendice contiene una breve presentazione del linguaggio *UML* e del programma *Poseidon* seguite da una descrizione non esaustiva del software sviluppato nel corso del lavoro di Tesi, descrizione data utilizzando il linguaggio UML ([BSL02] e [GW02], cfr. la sezione A.1.1).
La sezione è suddivisa in un certo numero di sottosezioni di descrizione del linguaggio, del programma e di ciascuno dei *tool*. Di ognuno dei *tool* viene data la struttura interna mediante uno o più diagrammi delle classi, tipici del linguaggio *UML* (cfr. la sezione A.1.1). I diagrammi che saranno utilizzati per la descrizione del software sono stati tracciati utilizzando il programma *Poseidon for UML Community Edition* ([BSS02]) sviluppato e distribuito dalla Società *Gentleware* (cfr. la sezione A.1.4).

A.1.1 Il Linguaggio *UML*

Il linguaggio *UML* (*Unified Modelling Language*) è uno strumento utilizzabile per l'analisi e la progettazione di sistemi *Object Oriented*. Nella presente sezione non si ha la pretesa di dare una descrizione del linguaggio, per la quale si rimanda a [BSL02] e [GW02], ma ci si limiterà ad introdurre gli elementi sintattici utilizzati nei diagrammi in modo da renderli leggibili indipendentemente dalla conoscenza del linguaggio.
Il linguaggio *UML* può essere utilizzato per la definizione di:

1. diagrammi dei *casi d'uso*,
2. diagrammi delle *classi*,
3. diagrammi di *collaborazione*,

4. diagrammi di *sequenza*,
5. diagrammi delle *attività*,
6. diagrammi di *stato*.

I diagrammi dei *casi d'uso* ([BSL02], cfr. la sezione A.1.2) possono essere usati nella fase di progettazione di un sisema nuovo o di analisi di un sistema esistente e contengono sia i *casi d'uso stessi* sia gli *attori* sia le associazioni che li legano. Un *caso d'uso* rappresenta sequenze di azioni svolte dal sistema descritto mentre gli *attori* sono gli *utenti del sistema* oppure *altri sistemi* con cui il sistema in esame interagisce.
I diagrammi delle *classi* ([BSL02]) permettono di evidenziare quali sono le classi che compongono un sistema od una sua parte e quali sono le relazioni fra le classi. Dato che saranno usati nel seguito, su di essi torneremo nella sezione A.1.3.
I diagrammi di *collaborazione* ([BSL02]) sono strumenti utilizzabili per realizzare i *casi d'uso*. Tali diagrammi permettono di evidenziare il lavoro di cooperazione svolto dagli oggetti, istanze di determinate classi, per lo svolgimento di un compito, corrispondente ad una funzionalità di alto livello, non eseguibile da nessuno degli oggetti singolarmente ma disponibile agli *attori*.
I diagrammi di *collaborazione* ([BSL02]) sono utilizzati principalmente per modellizare collaborazioni fra oggetti che contribuiscono alla funzionalità di un caso d'uso o di una operazione ma possono essere uati anche per descrivere scenari alternativi all'interno di un caso d'uso.
I diagrammi di *sequenza* ([BSL02]), come i precedenti di *collaborazione*, definiscono le interazioni fra gli oggetti, interazioni realizzate mediante *scambi di messaggi* fra oggetti ovvero mediante *invocazioni di metodi*.
Mentre i diagrammi delle classi permettono di modellizzare la *struttura statica* di un sistema i diagrammi di interazione ne modellizzano la *natura dinamica* dal momento che mostrano come gli oggetti interagiscono fra di loro evidenziando, nel caso dei diagrammi di *sequenza*, la sequanza temporale dei messaggi scambiati fra i vari oggetti in modo da realizzare una certa funzionalità di alto livello.
I diagrammi delle *attività* ([BSL02]) permettono di descrivere i flussi delle attività sia in fase di *progettazione* sia in fase di *analisi* e sono di complemento ai diagrammi di collaborazione e di sequenza. In tali diagrammi vengono rappresentate le *attività*, gli *stati* e le *transizioni* fra attività e stati.
I diagrammi di *stato*, infine, ([BSL02]) sono usati per descrivere il comportamento degli elementi dinamici di un modello mediante le modifiche del loro *stato*. Lo *stato*, nel caso di *oggetti* istanze di una *classe*, è costituito dai valori assunti dai campi dati dell'oggetto. I diagrammi di questo tipo descrivono *transazioni* fra *stati* a seguito del verificarsi di *eventi*. Per ulteriori dettagli si rimanda a [BSL02].

A.1.2 *UML e i diagrammi dei casi d'uso*

I diagrammi dei casi d'uso vengono di solito usati per spiegare il comportamento di una entità (sistema o sottosistema) senza specificare come tale comportamento sia effettivamente realizzato.
I diagrammi dei casi d'uso mostrano:

1. i casi d'uso,
2. gli attori,
3. le relazioni fra casi d'uso,
4. le relazioni fra casi d'uso ed attori,
5. le relazioni fra attori.

I casi d'uso sono sequenze di azioni svolte dal sistema mentre gli attori sono o utenti o altri sistemi con cui il sistema dato interagisce. Scopo delle sequenze di azioni e' quello di ottenere un risultato osservabile da un attore e di descrivere possibili scenari di utilizzo di un sistema. Uno scenario e' l'insieme delle possibili evoluzioni di tutte le istanze di un caso d'uso.
I diagrammi dei casi d'uso sono, di solito, uati per modellizzare cio' che avviene in un sistema esistente oppure in uno in fase di sviluppo/implementazione. Li si usa anche per avere una visione ad alto livello del comportamento di un sistema e rappresentano il punto di partenza per la definizione dell'interfaccia uomo-macchina. Li si puo' inoltre usare per definire scenari alternativi e per specificare casi d'uso che generano sequenze distinte di azioni.
I *casi d'uso* sono descritti utlizzando i diagrammi dei *casi d'uso* ([BSL02]) che contengono i casi d'uso stessi, eventualmente raggruppati in *sottosistemi*, gli attori e le associazioni fra gli attori e i casi d'uso.

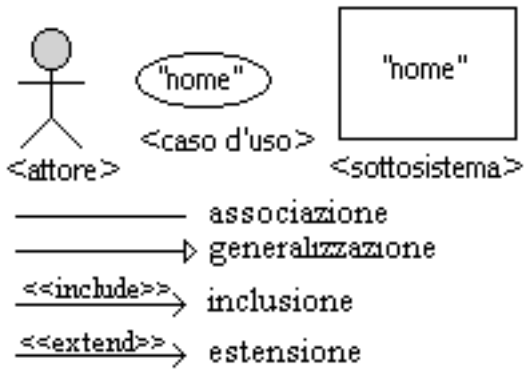

Figura A.1: *La notazione UML per i diagrammi dei casi d'uso*

La figura A.1 illustra i principali elementi sintattici del linguaggio *UML* utilizzabili per il tracciamento dei *diagrammi dei casi d'uso*. In tale figura sono presentati i simboli usati per definire gli attori, i casi d'uso e i sottosistemi (tipicamente raggruppamenti di casi d'uso che svolgono un compito finalizzato ad uno scopo). La figura presenta, inoltre, i simboli utilizzati per caratterizzare le relazioni fra attori, fra casi d'uso e fra attori e casi d'uso. Le relazioni possono essere di *associazione*, di *generalizzazione*, di *estensione* e di *inclusione*. Le relazioni di *associazione* stabiliscono un legame fra un attore e un caso d'uso, le relazioni di *generalizzazione* stabiliscono un legame fra attori o fra casi d'uso mentre le relazioni di *inclusione* e di *estensione* stabiliscono legami fra casi d'uso.

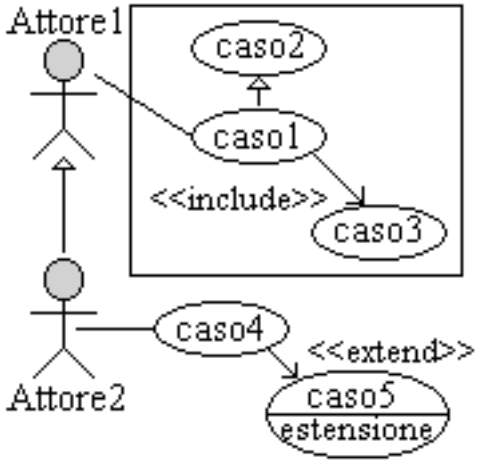

Figura A.2: *Alcuni esempi di utilizzo*

La figura A.2 presenta alcuni esempi di utilizzo della notazione. La figura (che non intende presentare un diagramma realistico) raffigura un diagramma dei casi d'uso caratterizzato da due attori e da cinque casi d'uso, tre dei quali sono raggruppati in modo da caratterizzare un sottosistema.
Il diagramma presenta una relazione di *generalizzazione tra attori*: "Attore1" rappresenta, nel contesto della figura A.2, un attore generico mentre "Attore2" è un attore specializzato che può svolgere tutti i compiti del primo più altri altri compiti suoi propri. La relazione fra i due attori è una relazione di tipo "is-a" (nel senso che "Attore2" "is-a" "Attore1" ovvero la classe di "Attore2" è una estensione della classe di "Attore1").
In modo analogo vengono definite le relazioni di *generalizzazione fra casi d'uso*. Ad esempio (cfr. la figura A.2) "caso2" rappresenta il caso generico mentre "caso1" rapresenta una sua specializzazione. Il caso generico e il caso specializzato sono legati, come nel caso precedente, da una relazione di tipo "is-a", nel senso che "caso1" "is-a" "caso2" ovvero la classe di "caso1" è una estensione o una

implementazione della classe di "caso2". Nel caso della generalizzazione fra casi d'uso, infatti, il caso d'uso generico può rappresentare una classe astratta oppure una interfaccia, ovvero un elemento che si limita a definire un modello per gli oggetti specializzati ma che non viene sviluppato per il sistema reale.
Oltre alle relazioni di generalizzazione fra i casi d'uso si possono definire le relazioni di *inclusione* (cfr. in figura A.2 la relazione fra "caso1" e "caso3") e di *estensione* (cfr. in figura A.2 la relazione fra "caso4" e "caso5").
Una relazione di inclusione (individuata con un arco orientato dal caso d'uso includente al caso d'uso incluso etichettata come << *include* >>) individua una relazione fra casi d'uso in cui un caso d'uso richiede ed utilizza le funzionalità di un altro caso d'uso, funzionalità che possono, in generale, essere accedute anche autonomamente. Nell'esempio di figura A.2 si ha che "caso4" fa uso della sequenza di attività di "caso5" durante lo svolgimento della propria sequenza di attività. Una volta che la sequanza di attività di "caso5" è terminata la sequenza di attività di "caso4" può proseguire dal punto in cui si era interrotta.
Una relazione di estensione (individuata con un arco orientato dal caso d'uso che estende al caso d'uso esteso etichettata come << *extend* >>) individua una relazione fra casi d'uso in cui un caso d'uso estende le funzionalità di un altro caso d'uso, che può essere detto di base. Ad esempio, in figura A.2 il "caso4" rappresenta una estensione del "caso5". La notazione prevede che le estensioni siano segnalate all'interno dell'ellisse che individua il caso d'uso di base sotto forma di *punti di estensione* (cfr. la sezione etichettata come *estensione* nel "caso5" della figura A.2) che elencano le funzionalità aggiunte ai casi estesi.

A.1.3 *UML e i diagrammi delle classi*

I *diagrammi delle classi* possono essere utilizzati in fase di analisi, in fase di progettazione e in fase di documentazione di un sistema software e consentono, mediante l'uso di una simbologia di tipo grafico:

1. di rappresentare le *classi* del sistema con i relativi *attributi* e *metodi*,
2. di illustrare le relazioni fra le classi.

Le relazioni fra le classi che sono rappresentate nei *diagrammi delle classi* sono associazioni, aggregazioni e gerarchie di specializzazione/generalizzazione. Un diagramma delle classi consente di rappresentare le classi astratte e le interfacce da cui derivano le classi concrete (ovvero caratterizzate da metodi completamente specificati) utilizzate all'interno del sistema software.
La notazione che l'*UML* definisce nel caso dei *diagrammi delle classi* (e che viene utilizzata nelle sottosezioni della sezione A.2) è molto ricca e verrà presentata in questa sezione solo in parte facendo ricorso ad un certo numero di figure esplicative.
La figura A.3, ad esempio, illustra due classi (ClasseA e ClasseB) in una relazione

di *associazione*. Le classi sono rappresentate con un rettangolo suddiviso in tre sezioni che contengono, rispettivamente, il nome, gli attributi e i metodi della classe.

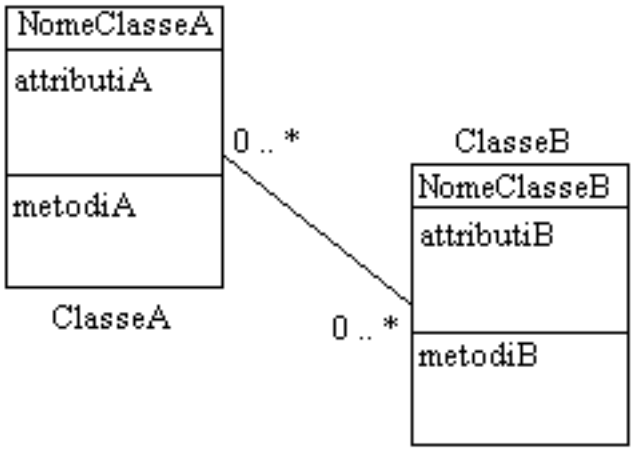

Figura A.3: *I diagrammi delle classi (1): la notazione*

Nella figura A.3, ad esempio, la *ClasseA* è caratterizzata da un nome *NomeClasseA*, da un certo numero di attributi (*attributiA*) e da un certo numero di metodi (*metodiA*). Il nome della classe permette di individuarla ed è utilizzato per la creazione di oggetti istanze della classe. Gli *attributi* rappresentano le strutture dati interne ad una classe con i relativi tipi e i modificatori d'accesso. I metodi, infine, rappresentano le operazioni che la classe è in grado di svolgere o di offrire ad altre classi attraverso gli oggetti istanze della classe.
Oltre alle sezioni rispettivamente contenenti il nome della classe, gli attributi e i metodi più una classe può essere descritta mediante altre sezioni opzionali che individuano gli eventi cui la classe può essere soggetta o che ne individuano i compiti della classe.
Le classi sono rappresentate mediante rettangoli suddivisi in sezioni rispettivamente contenenti il nome della classe, gli attributi e i metodi più altre sezioni opzionali che individuano gli eventi cui la classe può essere soggetta o che ne individuano i compiti della classe.
Le sezioni contenenti gli attributi e i metodi possono non essere rappresentate in un diagramma delle classi che, pertanto, può contenere classi descritte dalla sola sezione del nome o dalle sezioni nome e attributi o dalle sezioni nome e metodi.
Insieme al nome della classe possono essere presenti:

1. lo *stereotipo* (ovvero il modello di comportamento cui la classe aderisce) o il tipo generico cui la classe appartiene,

2. informazioni descrittive della classe,

3. informazioni sulle caratteristiche della classe: se la classe è *abstract*, se può essere generalizzata o meno, se eredita o meno da un'altra classe.

Le sezioni contenenti gli *attributi* e i *metodi* contengono i rispettivi elementi elencati uno per riga.
I singoli *attributi* sono descritti mediante:

1. un modificatore di accesso,
2. il nome,
3. il tipo.

Gli attributi, inoltre, possono essere caratterizzati da altre proprietà quali i *valori iniziali*, gli *attributi derivati* e la *molteplicità*. Per ciascun attributo è possibile, infatti, indicare il valore iniziale, se il suo valore deriva da quello di altri attributi o meno e il numero di valori distinti che un attributo può contenere (*clausola di molteplicità*). I *metodi* sono caratterizzati da:

1. un modificatore di accesso,
2. il nome e la "signature",
3. il tipo restituito.

La "signature" di un metodo è caratterizzata da un lista di coppie composte dal nome di un parametro e dal relativo tipo in cui i nomi dei parametri possono essere omessi. Oltre al nome e al tipo di ciascun parametro, la "signature" di un metodo contiene, per ciascun parametro, la così detta *clausola di genere* che può valere *in*, *out* e *inout*. Se la clausola di genere vale *in* il parametro è passato per valore, se vale *out* o *inout* il parametro viene passato per riferimento per cui rappresenta un effetto collaterale del metodo. Gli specificatori *out* e *inout* sono sottintesi nei diagrammi della sezione A.2.
I metodi possono essere raggruppati, all'interno della sezione relativa, in base al tipo di operazione svolta come appartenenti ad uno *stereotipo*. Gli stereotipi usati di solito per classificare i metodi sono i seguenti: *costructor* se il metodo è un costruttore della classe, *query* se il metodo si limita ad interrogare il valore di un attributo o *update* se il metodo modifica il valore di un attributo. Nei Capitoli 3 e 4 tali metodi sono stati caratterizati come "accessori".
I modificatori d'accesso sia per gli attributi sia per i metodi sono i seguenti:

1. $+$ per *public*,
2. $-$ per *private*,
3. $\sim$ per *package*,

4. ♯ per *protected*.

Una volta definiti i simboli con cui sono descritte le classi in un diagramma delle classi è necessario introdurre la simbologia per descrivere le *relazioni fra le classi*. Quando una istanza di una classe passa messaggi (invocazioni di metodi) a una istanza di un'altra classe si ha una *associazione* fra le due classi. Una relazione di associazione viene di solito definita una relazione del tipo "has-a" nel senso che una istanza di una classe contiene istanze di altre classi. In UML una associazione fra due classi è rappresentata con una linea continua (cfr. la figura A.3).
Una associazione fra una classe A e una classe B può essere caratterizzata da:

1. un nome,
2. una molteplicità,
3. un ruolo.

Il *nome* è una etichetta che indica la natura di una associazione fra le classi. Insieme alla etichetta può comparire anche la punta di una freccia che indica il verso in cui va interpretato il nome dell'associazione.
La *molteplicità* (cfr. la figura A.3) indica il numero di istanze della classe A per una sola istanza della classe B e viceversa. La molteplicità la si indica con $[m, n]$ in cui l'intero m indica il valore minimo della molteplicità e l'intero n indica il valore massimo. Al posto di un intero n si può avere il simbolo $*$ che stà per *qualunque valore intero maggiore di m*.
La molteplicità dell'associazione fra le classi compare sia vicino alla classe A, a indicare il numero di istanze della classe A per una istanza della classe B, e vicino alla classe B, a indicare il numero di istanze della classe B per una istanza della classe A.
Il ruolo, infine, chiarisce il ruolo giocato da una classe in un'associazione dato che più istanze di una stessa classe possono giocare ruoli diversi in associazione con classi diverse.
Una associazione fra due classi A e B esprime il fatto che gli oggetti di una classe A sono composti da oggetti di un'altra classe B.
Nei diagrammi della sezione A.2 sono utilizzate sia *associazioni bidirezionali* fra classi sia *associazioni unidirezionali* fra classi (caratterizzate da archi orientati dotati di freccia) sia *autoassociazioni*, ovvero associazioni fra una classe e sè stessa
In UML sono disponibili due tipi particolari di associazione:

1. l'*aggregazione*,
2. la *composizione*.

Una *aggregazione* (cfr. la figura A.4) indica che una istanza di una classe può comprendere istanze di altre classi.
La presenza di una aggregazione non altera la molteplicità del legame fra due

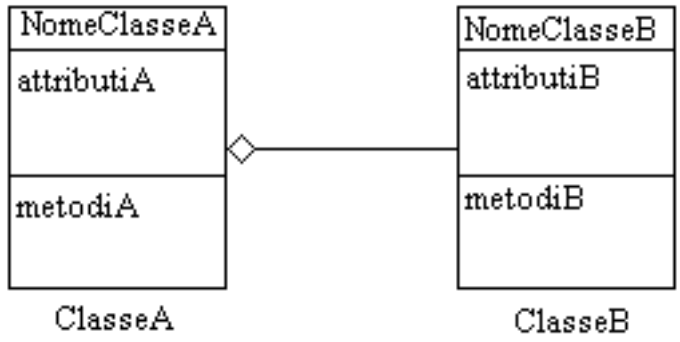

Figura A.4: *I diagrammi delle classi (2): aggregazione*

classi. Una aggregazione viene rappresentata con un segmento che inizia dal lato della classe componente ("ClasseB") e termina con un rombo bianco disegnato dal lato della classe composta o "globale" ("ClasseA"). La relazione può essere letta come segue: le istanze della "ClasseB" appartengono a istanze della classe "ClasseA" mentre istanze della "ClasseA" contengono istanze della "ClasseB".
Nel caso di una aggregazione le parti possono esistere senza il tutto mentre nel caso della *composizione* le istanze delle classi in relazione fra di loro hanno lo stesso ciclo di vita ovvero, in altre parole, le parti non possono esistere senza il tutto. In *UML* una relazione di *composizione* viene rappresentata con una notazione analoga a quella usata per l'*aggregazione* solo che il rombo è in colore nero.

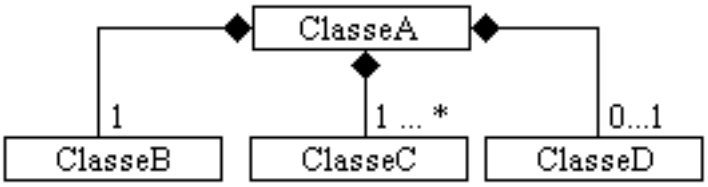

Figura A.5: *I diagrammi delle classi (3): composizione*

Con riferimento alla figura A.5 si ha che istanze della "ClasseA" sono composta da una istanza della "ClasseB", da un numero qualunque (non inferiore a 1), di istanze della "ClasseC" e da 0 o 1 istanze della "ClasseD".
Le classi possono essere legate fra di loro anche da una *relazione di generalizzazione* (cfr. la figura A.6) che esprime una relazione di ereditarietà fra *interfacce*

o fra *classi*.

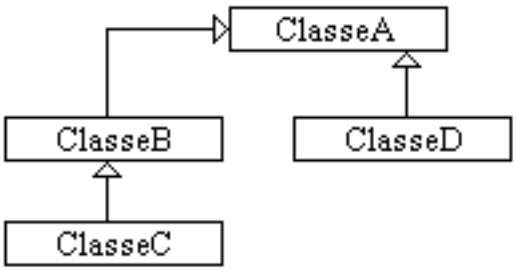

Figura A.6: *I diagrammi delle classi (4): ereditarietà*

Una relazione di generalizzazione può essere descritta come una relazione del tipo "is-a" e viene rappresentata con un segmento che termina con una freccia, orientato dalla classe specializzata (detta anche *sottoclasse*) alla classe generale (detta anche *superclasse*). In figura A.6 la "ClasseA" rappresenta la *superclasse* le cui sottoclassi dirette sono la "ClasseB" e la "ClasseD" per cui si può affermare che:

1. "ClasseB" *is-a* "ClasseA" oppure "ClasseB" *extends* "ClasseA",
2. "ClasseD" *is-a* "ClasseA" oppure "ClasseD" *extends* "ClasseA".

In modo analogo "ClasseB" è la superclasse di "ClasseC" (ovvero "ClasseC" *is-a* "ClasseB").
La *generalizzazione fra classi* consente l'*ereditarietà* degli attributi e dei metodi di una superclasse alla sottoclassi. Nell'esempio di figura A.6, le classi "ClasseB" e "ClasseD" ereditano metodi ed attributi dalla superclasse "ClasseA" ma possono ridefinirli oltre a definire metodi ed attributi loro propri.
La relazione di *generalizzazione* può coinvolgere classi che appartengono a *package* distinti. In questo caso il nome del package compare nella specificazione del nome della superclasse con gli elementi che lo compongono separate da coppie di :.
Una relazione di *generalizzazione* consente:

1. di estendere le proprietà di una classe,
2. di ridefinire le operazioni di una classe,
3. di definire classi di tipo *abstract* con metodi fittizi, detti *placeholder*, che devono essere implementati dalle classi derivate in modo che queste possano essere istanziate.

La notazione fornita da *UML* per le generalizzazioni è molto più ricca di quella presentata in questa Appendice e consente la introduzione di *commenti* mediante i quali si possono introdurre sia i *vincoli* sia i *discriminanti*. Per ulteriori dettagli si rinvia a [BSL02].
Oltre alle relazioni illustrate sinora si possono avere:

1. relazioni di dipendenza fra packages, che esprimono il fatto che le classi di un package usano le classi di un altro package dal quale il primo dipende,

2. relazioni di realizzazione fra interface e classi, che indicano quail classi implementano una data interfaccia,

per le quali si rimanda a [BSL02].

A.1.4 Il programma *Poseidon*

Il programma *Poseidon* ([BSS02]), nella versione *Community Edition* usata per la presente Tesi, rappresenta uno strumento utilizzabile per la stesura di modelli con il linguaggio *UML*, contiene tutti i diagrammi tipici del linguaggio *UML* descritti nella sezione A.1.1 e consente la generazione automatica di codice $Java^{TM}$.
Il programma è scritto interamente in $Java^{TM}$ e può essere utilizzato sia in ambienti $Windows^{TM}$ sia in ambienti $Linux^{TM}$ ovvero in tutti gli ambienti in cui sia disponibile un *Java Runtime Environment 1.3* o superiore o un *Java Development Kit 1.3* o superiore.
La versione Community Edition rappresenta la versione base pienamente funzionale del programma e può essere usata per *tracciare i diagrammi UML* e per operazioni di "reverse engineering" di codice Java scritto utilizzando altri ambienti di sviluppo.
Nell'ambito della presente Tesi il programma è stato utilizzato per:

1. tracciare i diagrammi dei casi d'uso,

2. ricavare i diagrammi delle classi e altre informazioni dal codice Java.

I diagrammi dei casi d'uso, utilizzati nel Capitolo 5, sono stati tracciati usando la simbologia standard UML allo scopo di illustrare quali siano le operazioni principali messe a disposizione degli utenti dai singoli Tool senza specificare

1. come tali operazioni siano implementate,

2. quali siano i passi che l'utente deve compiere per portare a buon fine ciascuna operazione.

I diagrammi delle classi (e le altre informazioni ad essi associate), che sono contenuti nella presente Appendice, invece sono stati ricavati sfruttando le funzionalità di "reverse engineering" del programma, funzionalità accessibili mediante il comando "Import Files" del menù "File".
Tale comando permette di analizzare un package con tutti i relativi sub-package e package importati e produce un modello che contiene:

1. i package,
2. le classi,
3. la loro interfaccia,
4. le associazioni,
5. un diagramma della classi per ciascun package.

I diagrammi delle classi così prodotti per i singoli package contengono:

1. le classi contenute nel package, ognuna caratterizzata dai metodi e dai campi dati (attributi),
2. i listener delle classi, se definiti come istanze di classi non anonime,
3. le associazioni fra le classi,
4. le relazioni di ereditarietà fra le classi

Gli attributi e i metodi sono caratterizzati dal loro modificatore di accesso ("+" per *public*, "-" per *private* "∼" per *package* e "♯" per *protected*). Gli attributi sono, inoltre, caratterizzati dal tipo ed i metodi sia dal tipo che restituiscono sia dalla relativa "signature".
Le relazioni fra le classi sono rappresentate mediante link di vario tipo che permettono di caratterizzare relazioni di *associazione*, *dipendenza* e *generalizzazione*.
Il modello ottenuto dal processo di "reverse engineering" permette di analizzare la struttura del package da più punti di vista che, nel gergo di *Poseidon*, sono detti:

1. Class Centric,
2. Diagram Centric,
3. Inheritance Centric,
4. Package Centric.

La differenza principale fra tali punti di vista risiede essenzialmente nel modo in cui il package viene presentato all'utente sotto forma di un modello e nel modo in cui l'utente può navigare nella struttura del modello passando attraverso le classi e i relativi metodi ed attributi. A tali punti di vista corrisponde, infatti, lo stesso diagramma delle classi di cui sono evidenziati di volta in volta gli aspetti significativi per un dato punto di vista.
Nel caso *Class Centric*, ad esempio, sono elencate le classi contenute nel/importate dal package e, per ciascuna classe, sono elencati:

1. gli attributi con i relativi metodi accessori,
2. i listener degli eventi associati alla classe,
3. i metodi,
4. le associazioni,
5. le dipendenze.
6. le generalizzazioni.

Nel caso *Inheritance Centric* vengono evidenziate le relazioni di ereditarietà fra le classi per cui sono elencate le classi non derivate e le super-classi contenute nel package e, per ciascuna super-classe, sono elencate le classi derivate. Oltre alle relazioni di ereditarietà sono elencate le sia $interfacce$ sia le classi *abstract* e le classi che le implementano
Ad esempio un package può contenere come super-classi le classi standard di *Java* $JFrame$, $JPanel$ e $JPopupMenu$ sotto ciascuna delle quali (nella gerarchia delle ereditarietà) compaiono le classi derivate (ad esempio $ClMainFrame$, $ArrowPanel$ e $PopUpMenu$).
Nei diagrammi delle sezioni che seguono sarà fatto uso dei diversi punti di vista in tutti i casi in cui questi semplificano la descrizione di un package, come avviene nel caso dei package più complessi quali quelli relativi ai Tool *Causal Loop Graphic Editor* e *Folw Diagram Graphic Editor*.

A.2 I diagrammi

Nelle pagine che seguono sono presentate, per ciascun package, le classi con le relazioni reciproche e la struttura interna. Di alcune classi, ad esempio quelle ancora in corso di sviluppo, la struttura interna viene omessa, lo stesso dicasi per le classi la cui struttura interna è assimilabile a quella di una classe analoga di un altro package, differendo le due solo per la implementazione dei relativi metodi, oppure per le classi che non giocano un ruolo chiave all'interno di un package.

