Honeywell
Honeywell Information Systems Italia

Via Laboratori Olivetti - Pregnana Milanese

UNIVERSITA'
DEGLI STUDI DI MILANO
ISTITUTO DI CIBERNETICA
Via Viotti, 5 - Milano

25/26

NOTE DI SOFTWARE

UNIVERSITA'
DEGLI STUDI DI MILANO
ISTITUTO DI CIBERNETICA

Honeywell
Honeywell Information Systems Italia

Note di software.
è un bollettino trimestrale, edito a cura del Centro di Ricerca e Progettazione della Honeywell Information Systems Italia e dell'Istituto di Cibernetica dell'Università di Milano.

Note di software.
intende costituire uno strumento per la rapida e informale circolazione di informazioni, idee ed esperienze nell'area della tecnologia del software. In particolare, intende stimolare il dibattito sulle metodologie orientate alla diminuzione del rapporto costo/qualità dei programmi, e sugli strumenti atti ad automatizzare l'attività di produzione del software in ogni suo aspetto.

La collaborazione a **Note di software** è libera.
In particolare, si sollecitano contributi nella forma di brevi monografie e bibliografie essenziali commentate sui seguenti temi: tecniche e strumenti per la programmazione strutturata e modulare, metodologie e strumenti per il testing e il debugging dei programmi, documentazione del software, aspetti organizzativi e gestionali nella costruzione di sistemi software di grandi dimensioni.

Chi desiderasse pubblicare un contributo è pregato di prendere contatto con il Comitato di Redazione (HISI - Via ai Laboratori Olivetti - Pregnana Milanese), o di inviare direttamente il dattiloscritto in triplice copia.
I contributi non dovrebbero superare le quattromila parole.

I lavori accettati per la pubblicazione vengono revisionati dal Comitato di Redazione, che si avvale del contributo di specialisti del settore. Ogni contributo pubblicato riflette, comunque, le opinioni dei suoi autori, e non necessariamente quelle del Comitato di Redazione. Eventuali revisioni saranno effettuate di comune accordo fra il Comitato di Redazione e i proponenti.

Note di software.
Viene distribuito ad aziende, Enti e specialisti del settore che ne facciano richiesta alla redazione.

Direttore responsabile: Paolo Ghelfi
Comitato di Redazione: Roberto Polillo - Wanda Anselmetti - Stefania Bandini
Redazione: Franco Soresini

25/26

NOTE DI SOFTWARE

Dicembre 1984

Indice:

QUESTO NUMERO... Pag. 1

CONTRIBUTI TECNICI:

- CONTROLLO DEI PROCESSI E RETI DI PETRI, di P. Salvaneschi » 3
- LA QUALITÀ DEL SOFTWARE DEL PERSONAL COMPUTER: UNA DISCUSSIONE, di R. Polillo » 27
- IL DOSSIER ELETTRONICO: UNO STRUMENTO DI INFORMATICA INDIVIDUALE PER L'UFFICIO, di R. Paulin, R. Polillo, P. Schiaro Campo » 35
- GUIDELINES TO CHECK COHERENCE IN EXPLANATORY TEXTS, di G. Tonfoni, B. Bruce » 47
- UN ESEMPIO DI AUTOISTRUZIONE GUIDATA CON IL CALCOLATORE, di F. Monzani » 56
- IL LINGUAGGIO SGML (STANADARD GENERALIZED MARKUP LANGUAGE) PROPOSTO DA ISO, di Le van Huu » 66

Autorizzazione del Tribunale di Milano n. 87 del 23 febbraio 1977

QUESTO NUMERO...

L'uso delle reti di Petri per la modellazione di processi, il software per le applicazioni di informatica individuale del personal computer, la documentazione tecnica, il CAI e gli standard dei linguaggi di elaborazione di testi sono gli argomenti di questo numero doppio di NOTE DI SOFTWARE.

Nel primo articolo Paolo Salvaneschi propone un'ampia introduzione di carattere teorico, ma corredata da alcuni esempi, sull'uso di reti di Petri per la modellazione di processi. In particolare, viene esaminato il problema del controllo dal punto di vista degli strumenti atti a imporre a processi il comportamento desiderato e ad evitare l'insorgere di comportamenti non desiderati.

Nel secondo lavoro, Roberto Polillo discute una serie di problemi connessi al controllo di qualità del software per personal computer, e, in particoalare, dei cosiddetti "office productivity tools": sistemi di word processing, spreadsheet, database, etc...
Vengono esaminati, relativamente a tali prodotti software, i principali attributi di qualità, e possibili metriche per misurarli.

Il terzo lavoro, di R. Paulin, R. Polillo e P. Schiavo Campo, descrive i concetti base utilizzati nella progettazione dell'interfaccia utente di DOSSIER, uno strumento software di informatica individuale per l'ufficio, disponibile su diversi personal computer di tipo professionale.

Graziella Tonfoni presenta poi nel quarto articolo un modello per il controllo e la verifica dei livelli di coerenza nei testi di carattere esplicativo (ad esempio, documentazione tecnica). Tale modello fornisce uno schema utilizzabile nella pratica sia per la identificazione di problemi di coerenza, sia per ristabilire tale attributo nei testi che ne sono carenti.

Gli aspetti metodologici utilizzati nella realizzazione di un corso CAI che permette l'addestramento di un utente finale senza alcuna conoscenza di informatica all'uso di un minicomputer vengono esposti in un lavoro di Federico Monzani. Tale corso è stato realizzato sugli elaboratori MicroSystem 6/10 e 6/20 della Honeywell.

Nell'ultimo lavoro, infine, Le van Huu presenta SGML (Standard Generalized Markup Language), una componente dello standard per le attività di trattamento di testi (Standard Text Processing Languages), presentato dall'ISO. SGML fornisce una sintassi non ambigua che permette all'utente di descrivere gli elementi logici della struttura del testo.

La Redazione

CONTRIBUTI TECNICI:

CONTROLLO DEI PROCESSI E RETI DI PETRI

Paolo Salvaneschi (°)

RIASSUNTO

Perchè serve un controllore di un processo? Per imporre al processo (che già esiste) un comportamento desiderato (tra quelli possibili) e per tenerlo lontano da un comportamento non desiderato.
Per costruire un controllore quindi, prima si costruiscono modelli del processo, del comportamento desiderato e di quello non desiderato, poi modelli del controllore in cui si inseriscono i modelli precedenti (la conoscenza e i desideri) e attività che li usano. Le Reti di Petri sono lo strumento utilizzato per costruire i modelli. Sono mostrati degli esempi tutti da sistemi di controllo di processi industriali.

ABSTRACT

Why something needs a process controller? To force an existing process to behave as closely as possible to a desidered behaviour (between possible ones) and as far as possible from a not desired behaviour. To build a controller we have to build a process model and both the desired and the not desired behaviour models; then we have to model the controller using the previous models (the knowledge and the wishes) and activities using them. The modeling tool are Petri Nets. We show then some example from industrial process control systems.

1. I PROCESSI

1.1 Processi

Una qualunque unità produttiva serve a produrre dei beni a partire da altri beni. All'interno dell'attività i beni che entrano vengono trasformati fino ad ottenere dei beni in uscita tra cui i prodotti per cui l'unità produttiva è stata costruita. I beni di cui parliamo possono essere i più vari (automobili, energie, denaro, idee e così via).

Un processo è il fluire dei beni e lo svolgersi delle attività di trasformazione all'interno dell'unità produttiva.

1.2 Uomini e macchine

Queste attività di trasformazione vengono svolte da uomini e da macchine.
Ci sono interazioni tra uomini (il caporeparto ordina un'azione ad un dipendente, due colleghi di lavoro comunicano tra loro,...), interazioni tra uomini e macchine (uso di una macchina, sorveglianza di una macchina, riparazione di una macchina,...) e interazioni tra macchine (linea di produzione fatta di più macchine).

1.3 Cooperazione e conflitti

Poichè una unità produttiva produce determinati beni, deve esistere una qualche forma di cooperazione tra uomini e macchine al fine di produrre quei beni; cioè, le relazioni tra uomini e macchine devono essere, almeno parzialmente, costruite in vista di un fine voluto.

Non per questo si può dire che esista solo cooperazione tra uomini e macchine.
Esistono dei conflitti dovuti innanzitutto all'esistenza simultanea di obiettivi diversi nel processo. Ad esempio, un lavoratore può avere come obiettivo quello di accrescere la propria professionalità oltre che quello di portare a casa uno stipendio e il padrone può avere come obiettivo l'uso delle conscenze già in possesso del lavoratore al fine di produrre denaro il più possibile e il più in fretta possibile.

(°) Paolo Salvaneschi ha lavorato presso il gruppo sistemi di controllo processo della società Dalmine s.p.a., ed ora lavora presso la società ISMES di Bergamo.

La risorsa in comune tra i due è il lavoro del dipendente e di volta in volta sarà usata per l'uno o per l'altro fine (cioè il conflitto si risolverà a favore dell'uno o dell'altro).

Esistono inoltre conflitti dovuti a informazioni parziali. Ad esempio un uomo o una macchina danno ordini contraddittori ad un'altra macchina a causa della non sufficiente conoscenza dello stato di quella macchina.

In generale si può dire quindi che esistono simultaneamente sull'insieme degli uomini e delle macchine più reti di relazioni parziali che variano col tempo.
L'effetto che si ottiene riguarda comportamenti diversi, e le regole per cui viene eseguito un comportamento al posto di un'altro possono essere molto complicate e solo parzialmente note.

1.4 Flussi

Lungo le reti di relazioni tra macchine e uomini scorrono flussi di oggetti.
Alcuni di questi sono i flussi che vengono trasformati (barre di acciaio, denaro, idee...).

Altri sono flussi di comunicazione. Gli oggetti trasmessi lungo i flussi di comunicazione (segnali elettrici, pezzi di carta scritta,...) hanno per chi li riceve (uomo o macchina) un qualche contenuto di informazione nel senso che sono capaci di giocare un ruolo nelle scelte del ricevitore (tra possibili azioni).

In particolare i flussi di comunicazioni possono essere utilizzati (da uomini o da macchine) per regolare il flusso dei materiali e cioè per agire, disponendo di piccole quantità di flusso, su grandi quantità di flusso.

1.5 Imprevedibilità e rischi

È noto che possono verificarsi malfunzionamenti su entrambe le componenti del processo: gli uomini e le macchine (le macchine si guastano, gli uomini si stancano,...); inoltre sui flussi di comunicazioni c'è rumore (chi ascolta può capire in modo differente da ciò che voleva comunicare chi parla, una pratica operativa è ambigua e imprecisa, un segnale elettrico è disturbato,...).

Il malfunzionamento è nella natura stessa del processo e non è eliminabile.
È possibile però dare valori diversi ai rischi connessi ad ogni malfunzionamento (riduzione della produzione, danneggiamento di una macchina, morte di un operaio, esplosione di una centrale nucleare,...).

2. MISURA E REGOLAZIONE

2.1 Misura e regolazione

Una qualunque possibilità di conoscere il processo e di agire su di esso, da parte di qualcosa o di qualcuno che non è parte del processo, è legata alla possibilità di inserirvi due classi di attività tramite le quali stabilire dei flussi (di comunicazione) da o per il processo: l'attività di misura e l'attività di regolazione.

2.2 Attività di misura

Un'attività di misura è inserita nel processo allo scopo di generare un flusso di comunicazioni che contenga informazioni su ciò che accade nel processo.

Il processo con l'attività di misura aggiunta è diverso da prima; per esempio, nell'impianto industriale c'è un oggetto (strumento di misura) in più.
In generale è necessario che l'attività aggiunta disturbi il meno possibile il processo preesistente.
Questo, in alcuni casi, può essere un problema: un sensore magnetico di prossimità che misura la presenza di una barra di acciaio su un bancale può essere aggiunto senza grossi problemi di alterazione del processo.
Un ispettore che misuri il funzionamento di un ufficio creerà qualche problema aggiuntivo; il comportamento dell'ufficio con o senza ispettore potrà essere significativamente diverso.

L'attività di misura genera un flusso di oggetti (segnali elettrici nel caso del sensore di prossimità, pezzi di carta nel caso dell'ispettore).
In generale le energie associate al flusso di oggetti saranno inferiori alle energie in gioco nel processo (un trasformatore di misura su una linea ad alta tensione genera tensioni e correnti di ordini di grandezza inferiori a quelle delle linee stesse).

2.3 Attività di regolazione

Un altro tipo di attività è inserito nel processo allo scopo di alterare i flussi esistenti o di alterare comunque il processo usando un flusso di comunicazione dall'esterno.

Una valvola inserita in un tubo in cui scorre acqua può, se opportunamente azionata, regolare il flusso di acqua.
Un relay o una catena di relays possono aprire o chiudere un interruttore di potenza su una linea ad alta tensione.
Un aumento di stipendio può convincere un impiegato a lavorare con maggiore impegno.
La valvola, il ralay, l'aumento di stipendio sono esempi di strumenti di regolazione.

Le attività di regolazione possiedono in generale la caratteristica di pilotare flussi a cui è associata un'energia maggiore di quella associata al flusso di comunicazione che connette l'attività di regolazione all'esterno del processo.

Le correnti che scorrono in un relay che pilota un interruttore su una linea ad alta tensione sono inferiori a quelle che interessano la linea stessa. L'aumento di stipendio è dato nella speranza di un maggiore ritorno di denaro.
In generale chi inserisce l'attività di regolazione nel processo lo fa nella speranza (non sempre convalidata) di poter pilotare il processo con un qualche criterio di utile, spendendo meno di quanto guadagna (qualunque cosa sia ciò che viene speso o guadagnato).

3. MODELLI E CONTROLLO

3.1 Obiettivi e controllo

Come abbiano detto prima, di un processo serve per produrre determinati beni.
Esiste cioè qualcuno che vuole qualcosa tramite il processo (obiettivi).

Ma poichè nel processo ciò che accade è molto più vasto (imprevedibile, parzialmente noto, anche rischioso) di ciò che deve accadere per raggiungere gli obiettivi, allora qualcuno vuole esercitare delle attività di controllo sul processo per costringerlo a seguire dei comportamenti i più vicini possibili a quelli necessari per raggiungere gli obiettivi.
Per questo motivo viene costruito un controllore che esercita delle attività di controllo nei confronti del processo.

In questa schematizzazione attività di controllo e processo sono distinti e tra i due scorrono flussi di comunicazioni. (fig. 1).

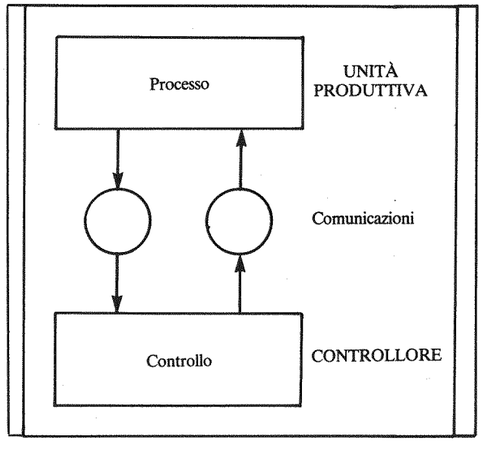


Fig. 1

3.2 Modelli e controllo

L'attività di controllo può inserire nel processo attività di misura per raccogliere informazioni e attività di regolazione per costringerlo a seguire definiti comportamenti.

Ma quali misure prelevare dal processo e quali azioni produrre su di esso? Dove mettere strumenti di misura e regolatori? In che modo usarli per raggiungere determinati obiettivi?

Ad esempio, il guidatore di automobile vuole mantenere l'automobile sul lato destro della carreggiata: quando l'auto si sposta o per una irregolarità della strada o perchè c'è una curva, il guidatore muove il volante fino a che l'automobile non ritorni nella posizione desiderata.
Ma perchè egli muove il volante e perchè lo muove in un certo modo (lentamente o velocemente) e non in un altro modo? Egli sa che così facendo l'auto si muove come egli desidera. Ma se la strada è coperta di ghiaccio il comportamento ottenuto può essere ben diverso da quello desiderato.
L'automobilista controlla il movimento del mezzo: egli innanzitutto sa quale è il comportamento che l'automobile deve seguire.
Egli inoltre conosce l'automobile, possiede cioè un modello mentale di come l'auto si comporta, ad esempio quando, ad una data velocità, gli muove il volante più o meno rapidamente.
Se la strada è ghiacciata, e l'automobilista non se n'è accorto, può darsi che egli esca fuori strada poichè il suo modello mentale di comportamento non è più adeguato al comportamento reale dell'automobile.

Un'attività di controllo deve quindi almeno possedere due modelli:

- ciò che può accadere (comportamento del processo, modello del processo, legge del processo);
- ciò che deve accadere (comportamento che l'attività di controllo vuole ottenere dal processo, comportamento desiderato).

Un problema di controllo è quindi prima di tutto un problema di modellizzazione.

3.3 La comunicazione dei modelli

Modelli del processo, dei suoi comportamenti e dell'ambiente attorno al processo vengono utilizzati durante le comunicazioni tra chi commissiona il controllare, il progettista e gli utilizzatori (fig. 2).