A.2.1 *TopLevel*

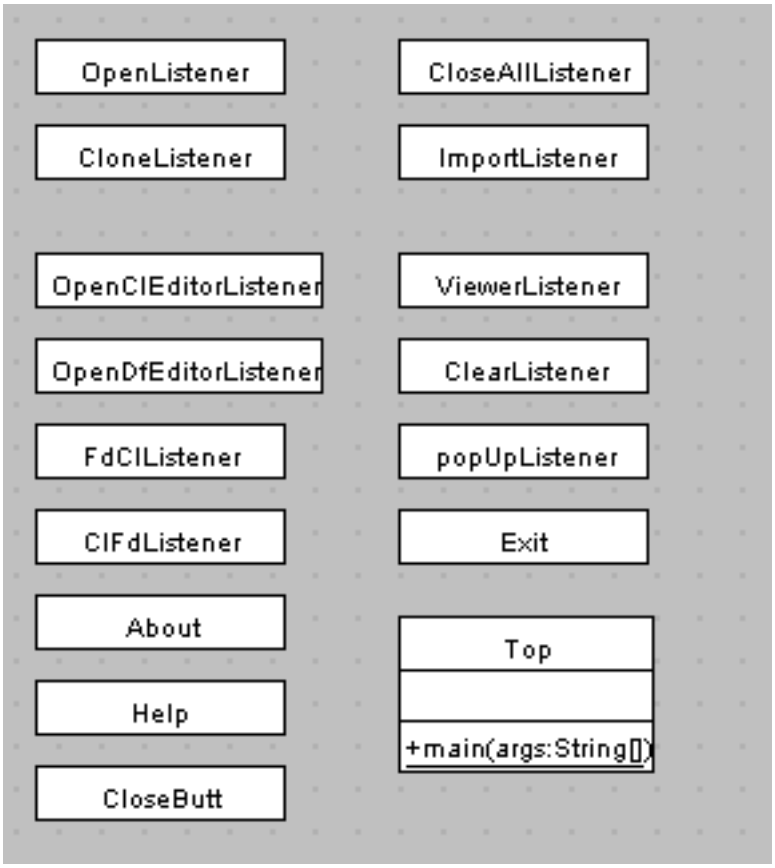

Figura A.7: *TopLevel: i listener, il main e altri metodi*

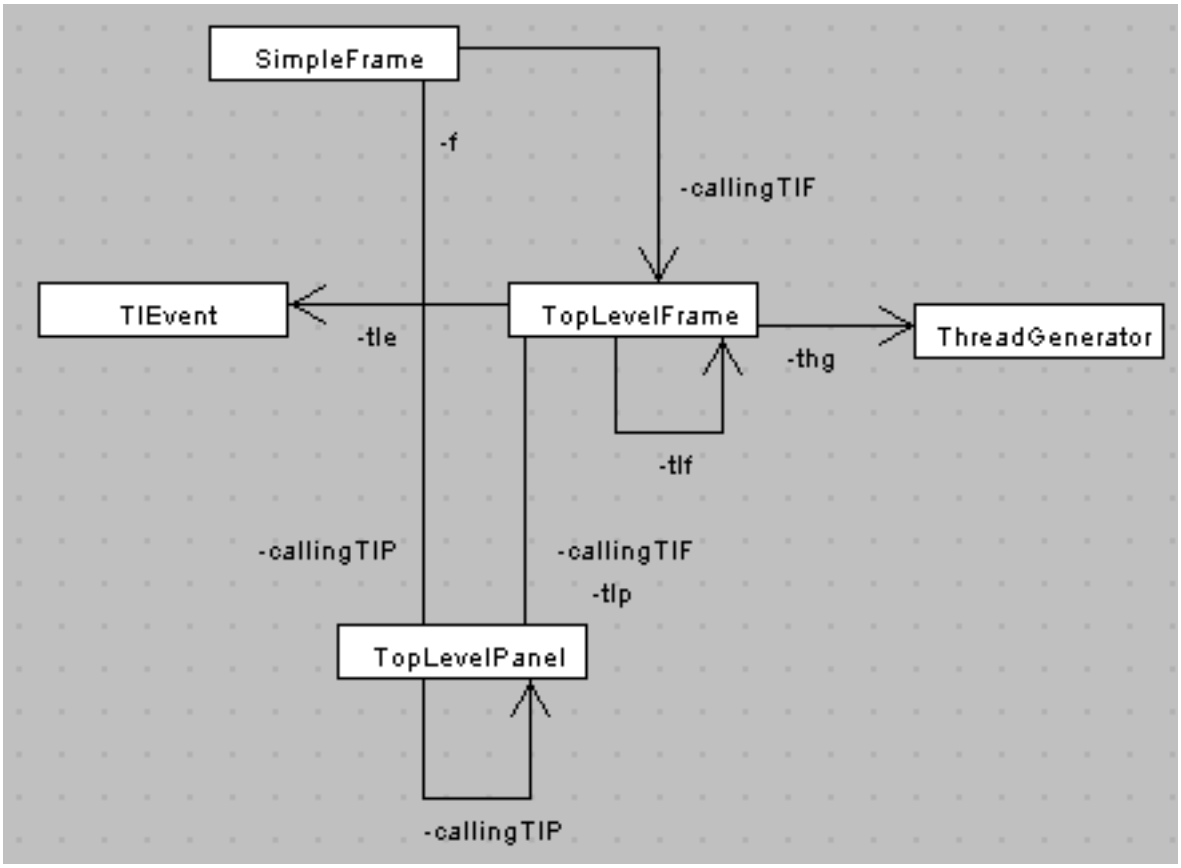

Figura A.8: *TopLevel: le classi principali, nessun dettaglio*

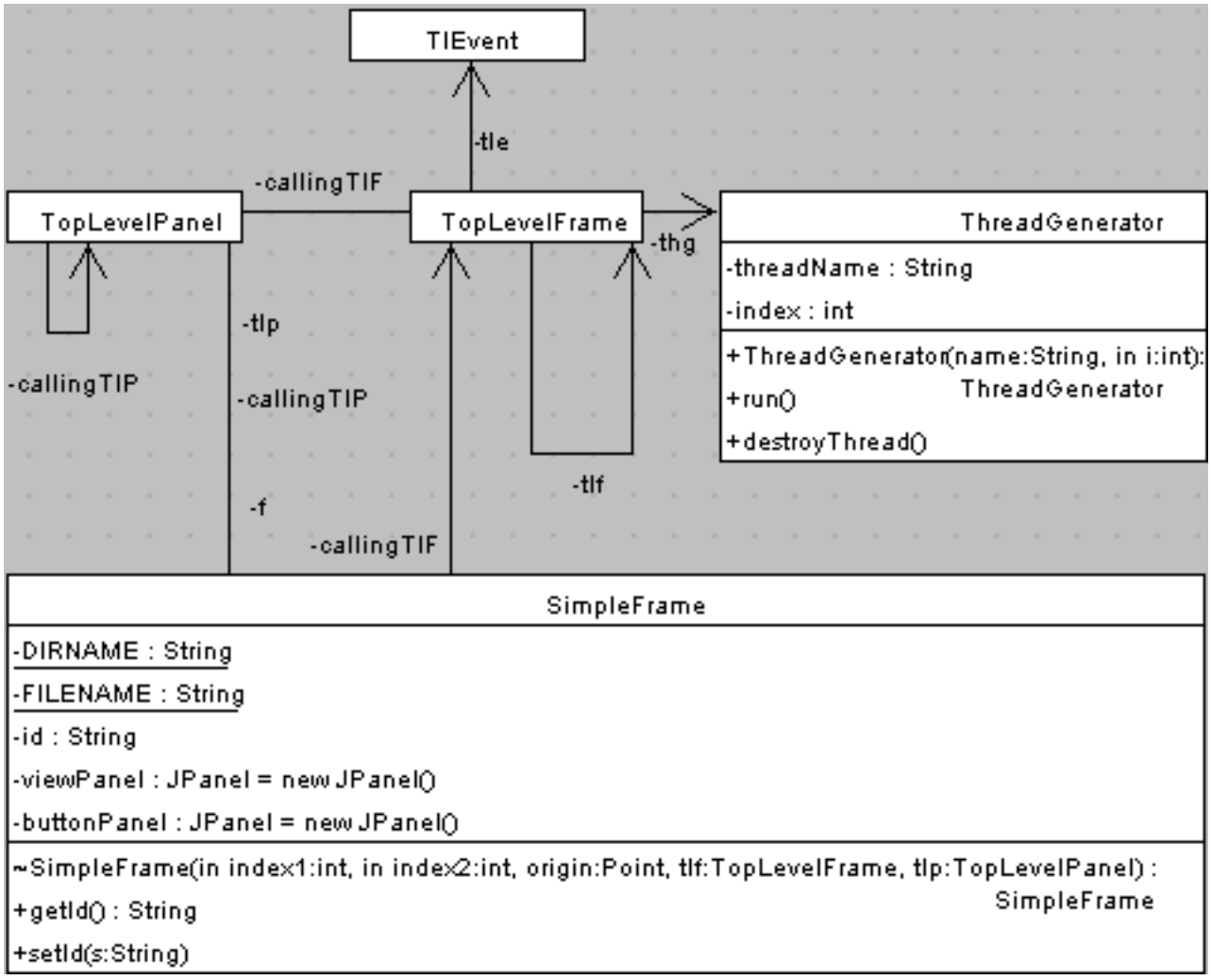

Figura A.9: *TopLevel: alcune classi principali in dettaglio*

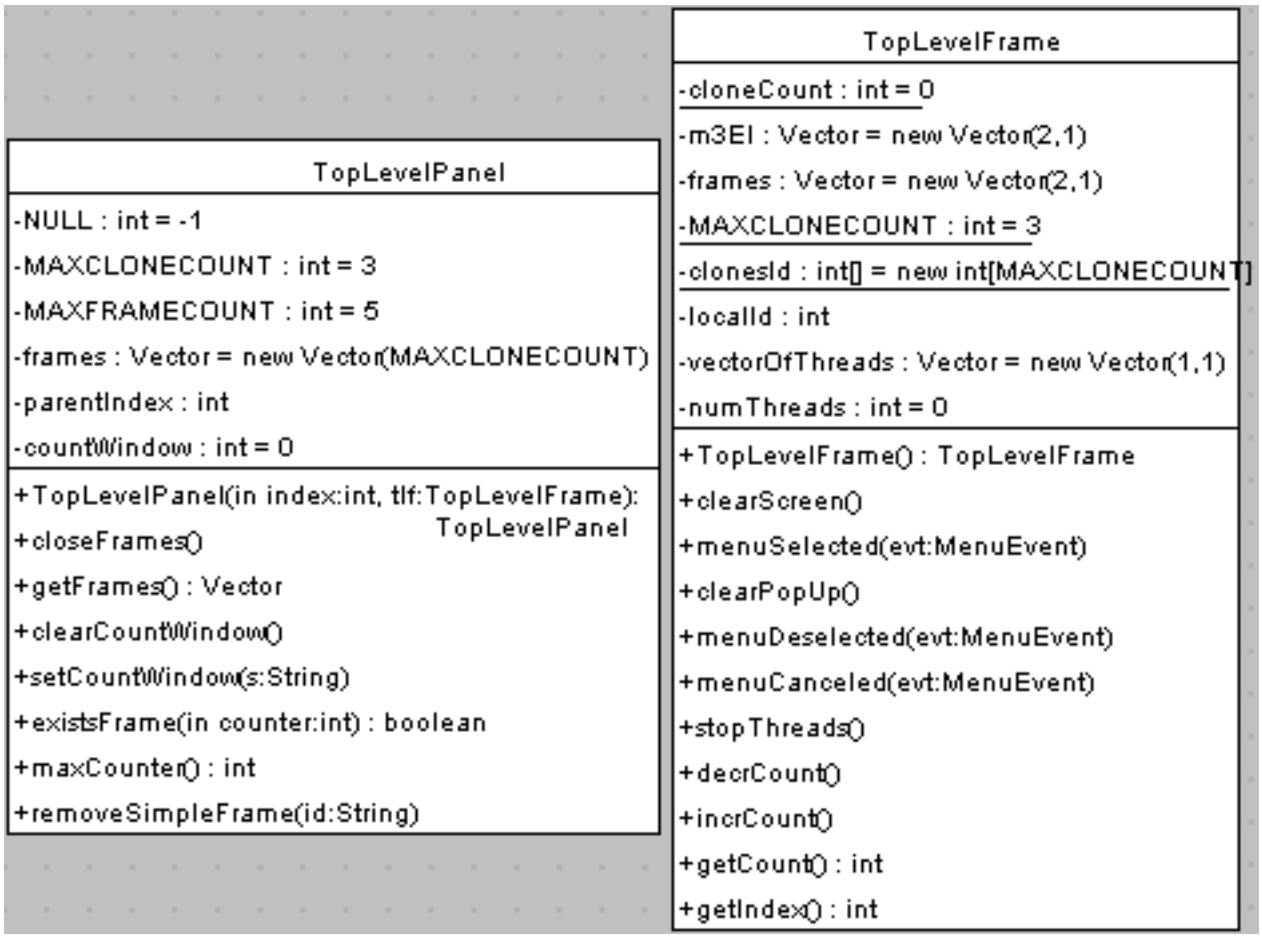

Figura A.10: *TopLevel: alcune classi principali in dettaglio*

A.2.2 *Causal Loop Graphic Editor (ClEdit)*

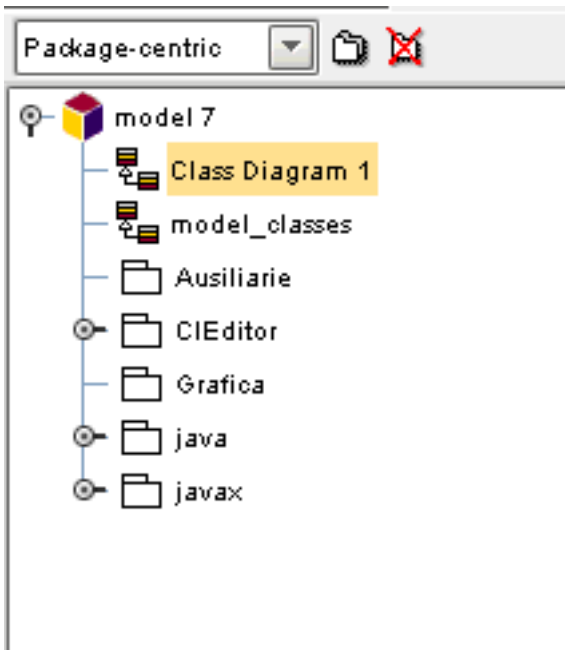

Figura A.11: *ClEdit: visione "package centric"*

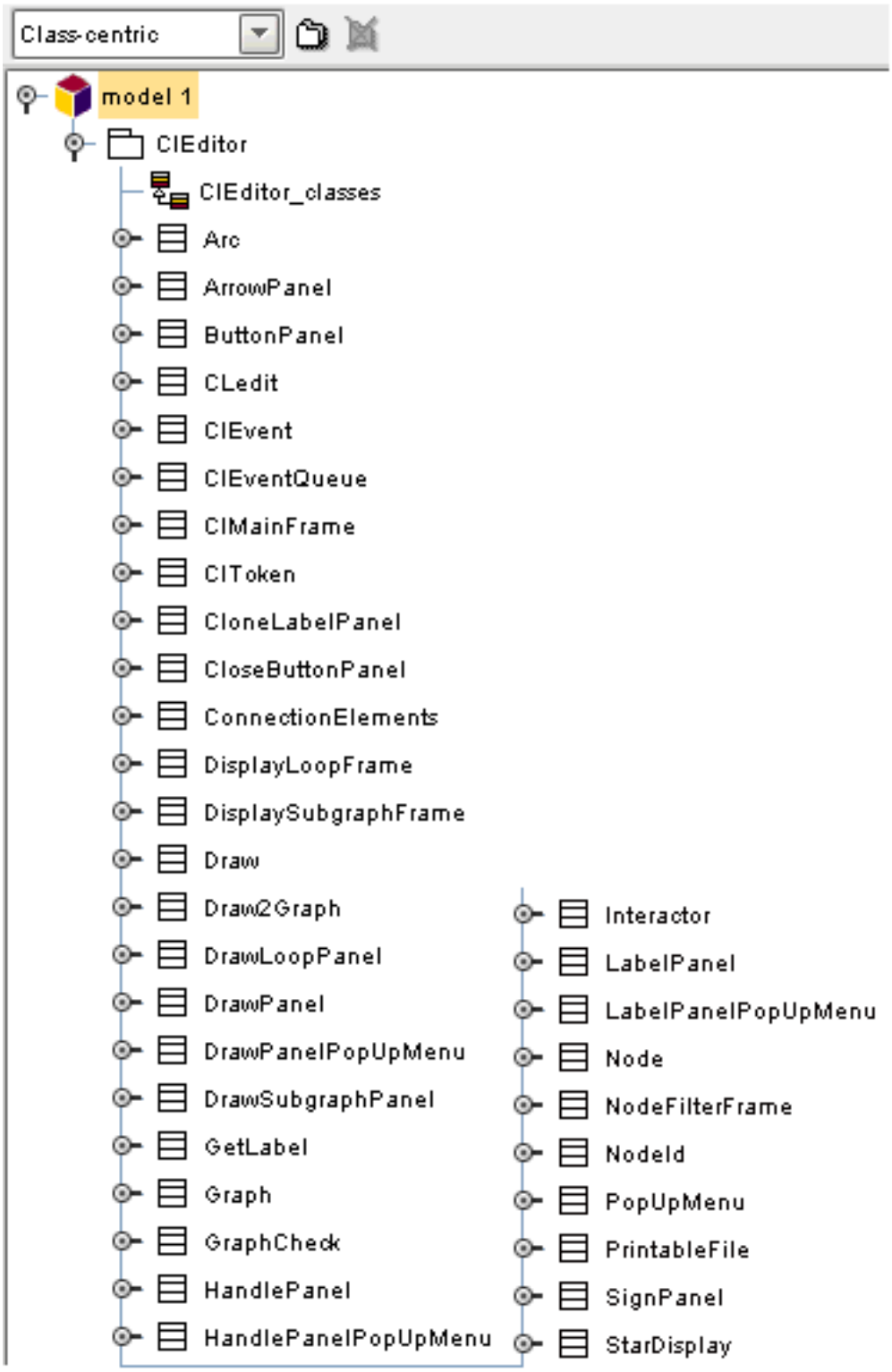

Figura A.12: *ClEdit: visione "class centric"*

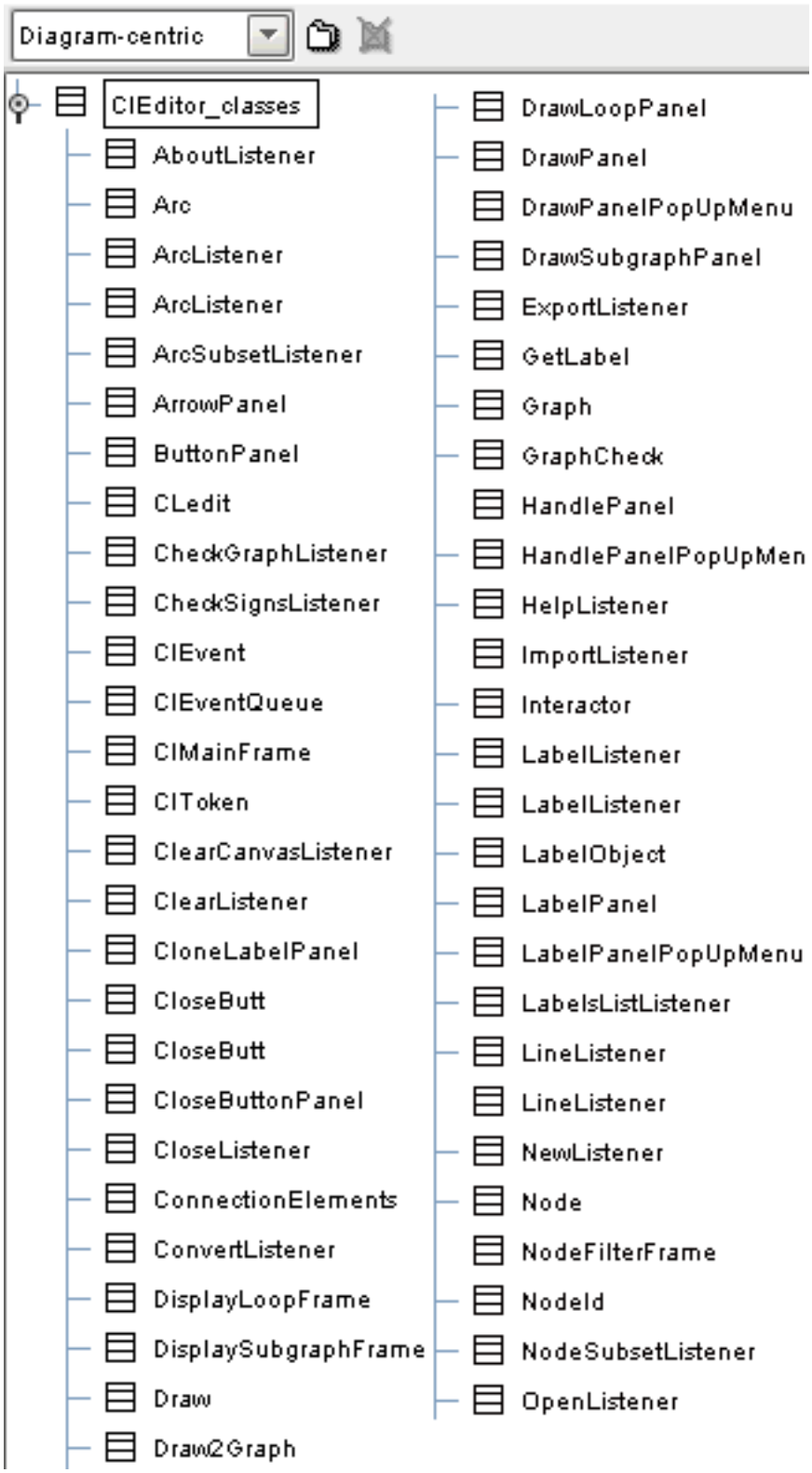

Figura A.13: *ClEdit: visione "diagram centric (1)"*

Figura A.14: *ClEdit: visione "diagram centric (2)"*

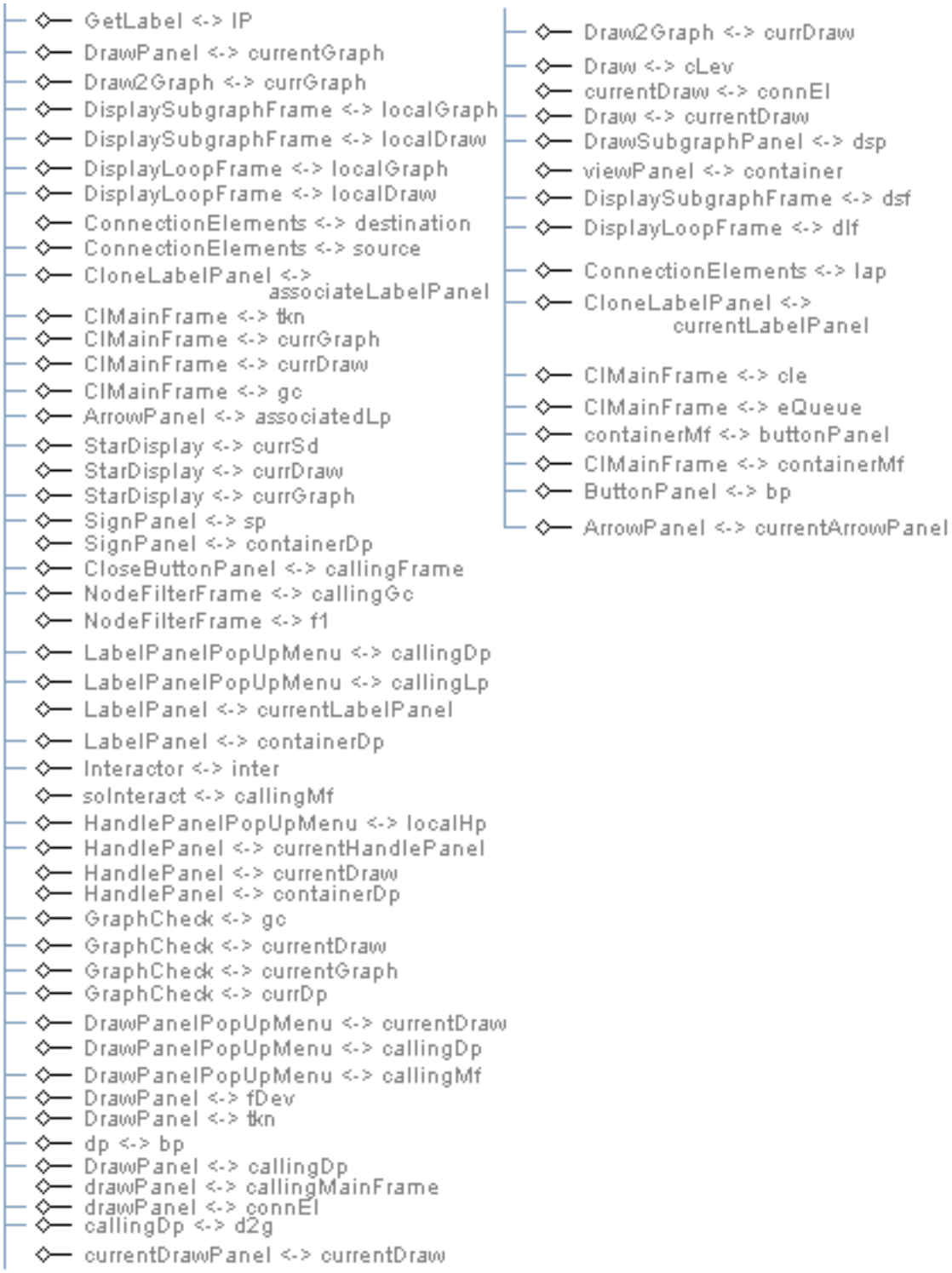

Figura A.15: *ClEdit: visione "diagram centric (3)"*

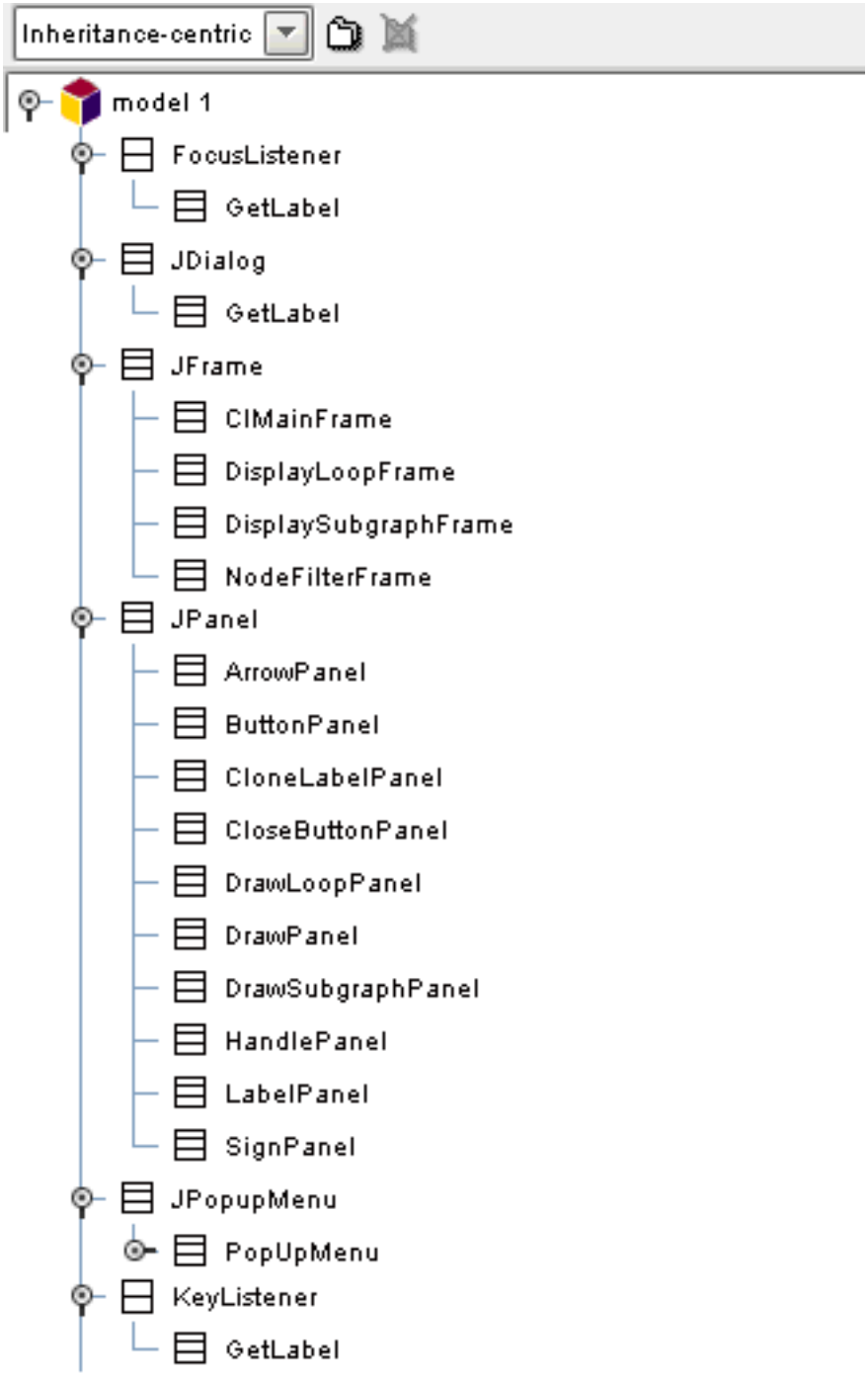

Figura A.16: *ClEdit: visione "inheritance centric (1)"*

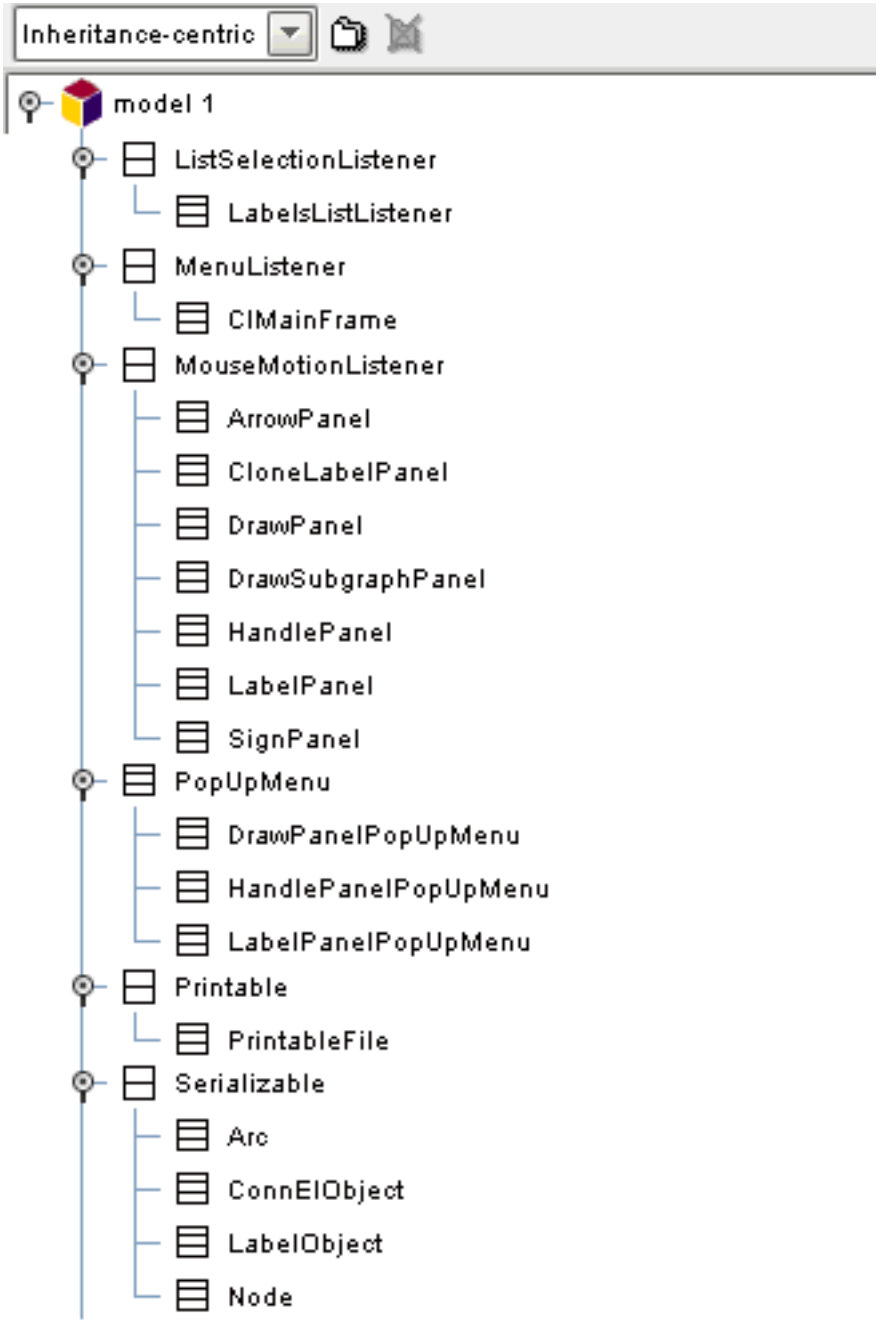

Figura A.17: *ClEdit: visione "inheritance centric (2)"*

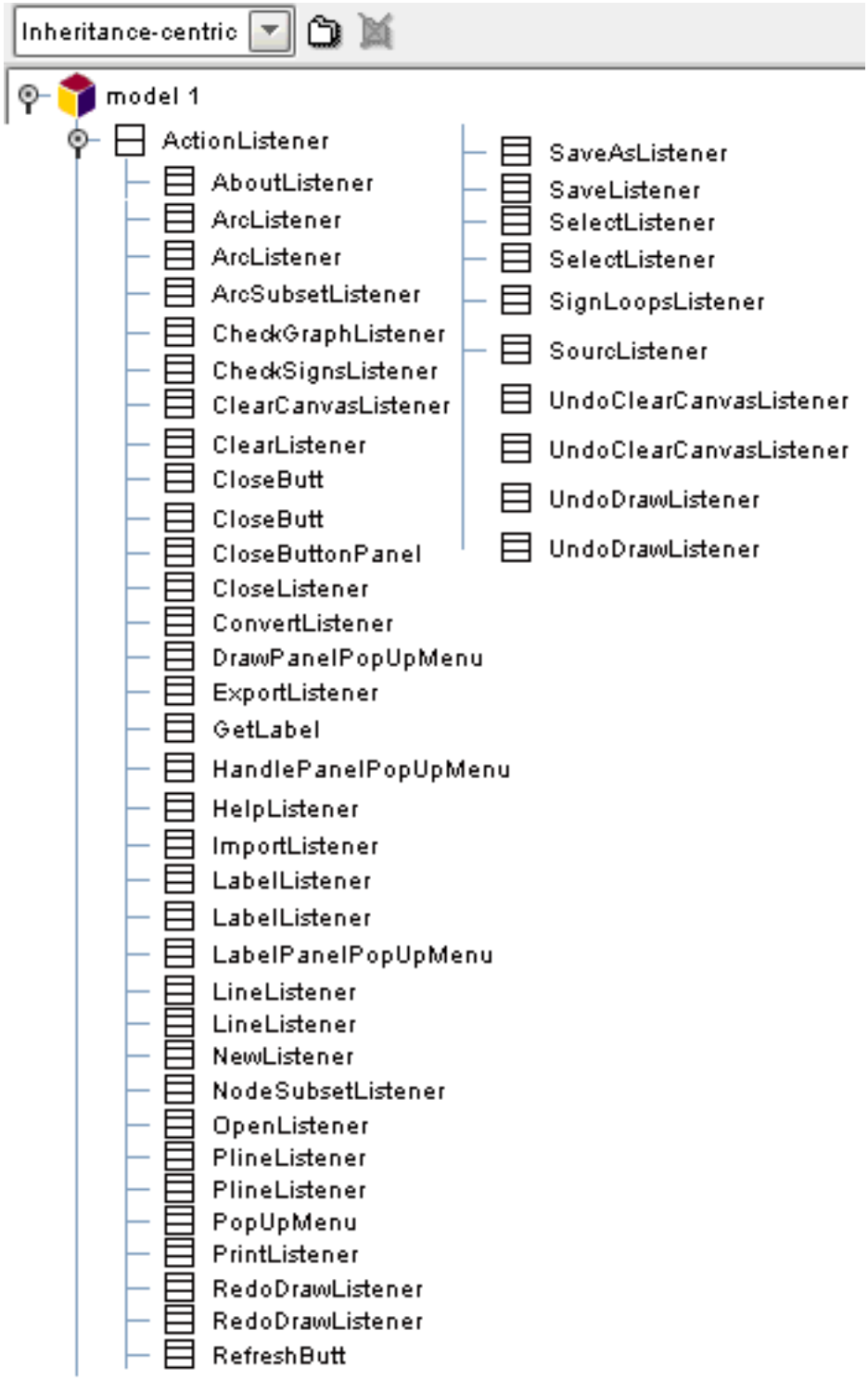

Figura A.18: *ClEdit: visione "inheritance centric (3)"*

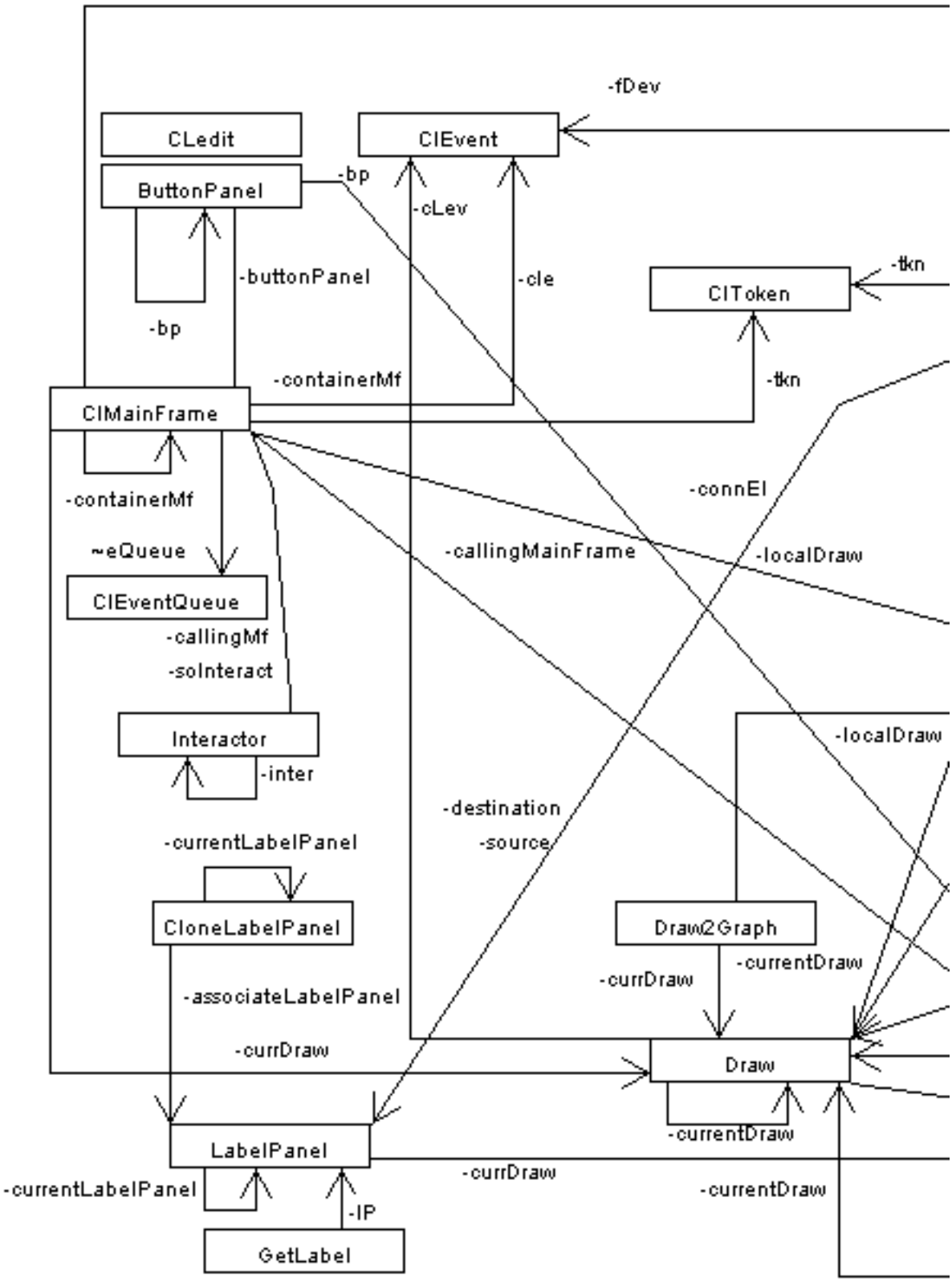

Figura A.19: *ClEdit: diagramma delle classi (1)*

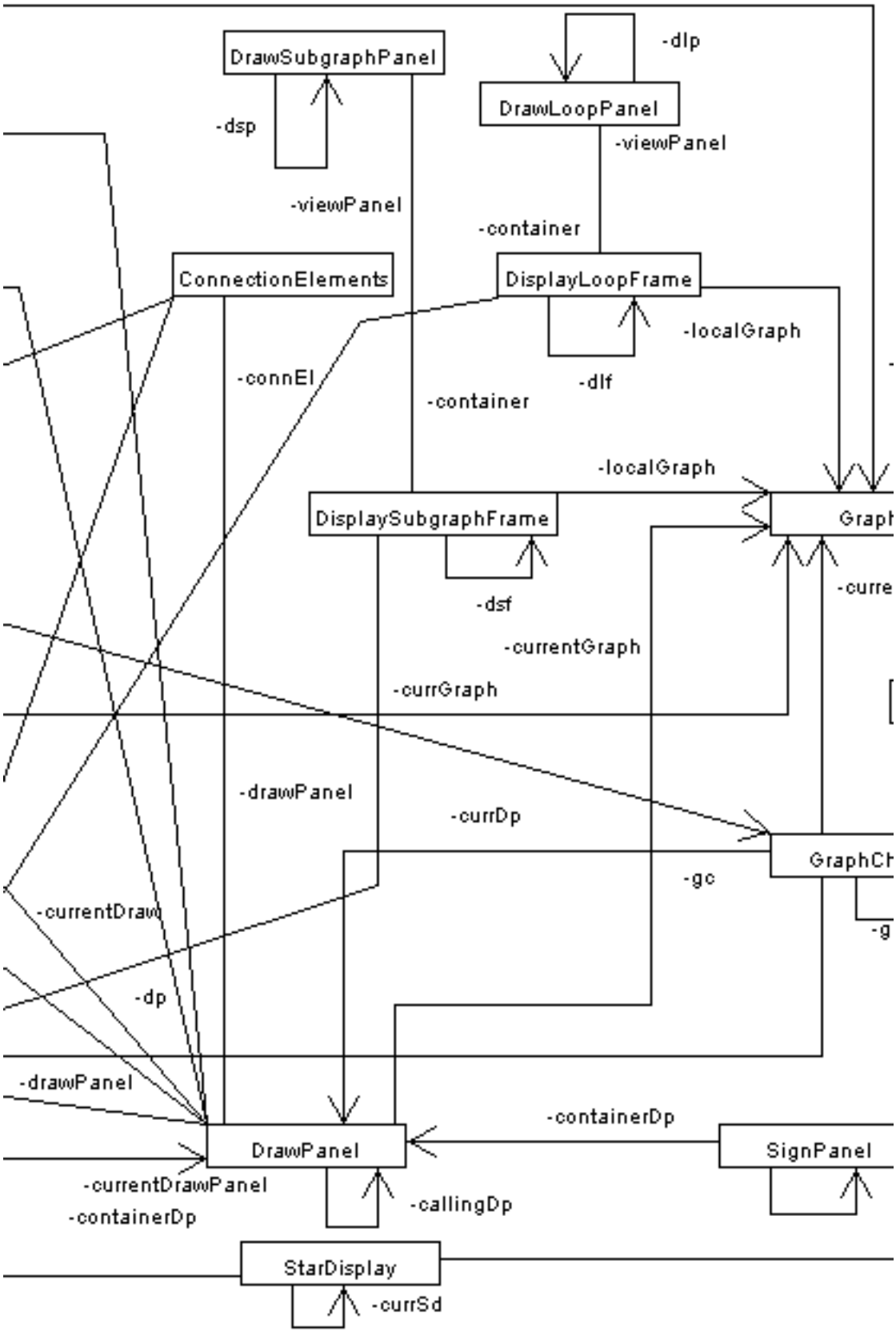

Figura A.20: *ClEdit: diagramma delle classi (2)*

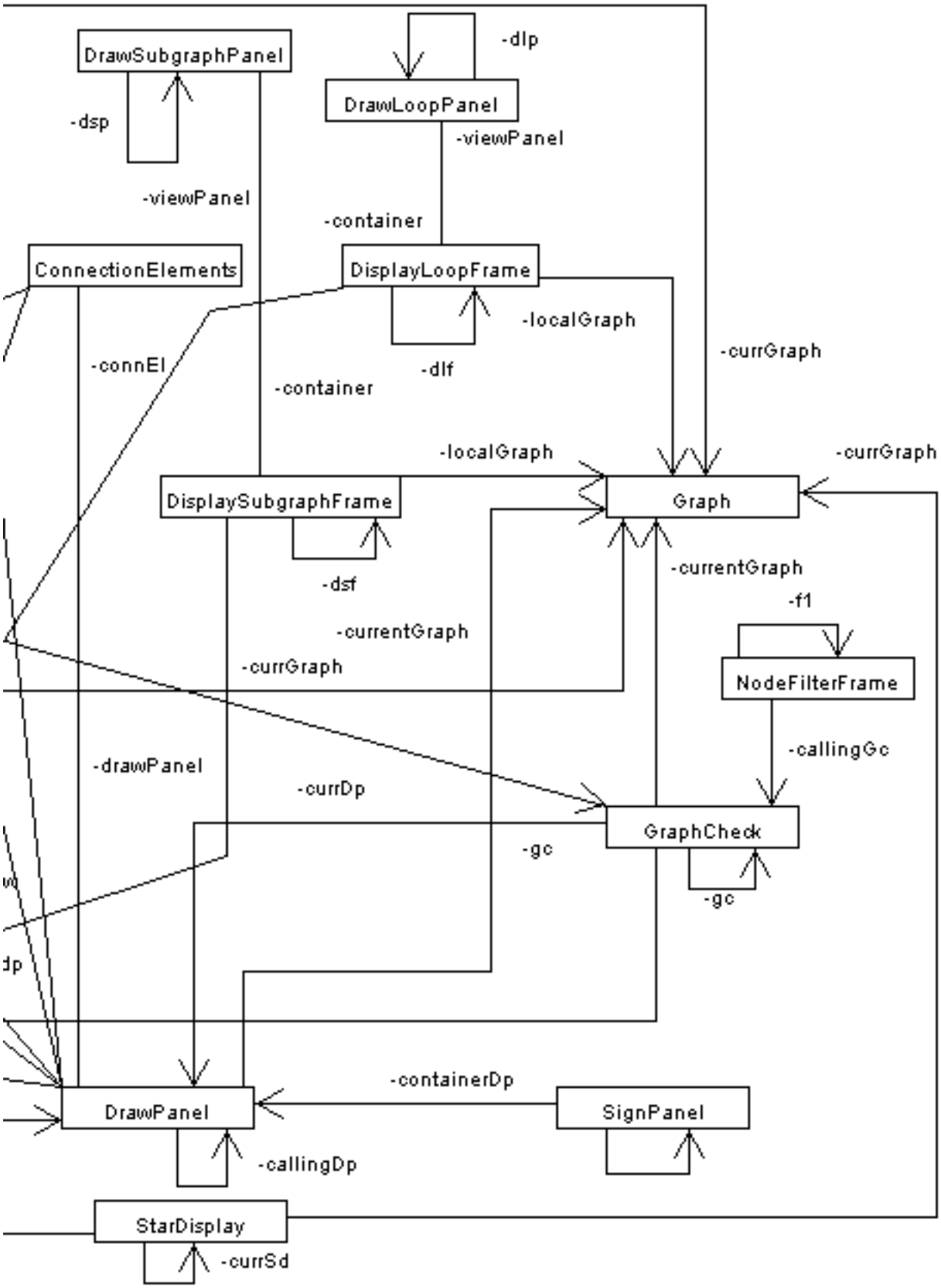

Figura A.21: *ClEdit: diagramma delle classi (3)*

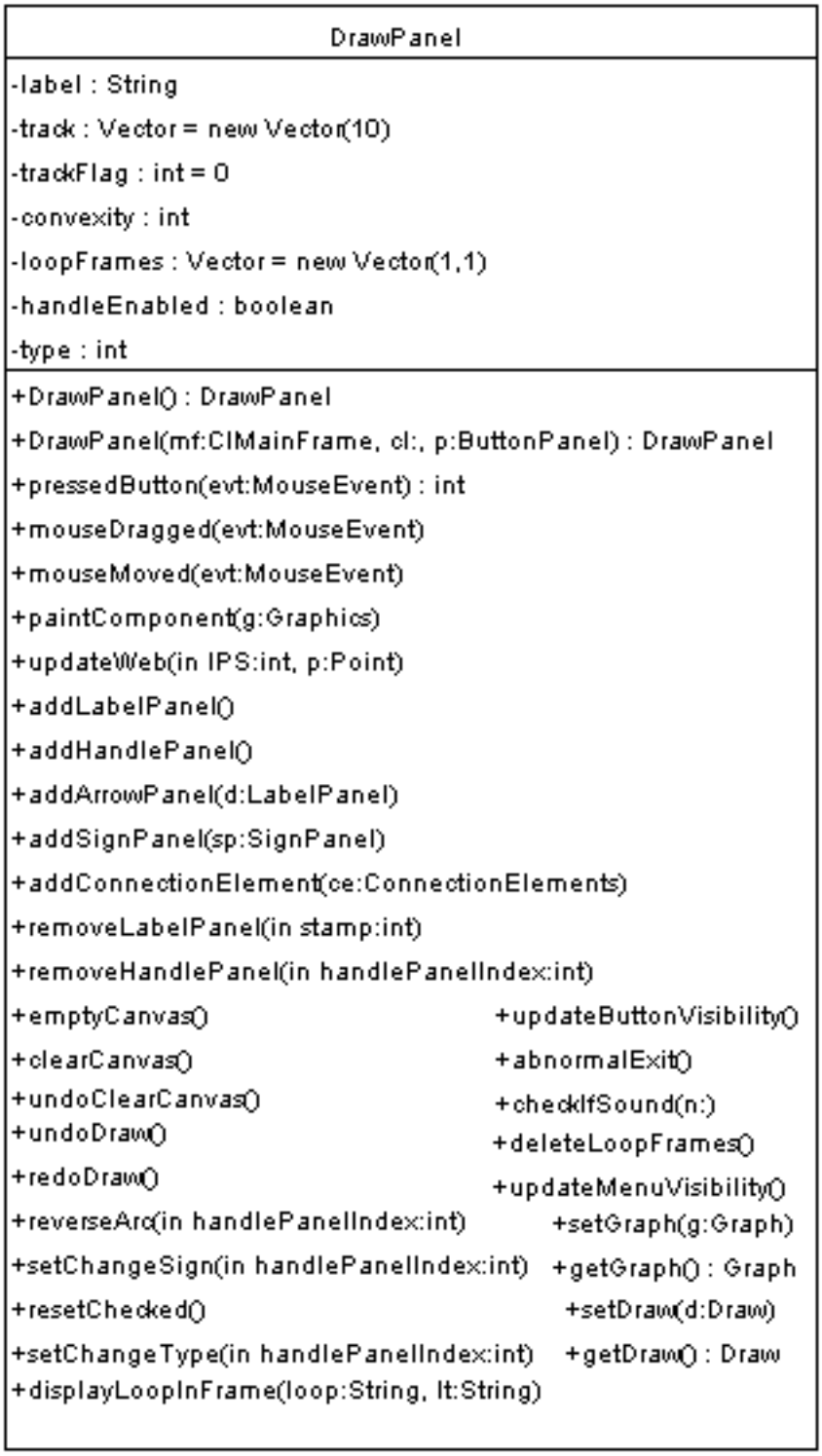

Figura A.22: *ClEdit: classi in dettaglio (1)*

Figura A.23: *ClEdit: classi in dettaglio (2)*

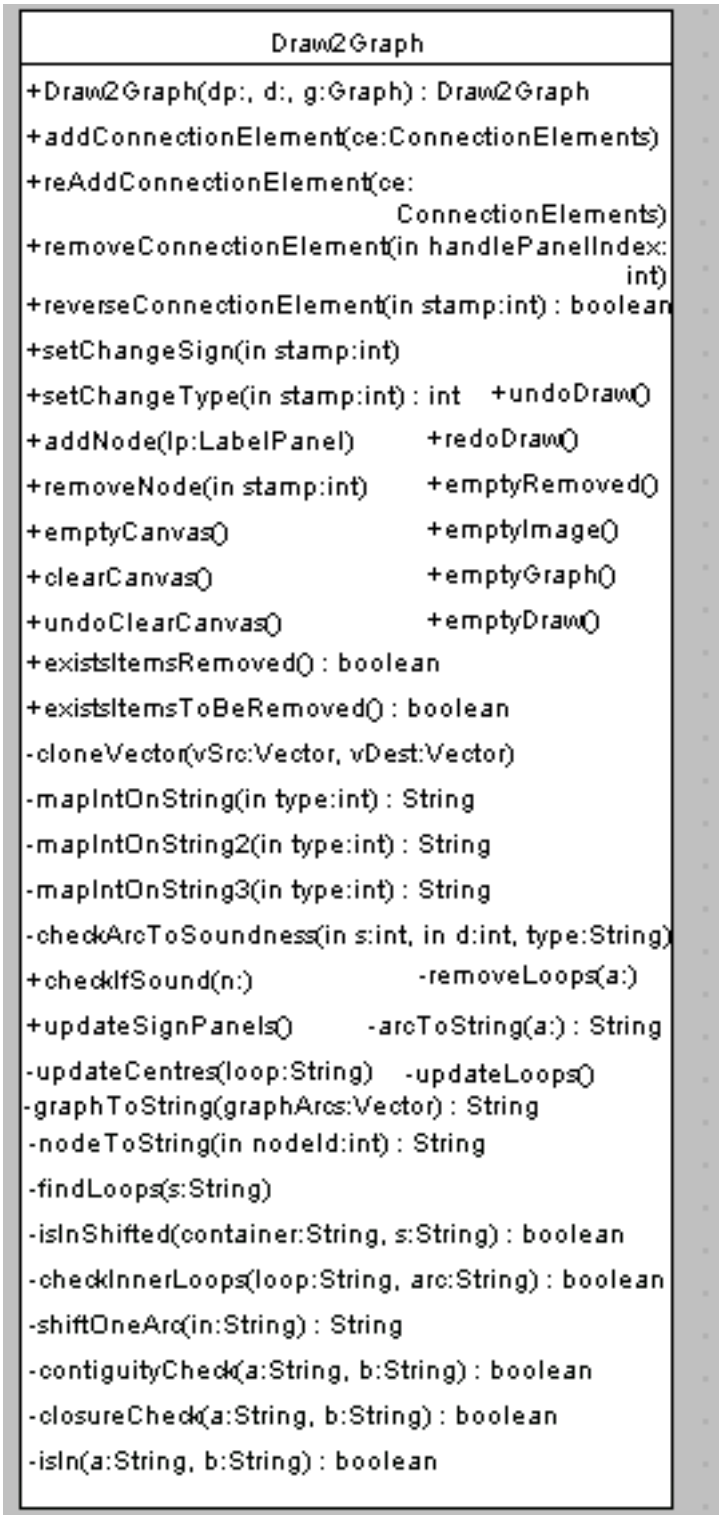

Figura A.24: *ClEdit: classi in dettaglio (3)*

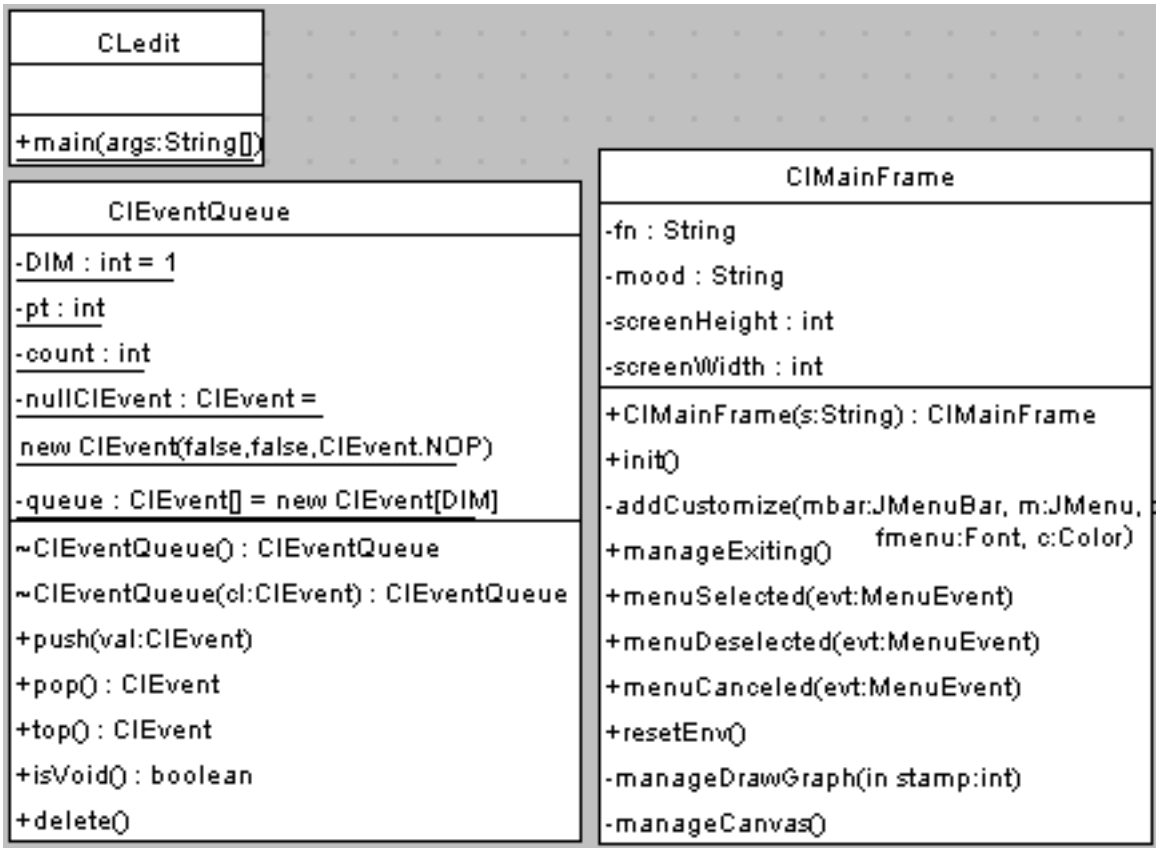

Figura A.25: *ClEdit: classi in dettaglio (4)*

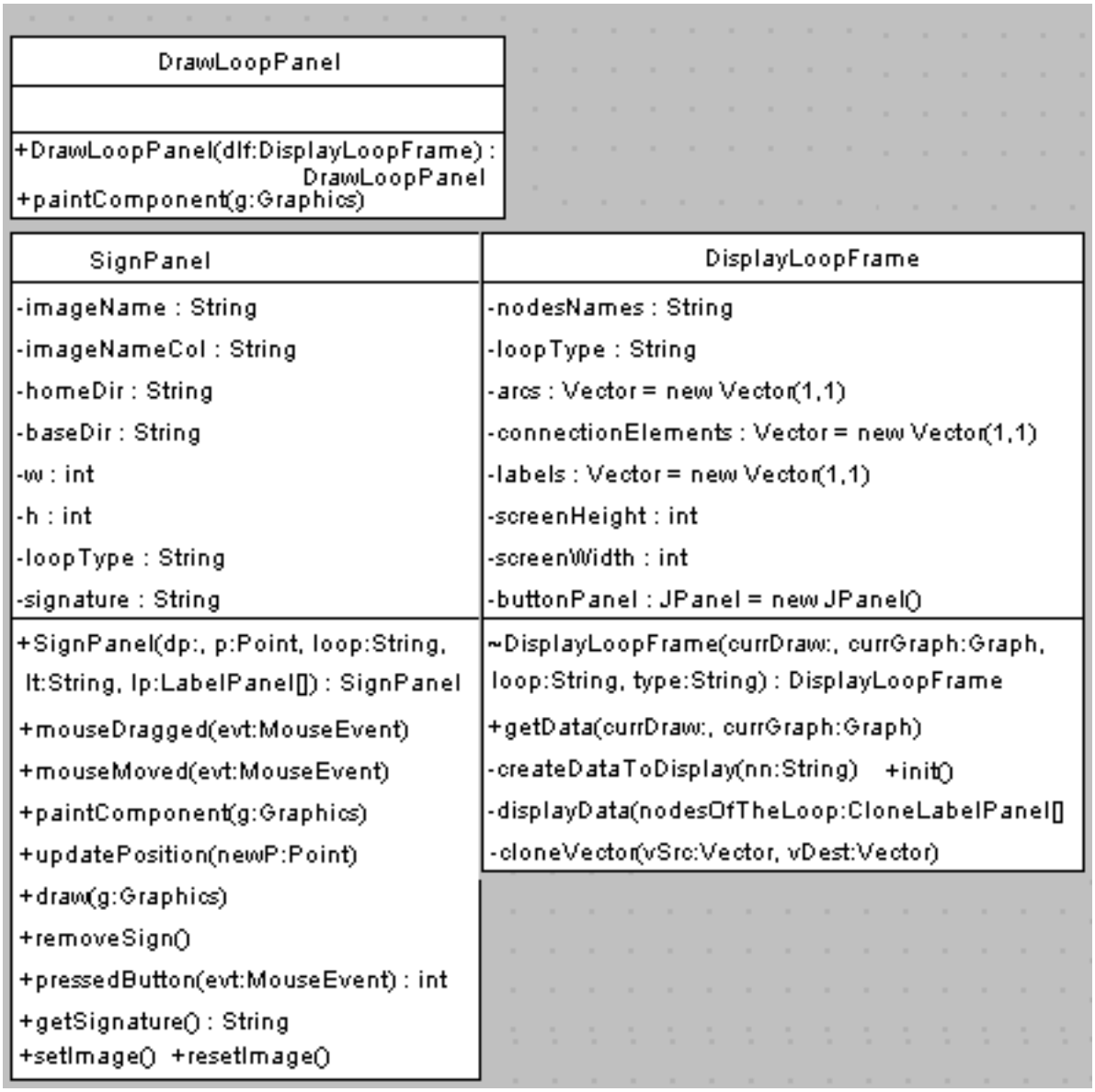

Figura A.26: *ClEdit: classi in dettaglio (5)*

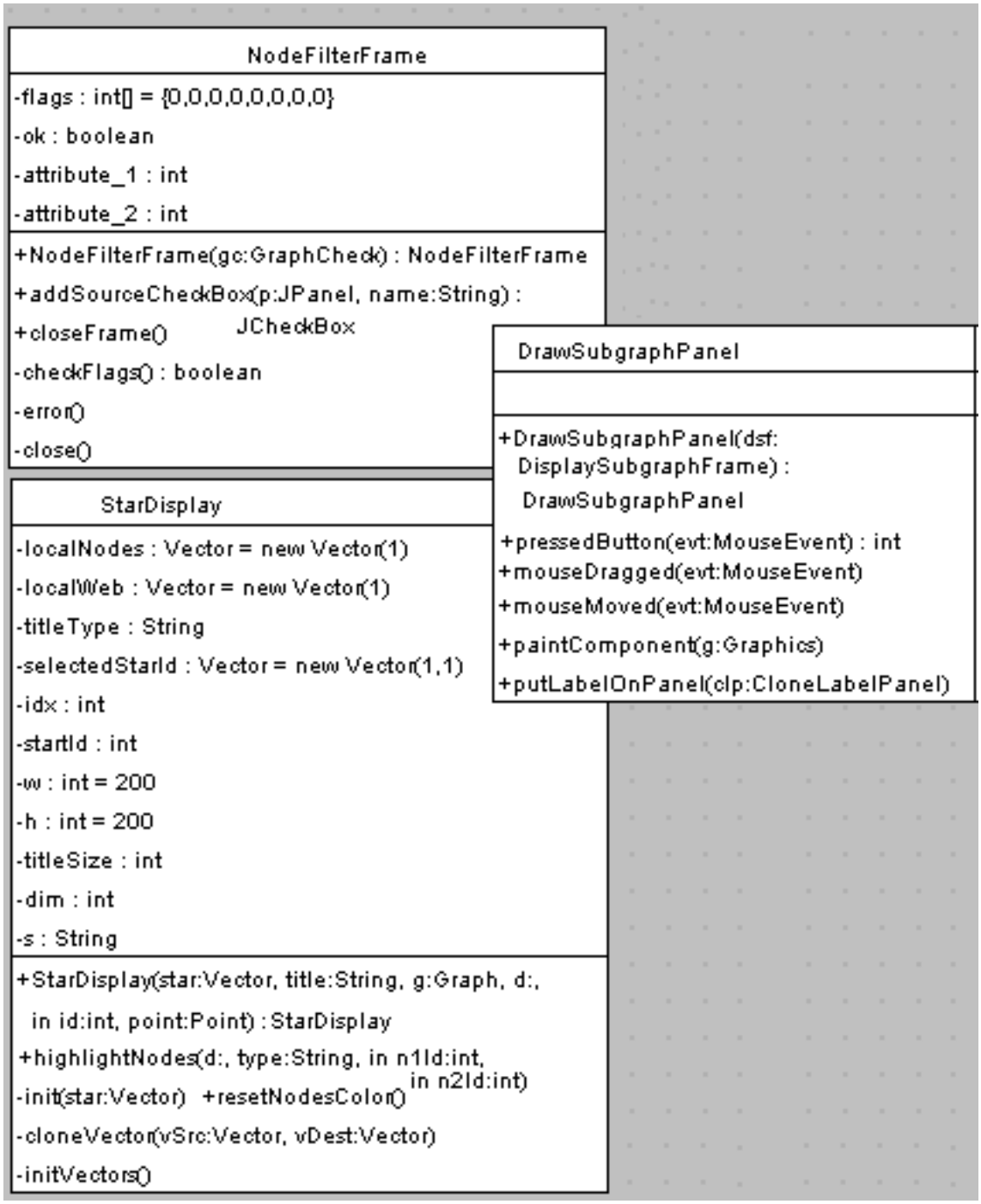

Figura A.27: *ClEdit: classi in dettaglio (6)*

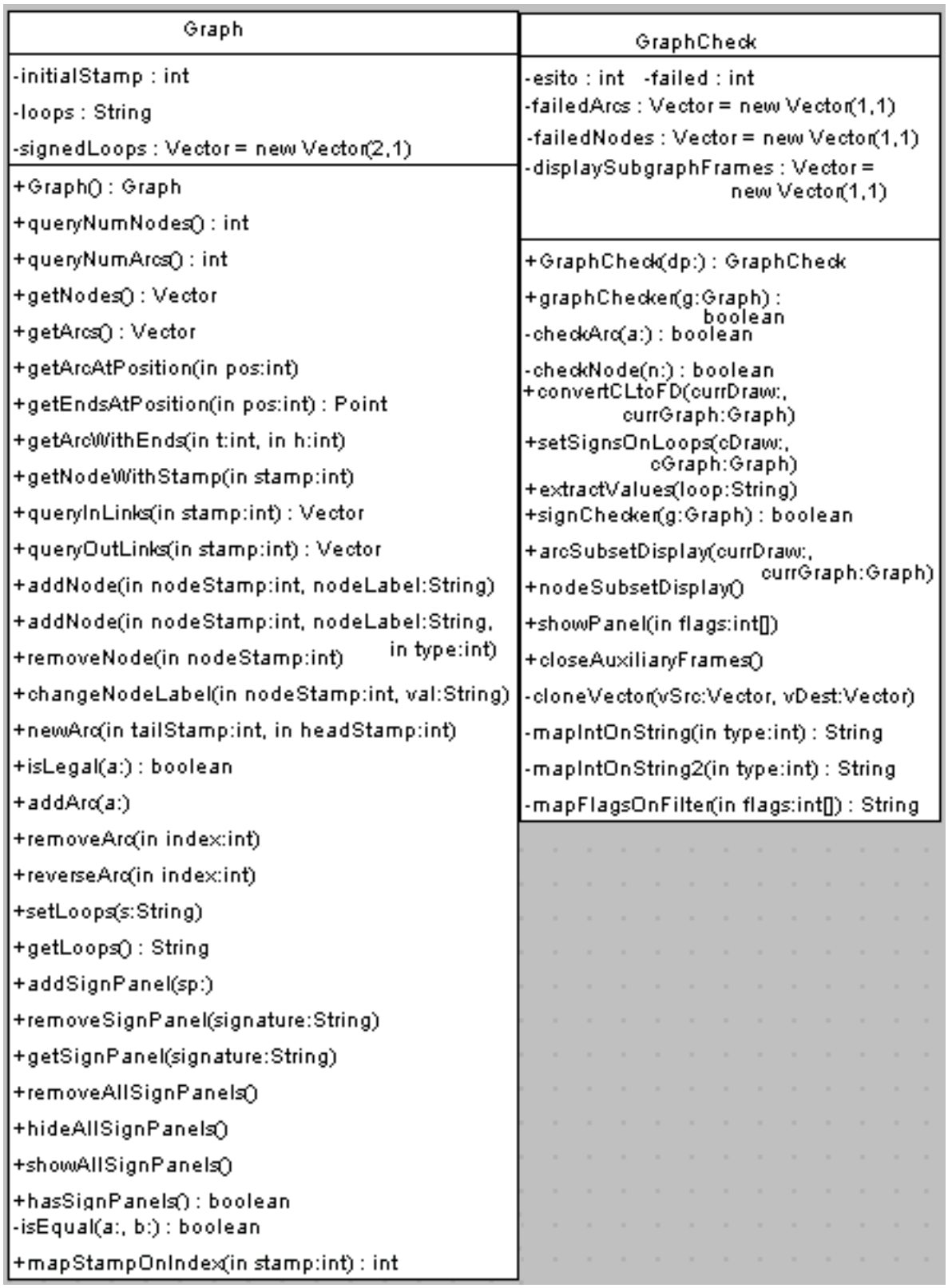

Figura A.28: *ClEdit: classi in dettaglio (7)*

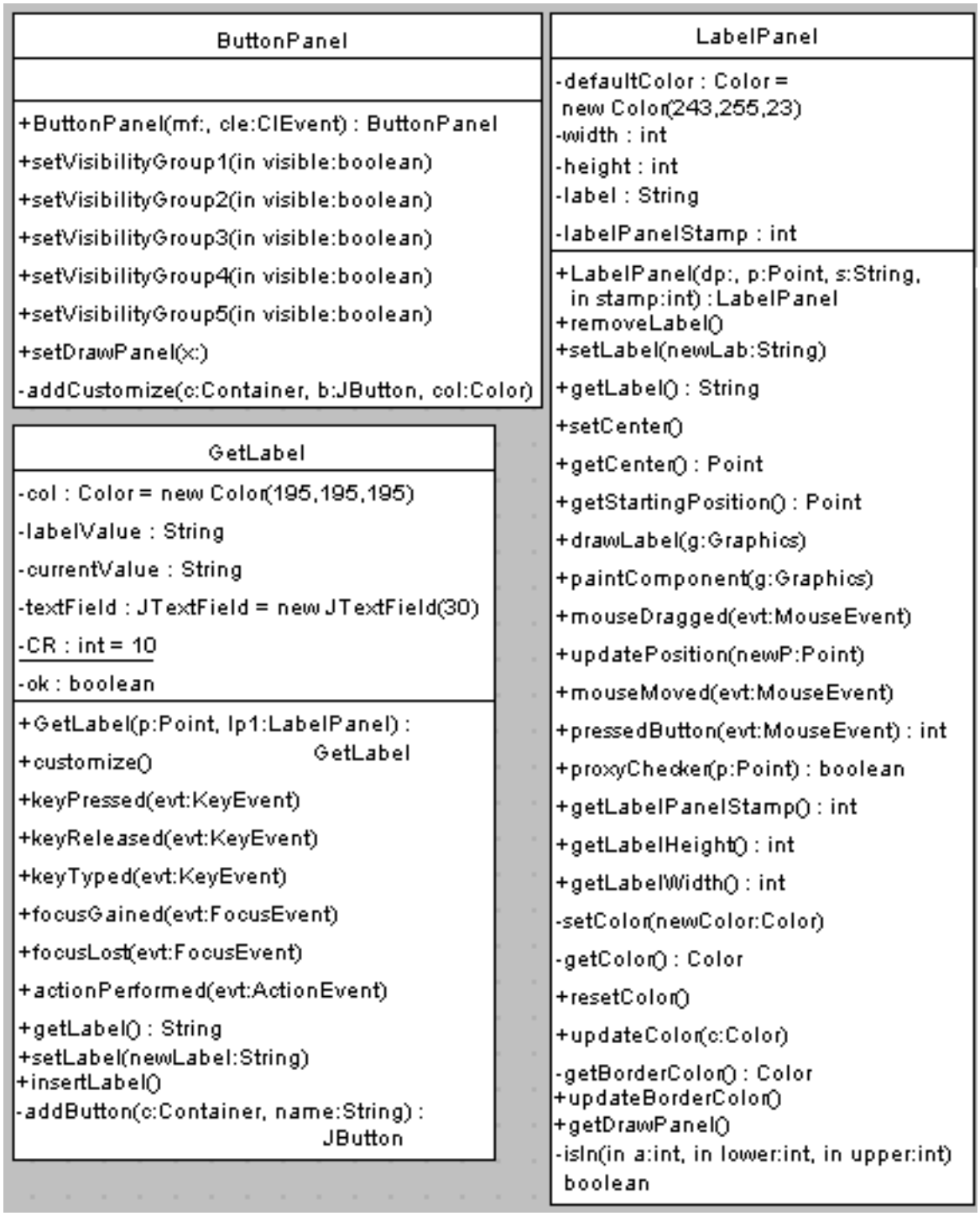

Figura A.29: *ClEdit: classi in dettaglio (8)*

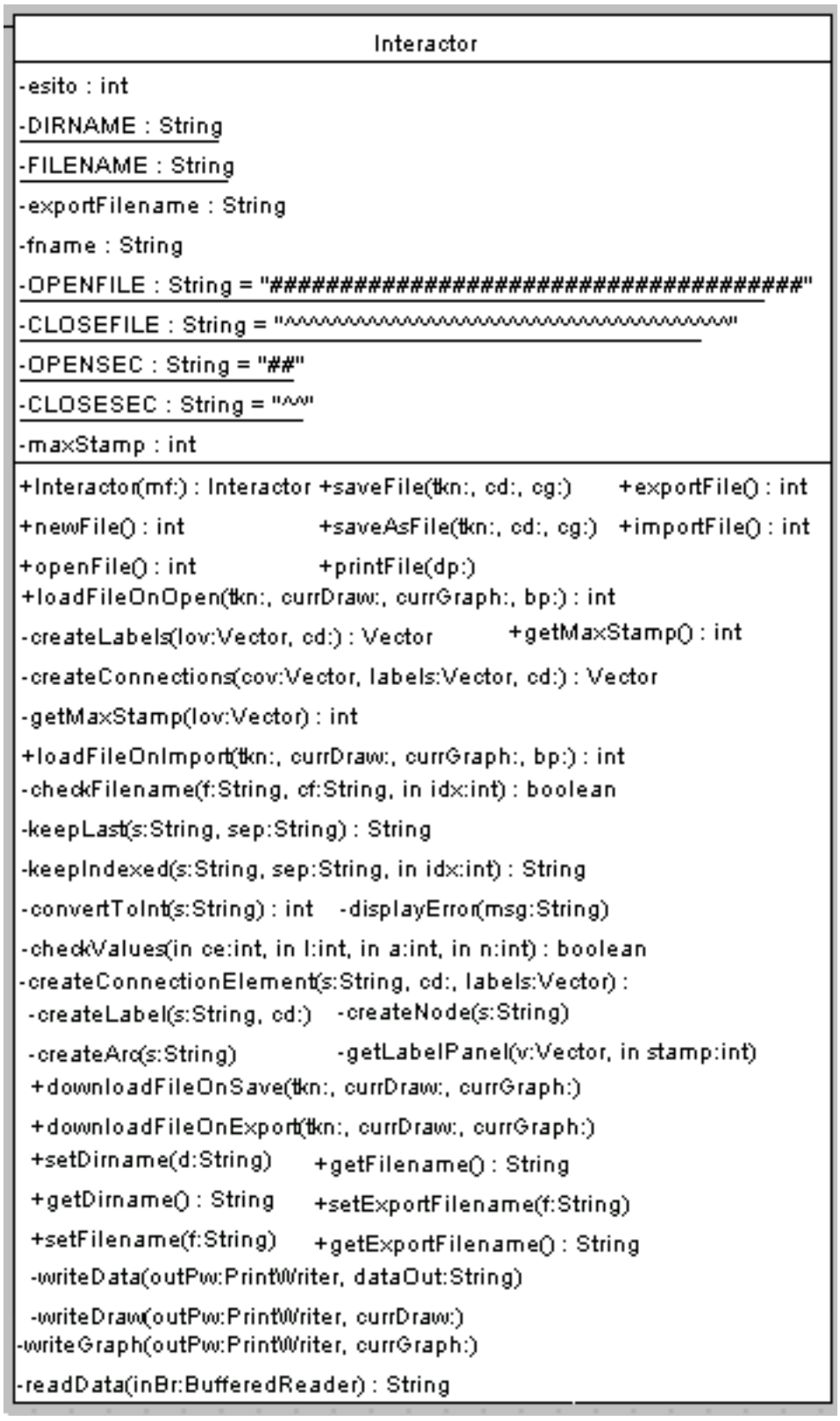

Figura A.30: *ClEdit: classi in dettaglio (9)*

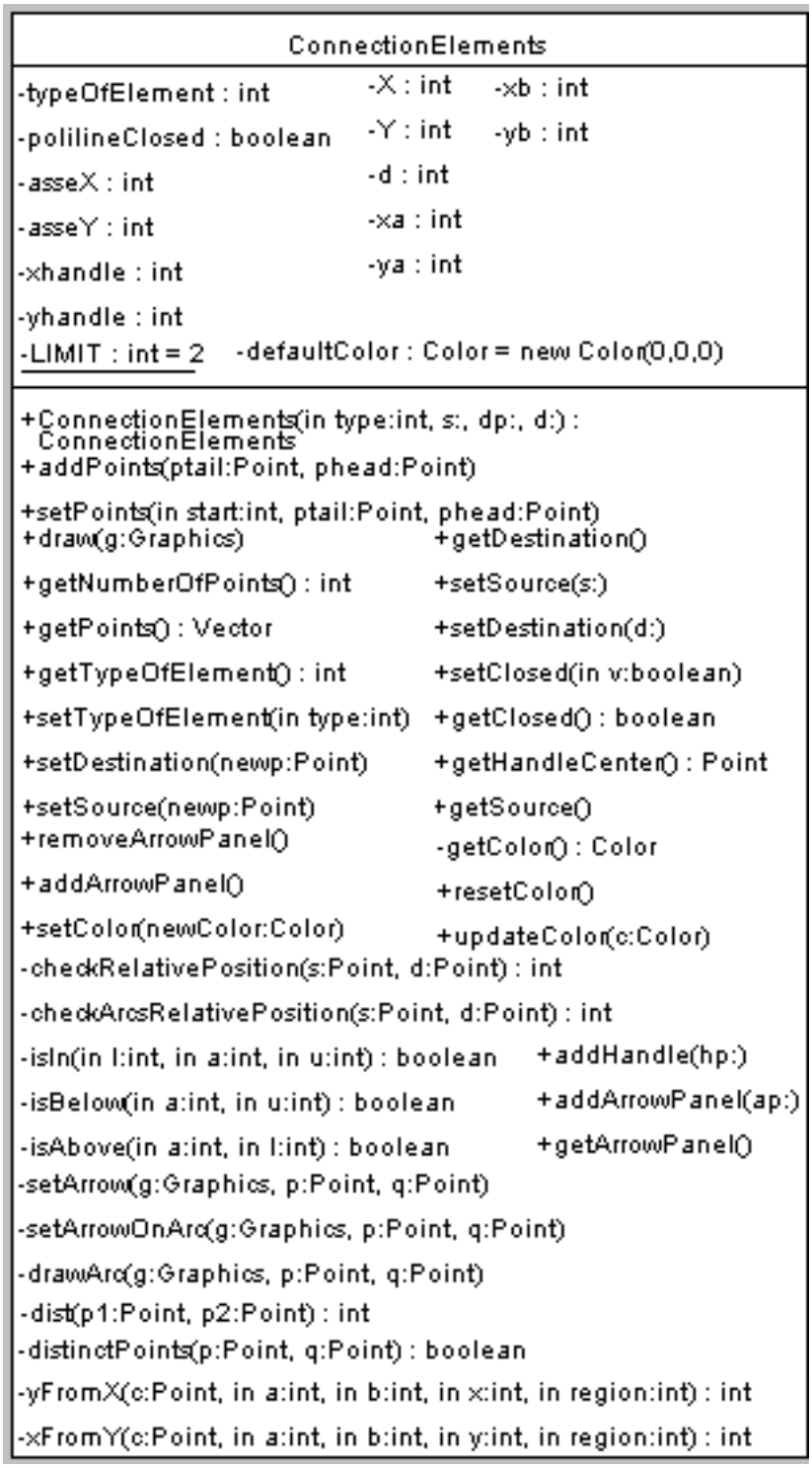

Figura A.31: *ClEdit: classi in dettaglio (10)*

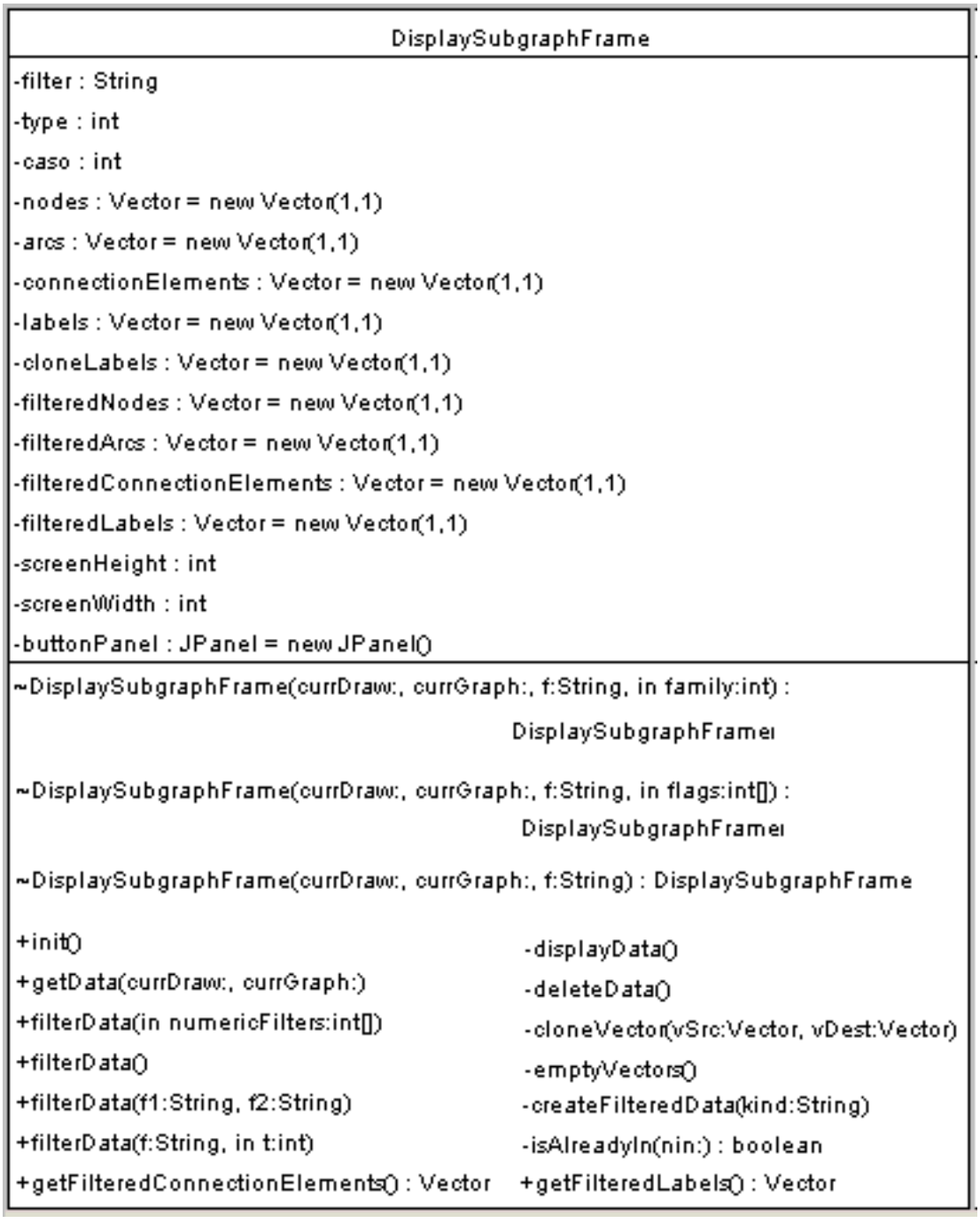

Figura A.32: *ClEdit: classi in dettaglio (11)*

A.2.3 *Flow Diagram Graphic Editor*

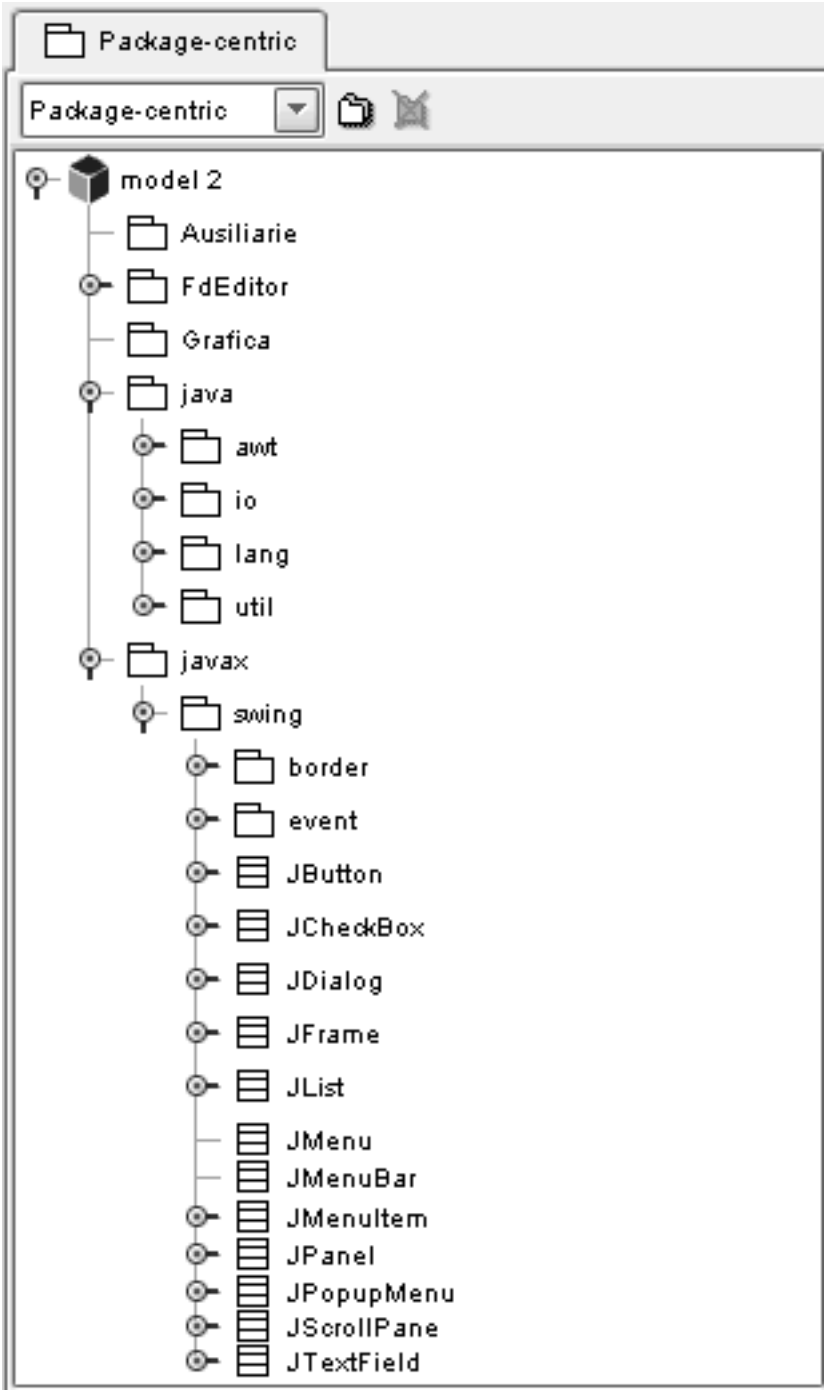

Figura A.33: *FdEdit: visione "package centric" (1)*

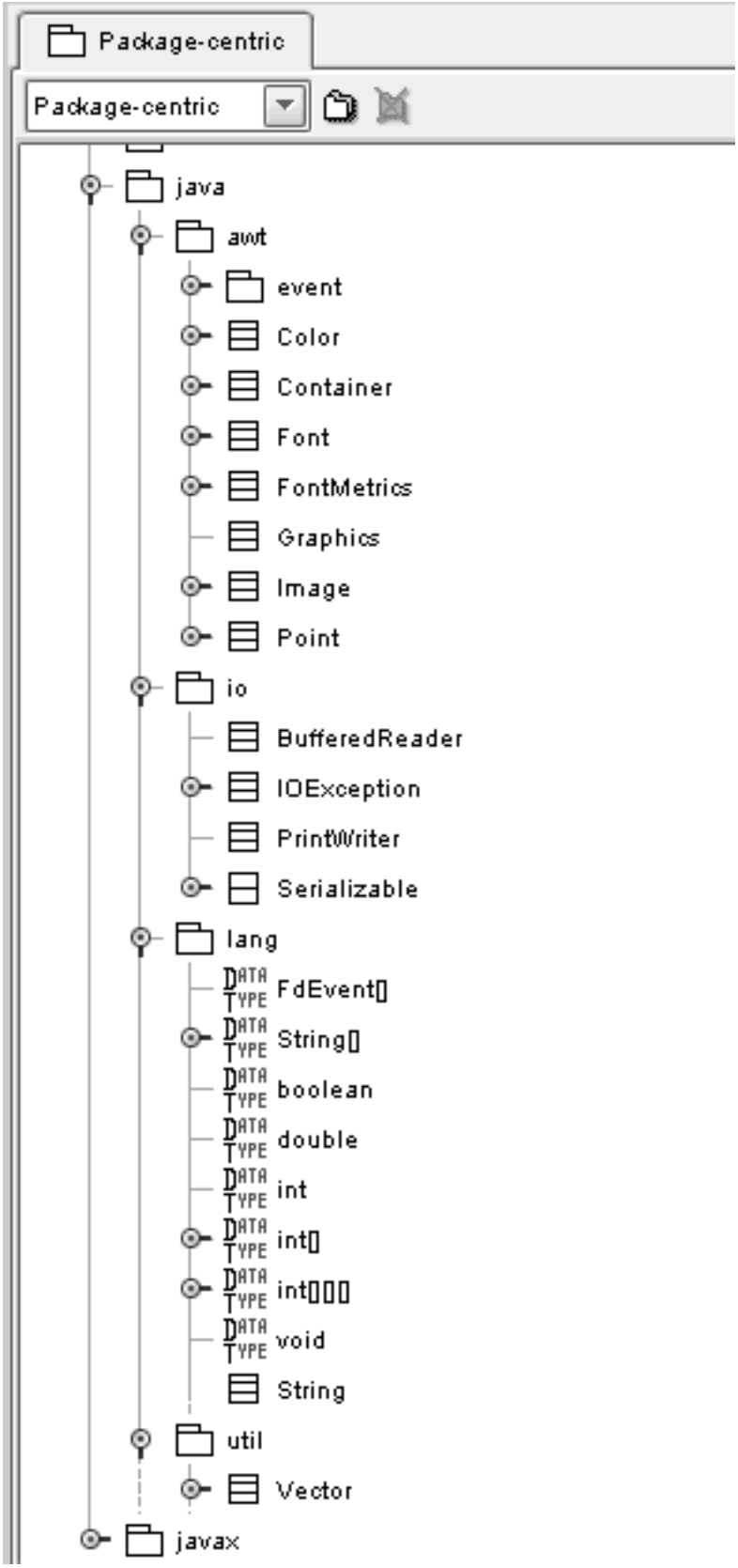