Durante la costruzione del controllare vengono utilizzati i modelli precedenti e il progettista produce più modelli del controllore stesso; alla fine modelli del nuovo processo controllato servono per chi usa la macchina per il controllo e la mantiene.

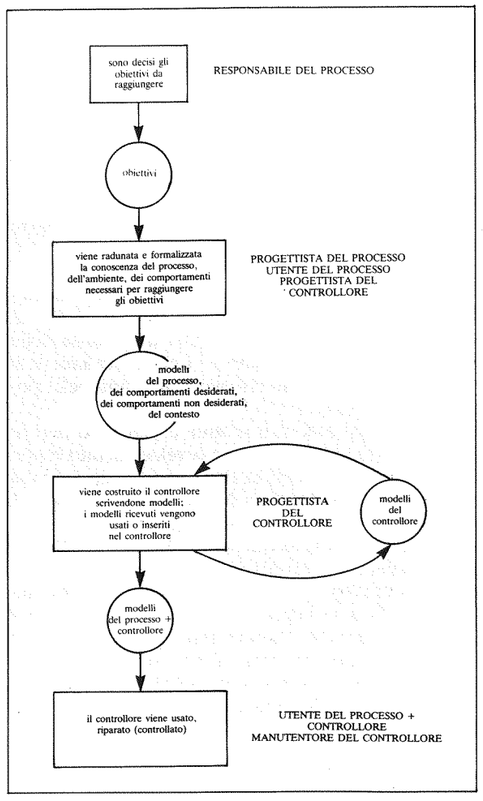


Fig. 2

3.4 Processo+controllo = processo, la necessità di uomini

La costruzione di un controllore è l'aggiunta di un insieme di attività e di connessioni al preesistente processo.
Ma il processo con l'aggiunta dell'attività di controllo è un nuovo processo e tutto quanto è stato detto per i processi vale anche per questo; in particolare valgono le considerazioni a proposito dei malfunzionamenti, dei comportamenti imprevedibili, e dei rischi connessi, e si pone di nuovo il problema di controllare il nuovo processo.
Non vi è altra possibilità che costruire un controllore del nuovo processo e così via.

Per questo motivo, qualunque sia il livello di realizzazione di attività di controllo mediante macchine, la presenza di uomini che svolgono attività di controllo è obbligatoria.

Questa asserzione è regolarmente verificata negli impianti industriali.
Prendiamo per esempio un controllo dell'energia consumata da una acciaieria mediante un calcolatore: ogni tanto viene chiamato il progettista dei programmi o il manutentore del calcolatore per mettere riparo ad un malfunzionamento. Ciò significa che mentre prima l'uomo controllava il consumo di energia manualmente, mediante l'ausilio di strumenti di misura e di azionamenti, ora controlla il calcolatore che controlla il consumo dell'energia.

3.5 Problemi di cultura e modelli di gestione dei processi

La necessità di operatori induce a considerare il ruolo degli uomini e delle macchine nei processi e l'atteggiamento generale (culturale) nei confronti del controllo dei processi.
A questo proposito un punto di vista sostiene che il controllo dei processi sia un problema di costruzione di macchine (nel senso che gli uomini sono secondari).
Ma poichè la presenza di uomini impegnati in attività di controllo è obbligatoria, un punto di vista più corretto afferma che il controllo dei processi è un problema umano di cui un aspetto è la costruzione di macchine (anche se pure la costruzione di macchine è un problema umano).

Risultano allora fondamentali problemi del tipo:

- progettare macchine per il controllo che siano utensili nelle mani di uomini (comprensione della macchina, interazione con la macchina,...);
- comunicare modelli (conoscenza) agli uomini (formazione, didattica, macchine per comunicare conoscenza,...);
- progettare relazioni tra i vari livelli di controllo umano (gerarchie) che massimizzino l'uso degli uomini come controllori (autonomia, mantenimento e minimizzazione dei flussi di informazione tra i vari livelli della gerarchia,...).

4. LA STRUTTURA DEL CONTROLLORE

Un'attività di controllo è composta da alcuni elementi sempre presenti, qualunque sia il processo da controllare.
Innanzi tutto l'attività di controllo deve contenere due modelli: il modello del comportamento desiderato del processo e il modello del processo.

4.1 Comportamento desiderato

Il modello del comportamento desiderato del processo descrive l'obiettivo del controllore (come il processo si deve comportare, ciò che deve accadere).

4.2 Modello del processo

Il modello del processo descrive invece il mondo del controllore (ciò che egli vede, ciò che per lui esiste, ciò che conosce, cioè ciò che può – e non può – accadere).

Il modello del comportamento desiderato serve da orientamento per descrivere la classe del fenomeni che deve essere rappresentata nel modello del processo. Questo (come qualunque modello) è costruito infatti dal punto di vista di ciò che si vuole.
Ad esempio, se si vuole costruire un controllore del flusso dei pezzi lungo un laminatoio, ci sarà un flusso desiderato (primo modello) e un insieme di flussi possibili (secondo modello). Chiaramente il secondo modello è costruito dal punto di vista del primo; banalmente esso modella flussi di pezzi e non flussi di molecole od altro.

I due modelli devono però essere distinti e il primo deve essere un sottoinsieme del secondo (il controllore deve vedere situazioni del processo non sotto controllo ma controllabili).

4.3 Comportamento non desiderato

Per costruire un buon controllore non è sufficiente definire un comportamento desiderato.
Come abbiamo visto, in un processo ci sono malfunzionamenti, comportamenti non previsti e rischi associati; ma tuttavia non tutti i comportamenti del processo diversi da quelli desiderati presentano gli stessi rischi associati.
È quindi possibile tentare di identificare un insieme di comportamenti ad alto rischio e costruire un nuovo modello che descriva comportamenti non desiderati.

L'attività di controllo assume allora il significato di costringere il processo a comportarsi nel modo il più possibile vicino al comportamento desiderato e lontano dal comportamento non desiderato.

Anche il comportamento non desiderato serve da orientamento per costruire il modello del processo; come pure il modello del comportamento non desiderato deve essere un sottoinsieme del modello del processo.

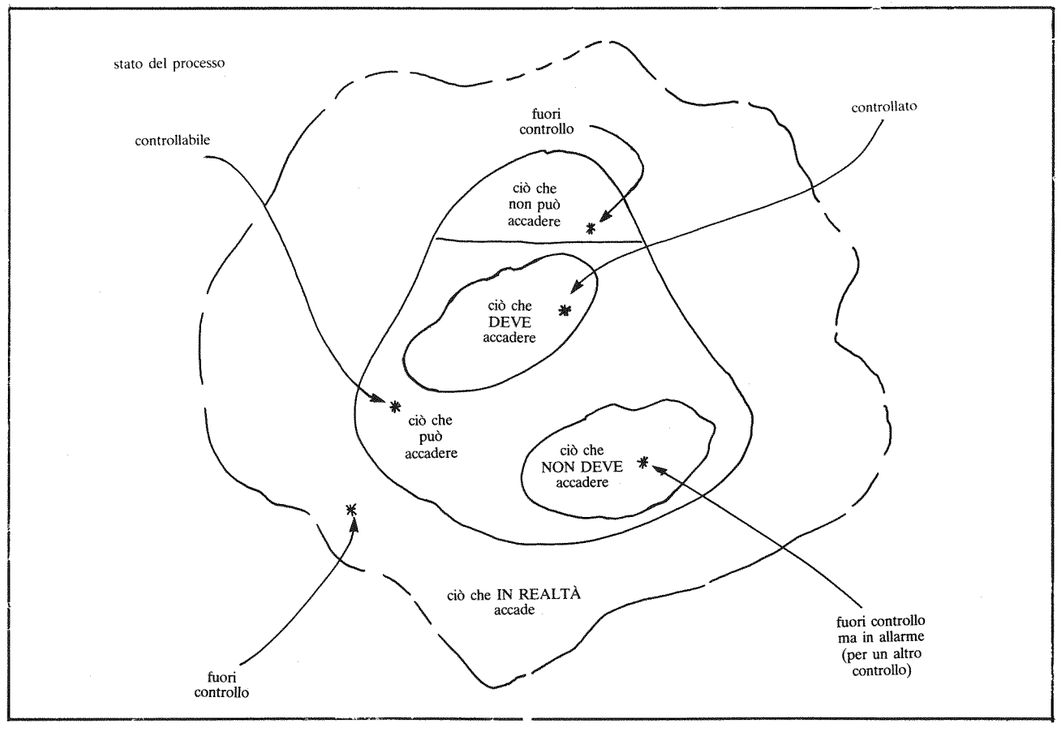


Fig. 3

Possiamo verificare che, anche se non è esplicitamente affermato e a volte non è compreso il significato, i due modelli (desiderato e non desiderato) esistono in pratica in tutti i controllori che si costruiscono realmente. Ad esempio, il controllore di una variabile di un processo contiene normalmente un valore di consegna (comportamento desiderato) e un valore limite di allarme (comportamento non desiderato).

Nelle azioni umane l'uso dei due modelli in attività di controllo si trova nel parlare comune; il vecchio detto del bastone e della carota descrive un'azione di controllo (con la carota) relativa ad un comportamento desiderato e un'azione di controllo (con il bastone) relativa ad un comportamento non desiderato.

La Figura 3 descrive le relazioni fra i tre modelli considerati e ciò che si può dire sullo stato del processo dal punto di vista dell'attività di controllo. Il modello del processo definisce i due insiemi di stati "ciò che può accadere e ciò che non può accadere"; i modelli dei comportamenti desiderati e non desiderati definiscono i due insiemi "ciò che deve e ciò che non deve accadere".
Si noti che l'insieme di "ciò che in realtà accade" è un insieme piu ampio dei precedenti e dai confini indistinti.

4.4 La sicurezza di un processo

Ci si può porre ora il problema di che cosa sia la sicurezza di un processo.
Diciamo che un processo è sicuro se non assume comportamenti non desiderati (fortemente rischiosi).

Il problema della sicurezza è quindi innanzitutto il problema di definire e modellare l'insieme dei comportamenti non desiderati del processo; per tentare di rendere sicuro un processo è necessario che il comportamento non desiderato faccia parte del controllore tanto quanto il comportamento desiderato.

Ma l'aggiunta di attività di controllo ad un processo costruisce un nuovo processo per il quale si ripresentano problemi di controllo e quindi:

- non esistono processi sicuri ma processi che si tenta di mantenere sicuri;
- il problema della sicurezza non è solo quello di identificare comportamenti non desiderati e di costruire macchine per il controllo affidabili, ma è un problema umano poichè la presenza dell'uomo è inevitabile.

4.5 Misure e azioni

Altri componenti del controllore sono i flussi di comunicazioni.

Di questi uno è diretto dal processo al controllore (misure) e serve per aggiornare il modello del processo in base a ciò che accade nel processo; un altro flusso di comunicazioni è diretto dal controllore al processo (azioni) e serve al controllore per agire sul processo in base al comportamento desiderato e al comportamento non desiderato.

Esistono inoltre altri due flussi di comunicazioni che connettono il controllore al controllore di livello superiore ed hanno egualmente il significato di misure (allarmi, informazioni diagnostiche,...) e azioni (accensione, spegnimento, blocco, taratura del controllore,...).

La Figura 4 mostra tutte le componenti del controllore fino ad ora descritte.

4.6 Decisioni e strategie

Una parte dell'attività del controllore è dedicata al confronto tra i modelli per decidere se è necessario intraprendere azioni di controllo.

L'attività di controllo confronta ciò che accade nel mondo del processo con il comportamento desiderato e con quello non desiderato e, in caso di diversità del comportamento attuale rispetto al desiderato o di

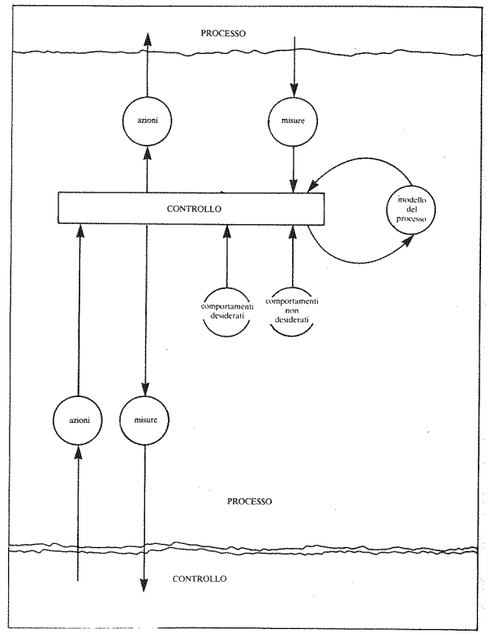


Fig. 4

eccessiva vicinanza al non desiderato, può decidere di intervenire.

Un'altra parte dell'attività di controllo è dedicata a realizzare strategie (quali azioni eseguire una volta presa la decisione che è necessario agire).

Possono essere realizzate sia strategie molto semplici che via via più "intelligenti" (simili ai modi umani di agire), ma un controllore deve comunque comprendere una strategia nei confronti del comportamento desiderato e una nei confronti di quello non desiderato.

Nei confronti del comportamento desiderato ad esempio:

- si applica una regola costante per generare le azioni (ad esempio una funzione dello scostamento tra comportamento desiderato e attuale);
- vengono eseguite simulazioni sul modello per valutare l'effetto di possibili azioni e viene di conseguenza decisa l'azione;
- Si tiene conto di cosa è successo in precedenti casi analoghi per decidere il tipo di azione;
  ...

Nei confronti del comportamento non desiderato ad esempio:

- si generano azioni di blocco e un allarme;
- si prova ad agire sul processo per spostarlo dal comportamento non desiderato;
- si opera un intervento di riconfigurazione sul processo;
  ....

5. LA COSTRUZIONE DEL CONTROLLORE

Vediamo ora quali criteri, relativi alla costruzione di un controllore, si possono ricavare dalle considerazioni precedenti.

5.1 Stile di lavoro e retroazione

Lo stile umano del lavoro è fondato sulla retroazione ed il problema è come usare utilmente la retroazione nella fabbricazione di un controllore (software) di un processo.

Tutti gli stili artigianali di fabbricazione fanno un uso estensivo della retroazione ma non è detto che ci sia sufficiente capacità di controllo della fabbricazione e che le successive iterazioni nella fabbricazione convergano ad un risultato di buona qualità.

Anche nei casi in cui il processo alla fine converge, possono esistere pesanti problemi di costo dovuti ad eccessive iterazioni, mentre un progetto nella maggior parte dei casi deve riuscire al primo colpo.

In pratica per tentare di risolvere il problema vengono usate le tecniche di trattare pochi oggetti alla volta e di suddividere in fasi sperabilmente senza ritorni il processo di fabbricazione (retroazione locale).

Trattare pochi oggetti alla volta vuol dire che innanzitutto ci devono essere degli oggetti e poi che devono essere pochi nonostante il problema vada considerato globalmente (non prendendone solo una parte).
Il trucco consiste nel costruire oggetti astratti (pochi) e decomporli progressivamente e ordinatamente in oggetti sempre meno astratti (molti), una parte alla volta.

Suddividere in fasi significa usare ancora in realtà meccanismi di retroazione, ma locali (ad esempio, ci si occupa prima del perchè di un controllore e poi di cosa deve fare).

Ogni fase si conclude con documenti completi, esaustivi di quella fase, che sperabilmente non richiedono di tornare a ripensare alla fase terminata durante le fasi successive, ma all'interno di ogni fase viene ancora usato il meccanismo della retroazione.

5.2 Strumenti

Il problema è allora quello di disporre di strumenti per aiutare il meccanismo della retroazione, di strumenti cioè che modellino la realtà (processi e controllori) con alcune caratteristiche:

- devono modellare oggetti reali molto diversi tra loro (macchine, pezzi di software, uomini);
- devono modellare la dinamica della realtà;
- devono prestarsi a generare più livelli di descrizione (da modelli generali a raffinamenti successivi);
- devono essere a più livelli di rigore (dalla capacità di descrivere situazioni imprecise, non ben note, alla capacità di descrivere situazioni sempre più note e formalizzabili rigorosamente);
- devono modellare la realtà da punti di vista diversi (il funzionamento, la struttura, gli oggetti);
- devono essere facili il più possibile per agevolare la comunicazione tra uomini con conoscenze diverse;
- devono essere eseguibili su macchine in modo che sia agevole per il progettista fabbricare e provare più modelli.

5.3 Il ciclo di vita di un controllore; fasi e documenti

Si può rappresentare la vita di un controllore come un ciclo in cui si alternano un'attività di produzione e una di uso.
Il ciclo è ripetuto più volte e la stessa prima produzione, quando il processo già esiste, può essere un riciclo relativo ad attività di controllo già esistenti e svolte diversamente.

L'attività di produzione può essere suddivisa in fasi ognuna delle quali produce un insieme di documenti che contengono modelli, fino alla fase di realizzazione in cui una parte di questi modelli viene incorporata in un calcolatore.

Tre fasi successive di lavoro rispondono a tre domande:

- perchè serve un controllore?
- cosa deve fare il controllore?
- come fa il controllore a fare ciò che deve?

Di seguito proponiamo per ogni fase un insieme di temi che devono essere esaminati e di modelli (documenti) che devono essere prodotti.

I documenti costiuiranno capitoli e paragrafi del raccoglitore memorizzato nel calcolatore e, dipendentemente dalle capacità della macchina, si potranno eseguire anche i documenti di livelli alti (ad esempio le reti delle specifiche funzionali) oppure solo quelli di livelli bassi (i programmi).

La prima fase usa la conoscenza del processo e dell'ambiente e, diretta da obiettivi, studia il processo e l'ambiente, e valuta gli obiettivi per produrre:

1. Delle descrizioni informali del processo, dell'ambiente e degli obiettivi, propedeutiche agli altri documenti.
2. La definizione degli obiettivi.
3. Il modello del processo.
4. Il comportamento desiderato (modello del processo voluto).
5. Il comportamento non desiderato (modelli dei processi assolutamente non voluti).
6. Il contesto (l'ambiente modellato come problema di controllo)
7. Le valutazioni dei modelli prodotti.

La seconda fase usa i documenti prodotti, le prestazioni richieste, i vincoli imposti al controllore e sintetizza un primo modello del controllore contenuto nei documenti:

8. Il modello del controllore come unica attività connessa al processo e al contesto.
9. Le funzioni svolte dalla attività di controllo.
10. Le comunicazioni tra attività di controllo e processo.
11. Le prestazioni e i vincoli.
12. La valutazione dei modelli prodotti.

La terza fase usa tutti i documenti prodotti nelle due precedenti, costruisce un'architettura funzionale e la realizza utilizzando specifiche tecnologie.
Vengono inseriti nel raccoglitore i documenti:

13. La struttura funzionale del controllore.
14. La realizzazione tecnologica (gerarchia di documenti comprendente la struttura tecnologica – programmi, dati, pezzi di hardware e relative connessioni –, il disegno di ogni programma, il testo di ogni programma, la specifica dell'hardware in cui sono ospitati i programmi e dell'hardware di connessione con il processo).
15. La valutazione.

6. PERCHÈ SERVE UN CONTROLLORE?

La prima domanda a cui rispondere è "perchè serve un controllore?"
Un controllore serve ad ottenere definiti comportamenti da un processo; allora il punto di partenza per costruire un controllore non possono che essere il processo, gli obiettivi che si vogliono raggiungere mediante il controllore e i comportamenti che si vogliono imporre al processo e da cui ci si vuole difendere.
Questo punto di partenza è veramente chiave del resto del lavoro ed è del tutto inutile preoccuparsi del controllore se non si conosce il processo e ciò che si vuole dal processo.

6.1 Descrizioni informali

Si comincia con delle descrizioni informali, ma scritte, (in italiano, con reti di Petri, con schizzi) che iniziano a radunare le idee e la conoscenza, sia a proposito degli obiettivi che di come funziona ed è fatto il processo. Queste descrizioni (scritte) sono strumenti di lavoro e di comunicazione che servono al chiarimento preliminare del problema.

6.2 Obiettivi

Bisogna descrivere quali sono gli obiettivi che il committente vuole raggiungere mediante l'inserimento del controllore nel processo.
In generale essi possono essere descritti come prestazioni associate al processo (ad esempio la qualità del prodotto deve migliorare).
Queste prestazioni dovranno essere il più possibile misurabili e dovrà essere data una stima di valori che si vogliono ottenere con l'inserimento del controllore. Esso è infatti lo strumento per raggiungere gli obiettivi e, alla sua entrata in servizio, sarà valutato sulla base del documento che li descrive.

6.3 Modello del processo

È necessario ora scrivere un modello del processo fornendo almeno tre tipi di descrizione:

- funzionale (come funziona il processo, quali trasformazioni opera);
- strutturale (da quali parti è composto e come sono connesse);
- una descrizione fisica (se il processo si svolge ad esempio in un impianto industriale come faccio ad associare le descrizioni funzionale e strutturale agli oggetti fisici che vedo).

Vanno inoltre descritte le possibili vie di azione e di misura sul e dal processo.

È chiaro che questo documento non può essere steso indipendente dagli altri poichè i comportamenti da imporre o da evitare servono come riferimento per sapere cosa modellare.

Lo strumento in generale proposto per modellare sono coppie di pagine; una pagina contiene una rete di Petri, la sua associata una descrizione in italiano che dettaglia e aggiunge informazioni sulla rete.
Possono anche essere utili schizzi e disegni per dare la descrizione fisica del processo.

È possibile che del processo si possieda o si riesca a costruire un modello matematico. In questo caso il modello funzionale del processo è rappresentato da un insieme, ad esempio, di equazioni differenziali.
Ma molto spesso non ci si trova nella situazione felice di poter modellare il processo con strumenti di questo tipo e in generale esisteranno parti e comporamenti del processo fuori dalla descrizione dei modelli matematici tradizionali.

Se del processo fanno parte degli uomini che comunicano tra loro e con delle macchine, bisognerà pure tenerne conto e modellare, anche se l'ambiente da modellare è impreciso, incerto, poco formalizzabile (qual'é la funzione di trasferimento di un operaio metalmeccanico?).

È possibile che di un processo continuo si costruisca un modello fondato su un sistema di equazioni. Ad esempio si può costruire il modello termico e chimico del comportamento del bagno di acciaio in un forno elettrico, ma questo modello non esaurisce il processo e non è sufficiente per descriverlo. Infatti attorno alla parte continua del processo esiste un processo discontinuo che va modellato (gli addetti al forno lo accendono, lo spengono, accadono delle situazioni di emergenza che interrompono il lavoro,...).

Da questo punto di vista si può pensare che lo strumento sopra proposto (reti di Petri e italiano) possa essere in generale usato e venga localmente sostituito da modelli matematici di altro tipo.
Si può raffigurare il modello del processo come una rete di cui alcune attività contengono relazioni matematiche.

Ovviamente il trucco di costuire modelli astratti più generali e di raffinarli successivamente vale anche per questo caso e, se il processo è complesso, il documento conterrà più raffinamenti successivi del modello.

6.4 Modello del comportamento desiderato del processo

Bisogna poi definire quali sono i funzionamenti voluti dal processo.
Il comportamento desiderato può essere descritto come insieme di asserzioni relative al modello del processo (se è vera una condizione, allora deve accadere qualcosa, oppure una viariabile deve assumere un insieme di valori) oppure come un modello a parte di un processo voluto.
Gli strumenti di descrizione proposti sono sempre quelle esaminati nel documento "modello del processo".

6.5 Modello del comportamento non desiderato del processo

Anche i comportamenti non desiderati vanno modellati come quelli desiderati, utilizzando gli stessi strumenti di descrizione.

6.6 Contesto

Una volta compreso il processo è necessario comprendere il contesto in cui va inserito il controllore.
Di solito la nuova attività di controllo si inserisce sul processo assieme ad altre attività di controllo già esistenti con le quali può comunicare. Conviene allora costruirsi un modello che descrive l'ambiente in cui va inserita la nuova attività.
L'ambiente va descritto come insieme di processi e di controllori e tutte le attività vanno modellate.
Ciò può risultare utile anche nel caso in cui gli obiettivi del progetto non si raggiungono attraverso la co-

struzione di un intero controllore, ma attraverso la modifica di una parte di un controllore già esistente (ad esempio, l'automazione della raccolta delle misure in un controllore in cui le decisioni sono prese da uomini).

6.7 Simulazione

A questo punto, se i modelli che abbiamo costruito sono eseguibili, possiamo simulare il comportamento del processo e del contesto, e valutare la correttezza dei nostri modelli e il grado di conoscenza raggiunto.
Possiamo eseguire le parti modellate con le Reti di Petri (a mano, oppure con un calcolatore di aiuto).

Dal punto di vista della correttezza, possiamo ad esempio verificare se il modello del processo esegue dei funzionamenti che esistono nel processo e se tra questi funzinonamenti si trovino quello desiderato e quello non desiderato.

Dal punto di vista del grado di conoscenza il problema di valutazione del lavoro già fatto e più complesso poichè si tratta di fare valutazioni sulla "bontà" dei modelli.
Bisogna farsi un'idea, tramite simulazioni e studio del processo reale, di com'è fatta la mappa di fig. 3 (che è poi la mappa della nostra conoscenza del processo).
Se ad esempio vediamo che l'insieme di ciò che conosciamo (ciò che può e non può accadere) è poco più ampio dell'insieme di ciò che vogliamo (ciò che deve accadere) e abbiamo detto molto poco su ciò che non deve accadere, non possiamo sentirci molto tranquilli, specialmente dal punto di vista della sicurezza del processo.
È probabile che, quando il nostro controllore andrà in servizio, accadranno molte cose di cui non abbiamo sospettato l'esistenza.

Per comodità l'attività di simulazione è qui descritta come l'ultimo passo della prima fase. In realtà il progettista eseguirà più volte il ciclo modello-simulazione man mano che la sua conoscenza cresce.

7. COSA FA IL CONTROLLORE?

Costruire il controllore significa ora prendere il modello (la rete) del processo e il modello del contesto e inserire una nuova attività con nuove connessioni (o modificare attività e connessioni già modellate).

7.1 Modello del controllore come unica attività

Il primo documento modella il controllore come una unica attività e un insieme di risorse connesse sia al processo che al contesto di controllo.

7.2 Specifica delle funzioni svolte dall'attività di controllo

Il secondo documento specifica le funzioni che l'attività deve svolgere senza definire però la struttura interna della stessa.
Bisogna passare cioè dalla definizione astratta "la funzione del controllore è far assumere al processo un comportamento desiderato e tenerlo lontano dal comportamento non desiderato" alla specificazione in un caso concreto della lista delle funzioni che il controllore deve svolgere.

In questa fase vanno definite, almeno a grandi linee, le due strategie (quella nei confronti del comportamento desiderato e quella nei confronti del comportamento non desiderato). La definizione deve essere sufficientemente precisa in modo da poter essere usata per definire successivamente i flussi di comunicazione tra il controllore, il processo e i controllori di livello superiore.

7.3 Specifica delle comunicazioni tra attività di controllo, processo e contesto

La nuova attività (o la preesistente attività modificata) sarà connessa al processo e alle altre attività di controllo con dei flussi di comunicazioni.

Le prime due sono le misure del processo e i comandi verso il processo.
Relativamente ad esse vanno definite le attività di misura e regolazione inserite nel processo e i flussi che le connettono al controllore.
In generale è bene tenere distinti i flussi di comunicazioni stabiliti per mantenere un comportamento desiderato da quelli utilizzati per evitare un comportamento non desiderato.

Essi infatti rispecchiano esistenze funzionali e tecnologiche diverse; ad esempio l'affidabilità richiesta su di un flusso di comandi per il normale funzionamento e quella su un flusso di comandi di blocco in situazioni di emergenza possono essere ben diverse.
Le altre due classi sono le misure dal controllore e i comandi per il controllore.

Si è detto che un controllore più un processo è un nuovo processo che necessita di attività di controllo; vanno quindi definiti quali sono le misure del nuovo processo (ad esempio gli allarmi generati dal controllore a fronte di un comportamento non desiderato del processo) e i comandi (ad esempio per inizializzare il controllore o disattivarlo o per cambiare parametri delle strategie).
Anche per queste due altre classi di flussi di comunicazione è bene distinguere ciò che è relativo ad un comportamento desiderato (del controllore più il processo) da ciò che è relativo ad un comportamento non desiderato.

In generale, per tutti i flussi di comunicazione descritti a questo livello, è bene che la descrizione sia funzionale e non tecnologica (o almeno non principalmente).

7.4 Prestazioni e vincoli

Qualifichiamo ora ulteriormente l'attività associandole delle prestazioni che deve rispettare e dei vincoli a cui deve sottostare.

Per le prestazioni si può chiedere ad esempio quanti cicli del processo devono essere memorizzati in un archivio storico o qual'è il tempo massimo ammesso tra la generazione di una situazione d'allarme sull'impianto e l'intervento del sistema di blocco, e così via.

Egualmente possono essere descritti dei vincoli che le successive fasi di produzione del controllore devono rispettare. Ad esempio il rispetto di protocolli di comunicazione verso altri calcolatori o l'uso di un definito hardware per la realizzazione.

7.5 Simulazione

Possiamo ora ad esempio verificare se, per svolgere le funzioni listate, sono state modellate tutte le comunicazioni necessarie con il processo.
Possiamo poi simulare dei comportamenti del processo e vedere quali flussi di comunicazione giungono al controllore. Poichè abbiamo modellato il comportamento desiderato e descritto uno schema della strategia, possiamo anche tentare delle simulazioni di comportamento del controllore e generare flussi di comanid verificando che cosa accade nel modello del processo.

È possibile anche simulare che cosa accade nel contesto a causa dell'inserimento del controllore (quali informazioni vengono fornite e a chi, quali modifiche organizzative nascono,...).

In pratica, anche se a livello molto generale ed eventualmente non rigoroso, possiamo però cominciare a renderci conto di come funzionerà il nuovo sistema processo + controllore nel contesto, in una fase del lavoro in cui siamo ancora molto lontani dal prodotto finito e abbiamo solo delle specifiche di ciò che dobbiamo costruire.

In questo modo si può tentare non solo di valutare la correttezza delle specifiche ma anche di verificare che non siano sfuggiti aspetti importanti.
Se ad esempio il contesto è complesso, con molti controllori a più livelli gerarchici fatti da uomini e da macchine e con più modi di funzionamento (manuale, semiautomatico, automatico,...) è molto utile il lavoro di comprensione di cosa accadrà nel contesto inserendo il nuovo controllore.

8. COME FA IL CONTROLLORE?

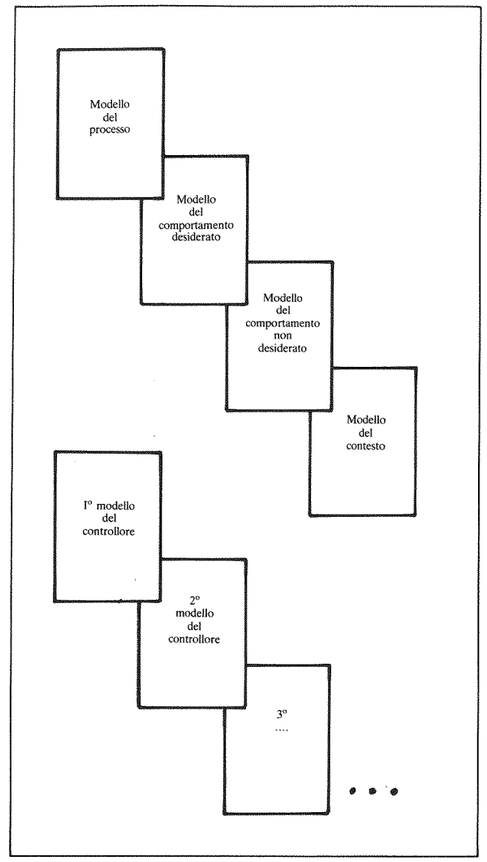


Fig. 5

Il lavoro svolto fino ad ora ha generato cinque modelli (vedi Fig. 5), i primi quattro dei quali contengono conoscenza del processo e dell'ambiente e desideri, mentre l'ultimo è la prima specifica del controllore.
Il problema è ora quello di costruire il controllore partendo dalla prima specifica e incorporandogli la conoscenza e i desideri.

8.1 Struttura funzionale

L'unica attività contenente una lista di funzioni va composta in una rete di risorse, di attività e di connessioni che realizzino le funzioni descritte (Fig. 6)

La struttura di un generico controllore ci suggerisce che questo lavoro di fabbricazione di un'architettura si può eseguire in due passi distinti (Fig. 7).

– Prima inseriamo nel controllore i modelli del pro-

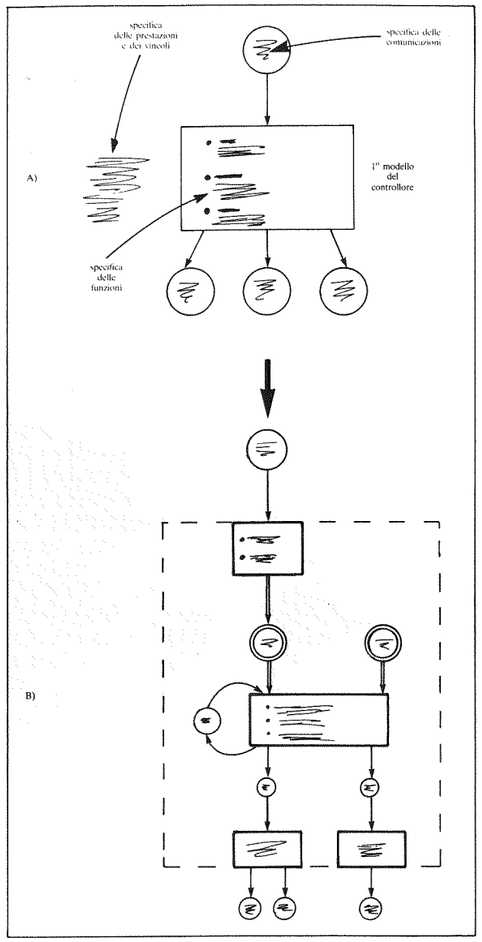


Fig. 6

cesso, del comportamento desiderato e del comportamento non desiderato.