Figura A.34: *FdEdit: visione "package centric" (2)*

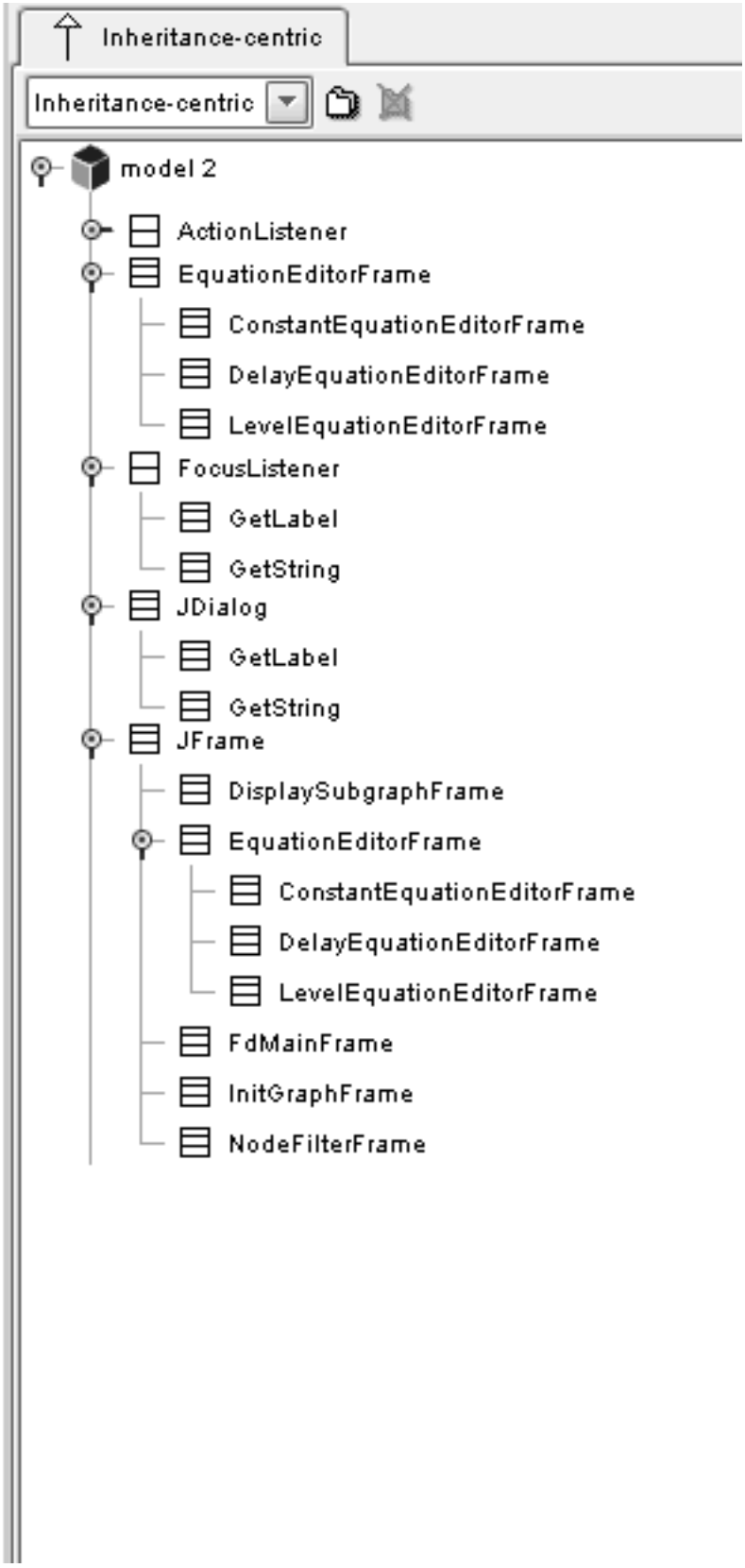

Figura A.35: *FdEdit: visione "inheritance centric" (1)*

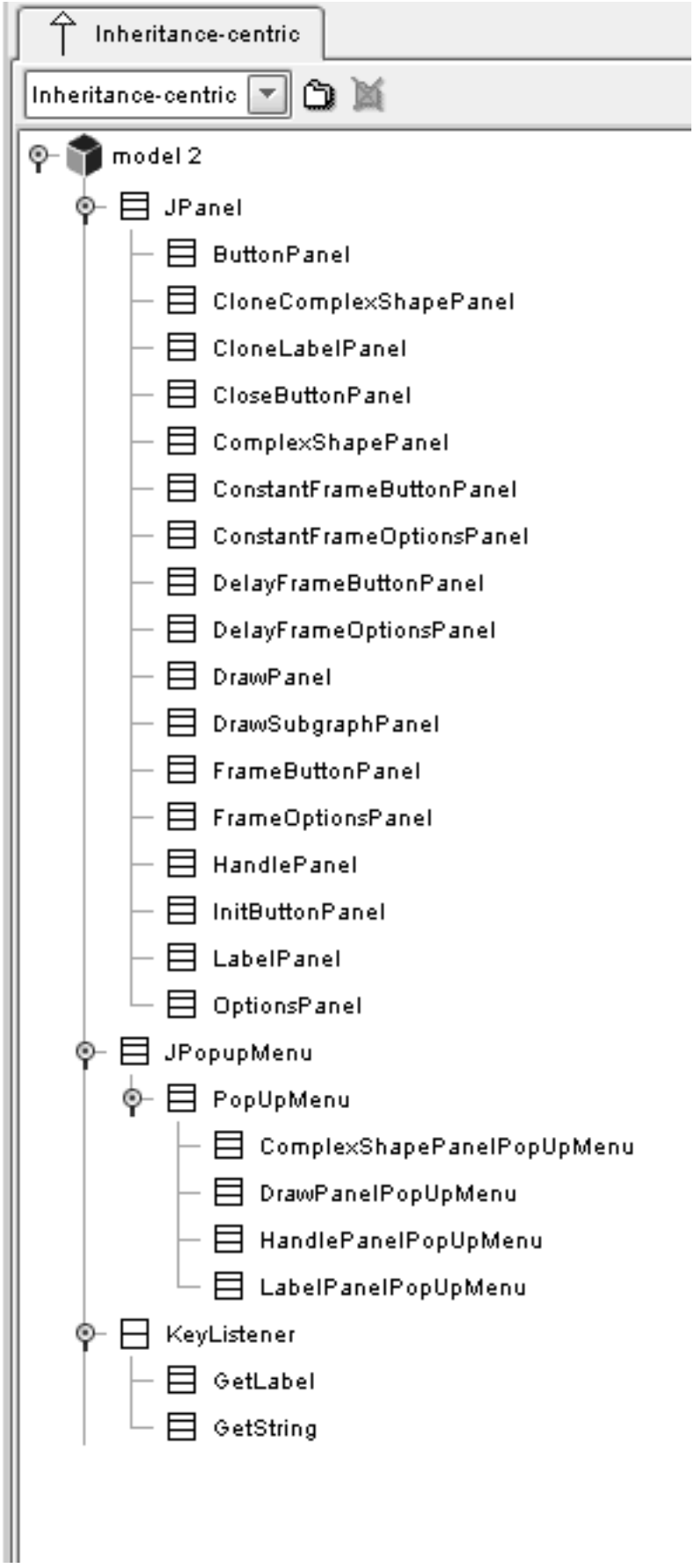

Figura A.36: *FdEdit: visione "inheritance centric" (2)*

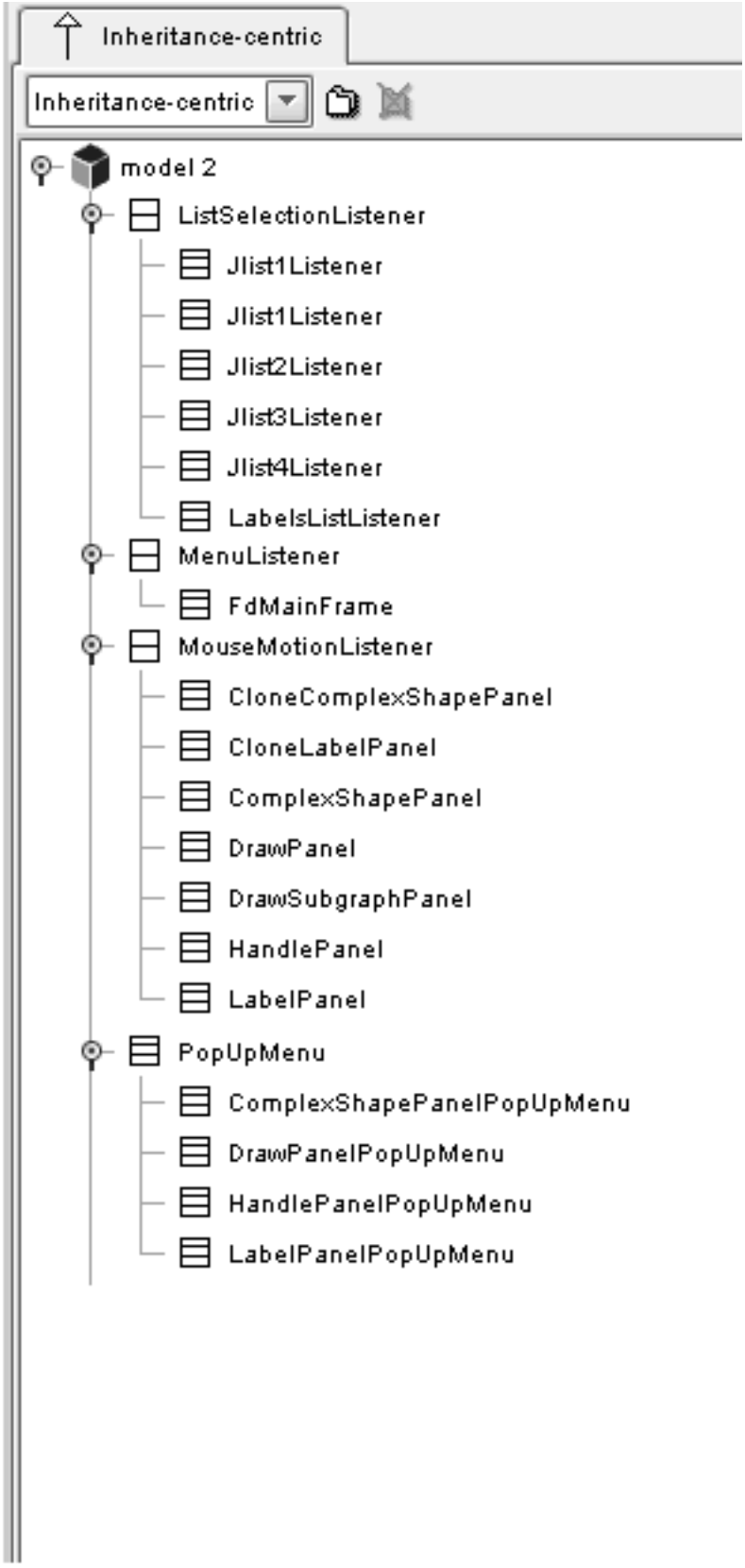

Figura A.37: *FdEdit: visione "inheritance centric"(3)*

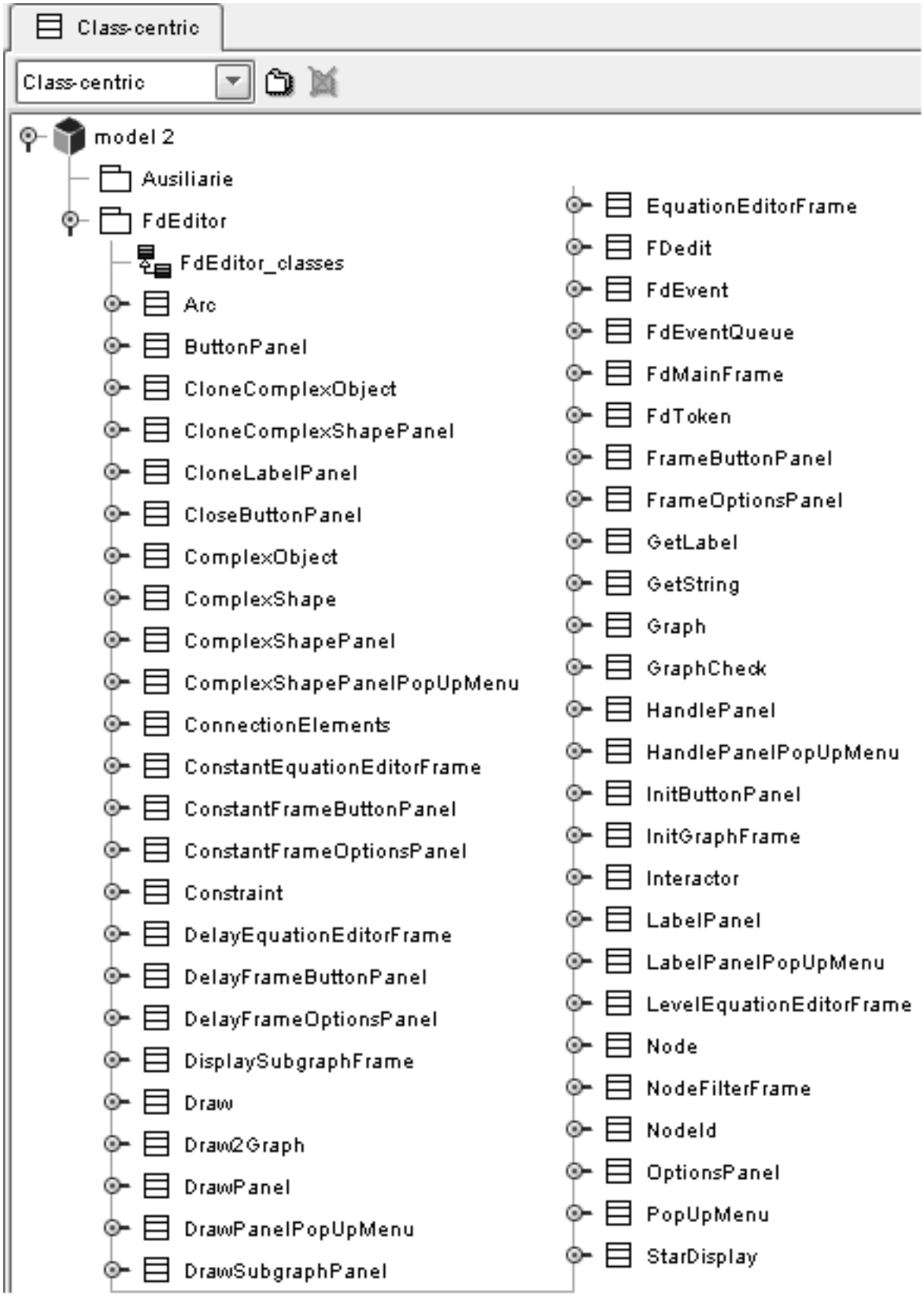

Figura A.38: *FdEdit: visione "class centric"*

Figura A.39: *FdEdit: le associazzioni (1)*

Figura A.40: *FdEdit: le associazioni (2)*

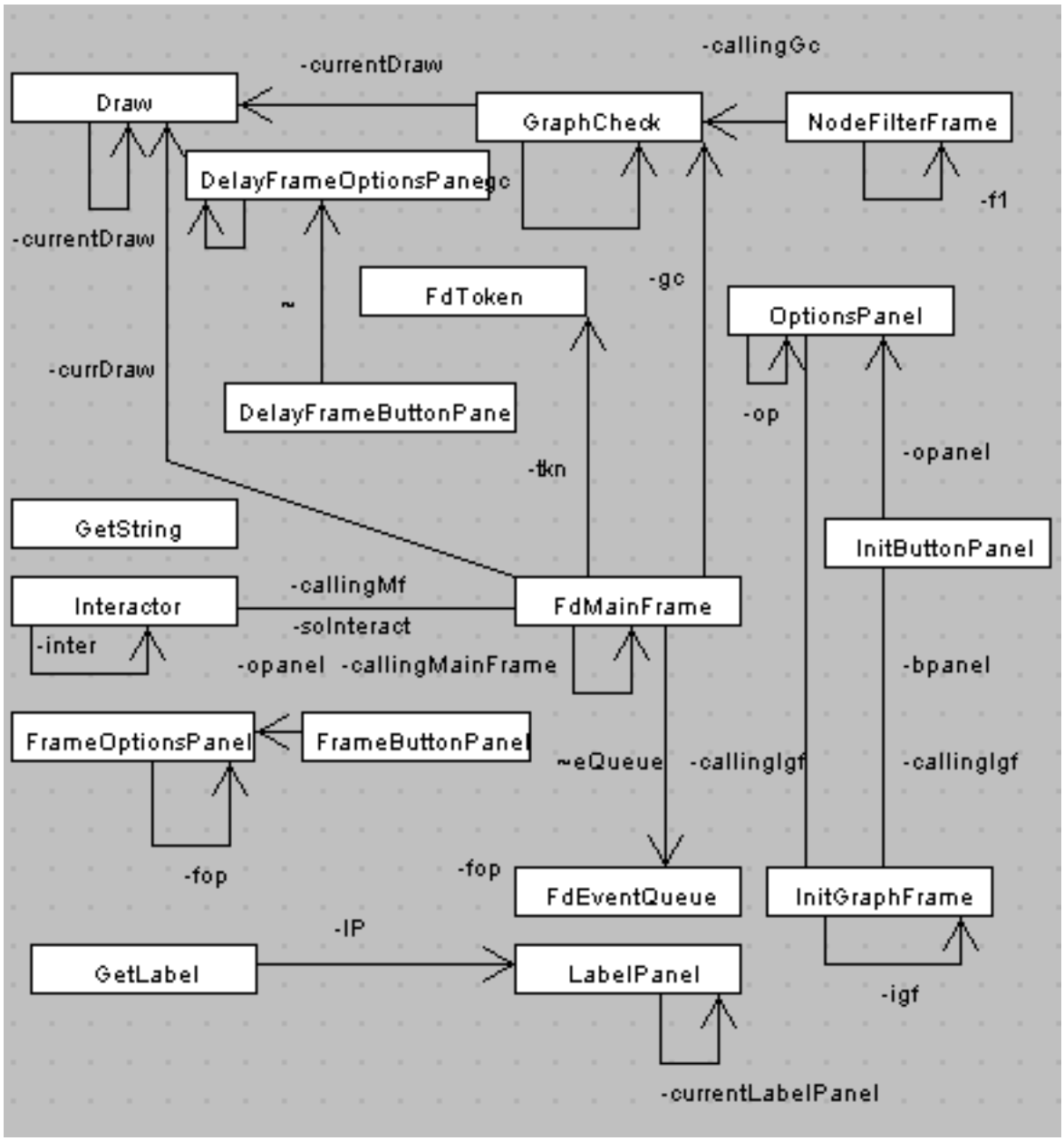

Figura A.41: *FdEdit: alcune classi, senza dettagli (1)*

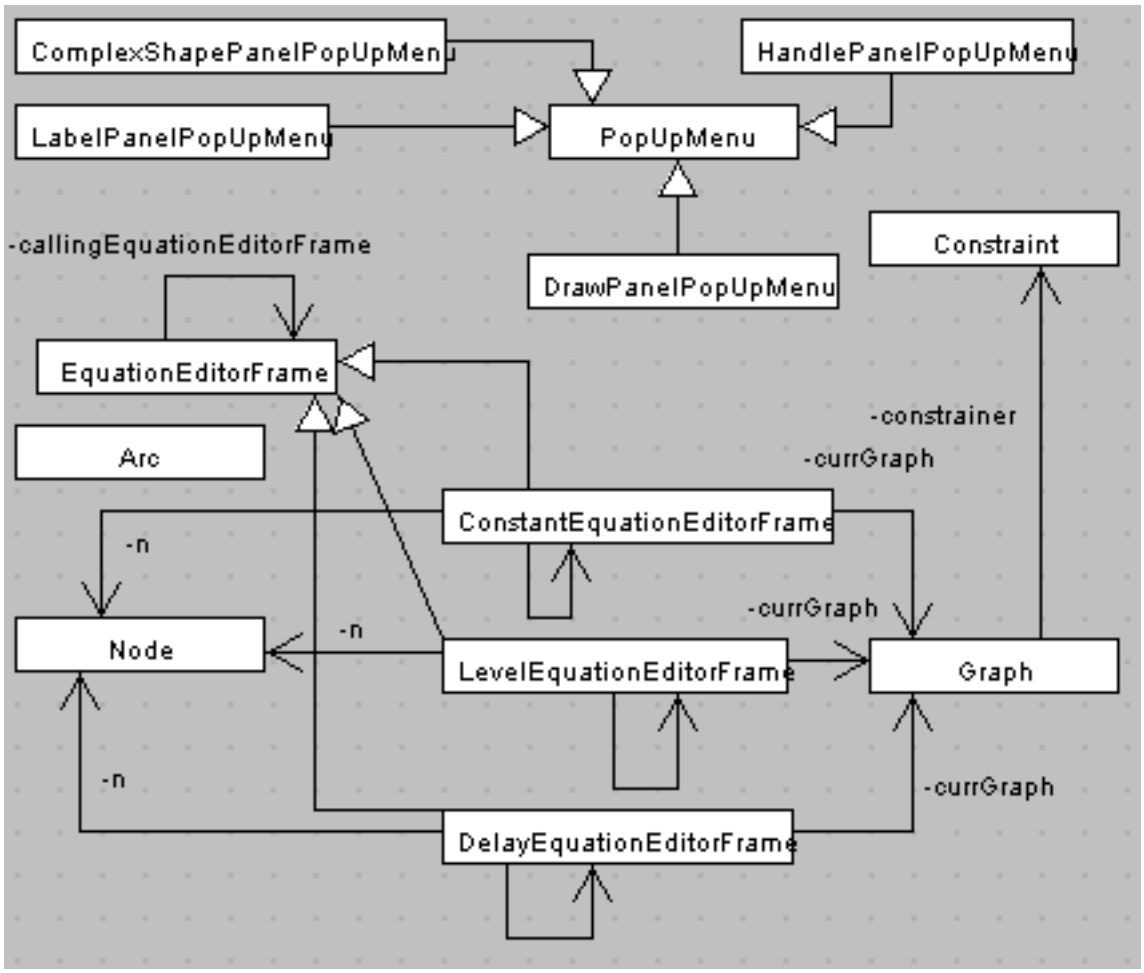

Figura A.42: *FdEdit: alcune classi, senza dettagli (2)*

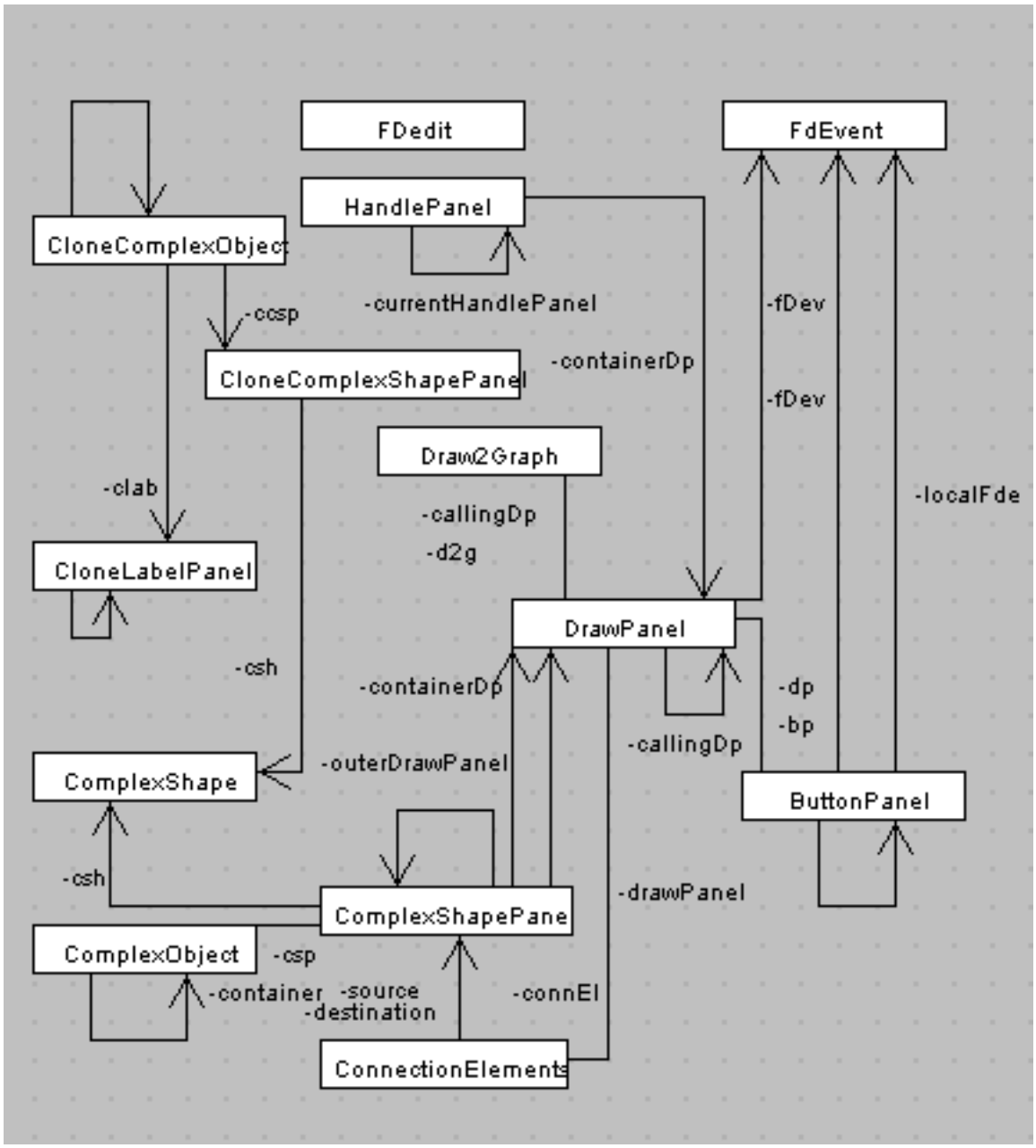

Figura A.43: *FdEdit: alcune classi, senza dettagli (3)*

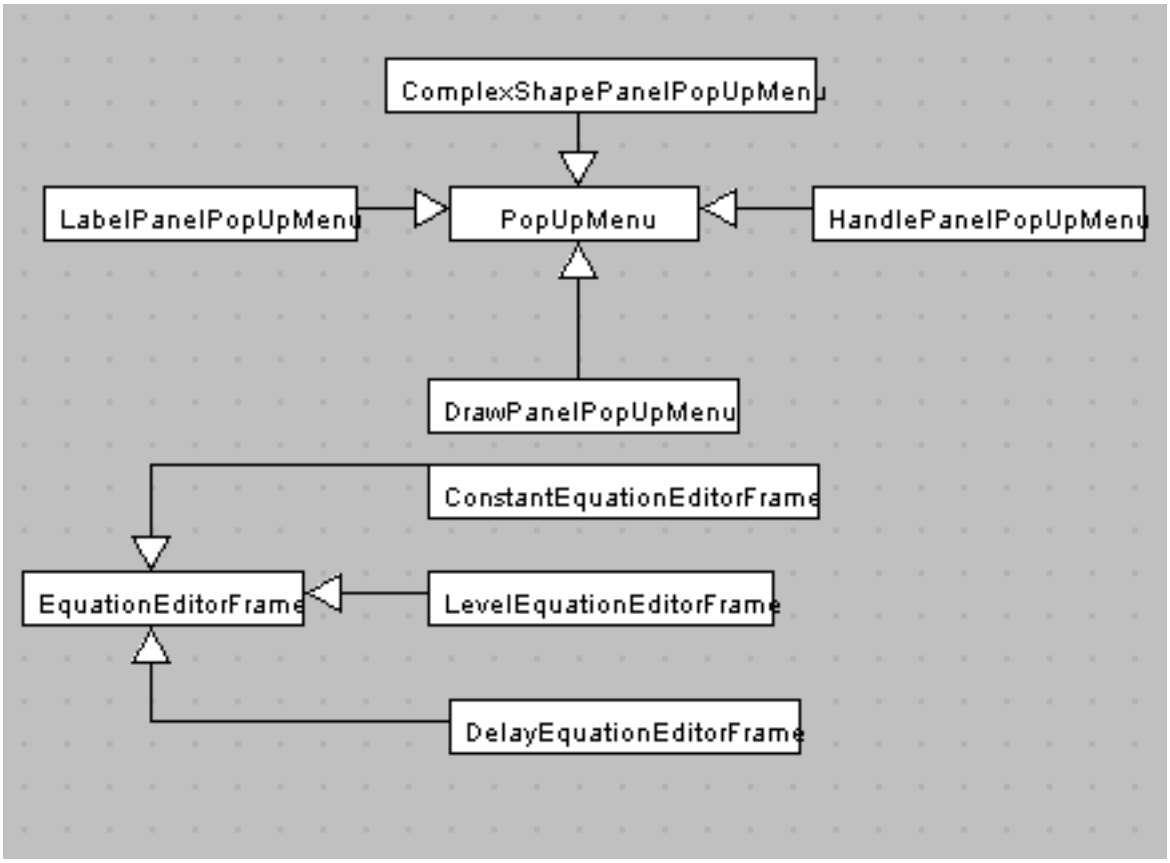

Figura A.44: *FdEdit: ereditierietà, senza dettagli*

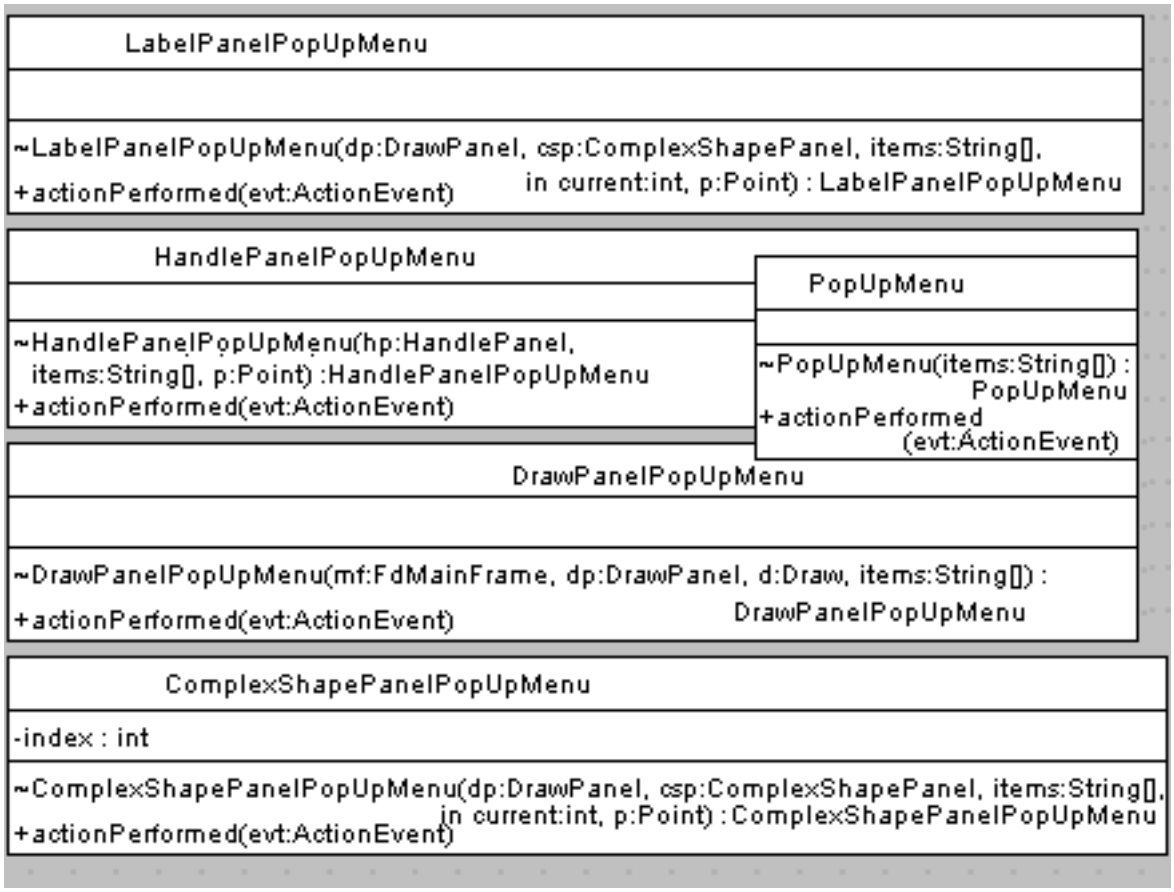

Figura A.45: *FdEdit: dettaglio delle classi (1)*

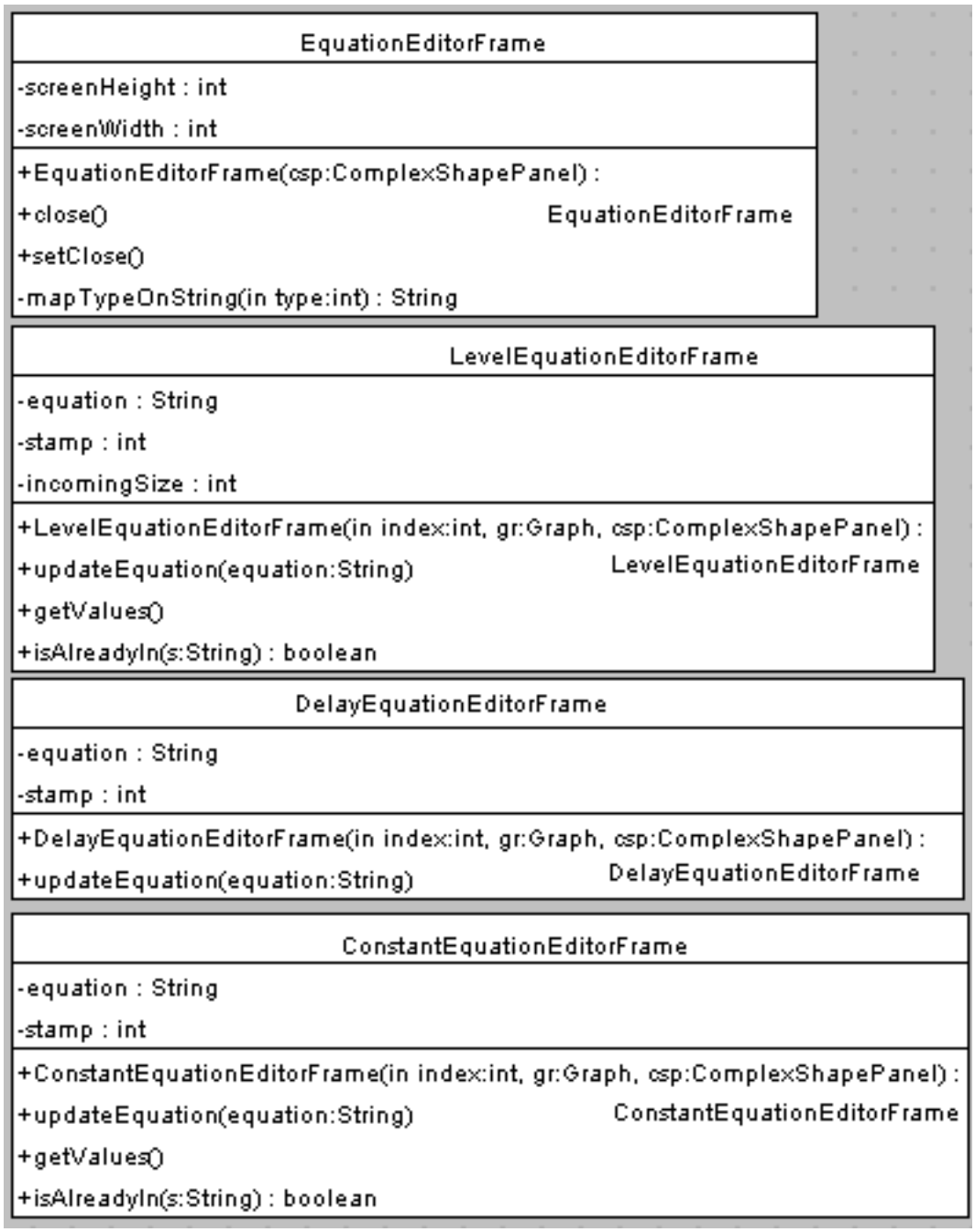

Figura A.46: *FdEdit: dettaglio delle classi (2)*

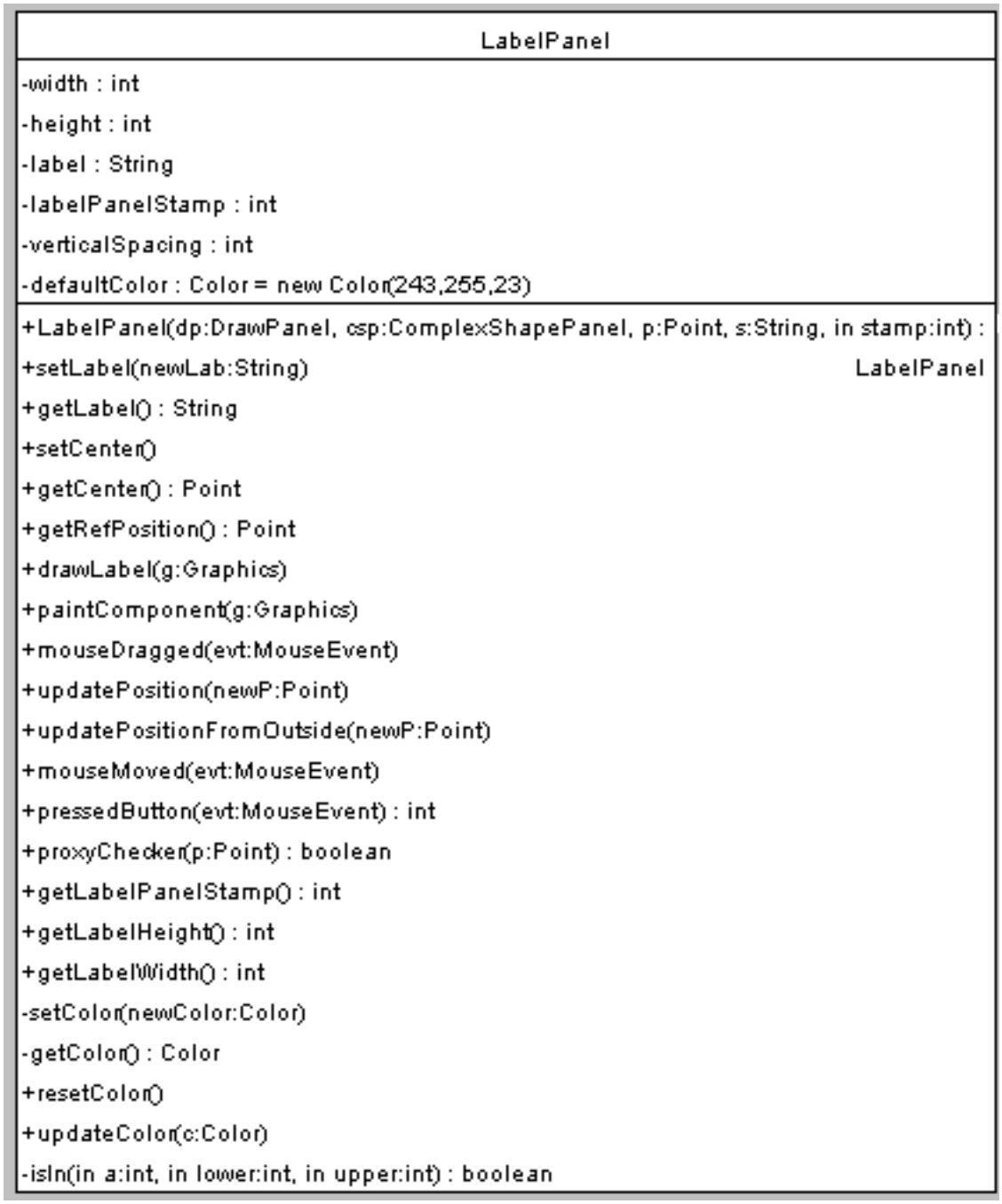

Figura A.47: *FdEdit: dettaglio delle classi (3)*

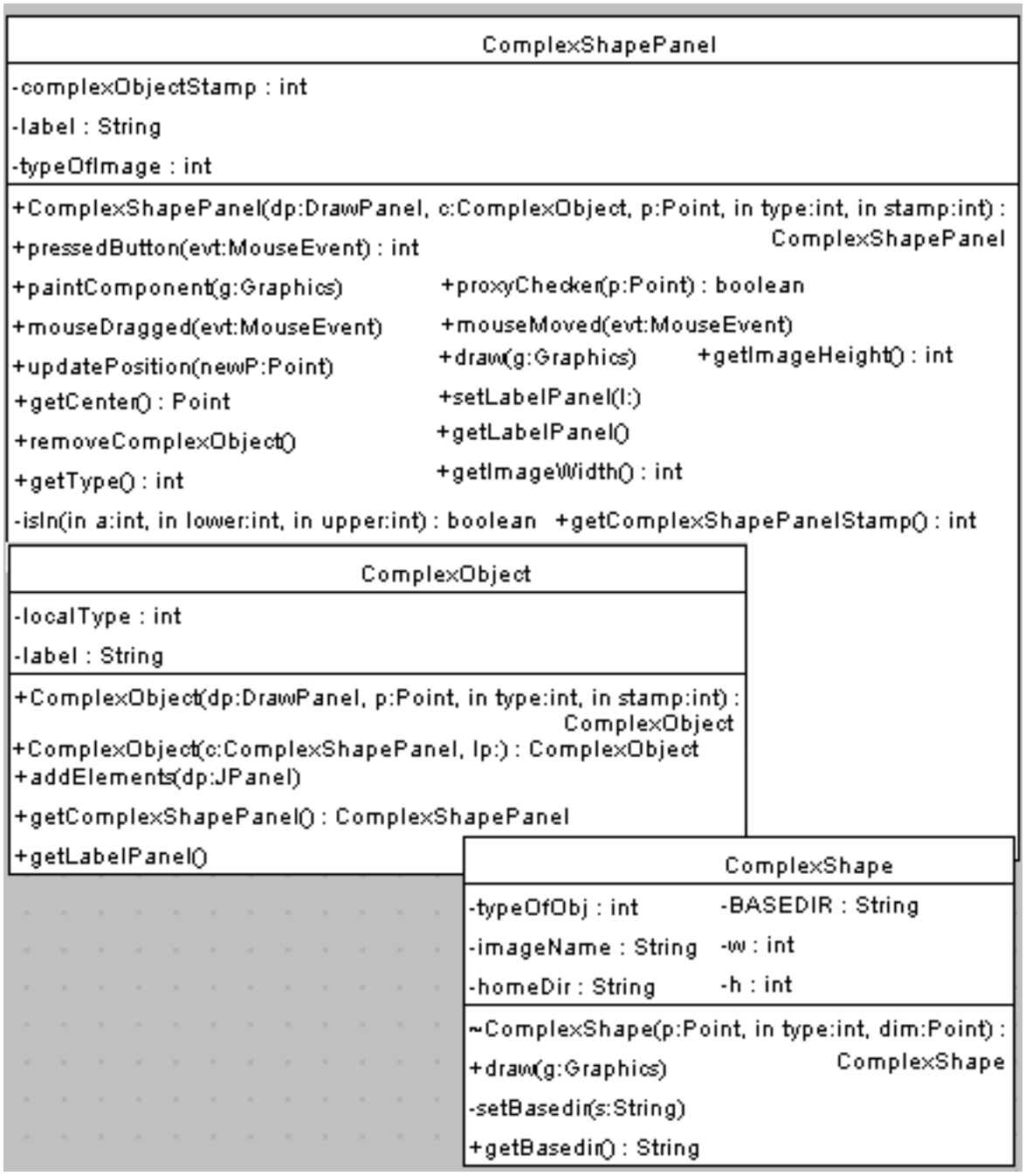

Figura A.48: *FdEdit: dettaglio delle classi (4)*