- Poi costruiamo tutte le attività (funzioni) che usano i modelli.

8.2 Inserire modelli

Nel processo vi sono attività di trasformazione svolte da uomini e macchine e flussi di beni tra le attività; egualmente si può pensare che nel modello inserito nel controllore vi siano attività di trasformazione svolte da programmi e flussi di dati.

Poichè il processo ha in generale una dinamica, non sono in generale sufficienti strutture di dati (che sono

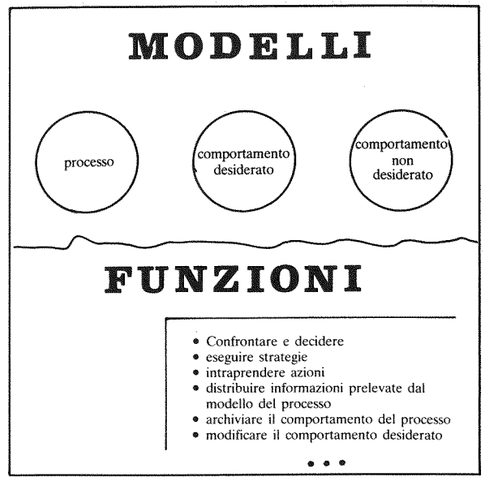


Fig. 7

una fotografia statica della realtà) ma servono anche programmi che muovono i dati.

Si può pensare che ad ogni attività svolta da una macchina (o uomo) corrisponda un'attività svolta da un programma e che ad ogni oggetto contenuto in un flusso corrisponda una struttura di dati contenuta in un opportuno contenitore per dati (Fig. 8).

A questo punto è bene non porsi (o porsi poco) il problema delle attività (funzioni) che useranno i modelli.
In evoluzioni successive del controllore potranno essere modificate, aggiunte o tolte funzioni ma senza modificare i modelli fino a che non vengano modificati o il processo o i comportamenti relativi ad esso.

La corrispondenza macchina/uomo ≡ programma e oggetto ≡ dati è abbastanza astratta e potrà essere necessario tradurre il modello in altri modelli di tipo diverso per poterlo effettivamente realizzare.

Ora il controllore possiede conscenza e desideri, ma la conoscenza non è aggiornata in base a ciò che accade nel processo e il controllore non la sa usare per soddisfare i propri desideri.

8.2 Costruire funzioni

Costruiamo adesso tutte le attività (e i flussi che le connettono) che realizzano le funzioni descritte dal documento "funzioni".
Alcune di queste attività sono esistenti in un qualunque controllore, mentre altre possono essere specifiche di un definito controllore; se poi il sistema da costuire è una parte di un controllore (ad esempio uno

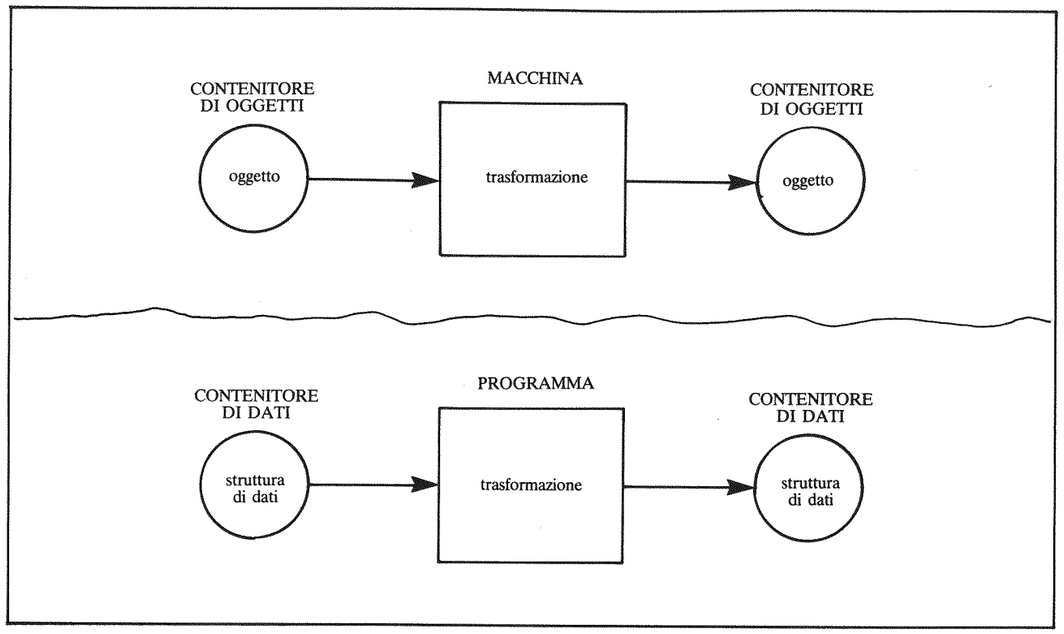


Fig. 8

strumento di misura anche se sofisticato) mancheranno delle attività tra quelle in generale presenti.

La Fig. 9 mostra le principali attività del controllore:

- L'osservatore preleva misure dal processo e aggiorna il modello che è nel controllore in base a ciò che accade nel processo.

- I decisori confrontano ciò che accade nel modello con il comportamento desiderato e il comportamento non desiderato e, se è il caso, attivano gli strateghi.

- Gli strateghi possono usare il modello del processo per simulare le conseguenze del loro intervento e possono usare i comportamenti desiderati e non per valutarne le conseguenze.
  Essi generano ordini per gli azionatori.

- Gli azionatori generano i comandi per il processo.

Aggiungendo funzioni si possono modificare i tipi di controllori (ad esempio può essere inserita una attività che permette ad un operatore di modificare il comportamento desiderato).

8.3 Realizzazione tecnologica

Il controllore a questo punto è modellato come una rete di risorse, attività e connessioni.
Si tratta ora di far eseguire le attività a qualcuno (un programma, un uomo, un pezzo di elettronica ad hoc,...), di mettere le risorse in qualche posto (aree di memoria, files, pezzi di carta,...) e di stabilire fisicamente le connessioni tra risorse e attività.

Non è detto però che il modello del controllore che possediamo sia immediatamente realizzabile fisicamente. Esso può richiedere delle trasformazioni; se ad esempio una attività viene svolta da un programma bisognerà dettagliarla fino a scrivere il codice del programma.

8.4 Decomposizione

Se il controllore è complicato non ci sarà un solo raffinamento del tipo di quello a fig. 6, ma la prima descrizione della struttura funzionale dovrà essere raffinata ulteriormente introducendo dettagli via via crescenti in modo ordinato.

Il criterio che possiamo seguire per raffinare le descrizioni è lo stesso usato per passare dalla fig. 6a che era un'unica attività con delle risorse connesse, alla fig. 6b che fornisce dettagli mediante un'espansione

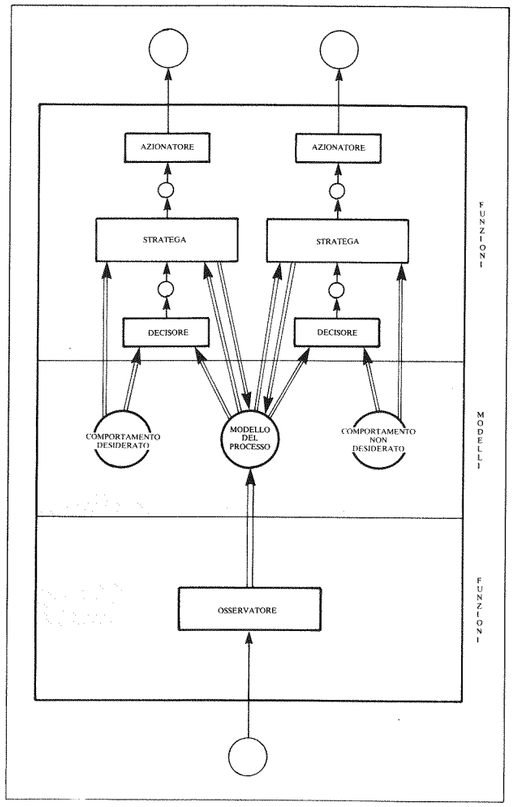


Fig. 9

dell'attività.
Si possono espandere sia attività che risorse e il procedimento può essere iterato più volte.
Durante il procedimento di raffinamento del modello si eseguono simulazioni e valutazioni.

Nel processo di decomposizione è necessario seguire delle regole legate formalmente al linguaggio di descrizione utilizzato, e il cui significato è garantire in qualche modo che la serie dei modelli sempre più dettagliati rappresenti sempre lo stesso sistema a livelli di astrazione diversi.

Il procedimento descritto è però solo formale, infatti molte strutture diverse possono essere costruite a partire dallo stesso modello iniziale.
La struttura di un generico controllore fornisce una guida generale nella identificazione delle attività principali e delle loro relazioni.
Si possono poi adottare dei criteri di assegnazione delle funzioni alle attività (e quindi ad esempio ai programmi che le realizzano) come ad esempio rendere il più possibile funzionalmente uniforme ognuna di esse e ridurre al minimo le connessioni tra le attività stesse.

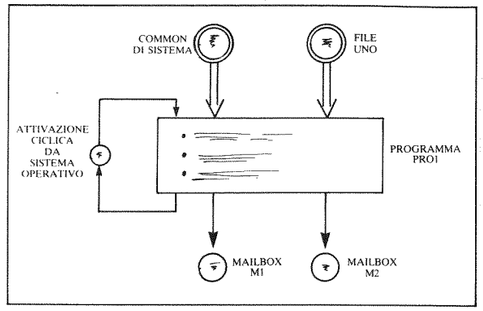


Fig. 10

8.5 Chi esegue le attività e dove sono le risorse.

Le architetture del tipo di quella a fig. 9 sono solamente funzionali.
Il passo successivo è scrivere accanto ad ogni attività qual'è l'oggetto fisico che la esegue e accanto ad ogni flusso di comunicazione qual'è il contenitore dei dati.
Bisogna cioè passare da un modello che potrebbe essere realizzato da oggetti fisici del tutto diversi ad una sua specifica realizzazione.
La Fig. 10 mostra ad esempio un frammento di realizzazione.

La decomposizione funzionale in attività, risorse e connessioni può procedere fino ai minimi dettagli; a un certo punto della decomposizione bisognerà definire gli oggetti concreti. Ma quando conviene passare dai modelli funzionali a quelli tecnologici?

Una realizzazione tecnologica è sempre una specifica realizzazione fatta usando certi mezzi messi a disposizione e non altri ed è a volte una mediazione tra struttura funzionale e strumenti disponibili.
Allora la definizione di una struttura funzionale va portata avanti fino a quando si ha una chiara comprensione del funzionamento completo del controllore. Solo in seguito vanno effettuate le mediazioni.
Nei documenti relativi alla realizzazione tecnologica di una struttura funzionale deve comunque apparire chiaramente dove e perchè esse sono state adottate.
Ciò rende possibile costruire una nuova versione tecnologica del controllore riutilizzando i modelli funzionali dello stesso.
Una volta definiti i componenti fisici del controllore i procedimenti costruttivi variano a seconda del tipo di componente.
Se il componente è un pezzo di elettronica si useranno specifiche tecniche di costruzione;
se invece è un programma, disponendo di metodi o strumenti di sintesi del programma a partire dalle sue specifiche, si usano quelli, altrimenti si può continuare la decomposizione funzionale del programma utilizzando ad esempio l'italiano strutturato, fino alla definizione delle aree di dati e alla stesura del codice in programmazione strutturata; se invece è un uomo che svolge un'attività bisognerà ad esempio occuparsi dei problemi organizzativi connessi.

8.6 Simulazione

Durante tutto il processo di fabbricazione dei modelli del controllore a partire dal primo modello ci si trova iteravamente ad eseguire l'operazione descritta in fig. 6.
La fig. 6a definisce delle specifiche di comportamento di una macchina, mentre la fig. 6b come è fatta la macchina per soddisfare le specifiche.
La prima rete descrive con un comportamento desiderato mentre la seconda ha un comportamento che deve soddisfare quello desiderato.

Se raffiniamo la rete b) essa stessa (i pezzi che isoliamo) diventa comportamento desiderato di un'altra rete che dobbiamo inventare e così via.
Quando scriviamo i programmi, se usiamo ad esempio reti di Petri fino a un certo dettaglio e poi pseudocodice e poi un linguaggio di programmazione, la rete è un comportamento desiderato della macchina descritta dallo pseudocodice, lo pseudocodice è il comportamento desiderato della macchina descritta in linguaggio di programmazione e il codice è il comportamento desiderato della macchina hardware che lo ospita.

Dopo ogni raffinamento devono essere eseguite delle simulazioni per valutare in quale misura il comportamento della macchina costruita è quello desiderato (descritto dalla macchina precedente).

9. DEFINIZIONI ED ESEMPI

Descriviamo ora brevemente come abbiamo usato le reti di Petri per costruire modelli di processi e controllori.
Seguono poi alcuni esempi tratti da applicazioni nell'industria siderurgica.

Esempio 1

Il primo esempio si riferisce ad un sistema che serve per inseguire in tempo reale il flusso di pezzi lungo un laminatoio per tubi senza saldatura e per estrarre un insieme di informazioni usate al fine di controllare l'avanzamento della produzione.

La fig. a.1 schematizza l'impianto costituito da un

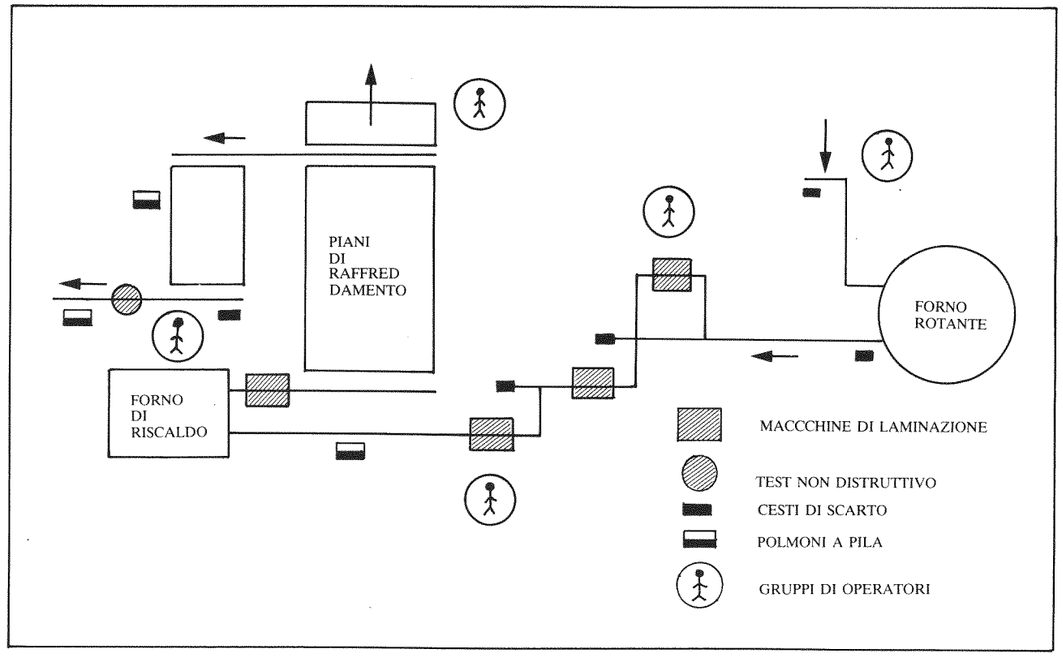
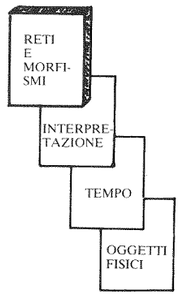


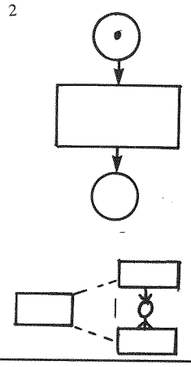
Fig. a.1 - Schema dell'impianto

1



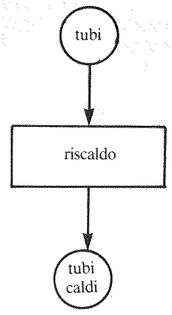
Una rete può essere vista come un disegno (fatto di cerchi, rettangoli ed archi orientati) su un foglio bianco, con l'aggiunta di più testi su fogli trasparenti sovrapponobili.
Il disegno rappresenta una struttura formale (rete non interpretata).
Ad uno stesso disegno possono essere associati più significati mediante dei trasparenti.

2



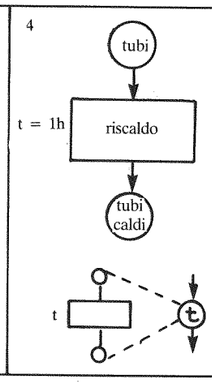
La rete è del tipo posti e transizioni e contiene posti, transizioni ed archi orientati.
Nella rete si muovono marche tutte uguali (non sono individui distinti) secondo una regola di simulazione.
Mediante morfismi la rete può essere espansa o concentrata.

3



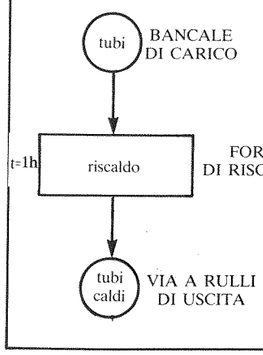
È aggiunto il trasparente della interpretazione.
I posti modellano contenitori di risorse e dentro i cerchi si possono scrivere i nomi e le descrizioni delle risorse che possono entrare nel contenitore.
Le transizioni (rettangoli) modellano contenitori di attività che possono essere descritte dentro i rettangoli. le marche diventano modelli (individuali) delle risorse e la regola di simulazione può tenere conto delle descrizioni di attività.

4



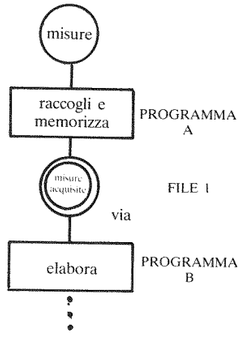
È aggiunto il trasparente del tempo (La attività di riscaldo si svolge in un tempo t = 1.h).
Possono essere utilizzati anche contenitori di risorse che contengono un tempo t (orologi). Quando una marca entra nel contenitore, l'orologio entra in funzione e, dopo un tempo t, rende disponibile la marca.

5



Il trasparente degli oggetti assegna delle corrispondenze tra oggetti fisici a rete.
Ad ogni contenitore di risorse è associato il nome dell'oggetto fisico modellato dal contenitore stesso e ad ogni contenitore di attività è associato il nome di chi la compie.

6



Utilizziamo anche una notazione particolare:

è un contenitore di risorse in cui c'è sempre una risorsa.

è un'operazione di scrittura (modifica della risorsa).

è un'operazione di lettura (copia della risorsa).

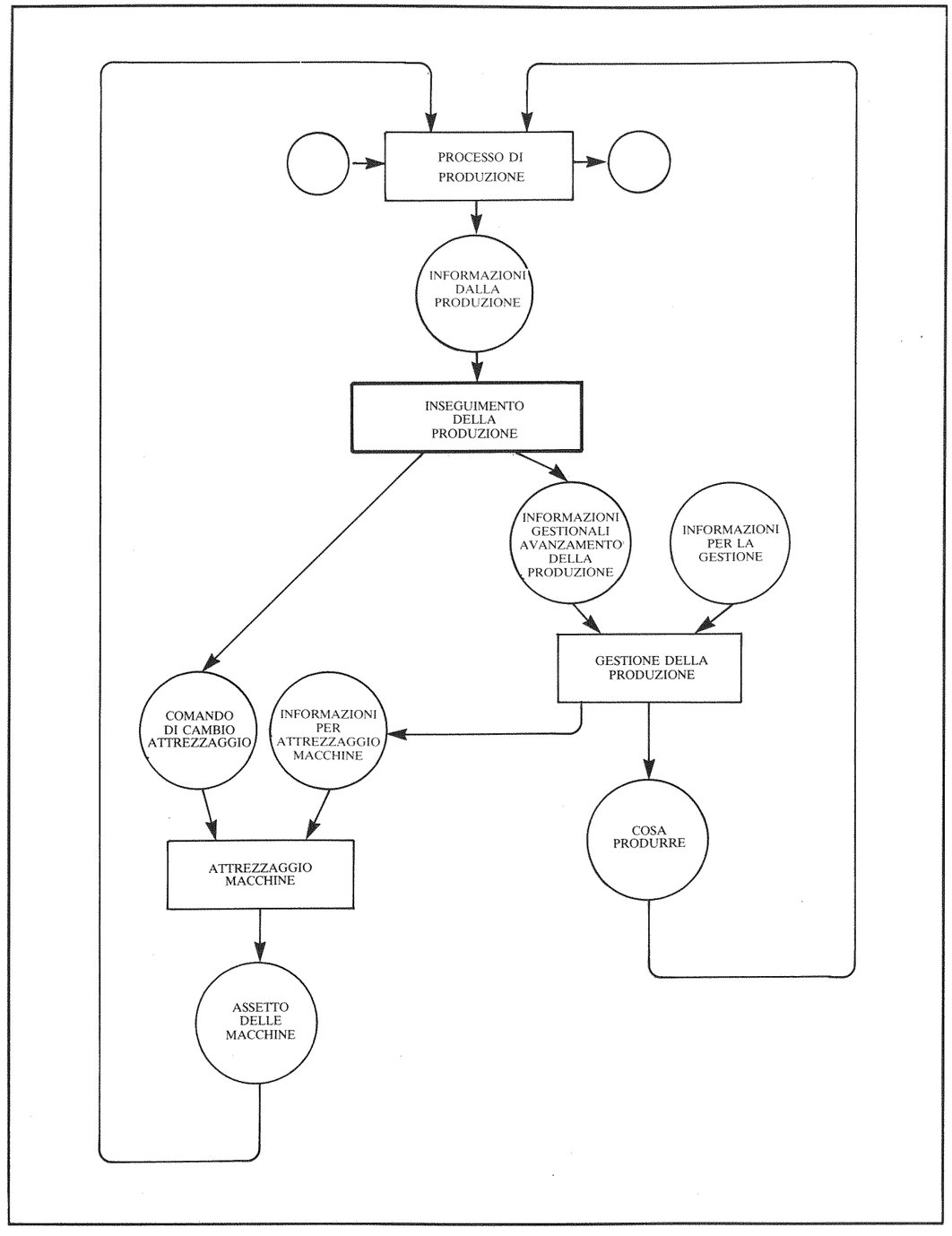


Fig. b.1 - L'ambiente di controllo

forno rotante, da più macchine di laminazione e da un forno di riscaldo collegati da vie a rulli, letti di raffreddamento e da altre macchine per movimentare i pezzi. Più gruppi di operatori, aiutati da più sistemi di automazione e raccolta dati, controllano il flusso e la lavorazione dei pezzi.

La fig. b.1 mostra l'ambiente di controllo connesso alla attività di inseguimento della produzione. Vediamo che l'inseguimento fornisce misure per due anelli di controllo: l'attrezzaggio delle macchine e la gestione della produzione.

Per costruire il sistema, l'impianto è stato suddiviso in zone e di ognuna sono stati prodotti uno schema tecnologico e la rete dei movimenti. In questo modo è stato modellato l'intero flusso dei pezzi sull'impianto.

La fig. c.1 mostra lo schema di una zona e la fig. d.1 è un frammento della rete che modella il processo che si svolge in questa zona.

La rete di fig. e.1 e la fig f.1 descrivono invece il sistema di inseguimento modellato come unica attività con tutti i flussi connessi e con la lista delle funzioni che deve realizzare. Questa attività è poi espansa (fig. g.1) in più attività e risorse. La parte chiamata in figura "mondo fisico" contiene il modello del processo ed una attività che, in base alle misure raccolte sull'impianto, mantiene aggiornato il modello; l'altra parte, chiamata "mondo applicativo" contiene tutte le attività ("applicazioni") che utilizzano le informazioni prelevate dal modello del processo.

Il modello del processo è stato implementato a partire dalle reti, utilizzando una struttura di records concatenati da puntatori (fig. h.1), e un programma che contiene le regola per il movimento dei records (ogni volta che giunge un segnale di movimento di un pezzo dell'impianto, un record viene mosso).
Le reti modello del processo sono poi state ampiamente utilizzate eseguendo delle simulazioni per decidere dove mettere i sensori per prelevare le informazioni di passaggio pezzo.

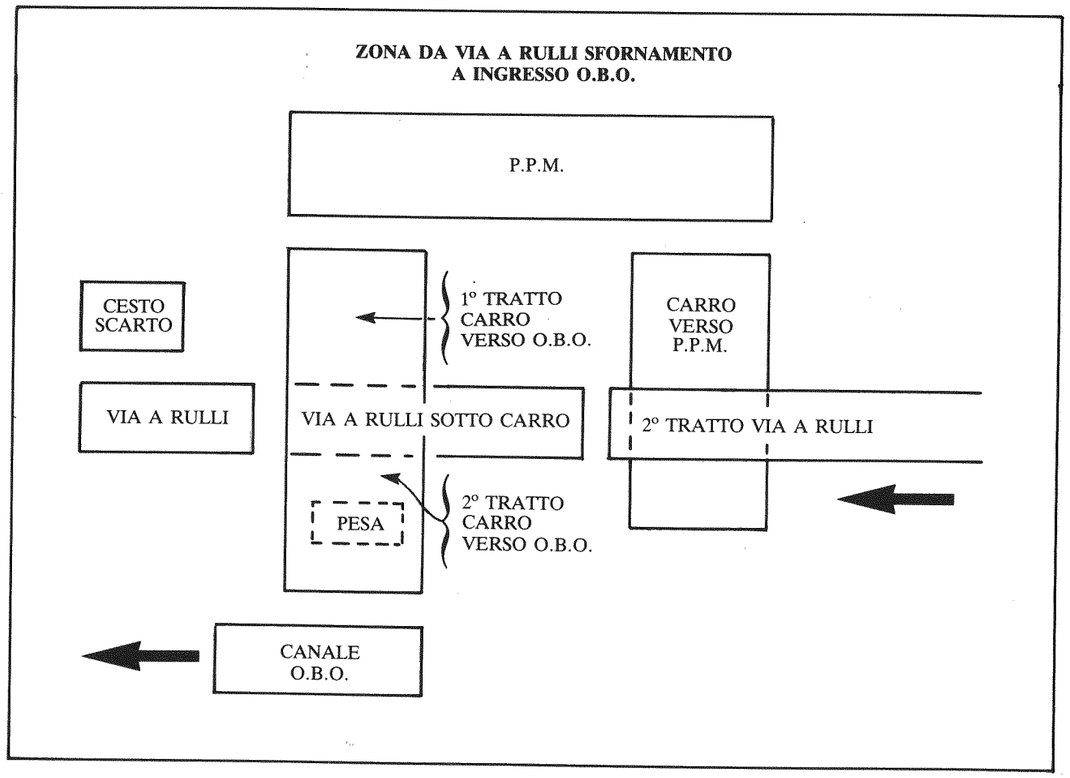


Fig. c.1 - Schema di una zona d'impianto (P.P.M. e O.B.O. sono le sigle di 2 macchine per la laminazione).

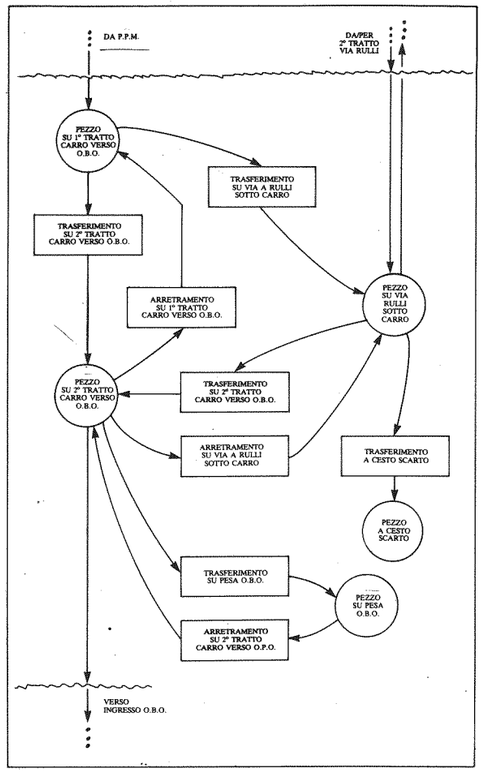


Fig. d.1 - Flusso dei pezzi nella zona

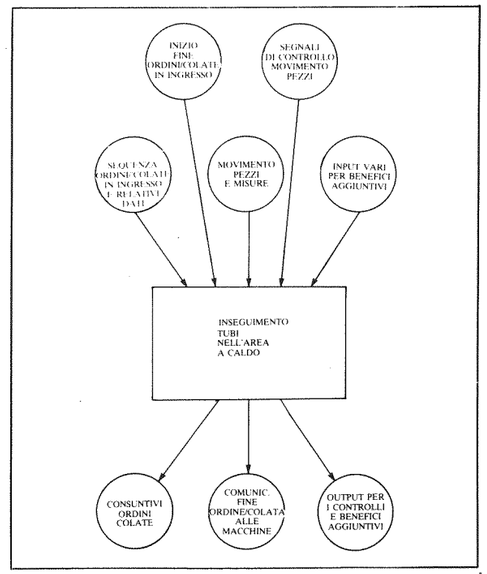


Fig. e.1 - Sistema di inseguimento

LISTA DELLE FUNZIONI DEL SISTEMA
DI INSEGUIMENTO:

- aquisizione delle informazioni di movimento pezzi e controllo movimento (RESET ERRORE) da impianto
- gestione avanzamento ordini di laminazione
- comunicazione agli operatori:
  . degli ordini di laminazione sull'impianto
  . della previsione di cambio attrezzaggio
- produzione di informazioni gestionali:
  . consuntivi ordini/colate
  . consuntivi giornate/turno
  . riepilogo interruzioni
- produzione del rapporto contenente le misure raccolte.

Fig. f.1 - Sistema di inseguimento

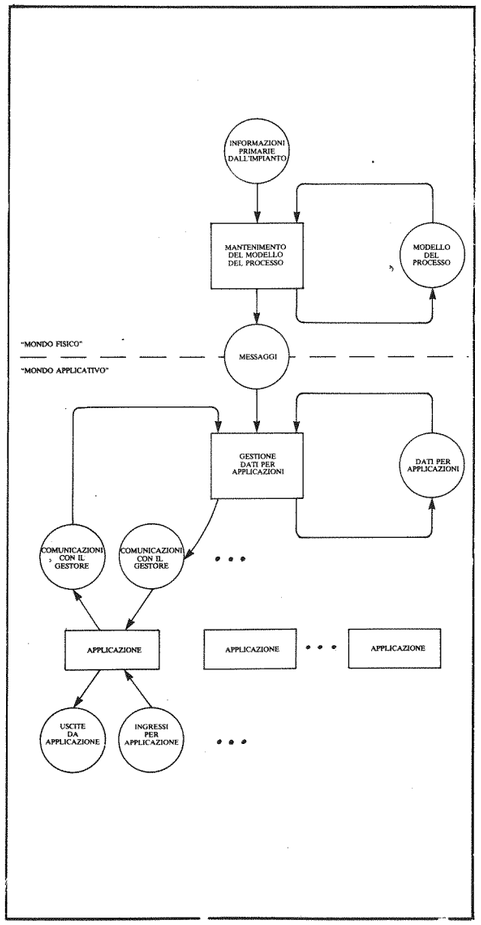


Fig. g.1 - architettura del sistema di inseguimento

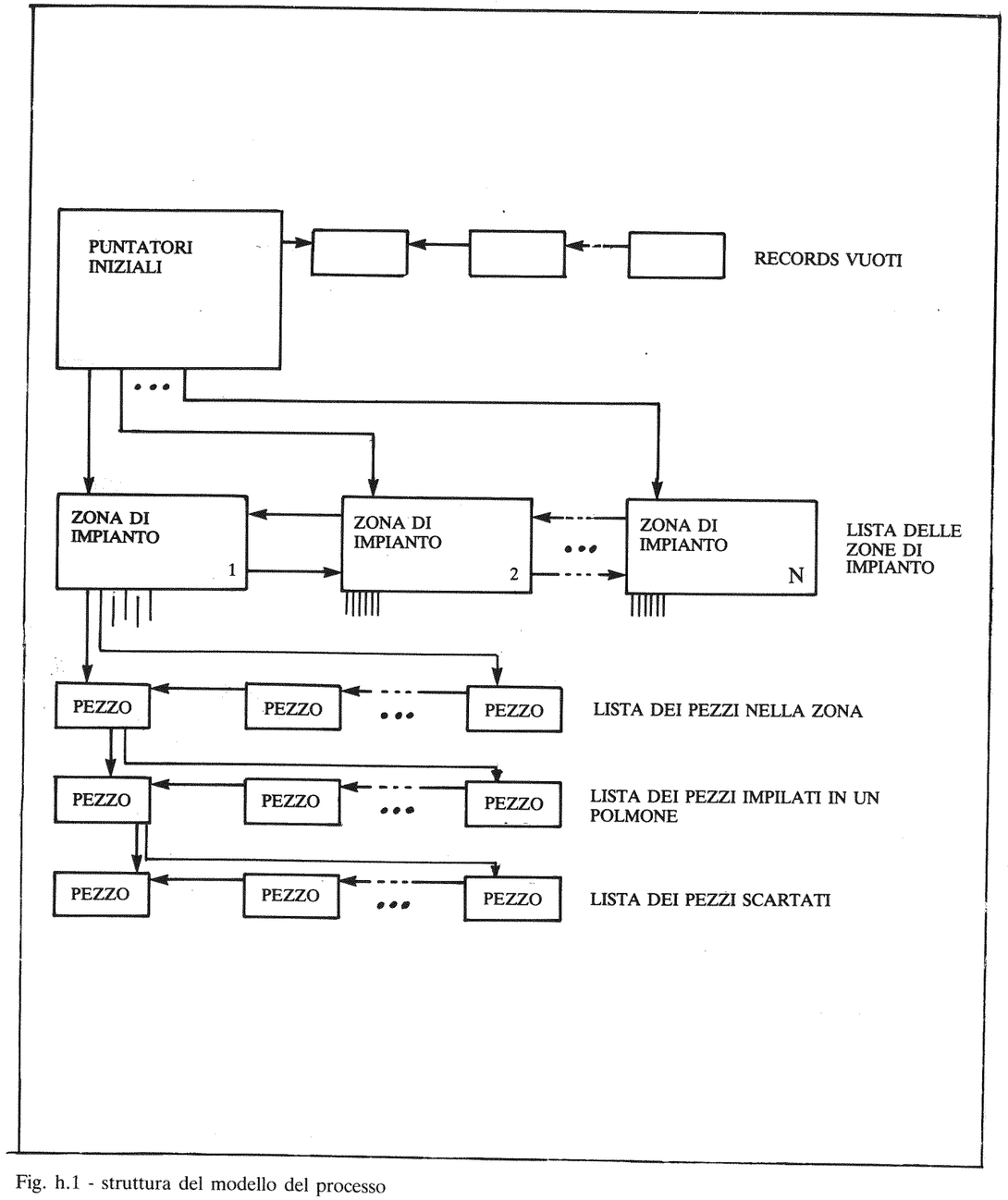


Fig. h.1 - struttura del modello del processo

La rete di fig. i.1 e la spiegazione associata (fig. l.1) mostrano un frammento della rete complessiva dei programmi che realizzano le "applicazioni".