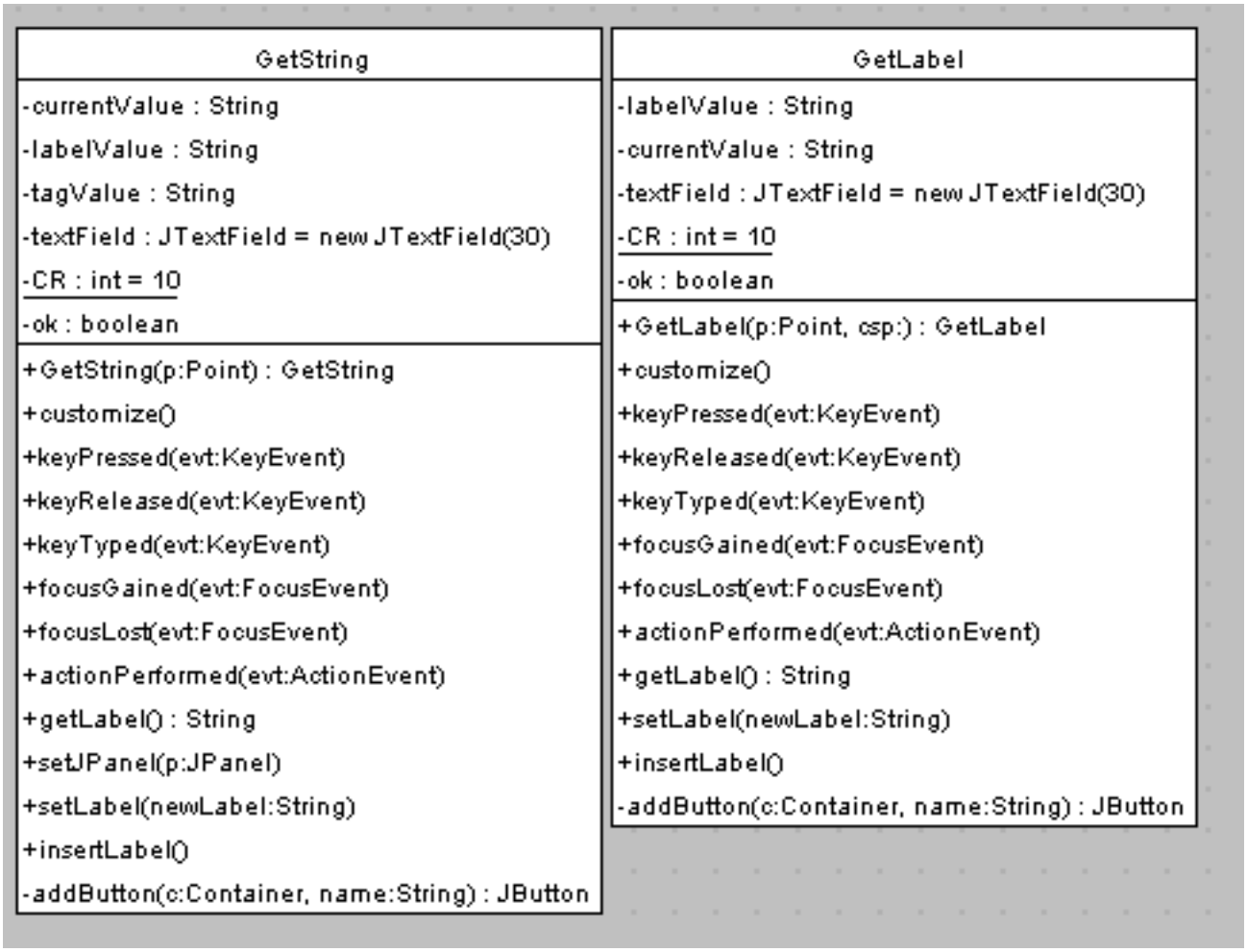

Figura A.49: *FdEdit: dettaglio delle classi (5)*

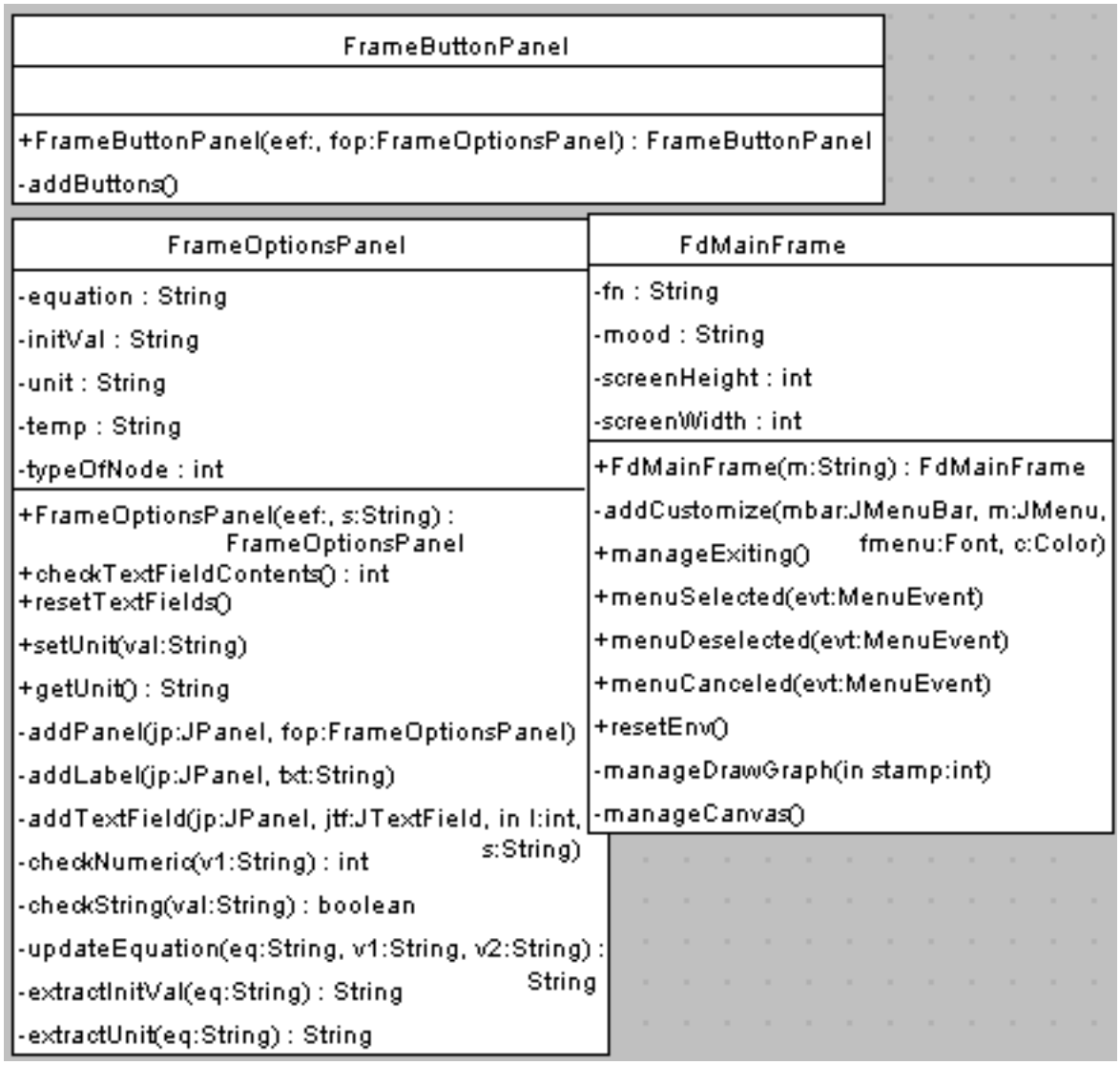

Figura A.50: *FdEdit: dettaglio delle classi (6)*

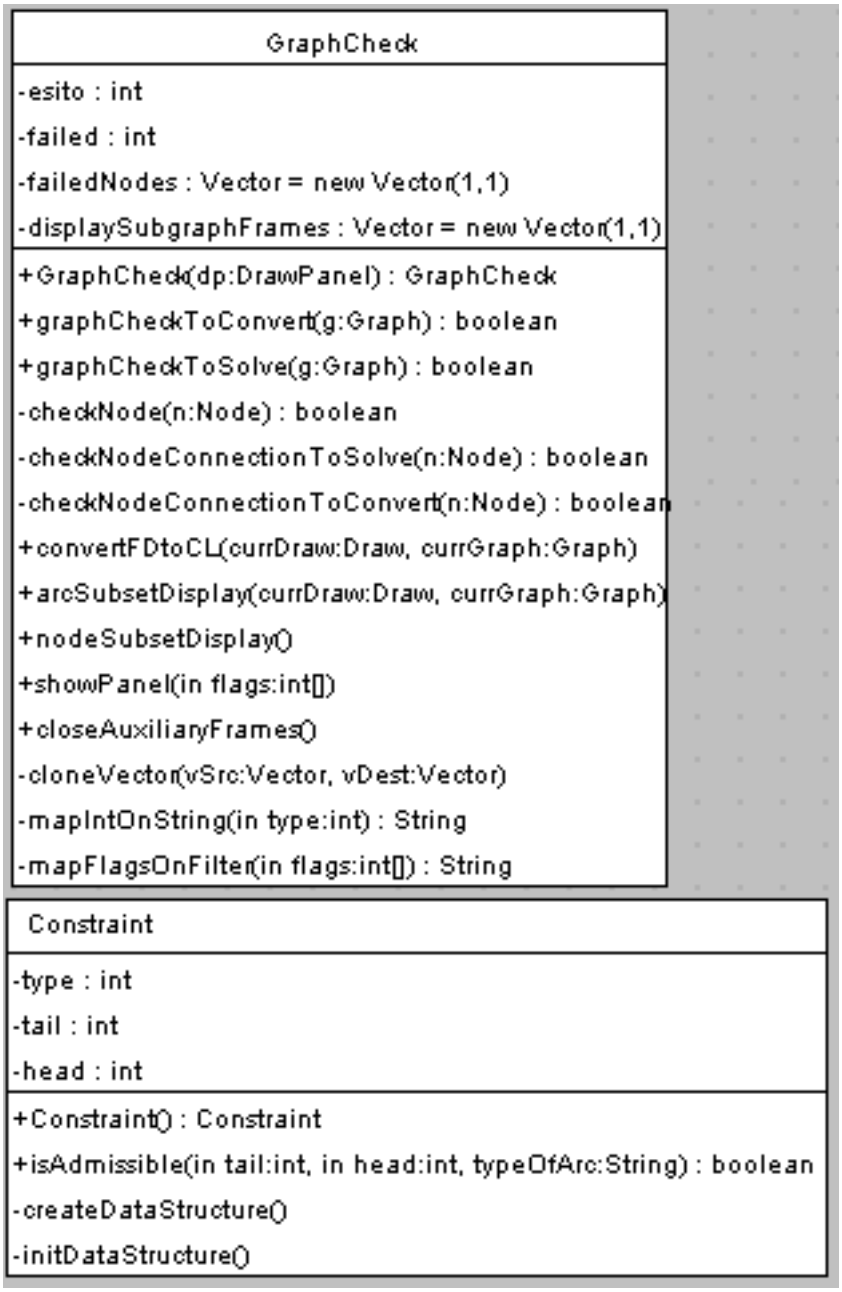

Figura A.51: *FdEdit: dettaglio delle classi (7)*

Figura A.52: *FdEdit: dettaglio delle classi (8)*

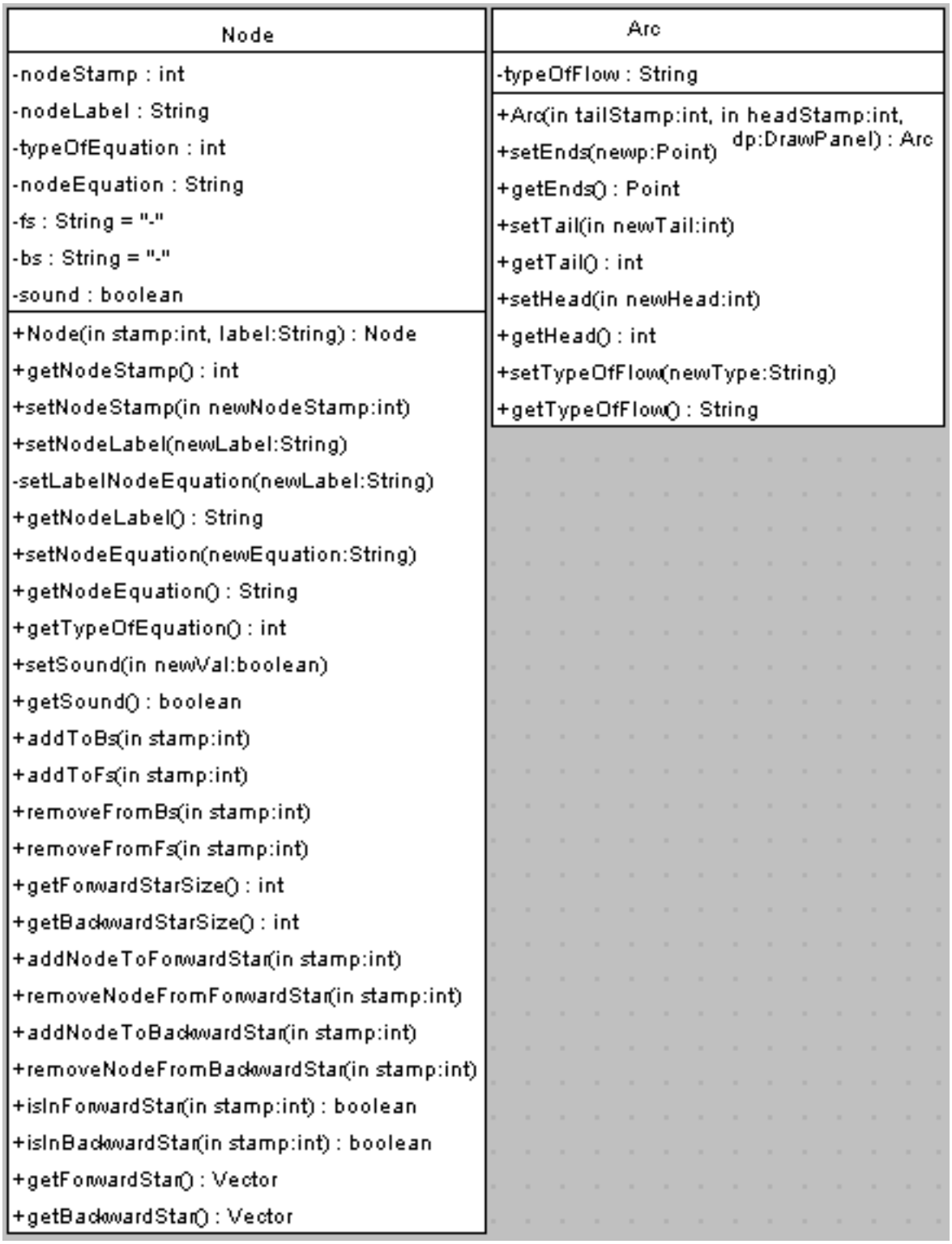

Figura A.53: *FdEdit: dettaglio delle classi (9)*

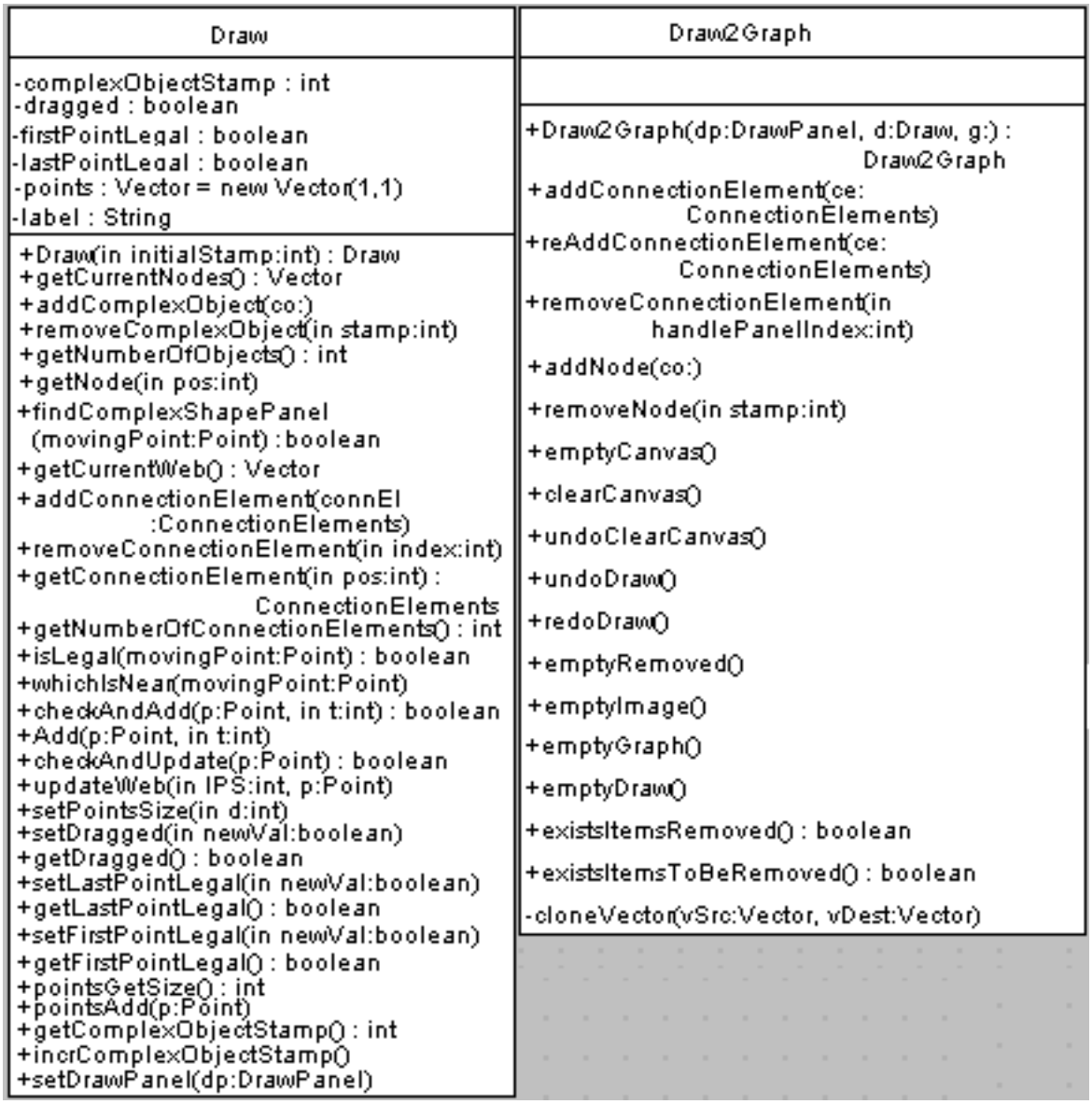

Figura A.54: *FdEdit: dettaglio delle classi (10)*

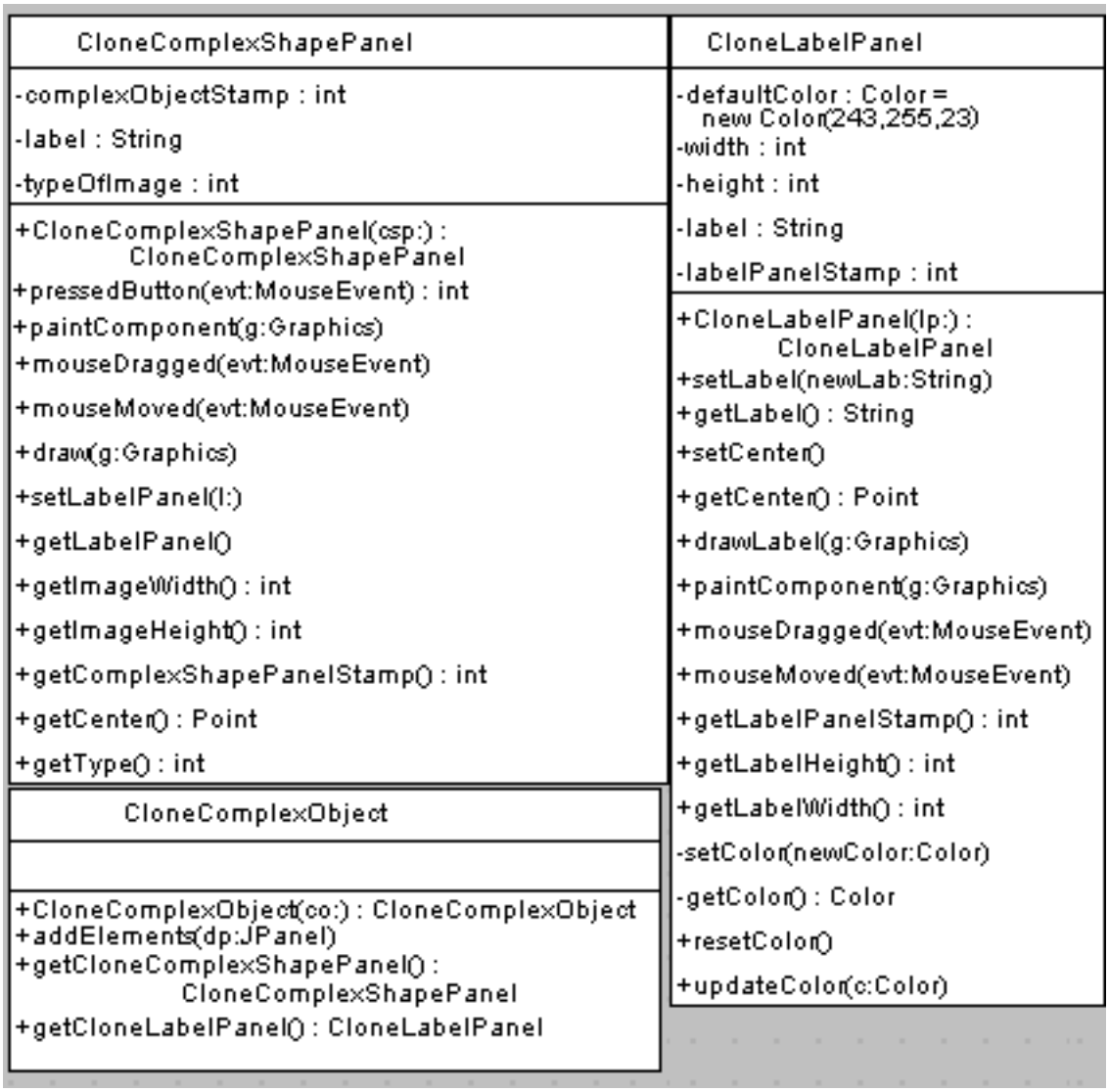

Figura A.55: *FdEdit: dettaglio delle classi (11)*

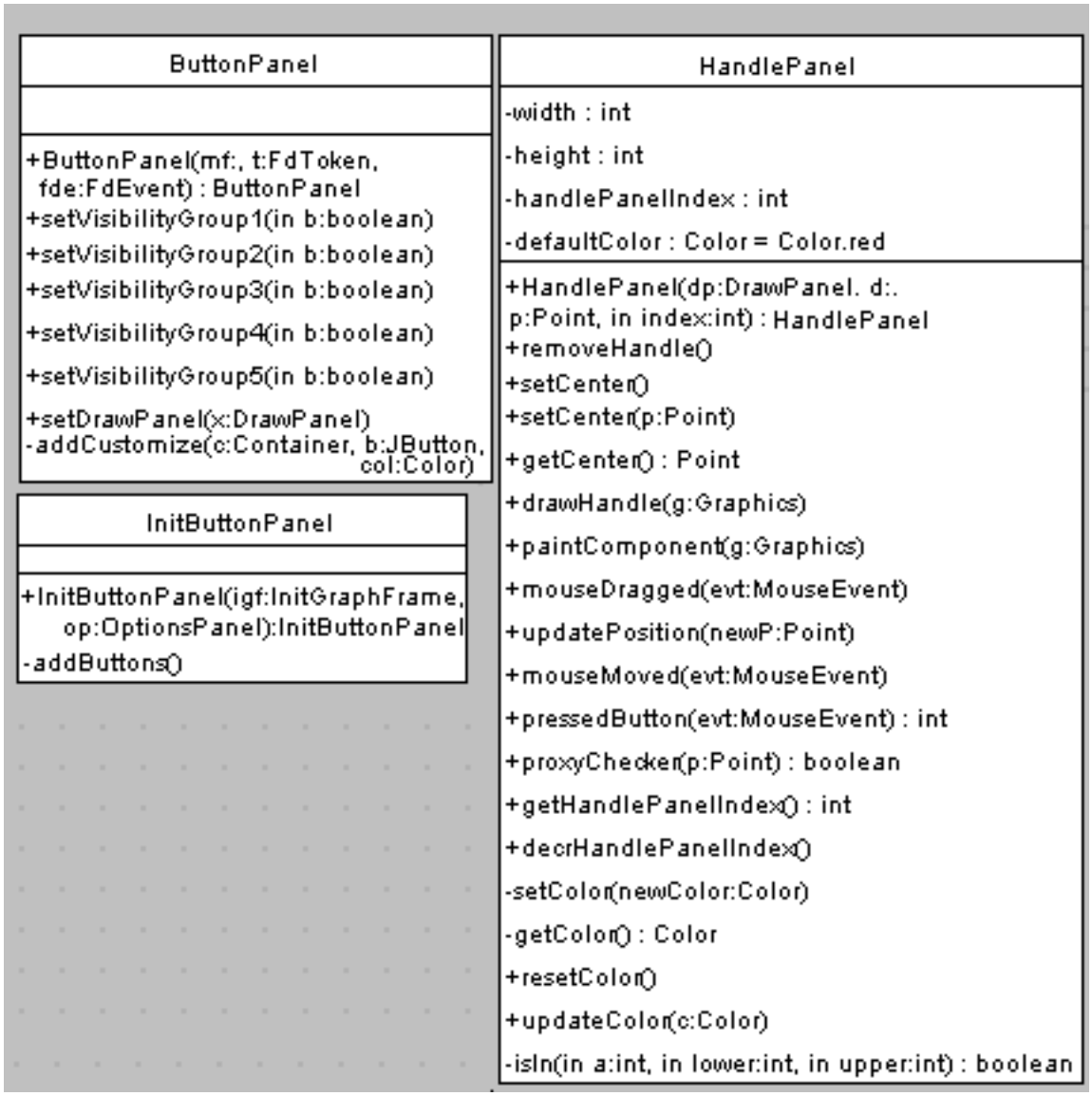

Figura A.56: *FdEdit: dettaglio delle classi (12)*

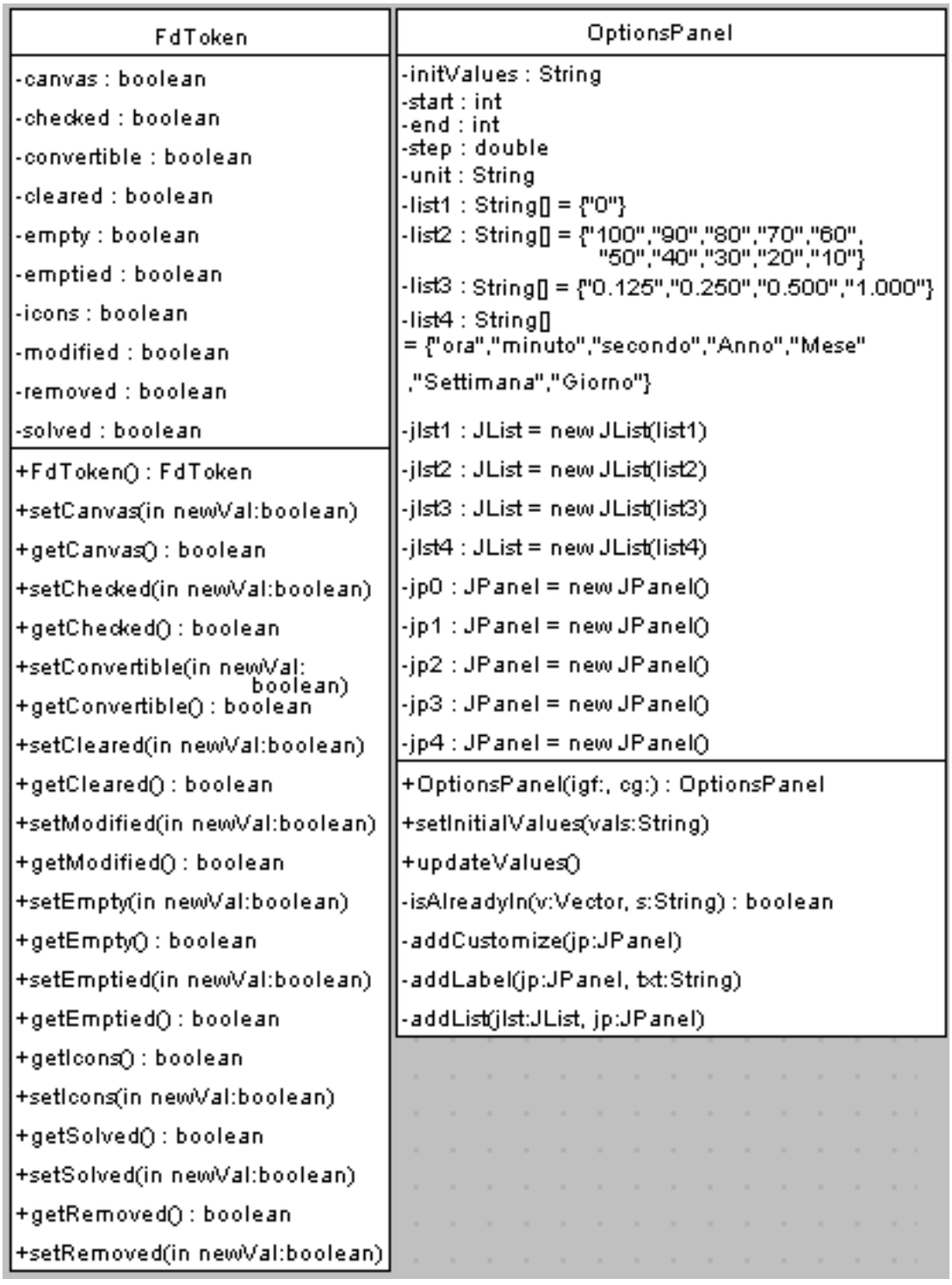

Figura A.57: *FdEdit: dettaglio delle classi (13)*

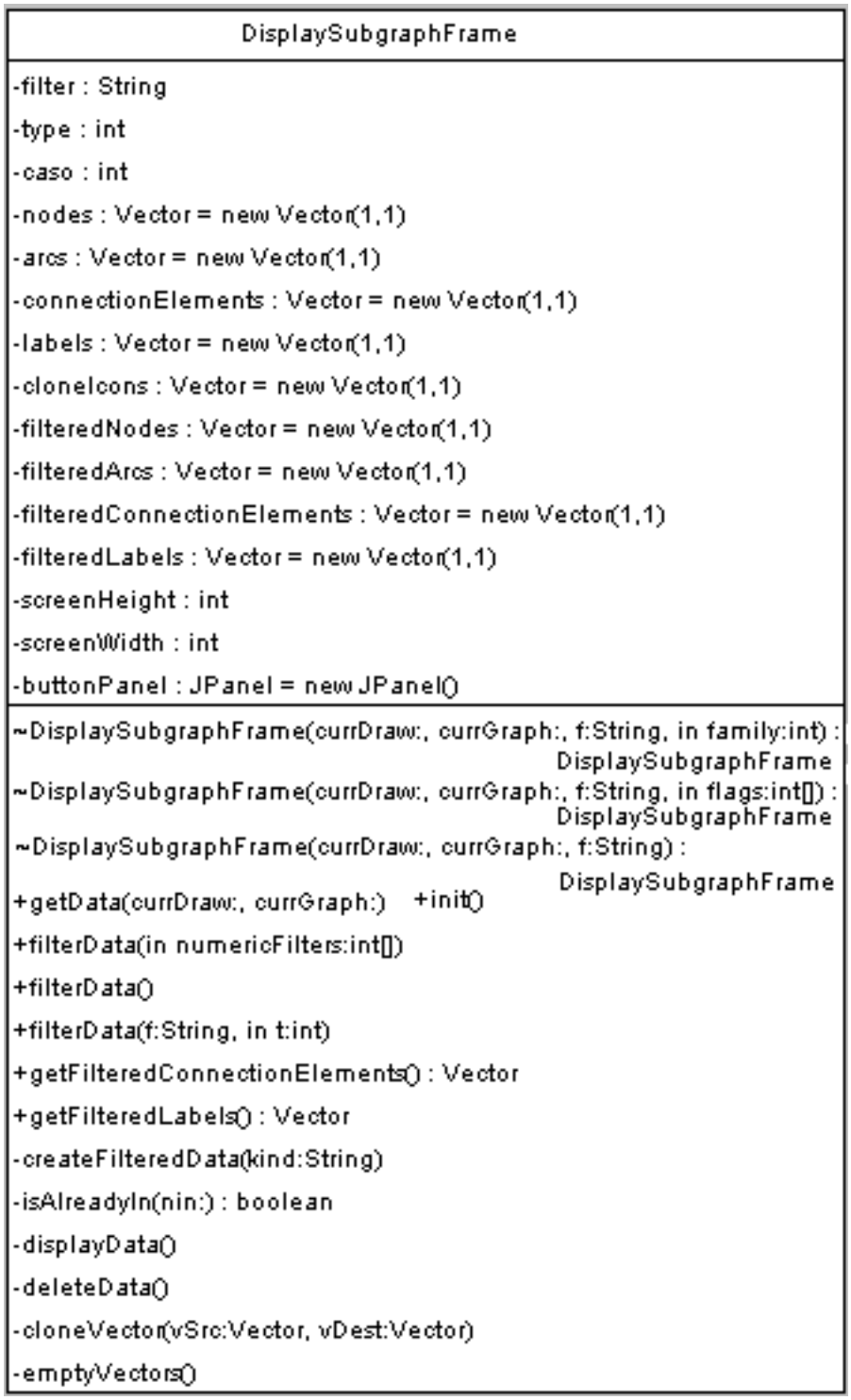

Figura A.58: *FdEdit: dettaglio delle classi (14)*

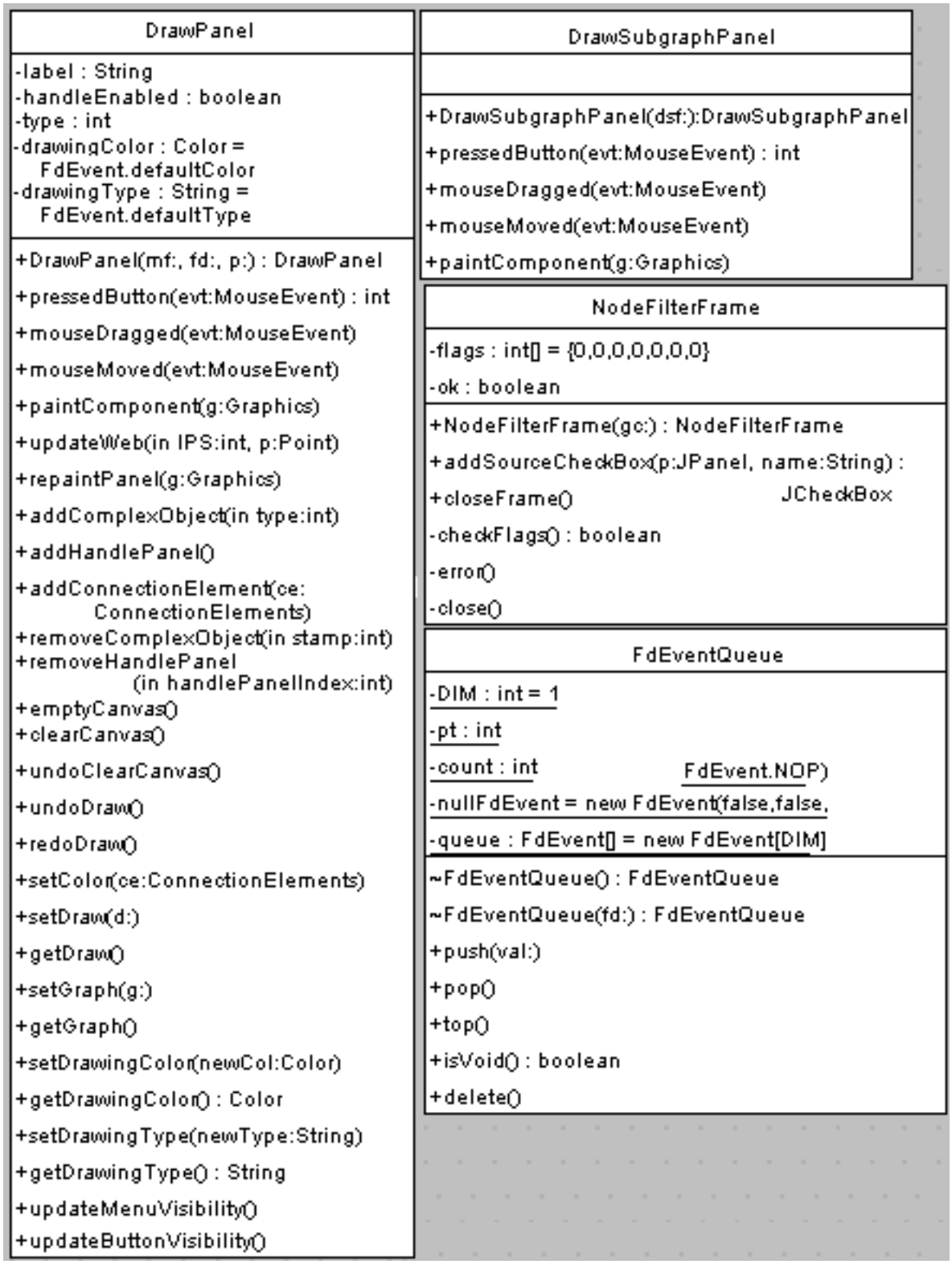

Figura A.59: *FdEdit: dettaglio delle classi (15)*

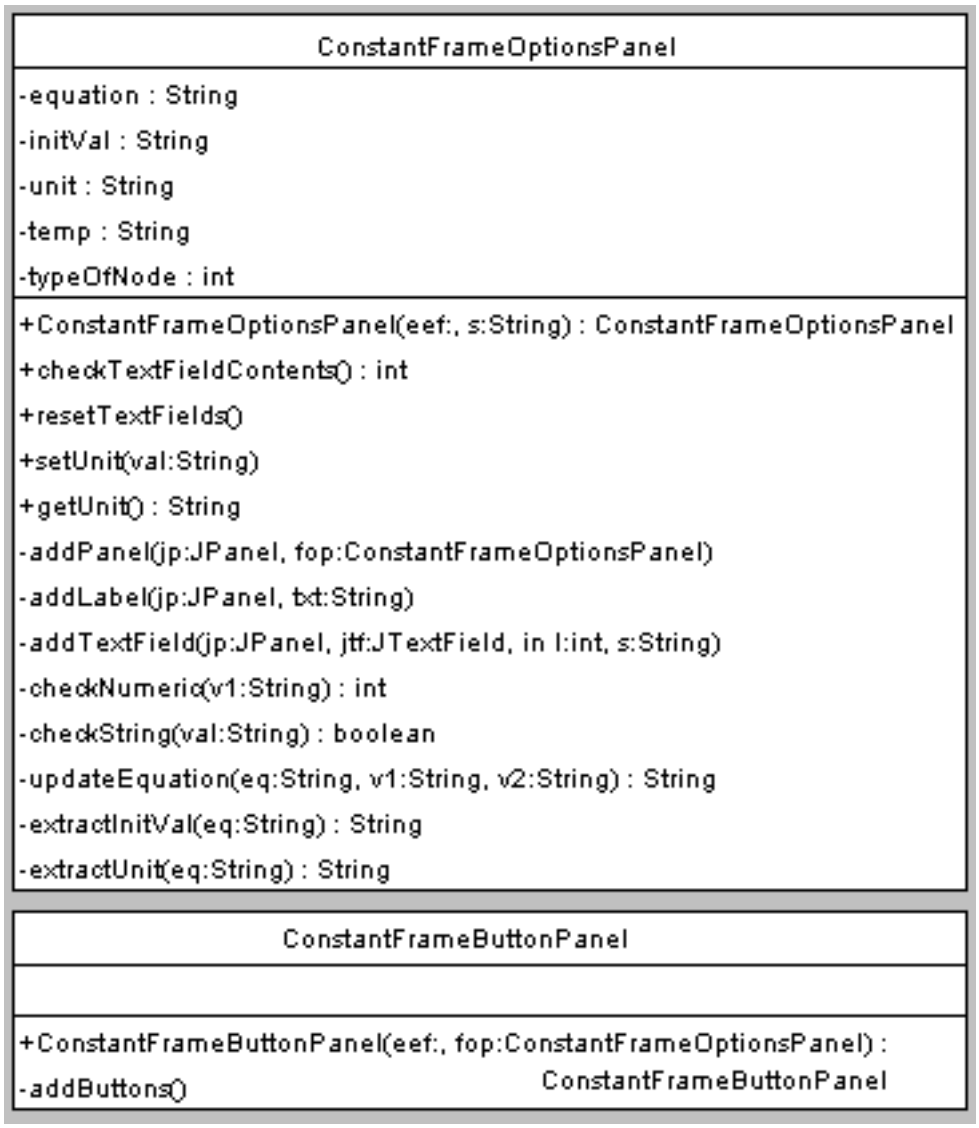

Figura A.60: *FdEdit: dettaglio delle classi (16)*

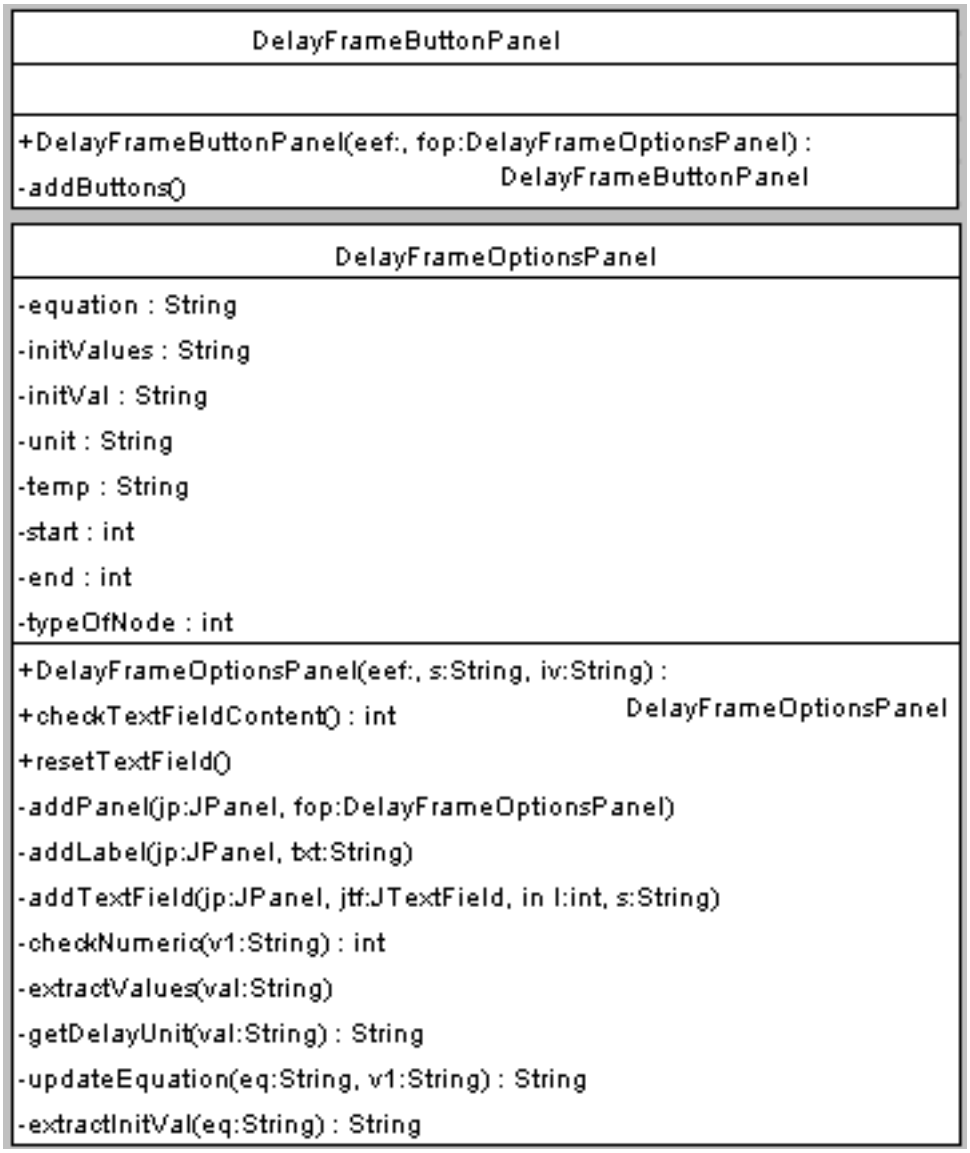

Figura A.61: *FdEdit: dettaglio delle classi (17)*

A.2.4 *Display*

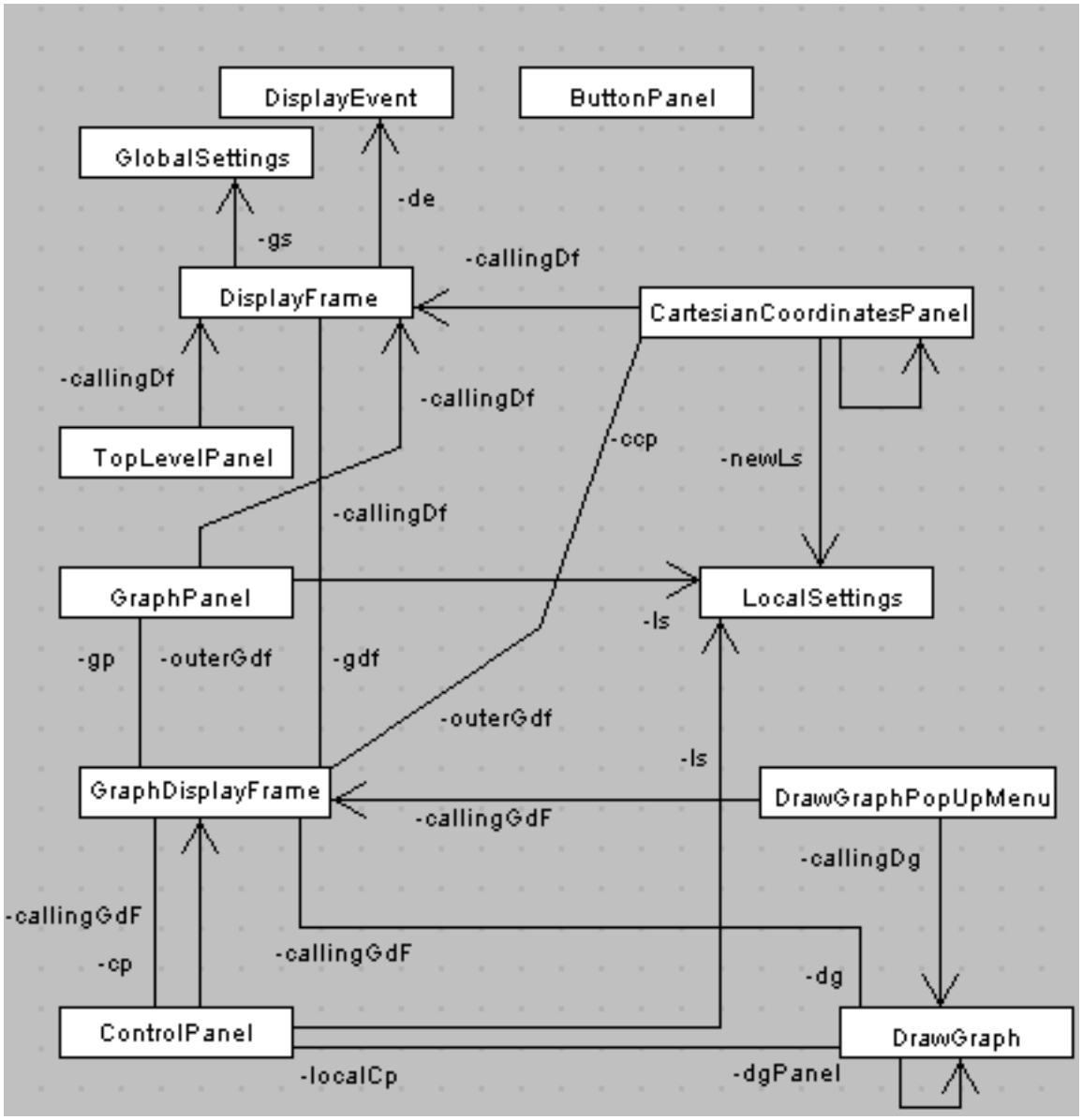

Figura A.62: *Display: le classi principali*

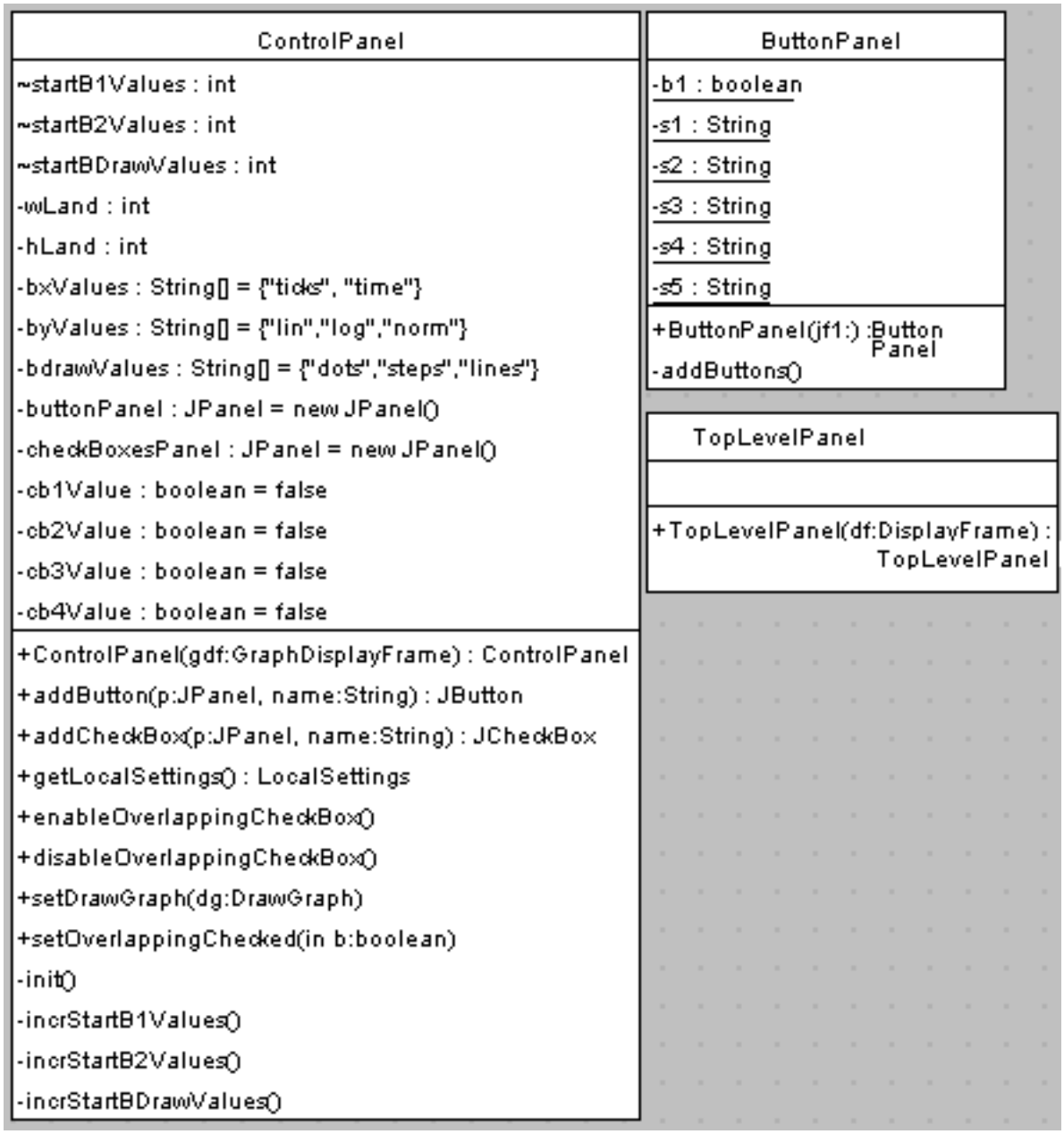