La rete e il testo sono le specifiche di un programma che permette all'operatore di inserire nel sistema e manipolare le informazioni sugli insiemi di pezzi (lotti) che vengono introdotti nell'impianto.

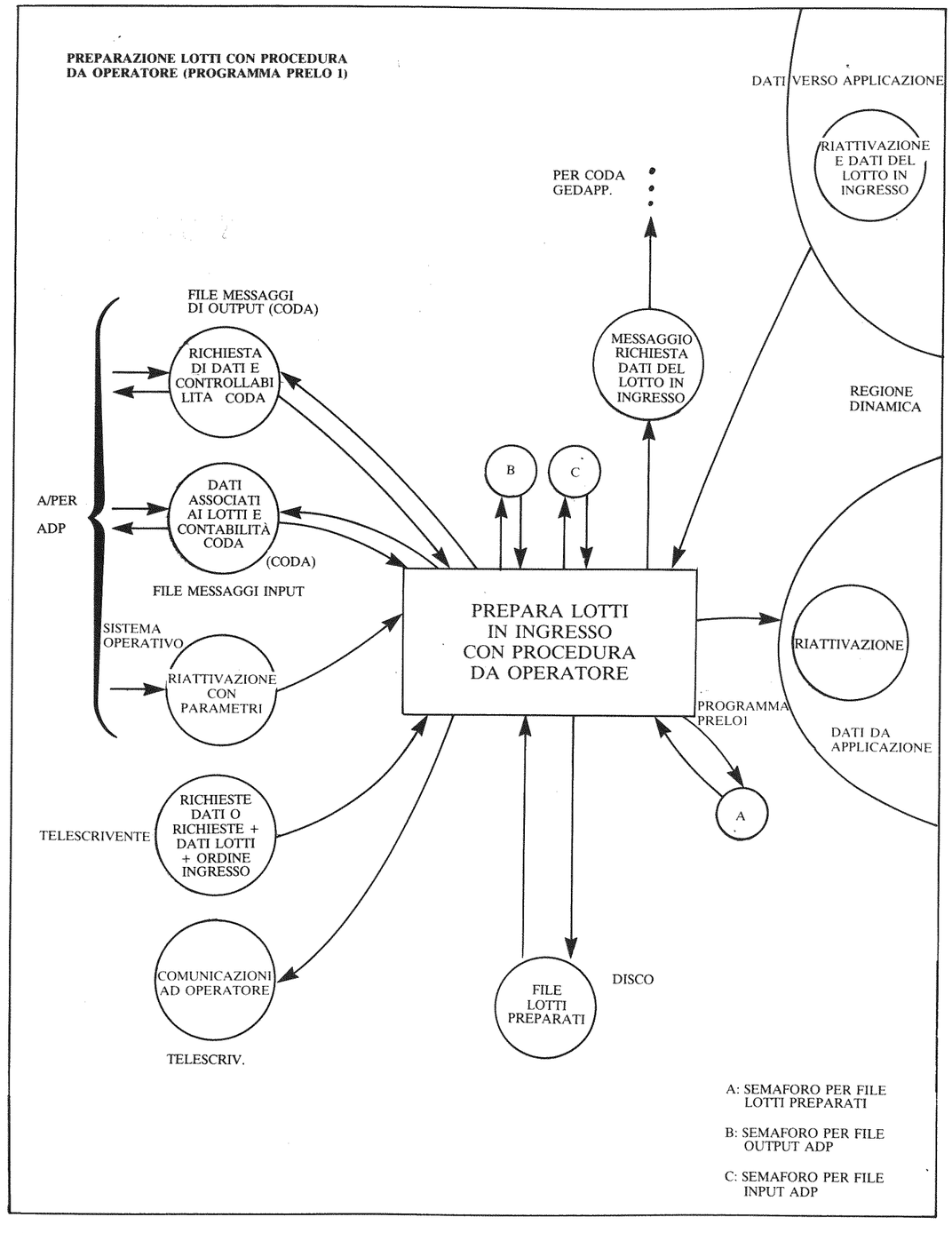


Fig. i.1 - Specifica di un programma (applicazione)

Funzionamento di PRELO1

Scopo di PRELO1 è la gestione da operatore del FILE "LOTTI PREPARATI" che contiene i pacchetti dati relativi ai lotti che devono entrare in lavorazione e la loro sequenza di ingresso.

Il programma PRELO1 può realizzare le seguenti attività:

1) Inserzione di un nuovo lotto
2) Lettura ed eventuale aggiornamento dei lotti esistenti

1) Per l'inserzione di un nuovo lotto i dati relativi sono prelevati dall'acquisitore dati di produzione (ADP) o comunicati da operatore.

   Il lotto preparato è accodato ai lotti esistenti mantenendo però come ultimo della coda il lotto di "default".

   Nota: il lotto di "default" è un lotto con dati arbitrari che deve permettere la gestione dei pezzi che entrano sull'impianto e per i quali non è stato preparato alcun lotto. L'associazione dei dati definitivi è fatto in un secondo tempo da operatore tramite la funzione di correzione dei lotti sull'impianto.

2) Per la lettura ed eventuale aggiornamento dei lotti esistenti non ancora entrati sull'impianto si prevedono le seguenti attività:
   - stampa lotti
   - correzione di un lotto
   - cancellazione di un lotto
   - cancellazione totale
   - cambiamento della sequenza di ingresso

Va distinto il caso particolare del 1° lotto del file lotti preparati.

Se dal 1° lotto (che è anche il lotto di ingresso) non sono ancora entrati pezzi sull'impianto esso può essere letto ed aggiornato come tutti gli altri lotti del file.

Se invece è già entrato anche solo 1 pezzo del 1° lotto del file esso è a tutti gli effetti trasparente all'operatore e non è più modificabile tramite il programma PRELO1 (Ciò vale anche se è il lotto di default).

L'informazione che è entrato almeno un pezzo del 1° lotto è prelevata dalle strutture dei dati applicativi tramite un colloquio del tipo lento (comunicazione del caso 3 tipo LA)

Questo programma è necessario anche se viene utilizzata la preparazione lotti con procedura da ADP sia in caso di caduta collegamento con ADP che in caso di ORDINI/COLATE non previsti a lancio operativo.

Fig. 1.1

Esempio 2

Eaminiamo ora un problema di controllo della laminazione di forati (un forato è un cilindro d'acciaio con un foro passante) per la produzione di tubi d'acciaio senza saldatura. La fig. a.2 mostra l'ambiente di controllo. L'attività di controllo non è completamente automatizzata ma è svolta parzialmente da uomini e parzialmente da macchine.

Il processo complessivamente è discontinuo (ogni laminazine di forato è un evento a sè stante) ma è continuo durante la laminazione di un singolo forato.
La fig. b.2 descrive il processo discontinuo mentre il processo continuo (l'attività "laminazione") è descritto da un sistema di equazioni.

Il comportamento desiderato del processo è illustrato in fig. c.2, considerando l'insieme di misure che si

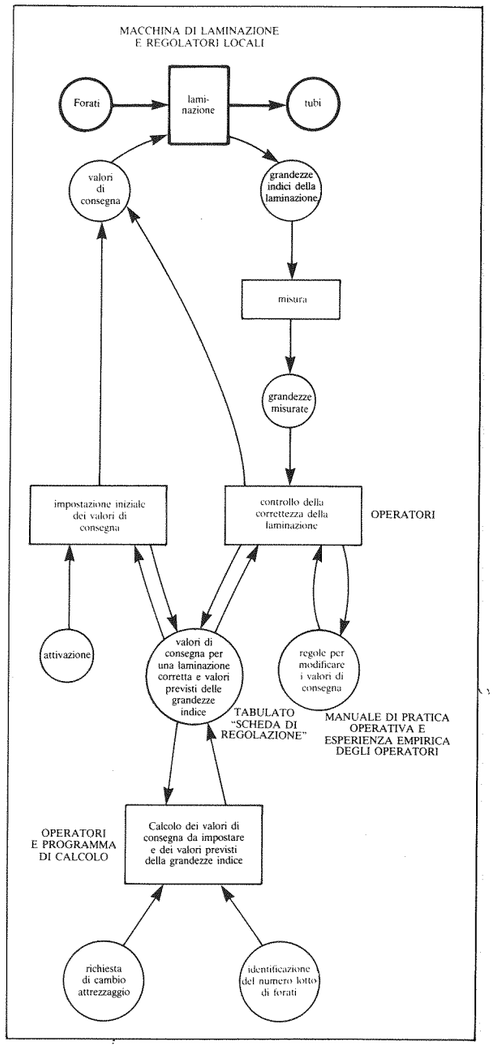


Fig. a.2 - Il problema di controllo

possono raccogliere durante la laminazione (ad esempio le forze di separazione tra le gabbie del laminatoio).
La fig. d.2 invece descrive qual'è il comportamento non desiderato del processo.

10. RINGRAZIAMENTI

Il punto di partenza di questo lavoro è stato l'articolo di G. degli Antoni, M. Maiocchi. R. Polillo. B. Zorita "How, What, Why" [1].
Fondamentale sono state le discussioni con G. Degli Antoni dell'Istituto di Cibernetica dell'Università di Milano.
Anche ad A. Beltramini dell'Istituto di Cibernetica dell'Unversità di Milano devo molte discussioni e stimoli.
Il modo di usare le Reti di Petri ad alto livello per modellare sistemi l'ho sostanzialmente appreso dal lavoro di G. Castelli, H. Haus, M. Maiocchi, "Una metodologia di stesura e verifica di specifiche funzionali di procedure e programmi software" [6].
Gli esempi riportati sono tratti in gran parte da lavori svolti presso la società Dalmine s.p.a. per la cui gentile concessione ho potuto pubblicare materiale di sua proprietà. Ho svolto buona parte del lavoro (e delle discussioni associate) con L. Tarantola e R. Fumagalli del gruppo Sistemi Controllo Processo della società Dalmine.
A tutte queste persone sono debitore e le ringrazio.

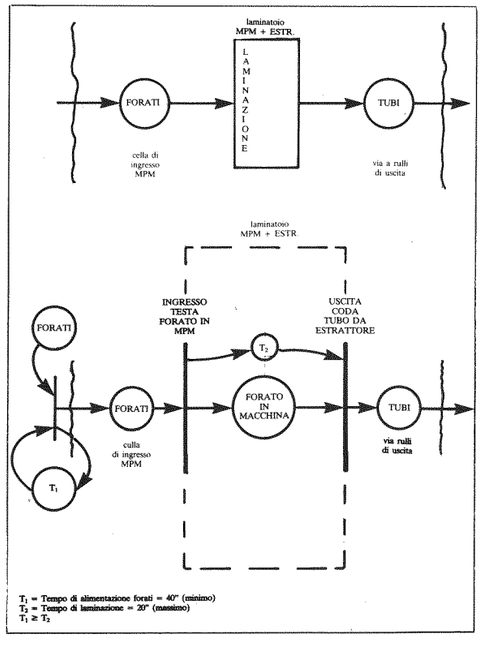
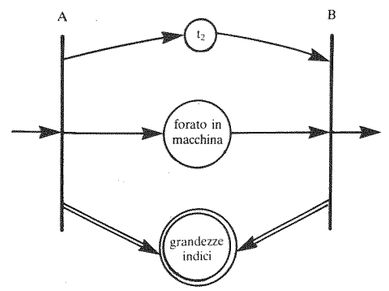


Fig. b.2 - Il processo di laminazione

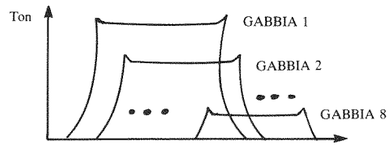
COMPORTAMENTO DESIDERATO



1) Una volta che il forato è entrato in macchina (è accaduta l'attività A) dopo un tempo t_2 deve essere prodotto il tubo in uscita (deve accadere B).

2) Se la 1) è rispettata, durante il tempo t_2 le grandezze indici devono assumere definiti valori:

   - le forze di separazione tra le gabbie devono avere la seguente forma:



   - Un insieme di valori medi e istantanei e di rapporti rispetto a valori limite, opportunamente estratti dalle grandezze indici, devono essere compresi entro definiti intervalli.

Fig. c.2 - Comportamento desiderato

COMPORTAMENTO NON DESIDERATO

1) Il forato resta in macchina.

2) Il forato esce ma le grandezze indici descrivono una laminazione pericolosa o per la qualità del tubo o per l'integrità dell'impianto:

   - la forma delle forze di separazione dice che il fluire del materiale tra le gabbie non è corretto

     laminatoio in trazione laminatoio in compressione

   - insieme di valori medi e istantanei estratti dalle grandezze indice è oltre dei fissati valori limite.

   - ci sono brusche variazioni tra le forze di separazione relative ad un tubo e quelle relative al successivo; si può sospettare la rottura della gabbie.

   - C'è una tendenza di valori estratti dalle grandezze indice a crescere, o decrescere, nel tempo; ciò può essere indice di progressivi degradi del laminatoio.

Fig. d.2 - comportamento non desiderato

11. BIBLIOGRAFIA

[1] G. Degli Antoni, M. Maiocchi, R. Polillo, B. Zonta, HOW, WHAT, WHY, Proc. IV Annual Honeywell International Software Conference, Minneapolis, 1980.

[2] C.A. Petri, GENERAL NET THEORY, FROM "COMPUTERING SYSTEM DESIGN". Proceedings of the Joint IBM Unversity of Newcastle Upon Type Seminar 7th-10th September 1976.

[3] C.A. Petri COMMUNICATION DISCIPLINES, FROM "COMPUTERING SYSTEM DESIGN". Proceedings of the Joint IBM Unversity of Newcastle Upon Type Seminar 7th-10th September 1976.

[4] C.A. Petri, MODELLING AS A COMMUNICATION DISCIPLINE, Measuring Modelling and Evaluating Computer Systems, North Holland Publ., 1977.

[5] S. Beer, R. Espejo, M. Grande, H. Schwember, IL PROGETTO CYBERSYN CIBERNETICA PER LA DEMOCRAZIA a cura di F. De Cindio e G. De Michelis, CLUP - CLUED Milano 1980.

[6] G. Castelli, G. Haus, M. Maiocchi, UNA METODOLOGIA DI STESURA E VERIFICA DI SPECIFICHE FUNZIONALI DI PROCEDURE E PROGRAMMI SOFTWARE, Note di Software n. 10/11 - 1979.

[7] A. Beltramini, UNA METODOLOGIA PER IL PROGETTO DI SISTEMI DI CONTROLLO DI PROCESSO BASATA SULLE RETI DI PETRI, Note di Software n. 10/11 - 1979.