Figura A.63: *Display: dettaglio delle classi (1)*

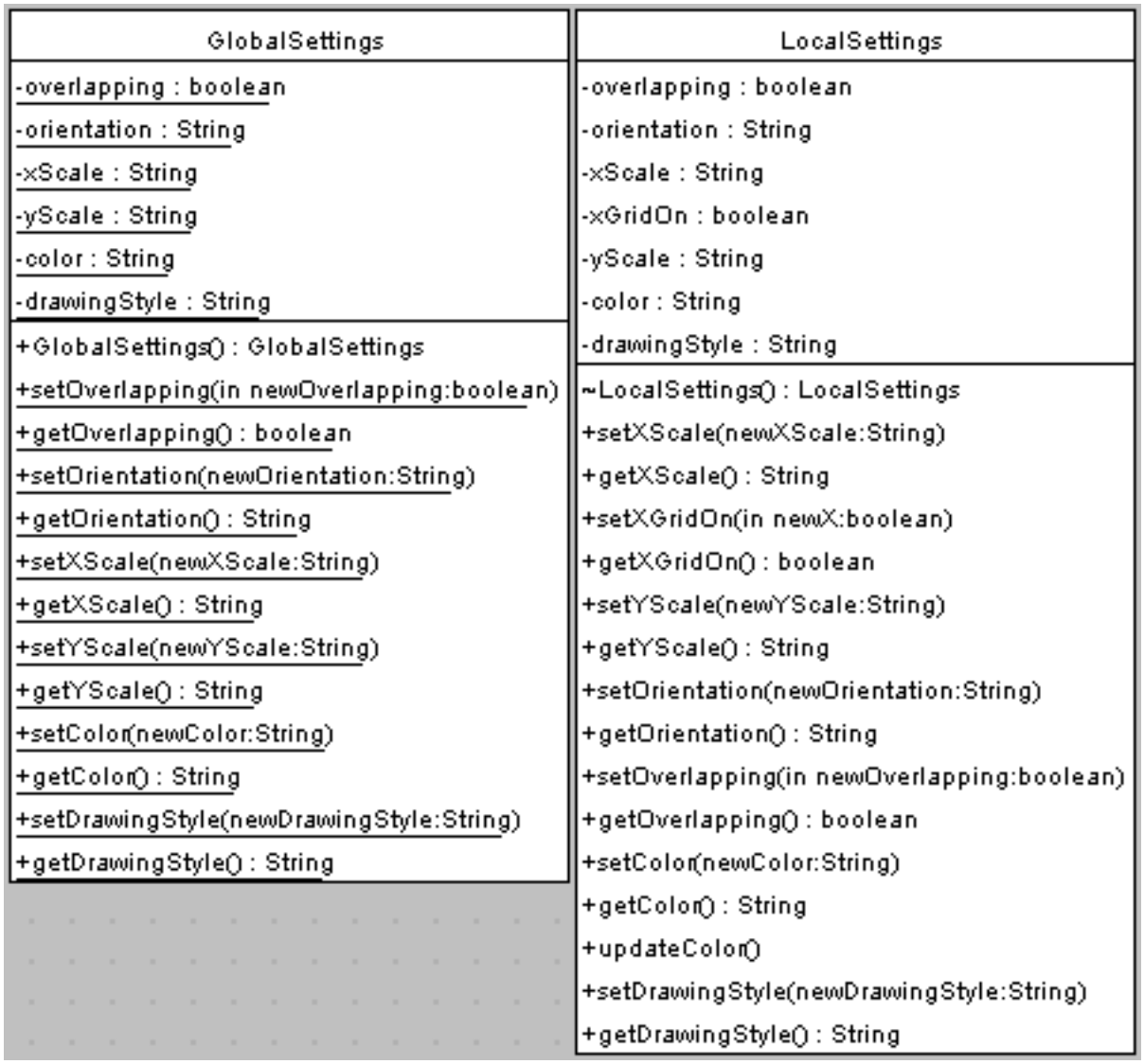

Figura A.64: *Display: dettaglio delle classi (2)*

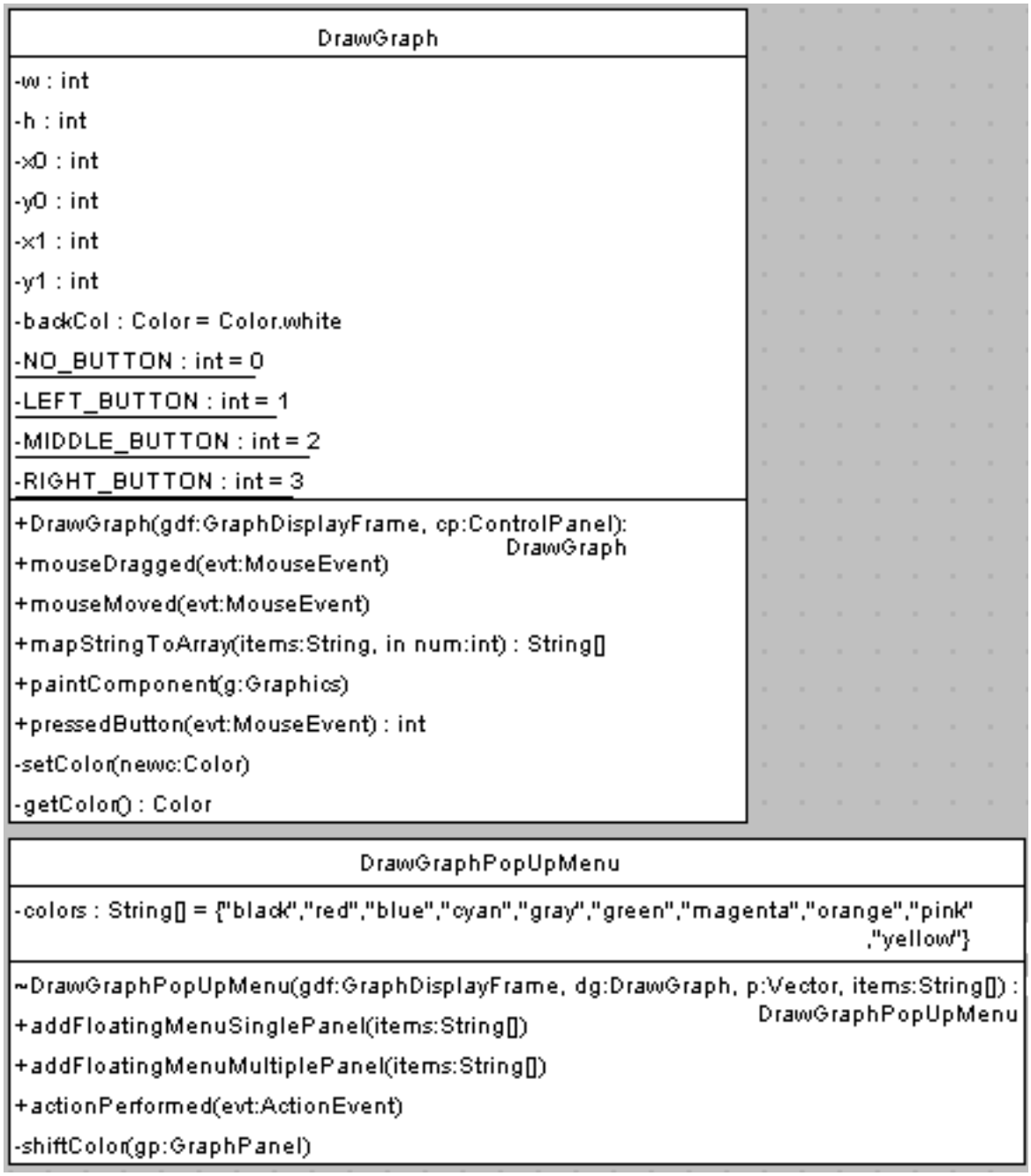

Figura A.65: *Display: dettaglio delle classi (3)*

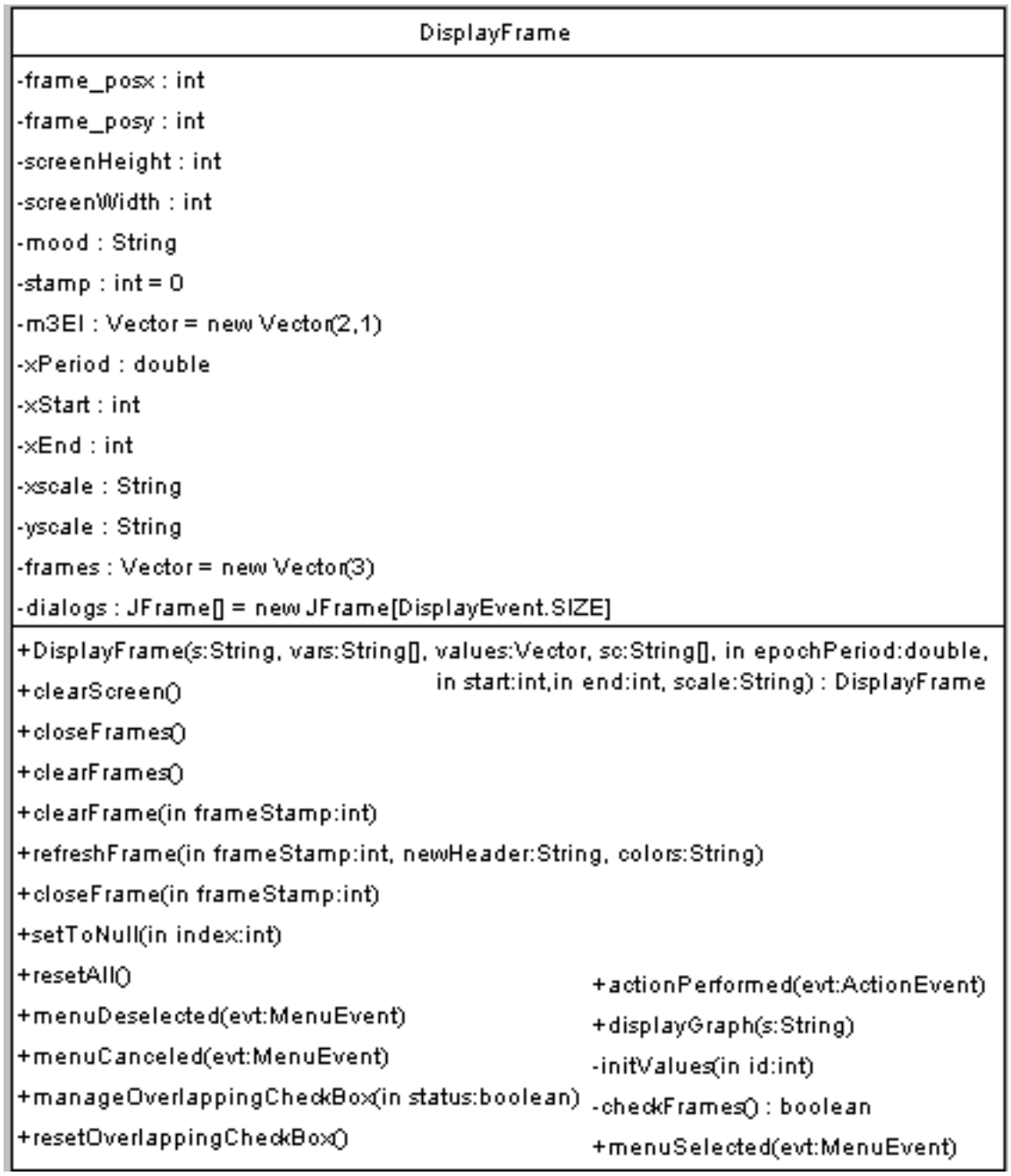

Figura A.66: *Display: dettaglio delle classi (4)*

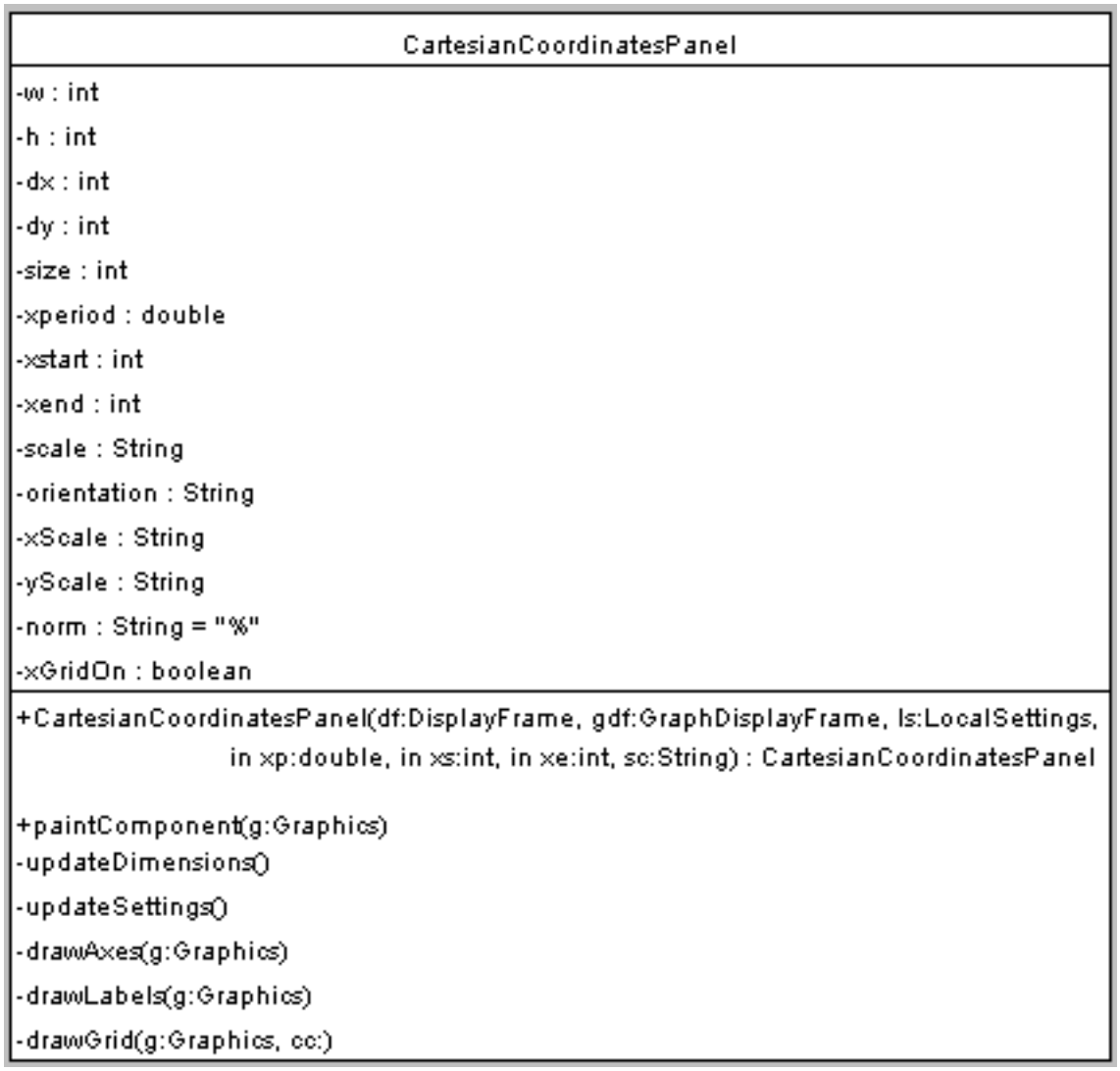

Figura A.67: *Display: dettaglio delle classi (5)*

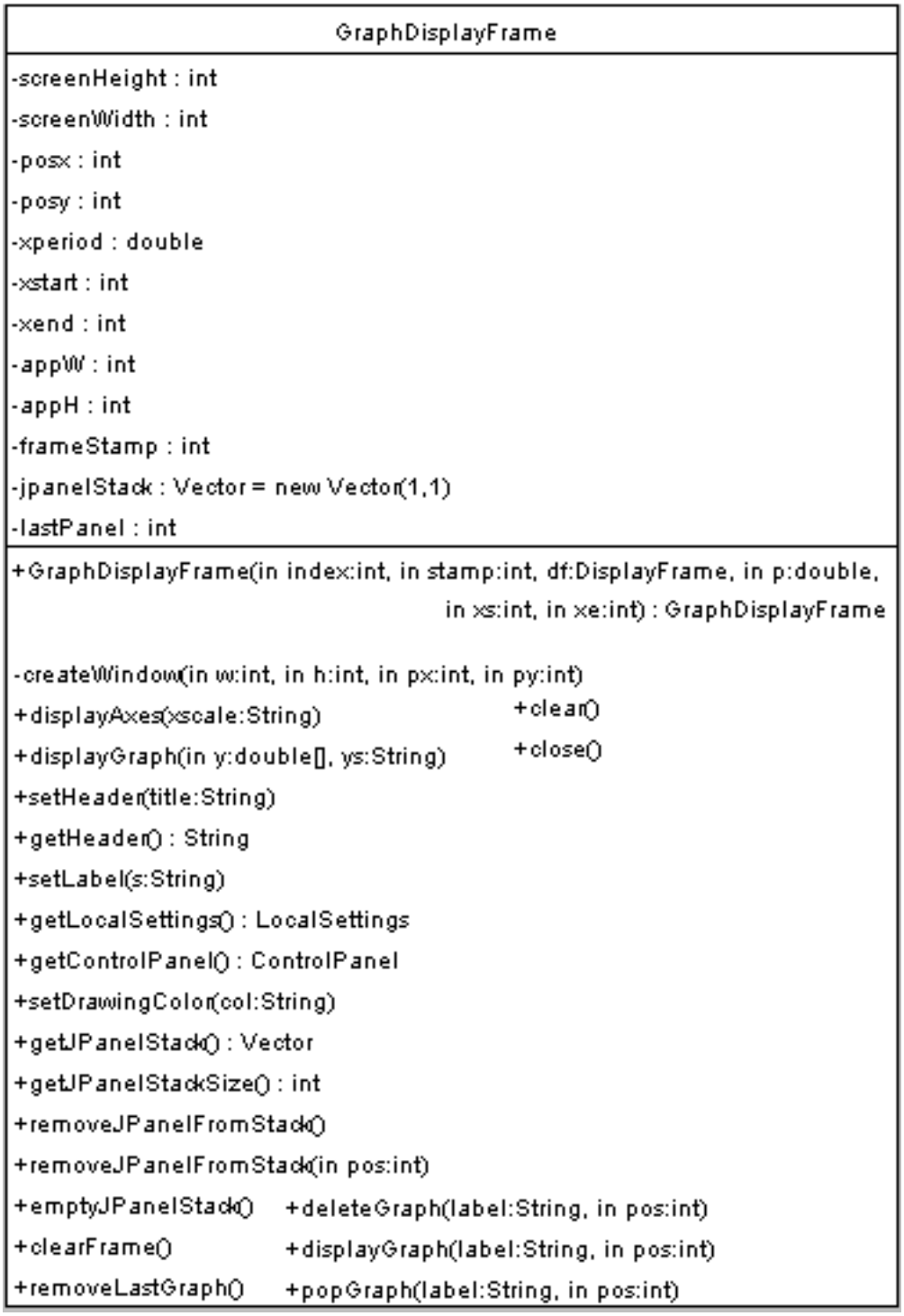

Figura A.68: *Display: dettaglio delle classi (6)*

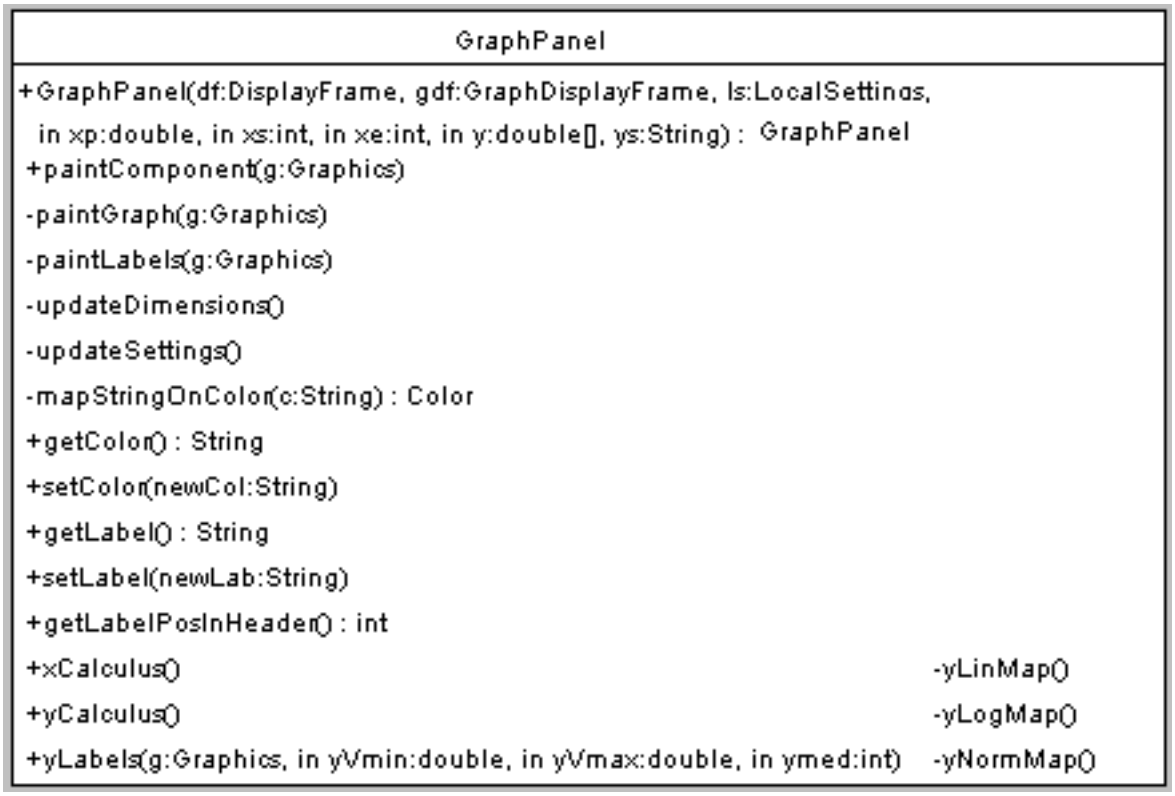

Figura A.69: *Display: dettaglio delle classi (7)*

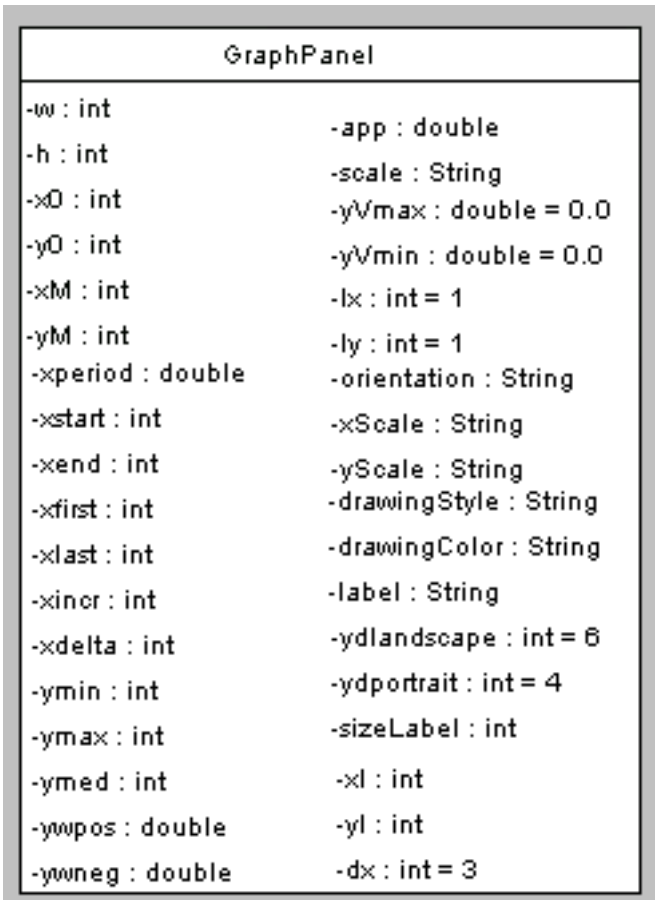

Figura A.70: *Display: dettaglio delle classi (8)*

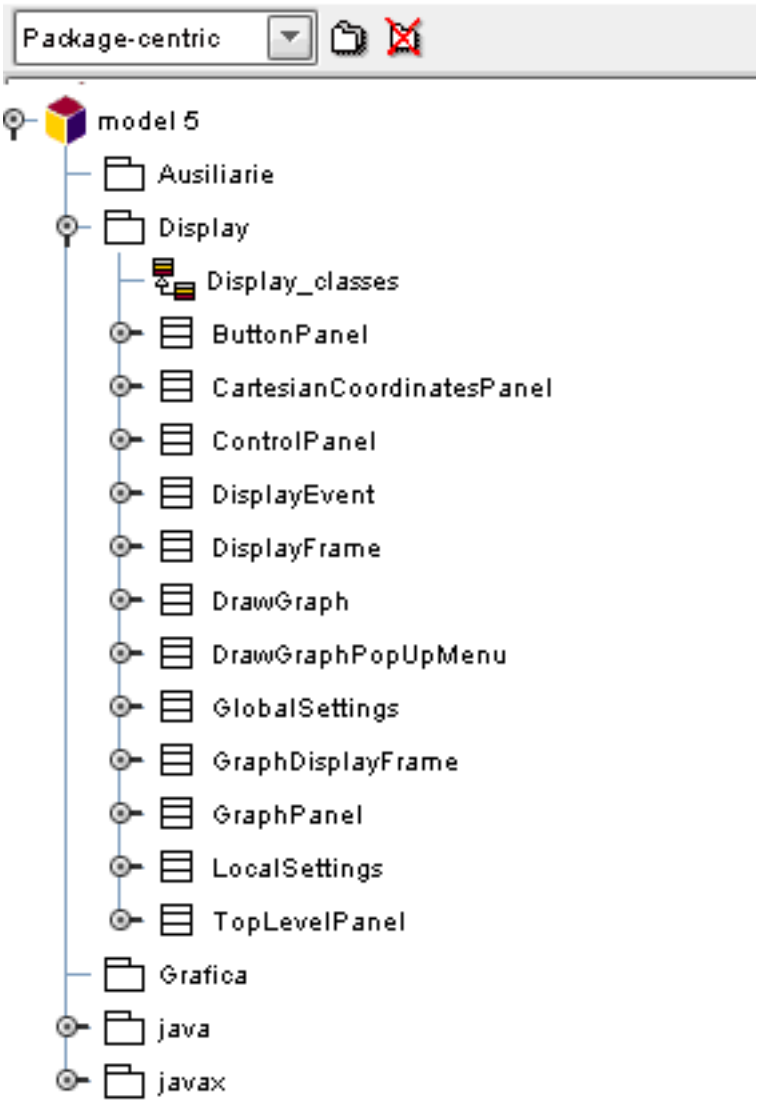

Figura A.71: *Display: "package centric"*

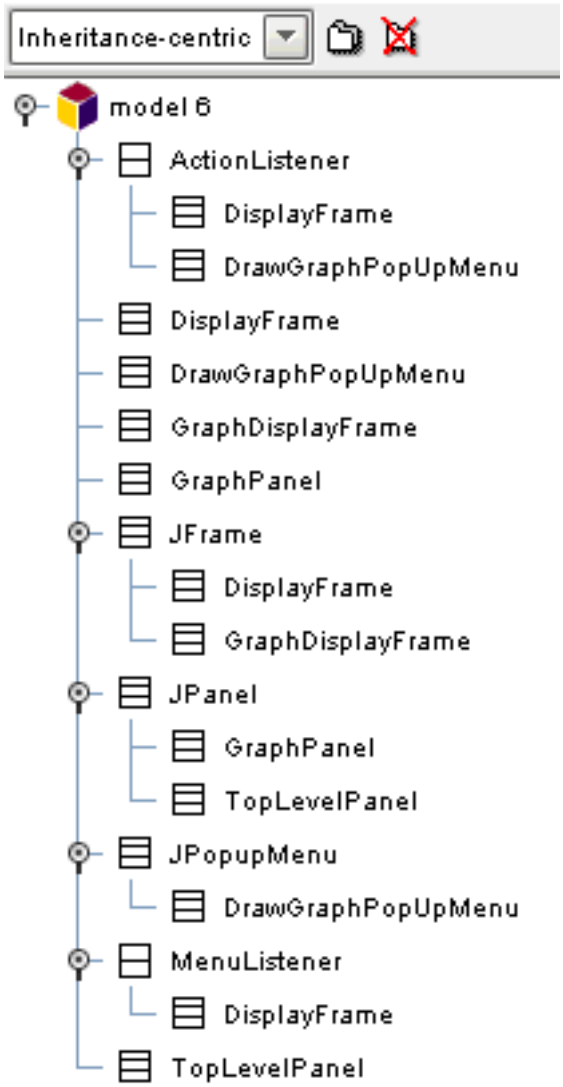

Figura A.72: *Display: "inheritance centric"*

A.2.5 *Equation Solver*

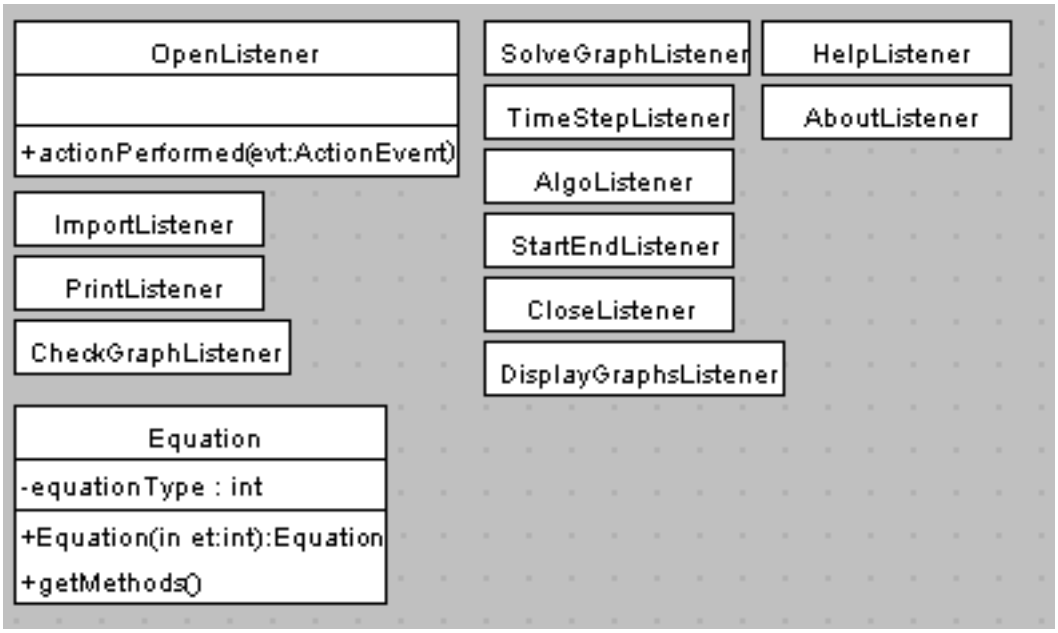

Figura A.73: *Equation Solver: i listener e alcune classi*

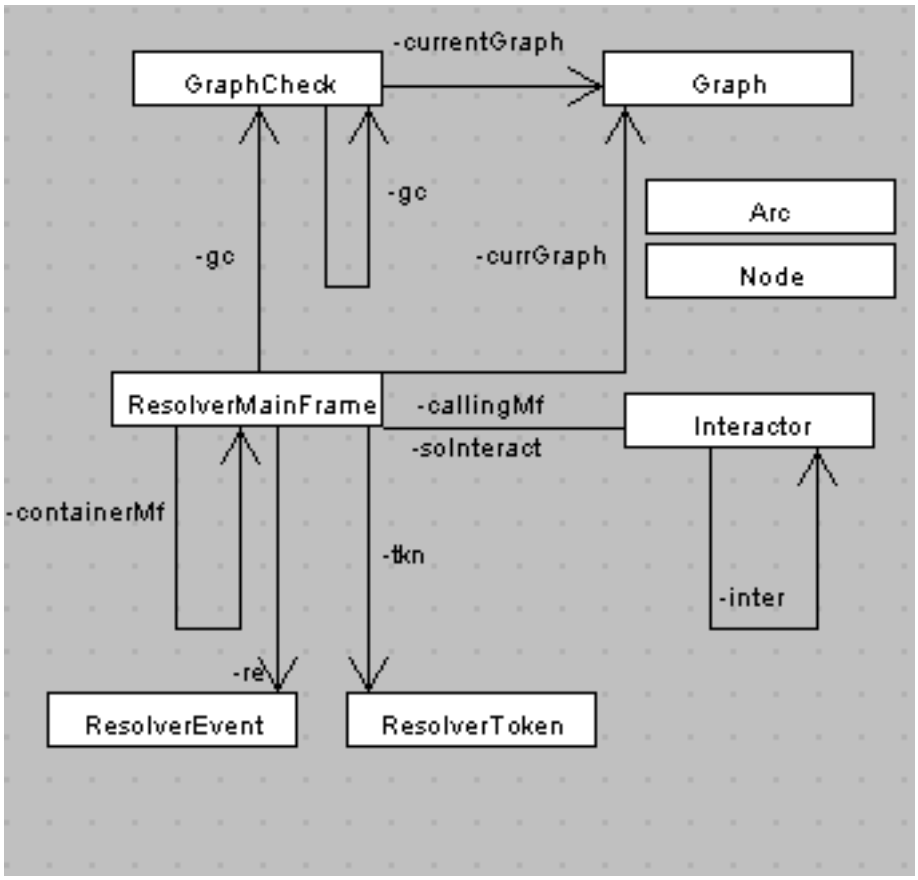

Figura A.74: *Equation Solver: le classi principali*

Figura A.75: *Equation Solver: classi e associazioni*

A.2.6 *I convertitori da CL a FD e viceversa*

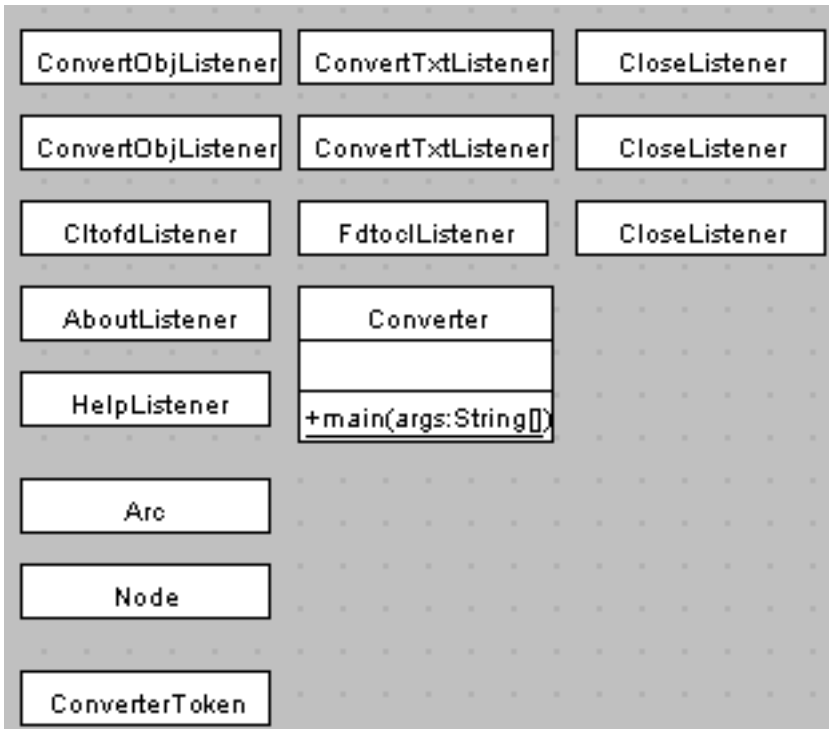

Figura A.76: *I convertitori: i listener e alcune classi*