[8] Mohamed Moalla, SPECIFICATION ET CONCEPTION SURE D'AUTOMATISMES DISCRETS COMPLEXES, BASEES SUR L'UTILISATION DU GRAFCET ET DES RESEAUX DE PETRI, These présentée a l'Université Scientifique et Medicale de Grenoble et a l'Istitut National Blytechnique de Grenoble Juillet 1981.

[9] S. Perego, PROGETTO PER L'ABBINAMENTO DI DUE PRESSE PIEGATRICI IDRAULICHE, Tesi presso l'Università degli studi di Milano, Facoltà di Scienze, Istituto di Fisica, Anno Academico 1982.

[10] P. Salvaneschi, AN AUTOMATIC PIPES TRACKING SYSTEM AT NEW DALMINE ROLLING MILL. SYSTEM DESCRIPTION AND USED METHODS BASED ON PETRI NETS MODEL, Proceedings of the Digital Equipment Computer Users Society Hamburg - September 1981.

[11] GESTIONE A PASSO VARIABILE DEL FORNO NTM PER OTTIMIZZARE LA CADENZA DI SFORNAMENTO CON BILLETTE DI GROSSA SEZIONE, Documento inerno Sistemi Controllo Processo - Dalmine spa, Maggio 1983.

[12] ARRESTO AUTOMATICO DELLO SFORNAMENTO PER RIDURRE IL RISCHIO DI DANNI IMPIANTISTICI AL CAMBIO PASSO (SCACCHIERA/PARALLELO), Documento interno Sistemi Controllo Processo - Dalmine spa, Dicembre 1983.

[13] S. Tosato, A Fornaro, R. Fumagalli, P. Salvaneschi, IMPOSTAZIONE E CONTROLLO DEL PROCESSO DI FABBRICAZIONE DEI TUBI SUL LAMINATORIO CONTINUO A MANDRINO TRATTENUTO DELLO STABILIMENTO DI DALMINE (NUOVO TRENO MEDIO), Bollettino Tecnico Finsider n. 392 - settembre/dicembre 1981.

LA QUALITÀ DEL SOFTWARE PER PERSONAL COMPUTER: UNA DISCUSSIONE (°)

Roberto Polillo
Istituto di Cibernetica - Università di Milano
Etnoteam Spa

RIASSUNTO

La presente nota esamina i principali attributi di qualità di una particolare classe di prodotti software per personal computer: i cosiddetti "office productivity tools" (sistemi di word-processing, spreadsheet, data base e file management systems, eccetera).
Tali attributi di qualità sono una conseguenza di vari elementi: tipo di tecnologia impiegata da tali prodotti software, tipo di utente, tipo di applicazione e tipo di mercato.
Le caratteristiche di mercato di tali prodotti, richiamate brevemente nel primo paragrafo, suggeriscono che tutti questi attributi siano sottoposti ad accurata valutazione, prima del rilascio finale, da parte dei gruppi di controllo qualità che vedono in tal modo alquanto ampliato il "taglio" del loro intervento.
La valutazione di tali attributi, infatti, coinvolge anche la impostazione e la "misura" di attività svolte da campioni di utenti, la analisi di dati di mercato, la sperimentazione di prodotti analoghi e/o concorrenti.
Tale valutazione può avvalersi di semplici "metriche", di cui si accenna nel quarto paragrafo.

ABSTRACT

The present paper discusses some of the most important "quality attributes" of software "office productivity tools" for personal computers (word processing and spreadsheet packages, data base and file management systems, and so on).
These quality attributes are the consequence of a number of elements, such as: type of technology of "business" personal computers, type of user, type of application and type of market. For all the discussed quality attributes, a simple "metrics" can be defined.

(°) Questo lavoro è stato presentato al convegno sul tema "La certificazione del software: principi, organizzazione, metodi, strumenti ed esperienze", organizzato nell'ambito del Progetto Finalizzato per l'Informatica del CNR, Sottoprogetto P1, Obiettivo METOD, Milano, 4-6 aprile 1984, FAST.

1. SOFTWARE PER PERSONAL COMPUTER: UN MERCATO NUOVO, UN PRODOTTO NUOVO

La diffusione dei personal computer ha creato (e sta creando) un profondo rinnovamento nel mercato del software. Da prodotto ad alto prezzo e, spesso, a bassa "tiratura", quale era ed è tuttora per i minielaboratori e per i mainframe, il software applicativo per gli elaboratori personali sta oggi assumendo le caratteristiche del prodotto di largo consumo.

I prodotti software per personal computer hanno, infatti, un prezzo finale molto contenuto (tipicamente, poche centinaia di dollari per i prodotti a più ampio campo di applicazione) e tirature che, nei prodotti più noti, arrivano a diverse decine di migliaia di copie e, nei "best sellers", anche a parecchie centinaia di migliaia di esemplari (fig.1).

Anche nella confezione il prodotto software si trasforma, assumendo via via una veste che richiama quella del libro: l'utente acquisterà oggi, nel computer shop o perfino nel grande magazzino (almeno nei paesi a mercato consolidato, come gli Stati Uniti), un kit completo, comprendente uno o più dischetti sui quali è registrato il programma (oltre che, molto spesso, un "tutorial" interattivo), e il manuale d'uso.

Quest'ultimo, spesso riccamente illustrato, è orientato sempre più all'utente non specialista, cioè a chi desideri utilizzare il personal computer come "strumento", senza conoscere le caratteristiche tecniche e le modalità di funzionamento interno e di programmazione (così come, ad esempio, l'acquirente di un televisore, di un videoregistratore o di un giradischi non è in generale interessato a conoscerne gli schemi elettrici).

In molti casi, il prodotto software è messo sul merca-

"BEST SELLER" SOFTWARE (Applicazioni Professionali)

Nome del prodotto	Editore	Copie vendute	Data di immissione sul mercato
WORDSTAR	Micropro	1.000.000	6/79
VISICALC	Visicorp	700.000	10/79
SUPERCALC	Sorcim	300.000	8/81
PFS: FILE	Software Publishing Co.	300.000	9/80
SCRIPSIT	Tandy	300.000	estate 79
1-2-3	Lotus	170.000	1/83
dBASE II	Ashton-Tate	170.000	1/81
PFS: REPORT	Software Publishing Co.	100.000	5/81
PERFECT WRITER	Perfect Software	100.000	primavera 82
MULTIPLAN	Microsoft	100.000	8/82
PERFECTCALC	Perfect	80.000	autunno 82
MULTIMATE	Software Systems	72.000	3/83
SPELLBINDER	Lexisoft	62.000	estate 78
EASYWRITER	IUS	50.000	5/82
PFS: GRAPH	Software Publishing Co.	30.000	5/82
PFS: WRITE	Software Publishing Co.	25.000	6/83

Fig. 1 - Un elenco dei principali "best seller" software per personal computer, con una stima del numero di copie vendute. (Fonte: International Data Corporation, cfr. LIST, feb. 1984)

to non da chi è stato sviluppato, ma da "software publisher", che lo acquisiscono da autori esterni, e provvedono a tutti gli aspetti di carattere "editoriale" (dalla cura della documentazione alla confezione del package, fino al controllo di qualità complessivo), al lancio e alla distribuzione del prodotto.

In effetti, via via che il prodotto software consolida le proprie caratteristiche di prodotto di largo consumo, questi aspetti vengono ad incidere sempre di più sui costi complessivi del prodotto, fino a superare abbondantemente i costi del puro sviluppo software (si pensi, ad esempio, agli ingenti investimenti in pubblicità effettuati dalla Lotus per il lancio del package 1-2-3, valutabili, a detta di molti osservatori, in 3 milioni di dollari per il solo primo anno di vita per prodotto).

Val la pena ricordare, tra l'altro, che in tempi recenti tale tipo di prodotto ha interessato gli editori di libri, ed oggi molti editori offrono, oltre ai propri prodotti tradizionali, prodotti software. Tale fenomeno, sensibile negli Stati Uniti (Prentice -Hall, McGraw Hill, Addison Wesley, e molti altri) è presente anche in Italia (Mondadori, Jackson,...).

Questo processo di trasformazione delle caratteristiche di mercato del prodotto software, assai recente e tuttora in atto, ha introdotto, e più ancora introdurrà nel prossimo futuro, profonde innovazioni nel modo di produrre software.

L'alto numero delle copie vendute di ogni singolo prodotto (di successo) e il bassissimo prezzo di vendita hanno infatti reciso, di necessità, il tradizionale "cordone ombelicale" che legava, nel mondo dei minielaboratori e dei mainframe, l'utente al produttore dei programmi. I margini economici per interventi di installazione del prodotto software su una specifica macchina, e per la sua manutenzione, non esistono in prodotti che molto spesso hanno un prezzo di vendita, per l'utente finale, al di sotto dei cinquecento dollari. Inoltre, la dimensione del parco utenti di ogni singolo prodotto di successo (come si è detto, anche centinaia di migliaia di utenti) e la sua distribuzione geografica impedirebbe comunque, anche alla organizzazione più ramificata, un'attività di assistenza diretta.

D'altro canto, se il numero di utenti del personal computer aumenta, la loro "competenza informatica" media diminuisce. I prodotti software del personal computer con il tasso di crescita di mercato più elevato sono i cosiddetti "office productivity tool": quei prodotti, cioè, destinati ad aumentare la produttività individuale di un largo numero di utenti non specialisti di informatica: professionisti, dirigenti, personale tecnico. Si parla, qui, del software per la gestione di fogli elettronici, per l'achiviazione e il reperimento di informazioni, per word-processing, per la creazione di grafici di tipo economico, e così via.

Le caratteristiche principali che si richiedono a tali

prodotti software sono sostanzialmente due: la flessibilità applicativa (si tratta, infatti, di prodotti di uso generale, che devono adattarsi facilmente a una larga gamma di applicazioni ed esigenze particolari) e la facilità d'uso.

Quest'ultimo termine, di cui si è spesso abusato, va qui inteso nel senso più ampio, tenendo soprattutto conto della impossibilità di una assistenza diretta all'utente da parte del produttore. "Facilità d'uso" sarà qui inteso, quindi, come quel complesso di caratteristiche che permettono all'utente non specialista di lavorare bene e senza problemi con il prodotto, senza alcuna assistenza specialistica.

Tutto questo impone, per questi prodotti software, un ripensamento del concetto di "qualità", e un ampliamento degli obiettivi del "controllo qualità", che non può limitarsi a verificare la coerenza del prodotto rispetto alle sue specifiche funzionali e di prestazioni, ma deve sottoporre il prodotto a una serie alquanto più ampia di verifiche e controlli, che tendano a una valutazione più complessiva del prodotto rispetto alle esigenze della fascia di mercato a cui questo è destinato.

Questo ampliamento degli obiettivi del controllo di qualità è di fondamentale importanza, tenuto anche conto dei seguenti elementi:

- la considerevole entità degli investimenti necessari per portare un prodotto sul mercato, una volta che il software sia stato realizzato (packaging, promozione, distribuzione, supporto);

- l'alto livello di concorrenza nell'offerta di software per personal computer esistente sui mercati consolidati (tipicamente: USA), che tende a determinare un sostanziale allineamento funzionale di molti prodotti software che si rivolgono alla medesima fascia di applicazioni/mercato (per cui il successo o il fallimento di un prodotto può dipendere dalla accurata individuazione e verifica di alcuni "plus" rispetto alla concorrenza), e una rapida obsolescenza dei prodotti stessi;

- la esistenza di una "critica" dei prodotti software, che orienta il consumatore, nelle sue scelte, in modo massiccio, attraverso riviste e pubblicazioni specializzate. Ci si riferisce qui, ancora, soprattutto alla situazione americana (i periodici che, negli USA, si indirizzano al mercato dei personal computer - utenti, produttori, distributori, publisher, home, game e business - sono ormai alcune centinaia. Tale stampa specializzata pubblica ogni mese una impressionante quantità di valutazioni, confronti, tabelle, applicazioni, classifiche dei prodotti software disponibili, che orientano in modo decisivo il mercato).

Si comprende, per tutto questo, la importanza di un "rapporto finale di qualità", che contenga una valutazione "complessiva" sul mercato (e su un mercato così complesso) prima che il prodotto software, sfugga, per così dire, di mano al suo produttore.

Obiettivo di questa nota è quello di stimolare la discussione sugli obiettivi del controllo di qualità del software per personal computer, individuando alcuni degli attributi di qualità che, secondo la esperienza di chi scrive, assumono una importanza fondamentale per il tipo di software in esame (strumenti di produttività individuale per l'ufficio).

2. SOFTWARE PER PERSONAL COMPUTER: LA QUALITÀ

Esaminiamo brevemente, in questo paragrafo, i principali "attributi di qualità" dei prodotti software in esame.

Possiamo innanzitutto caratterizzare meglio il tipo di software al quale rivolgiamo la nostra attenzione come segue:

a) *Tecnologia*
Si tratta di prodotti software operanti su microelaboratori a 8 o più spesso a 16 bits, con memoria centrale espandibile a partire da 64 Kbytes, o da 128 Kbytes e oltre, video standard a 80 colonne, spesso grafico (monocromatico o a colori), una o (più spesso) due unità a dischetti, possibilità di collegamento di un disco rigido.
Periferiche stampanti molto varie (stampanti a matrice di punti, a margherita, plotter...).

b) *Utente*
Utente non programmatore, dal quale non si assume alcuna esperienza di informatica.
Utilizza il personal computer soltanto occasionalmente, oppure spesso e sistematicamente, ma sempre come strumento di lavoro per ottenere risultati nella specifica area di attività, e non come oggetto primario di interesse.

c) *Applicazioni*
Applicazioni fortemente interattive orientate al calcolo o all'archiviazione personale, alla creazione ed elaborazione di testi e grafici, alla gestione di rapporti, di modelli economici, simulazioni di tipo "what if...", eccetera. Testo e grafica di alta risoluzione spesso integrati nella stessa applicazione; modalità di utilizzo al di fuori di procedure rigide precostituite; uso non strutturato e adattato di volta in volta alle diverse situazioni.

d) *Mercato*
Mercato fortemente competitivo, ricco di prodotti software concorrenti, funzionalmente analoghi. Necessari larghi investimenti di marketing;

reti di distribuzione vaste e capillari; distribuzioni su molti paesi (lingue diverse).

Gli attributi di qualità da considerare saranno allora conseguenza delle caratteristiche elencate (tecnologia, utente, applicazione, mercato). Esaminiamo, qui di seguito, solo gli attributi di qualità più importanti.

2.1. Attributi di qualità che sono conseguenza del tipo di utente

Si tratta, qui, del concetto intuitivo di "facilità d'uso", che può essere meglio precisato identificando i seguenti attributi:

. *Facilità di installazione*

Prima di poter utilizzare un prodotto software, l'utente deve, molto spesso, eseguire alcune operazioni di "installazione" che hanno in genere i seguenti scopi:

- caricare sul dischetto sul quale risiede il programma i componenti di sistema operativo utilizzati dal programma stesso, in modo da permetterne il funzionamento;
- fornire al programma i parametri che definiscono la particolare configurazione hardware utilizzata;
- in alcuni casi, effettuare le copie di back-up dei dischetti contenuti nella confezione del prodotto.

Ciò comporta in genere che l'utente esegua una procedura opportunamente predisposta, seguendo passo passo le istruzioni fornite e rispondendo alle domande che vengono poste sul video.

La facilità di installazione è un attributo di qualità molto importante: l'utente deve essere in grado di installare senza difficoltà il prodotto sul proprio sistema da solo, "a casa propria", senza assistenza da parte del rivenditore o di personale specializzato, e senza conoscere le modalità d'uso del prodotto stesso (tipicamente, l'utente imparerà ad usare il prodotto eseguendo al calcolatore gli esercizi suggeriti dal manuale d'uso; per far questo, dovrà prima completare le operazioni di installazione).

Tipicamente, i prodotti meglio concepiti forniscono le istruzioni di installazione in un fascicoletto separato dal manuale ("READ THIS FIRST"), in modo che l'utente possa effettuare le operazioni richieste senza nemmeno aprire il manuale d'uso.

Proprio perchè effettuato da un utente ancora insicuro sul prodotto (e, a volte, prima di aver effettuato le copie di back-up), questo dovrà essere particolarmente protetto da eventuali errori operativi dell'utente (robustezza).

L'utente dovrà, in ogni caso, poter interrompere senza danni le operazioni di installazione in qualsiasi istante (ad esempio per documentarsi meglio, o per richiedere consiglio a personale specializzato).

Possiamo considerare le operazioni volte ad ottenere una copia di riserva del prodotto software (sia che si tratti di spedire al produttore una cartolina di richiesta, sia che si tratti di eseguire una opportuna procedura di duplicazione, o altro), come parte delle operazioni di installazione, e con gli stessi requisiti di semplicità operativa, e protezione dagli errori ("facilità di back-up").

. *Facilità di apprendimento*

Si tratta, qui, del grado di facilità con cui l'utente "medio" (quello a cui il prodotto è destinato) è in grado di ottenere i primi risultati "reali" dall'uso del prodotto.