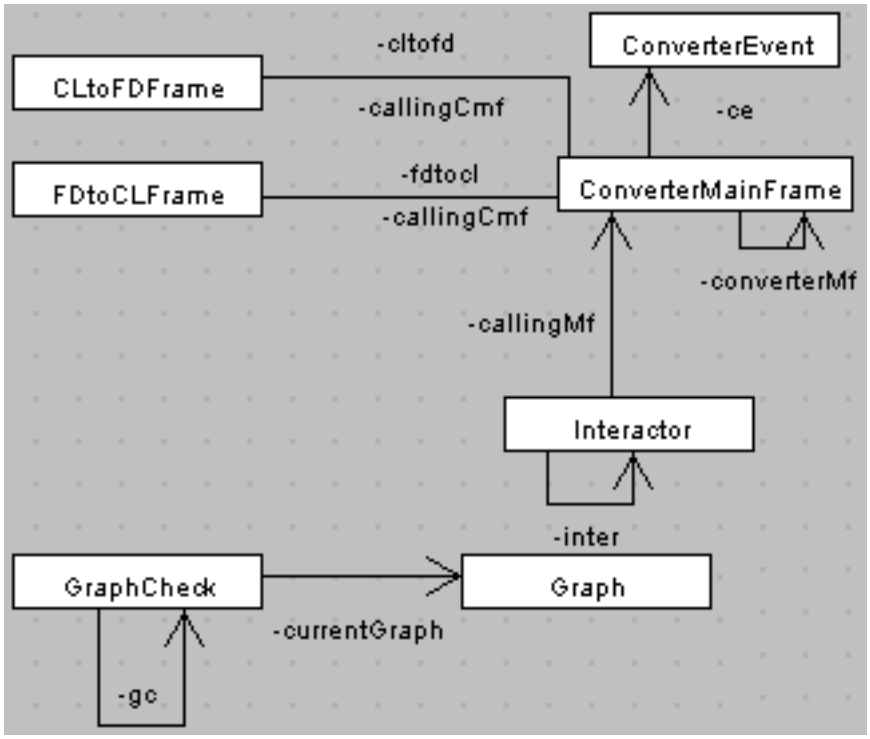

Figura A.77: *I convertitori: le classi principali*

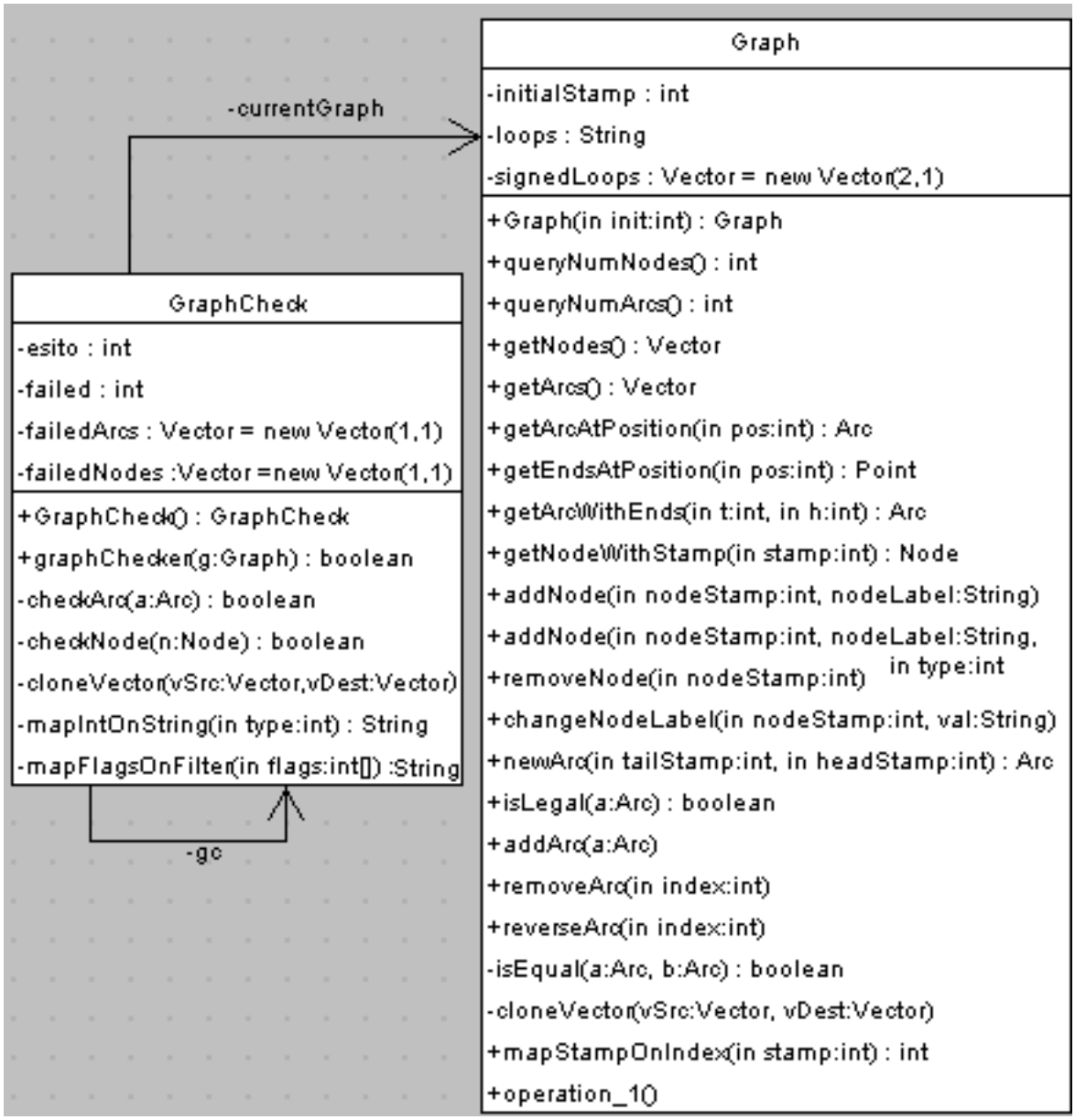

Figura A.78: *I convertitori: dettaglio delle classi (1)*

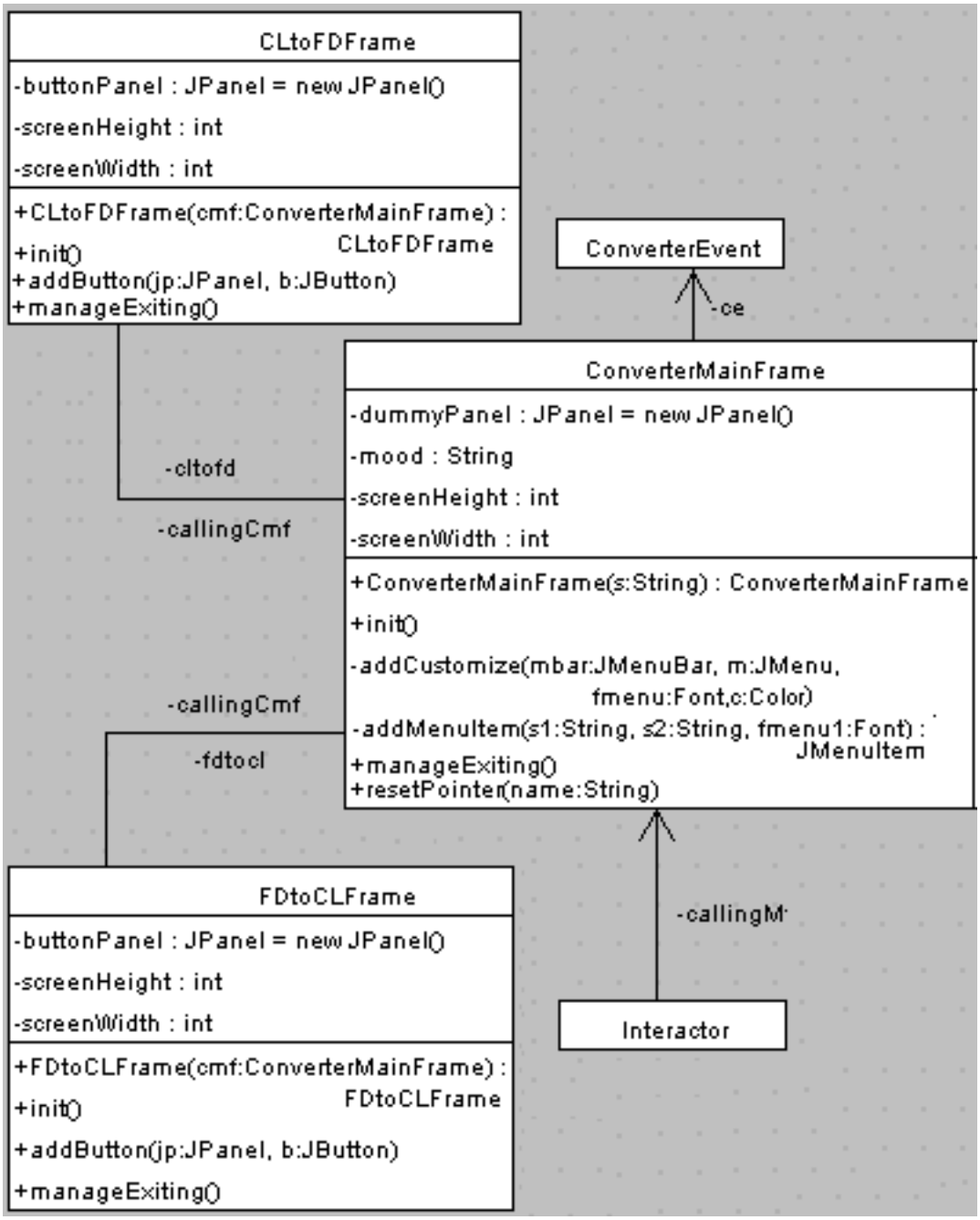

Figura A.79: *I convertitori: dettaglio delle classi (2)*

A.2.7 *Classi ausiliarie*

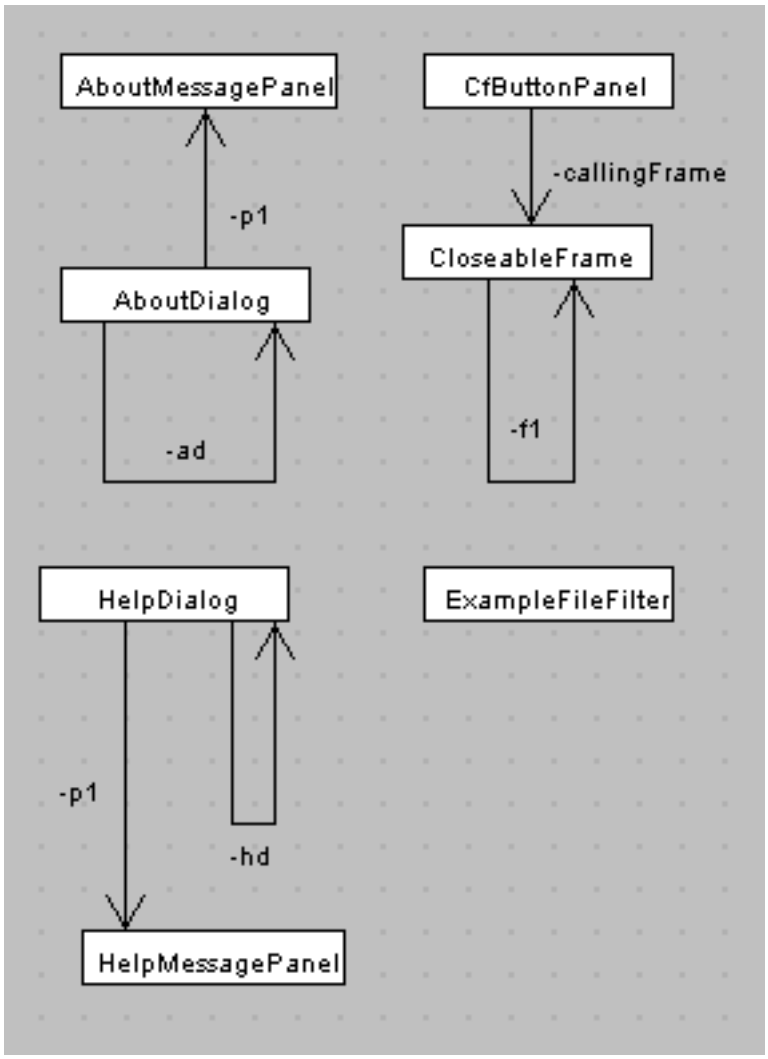

Figura A.80: *Classi ausiliarie*

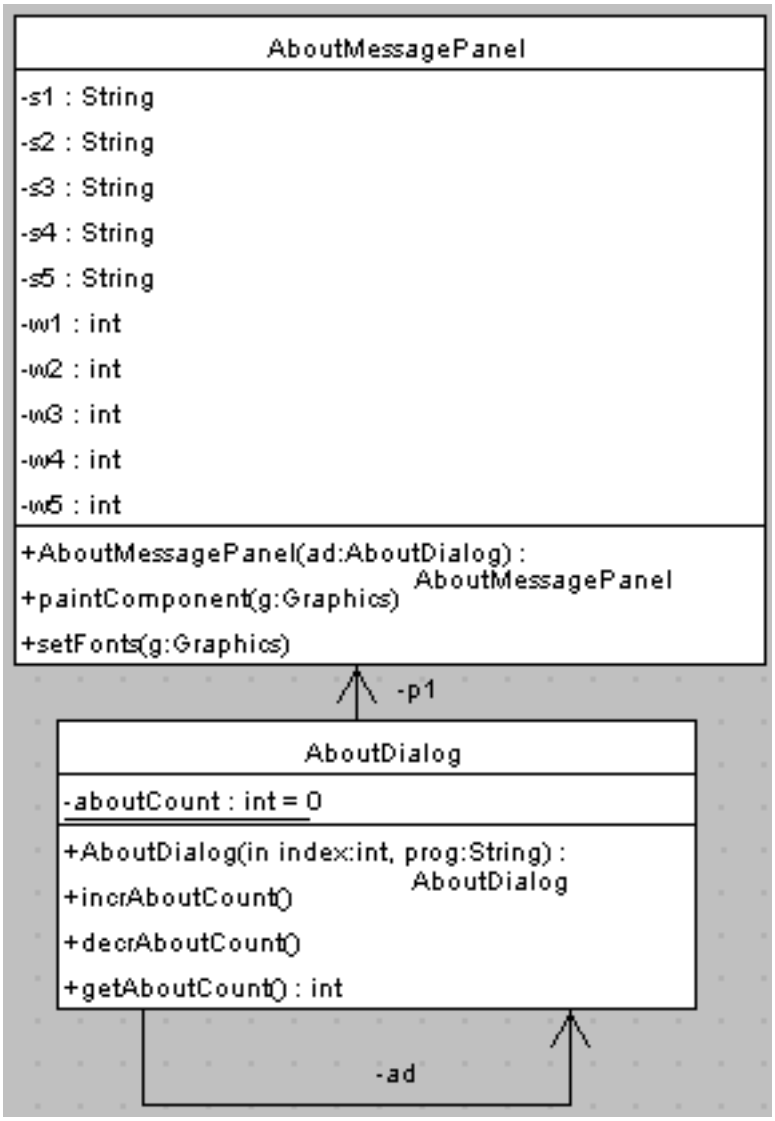

Figura A.81: *Classi ausiliarie: dettaglio (1)*

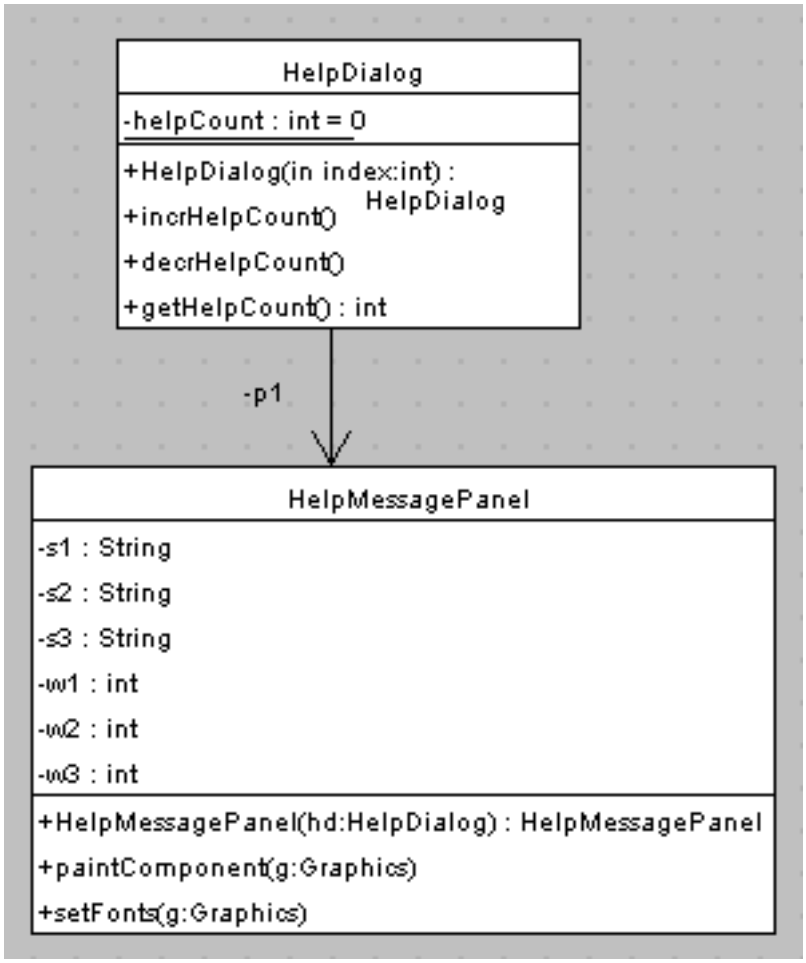

Figura A.82: *Classi ausiliarie: dettaglio (2)*

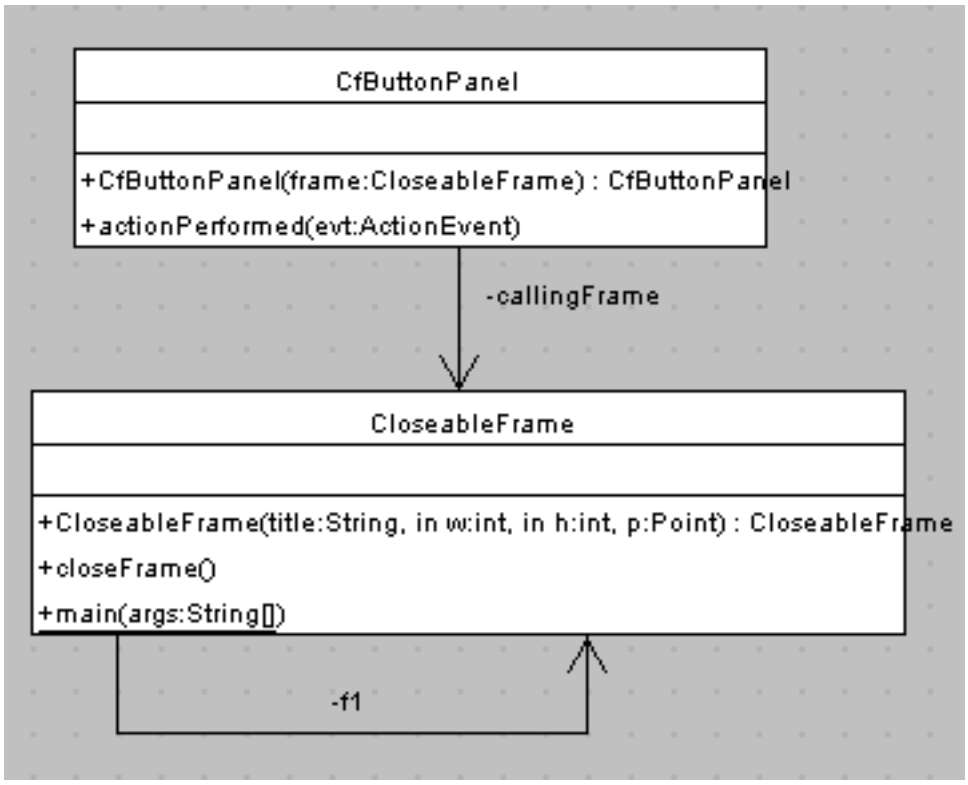

Figura A.83: *Classi ausiliarie: dettaglio (3)*

A.2.8 *Classi grafiche*

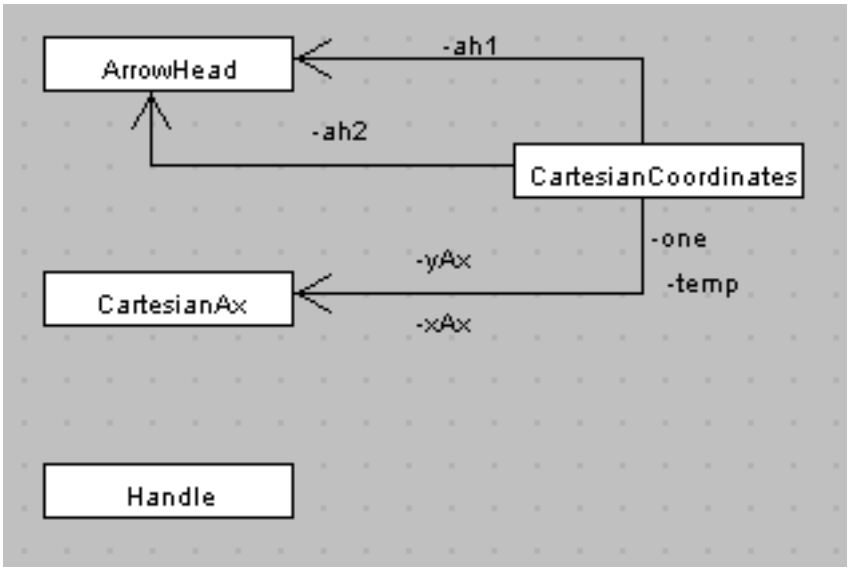

Figura A.84: *Classi grafiche*

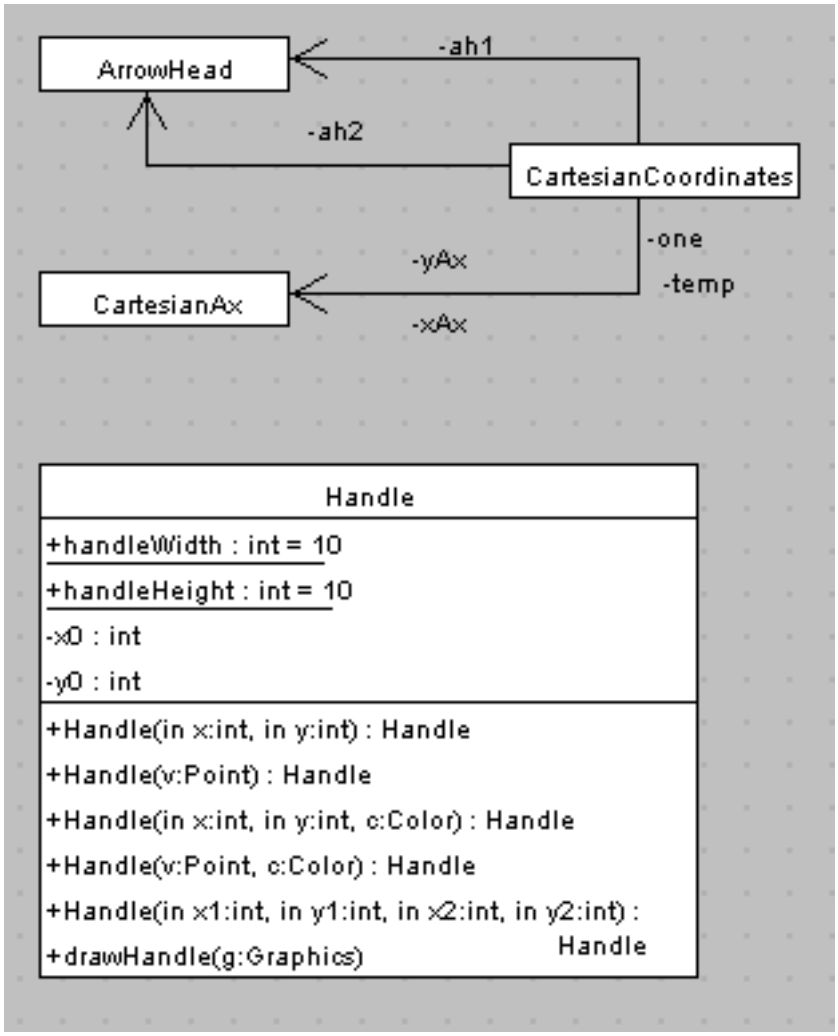

Figura A.85: *Classi grafiche: dettaglio (1)*

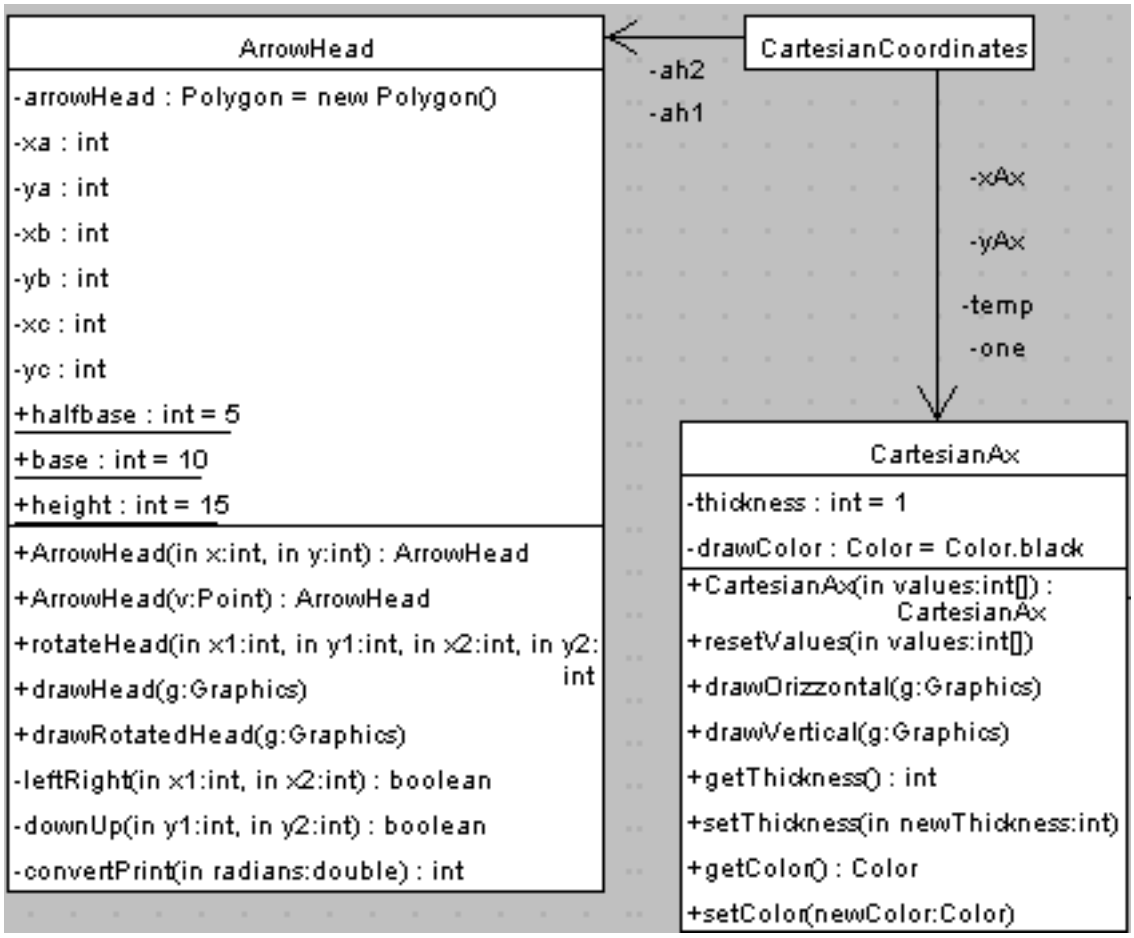

Figura A.86: *Classi grafiche: dettaglio (2)*

Riferimenti Bibliografici

[Bis99] Judy Bishop. *Java Gently, Corso introduttivo.* Addison-Wesley, 1999.

[BS91] Giuseppe Biondo e Enrico Sacchi. *Manuale di Elettronica e Telecomunicazioni.* Hoepli, 1991.

[BSL02] Simon Bennet, John J. Skelton e Ken Lunn. *Introduzione a UML.* McGraw Hill, 2002.

[BSS02] Marko Boger, Thorsten Sturm e Erich Schildauer. *Poseidon for UML Users Guide.* GentleWare, 2002.

[Eck98] Bruce Eckel. *Thinking in Java.* Prentice Hall, 1998.

[GMS94] Michel Goossens, Frank Mittelbach e Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, 1994.

[GW02] Ian Graham e Alan Willis. *UML, A tutorial.* MMI Trireme International, 2002.

[HC99] Cay S. Horstmann e Gary Cornell. *Java 2: i fondamenti.* Mc Graw Hill, 1999.

[Iaz75] Giuseppe G. Iazeolla. *Simulazione di Sistemi.* ETS, 1975.

[Kir98] Craig W. Kirkwood. *System Dynamics Methods: A Quick Introduction.* College of Business, Arizona State University, 1998.

[Lam94] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison-Wesley, 1994.

[Mar81] Giovanni Marro. *Controlli automatici.* Zanichelli, 1981.

[Ore76] Oystein Ore. *I Grafi e le loro applicazioni.* Zanichelli, 1976.

[Pau93] Frances N. Paulisch. *The Design of an Extensible Graph Editor.* Springer-Verlag, 1993.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling e Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[RAD+83] Nancy Roberts, David Andersen, Ralph Deal, Michael Garet e William Shaffer. *Introduction to Computer Simulation. A System Dynamics Modeling Approach.* Addison Wesley, 1983.

[WM99] Kathy Walrath e Campione Mary. *The JFC Swing Tutorial. A guide to Constructing GUIs.* Addison-Wesley, 1999.