Molti elementi contribuiscono a determinare la facilità di apprendimento di un prodotto; tra gli aspetti più rilevanti ricordiamo:

- la disponibilità di "tutorial" interattivi
- la comprensibilità del manuale d'uso
- la coerenza fra prodotto e documentazione
- la disponibilità di esempi preconfezionati su dischetto
- la esistenza di una "trouble-shooting guide"
- la esistenza di quadri sinottici, pocket guide o reference card
- la esistenza di indici analitici per un facile accesso all'informazione
- il linguaggio utilizzato nel prodotto e nella sua documentazione
- la "naturalezza" dell'interfaccia utente e la sua consistenza
- la disponibilità di messaggi di "help" contestuale
- la chiarezza dei messaggi diagnostici
- la disponibilità di un servizio di assistenza telefonica.

. *Facilità d'uso*

Questo attributo viene spesso confuso con il precedente (facilità di apprendimento). Con facilità d'uso intendiamo qui, invece, la "naturalezza" con la quale un utente già mediamente esperto nell'uso del prodotto è in grado di utilizzarlo per effettuare compiti di "normale" difficoltà.

Facilità d'uso e facilità di apprendimento possono pertanto, in molti casi, risultare in conflitto.

Si pensi, come esempio tipico, ai sistemi basati su menù: il menù può risultare molto comodo in fase di apprendimento, quando è necessario proporre

all'utente, in chiaro e in modo strutturato gerarchicamente, tutte le alternative che il sistema è in grado di offrirgli. I menù possono, però, risultare di intralcio quando l'utente, ormai esperto nell'uso del sistema, troverebbe più comodo poter indicare direttamente la operazione desiderata, senza doverla "raggiungere" operando selezioni successive in una gerarchia (anche profonda) di menù.

Anche qui, molti elementi contribuiscono a determinare la facilità d'uso di un prodotto, ed è difficile indicare criteri generali che prescindano dalla tipologia del prodotto stesso.

Tra gli elementi più tipici, potremo tuttavia ricordare i seguenti (già visti a proposito della facilità di apprendimento):

- la disponibilità di messaggi di "help" contestuale
- la chiarezza dei messaggi diagnostici
- la esistenza di una "trouble-shooting guide"
- la esistenza di quadri sinottici, pocket guide o reference card
- la esistenza di indici analitici della documentazione, per un facile accesso all'informazione
- la disponibilità di un servizio di assistenza telefonica;

inoltre:

- la facilità con cui l'utente può passare da un contesto a un altro contesto durante l'uso del prodotto
- la libertà che l'utente ha di "cambiare idea" durante l'uso del prodotto
- la essenzialità nella comunicazione utente-sistema (la ridondanza della comunicazione, spesso utile in fase di apprendimento, può risultare d'impaccio "a regime")

e così via.

2.2 Attributi di qualità che sono conseguenza del tipo di tecnologia

. *Prestazioni*

Si tratta, in sostanza, dei tempi di risposta del prodotto.

Tale attributo di qualità deve essere sottoposto a valutazioni particolarmente accurate nel tipo di software in esame. Infatti, la potenza relativamente limitata dei microprocessori oggi più diffusi e l'uso prevalente, in tali prodotti software, di supporti lenti come memorie di massa (floppy-disc), possono introdurre appesantimenti notevoli nelle prestazioni del software (al quale, d'altra parte, si richiede di svolgere funzioni sempre più complesse e sofisticate).

La scelta di un particolare algoritmo o di un particolare linguaggio può avere conseguenze drastiche sulle prestazioni di un prodotto software, che andranno pertanto monitorate fin dalle prime fasi di sviluppo, quando modifiche di architettura (o di linguaggio) risultano ancora possibili.

Si osservi inoltre che, in applicazioni fortemente interattive come quelle in esame, la risposta del sistema ad alcuni tipi di sollecitazione dell'utente dovrà essere "istantanea". Alcuni esempi tipici (si pensi ad applicazione di word processing o di spreadsheet) sono i comandi di movimento cursore, e i comandi di scroll: una risposta non istantanea a tali comandi riduce drasticamente la maneggevolezza del testo o della tabella su cui si opera, e quindi la "facilità d'uso" complessiva.

. *Compatibilità (sistemistica)*

Si intendono, qui, due diversi tipi di compatibilità:

- compatibilità del prodotto software con le diverse configurazioni del sistema hardware e della sua periferia (memoria centrale, unità a dischetti, disco rigido, tipo di monitor, tipo di stampante/plotter, tipo di keyboard);
- compatibilità del prodotto software con diversi sistemi hardware appartenenti a una stessa famiglia tecnologica, che siano forniti o meno dalla stessa casa costruttrice (ad esempio, "sistemi CP/M", "sistemi MS-DOS", "sistemi IBM-compatibili", e così via).

Entrambi i tipi di compatibilità sono molto spesso ottenuti mediante opportune procedure di configurazione del software (vedi "facilità di installazione").

2.3 Attributi di qualità che sono conseguenza del tipo di applicazione

. *Flessibilità*

I prodotti software in esame, destinati a soddisfare le esigenze di informatica individuale di una vasta tipologia di utenti, devono essere utilizzabili in una larga classe di applicazioni, e con modalità operative flessibili e a priori non predeterminate.

Si tratta, questo, di un attributo assai difficilmente definibile in generale, senza far riferimento a una specifica classe di prodotti software.

Si può comunque osservare che un elemento fondamentale - e molto importante - da considerare nella valutazione di questo attributo è l'insieme dei parametri dimensionali minimi e massimi del prodotto software.

Tipici esempi sono i seguenti:

- in un programma per la gestione di spreadsheet: dimensioni massime della tabella, lunghezza massima di ogni cella, numero di cifre significative in un valore numerico, complessità di una formula, ecc.

- in un data-base o file-management system: numero massimo di record per file; numero massimo di campi per record; lunghezza massima di files aperti contemporaneamente, ecc.

- in un package di business graphics: numero massimo di diagrammi sovrapponibili; numero massimo di punti individualmente leggibili in un diagramma; ecc.

. *Compatibilità (applicativa)*

Si intende, qui, la capacità posseduta da un prodotto software di scambiare dati con altri prodotti software a larga diffusione (ad esempio, con i "best-seller" indicati in fig. 1).

Oggi molti prodotti software a larga diffusione, anche di produttori concorrenti, sono fra loro compatibili in tal senso: è ad esempio molto diffusa la capacità di leggere - o generare - file del formato richiesto da VISICALC, o da dBASE II, o da LOTUS 1-2-3, e così via.

Ciò permette, evidentemente, all'utente di scegliere i prodotti che più si adattano alle proprie esigenze, anche da più fornitori diversi, e di essere tuttavia in grado di "far comunicare" tali prodotti (ad esempio, per inserire in un documento elaborato con un package di word processing una lista creata con un sistema data-base, e così via).

2.4 Attributi di qualità che sono conseguenza del tipo di mercato

Si è già detto della particolare competitività del mercato dei prodotti software in esame. Il successo o meno di un prodotto, in tale situazione, è legato a una quantità di elementi che non hanno attinenza diretta con la qualità del prodotto: investimenti in pubblicità, rete di distribuzione, e così via.

Certamente, però, gli attributi di qualità giocano un ruolo importante nel determinare la collocazione del prodotto sul mercato. Oltre agli attributi già esaminati, vogliamo qui aggiungere i seguenti:

. *Sex-appeal*

Si vuole intendere, qui, in assenza di termini migliori, la gradevolezza complessiva del prodotto (non soltanto la sua confezione esteriore, ma anche, e soprattutto, come il prodotto si manifesta durante l'uso), o la esistenza di particolari elementi di "fascino" o di "divertimento" nel suo uso.

In effetti, una componente di "gioco" è spesso presente nel rapporto fra utente e personal computer, e ne costituisce un elemento di stimolo all'apprendimento e alla creatività.

Elementi che rafforzano questa componente ludica possono avere una notevole influenza nel creare una immagine favorevole al prodotto. Alcuni elementi che possono contribuire a determinare il "sex-appeal" complessivo sono i seguenti:

- colore

Oggi il colore è normalmente supportato dai prodotti software più diffusi che richiedono un video con capacità grafiche.

Le potenzialità spettacolari di un monitor RGB sono elevate; il loro sfruttamento da parte dei package attuali ancora molto modestro.

- grafica

(la grafica ad alta risoluzione viene utilizzata sempre più di frequente, anche in applicazioni che di per sè non la richiederebbero necessariamente, ad esempio per sistemi di "window").

- "design" complessivo delle modalità di interazione uomo-macchina

(es.: mouse, touch-screen, window, ecc.)

- "gadget"

(si pensi ai molti gadget presenti in sistemi come Lisa, o Macintosh).

- tempi di risposta

(si pensi ai molti gadget presenti in sistemi come Lisa, o Macintosh).

. *Localizzazione*

Si intende, qui, la compatibilità del prodotto con le esigenze delle diverse lingue nazionali nei mercati a cui si rivolge (tastiere nazionali, set grafici, testi di help, menù e prompt, comandi).

3. SOFTWARE PER PERSONAL COMPUTER: LE MISURE DI QUALITÀ

L'attività di controllo qualità ha lo scopo di produrre delle *misure* degli attributi di qualità prescelti, misure che possano essere valutate per la decisione finale sulla rilasciabilità del prodotto.

Per ognuno degli attributi di qualità che si ritengono rilevanti per tale decisione, occorrerà allora definire una opportuna metrica.

Qui di seguito si forniscono alcune indicazioni di massima sulla metrica utilizzabile per ciascuno degli attributi individuati nel paragrafo precedente.

. *Facilità di installazione*

Metrica : Tempo medio di installazione, da parte di utenti "tipici", inesperti sull'uso del prodotto e senza assistenza (raffrontato col tempo medio di prodotti analoghi).

Osservazioni: Inoltre, occorre verificare la robustezza del sistema a fronte di errori operativi (checklist di "azioni distruttive").

. *Facilità di apprendimento*

Metrica : Tempo medio occorrente a utenti "tipici", inesperti nell'uso del prodotto e senza assistenza, per realizzare applicazioni "tipiche" predefinite.
Ad esempio: battitura e correzione di una lettera campione (per prodotti di word-processing); creazione di un semplice modello economico già realizzato sulla carta (per prodotti spreadsheet); ecc.
Eventuale raffronto con identiche applicazioni realizzate con prodotti analoghi.

. *Facilità d'uso*

Metrica : Tempo medio occorrente, a utenti "tipici" già completamente padroni nell'uso del prodotto, per realizzare applicazioni "tipiche" predefinite.

Osservazioni: In un certo senso, misurare la facilità di uso di un prodotto equivale, secondo le definizioni poste, a misurare le "prestazioni" complessive del sistema "prodotto + utente" in condizioni e applicazioni tipiche.

. *Prestazioni*

Metrica : Tempi di risposta a "sollecitazioni tipiche" da parte dell'utente.

Osservazioni: Importante una valutazione anche dei tempi di risposta *soggettivi* (percepiti dall'utente), che possono essere notevolmente diversi dai tempi di risposta reali.

. *Compatibilità (sistemistica)*

Metrica :

$$\frac{\sum_{i=1}^{m} \delta_i n_i}{\sum_{i=1}^{m} n_i}$$

Tale formula misura il grado di compatibilità con le diverse configurazioni di un sistema hardware. In essa, n_i è il numero dei sistemi (su un certo mercato) di configurazione i, e, $\delta_i = 1$ se la configurazione i è supportata, Ø altrimenti.

Analogo discorso per la compatibilità con sistemi diversi.

. *Flessibilità*

Metrica : Valutazione soggettiva da parte di un gruppo di utenti esperti, espressa con un *voto* globale, risultante da voti assegnati a specifici attributi, sulla base di una check-list preventivamente predisposta sulla base del tipo di prodotto. Il voto va tarato mediante valutazione di prodotti analoghi.

. *Compatibilità (applicativa)*

Metrica : Copertura di una check-list opportuna

. *Sex appeal*

Metrica : Valutazione soggettiva da parte di un gruppo di utenti (tra cui, possibilmente, anche esperti di comunicazione), espressa con un voto (sulla base di una check-list preventivamente predisposta: colore, grafica, ecc.).

. *Localizzazione*

Metrica : copertura di una check-list opportuna.

4. CONCLUSIONI

La presente nota ha descritto alcuni attributi di qualità che, sulla base della esperienza di chi scrive, sono di importanza fondamentale per una larga classe di prodotti software per personal computer ("productivity tools").

Tali attributi di qualità sono una conseguenza del tipo di tecnologia impiegata da tali prodotti software, del tipo di utente, di applicazione e di mercato.

Le caratteristiche di mercato di tali prodotti, richiamate brevemente nel primo paragrafo, suggeriscono anche che tali attributi siano sottoposti ad accurata valutazione, prima del rilascio finale, da parte dei gruppi di controllo di qualità, che vedono in tal modo alquanto ampliato il taglio del loro intervento.

La valutazione di tali attributi, infatti, coinvolge la impostazione e la "misura" di attività svolte da campioni di utenti, la analisi di dati di mercato, la sperimentazione di prodotti analoghi e/o concorrenti.

Tale valutazione può avvalersi di semplici metriche, di cui si è accennato nel quarto paragrafo.

5. RINGRAZIAMENTI

A quanto espresso nella presente nota hanno contribuito in modo determinante le esperienze effettuate, in ambito Etnoteam, nello sviluppo e nel controllo di qualità delle varie release dei prodotti DOSSIER e DOSSIER 128, per personal computer Apple II e IBM PC/XT rispettivamente.
Tali progetti sono stati realizzati da un gruppo di persone, e in particolare: Riccardo Paulin, Piero Schiavo Campo, Luisa Reina, Alessandro Mazzetti, Paolo Ferrara, Dario Brocca.

IL DOSSIER ELETTRONICO: UNO STRUMENTO DI INFORMATICA INDIVIDUALE PER L'UFFICIO

R. Paulin (+) - R. Polillo (°,+) - P. Schiavo Campo (+)

RIASSUNTO

Il presente articolo descrive DOSSIER, uno strumento software per l'ufficio basato sul concetto di "spread-sheet", orientato all'uso da parte di professionisti e dirigenti sulla fascia dei "personal computer".
Vengono illustrate le funzionalità dello strumento e una schematica descrizione della sua architettura software.

ABSTRACT

The present paper describes DOSSIER, a software office tool based on the "spread-sheet" concept directed to professional and manager users on personal computers.
The functionalites of this tool are described, together with a sketchy presentation of its software architecture.

1. INTRODUZIONE

Nel corso degli ultimi due-tre anni, è stato immesso sul mercato un numero via via crescente di elaboratori a basso costo, ma di tipo "professionale", caratterizzati spesso da processor a 16 bit, memoria centrale espandibile a partire da 128 K bytes, video di dimensioni standard (tipicamente, 24 x 80 caratteri) dotato di capacità grafiche a risoluzione medio-alta, in bianco e nero o a colori, tastiera professionale corredata di numerosi tasti funzione, floppy-disc di discreta capacità, possibilità di interfacciamento a dischi rigidi.

Tali sistemi si collocano, principalmente, nell'area dell'informatica individuale per l'ufficio.

(°) Istituto di Cibernetica dell'Università di Milano
(+) Etnoteam SpA, Milano

Il presente lavoro descrive DOSSIER, un prodotto per l'informatica individuale nell'ufficio, progettato con l'obiettivo di fornire, all'utente inesperto di elaborazione dati, uno strumento innovativo, di utilizzo semplice e naturale, che sfrutti appieno le potenzialità degli elaboratori personali professionali a basso costo della corrente generazione.

DOSSIER, prendendo le mosse dal concetto di foglio elettronico (1, e i molti derivati), dalla metodologia di documentazione PSPN (4), e dalle ricerche sulle modalità di colloquio uomo-macchina sviluppate da più parti nel corso degli anni settanta (ad es., (2)), realizza il concetto di "dossier elettronico". Questo può essere immaginato come un usuale raccoglitore a fogli mobili, contenente modulistica d'ufficio, pagine di testo, grafici, mediante il quale l'utente può organizzare dei "rapporti" fortemente strutturati, "vivi" nel senso che un aggiornamento di un dato presente in una pagina del dossier produce, automaticamente, l'aggiornamento di tutti i dati ad esso relazionati, nell'ambito dell'intero dossier.

Il dossier elettronico è uno strumento di tipo generale, che può essere utilizzato in una varietà di applicazioni. Il principale obiettivo del progetto, tuttavia, è stato quello di definire uno strumento orientato innanzitutto a professionisti, imprenditori e dirigenti, che permettesse di tenere sotto controllo, in modo semplice e ordinato, insiemi di dati *di sintesi* relativi all'andamento aziendale (piani di fatturato, finanza, gestione del personale, ecc).

Nel seguito, dopo una introduzione al concetto di dossier elettronico (paragrafo 2), viene fornito un breve inquadramento delle funzioni principali fornite dal sistema, e dell'approccio seguito per la progettazione dell'interfaccia utente (paragrafo 3). Il paragrafo 4 descrive quindi, più in dettaglio, la struttura dei "moduli" costituenti un dossier elettronico, e del lin-

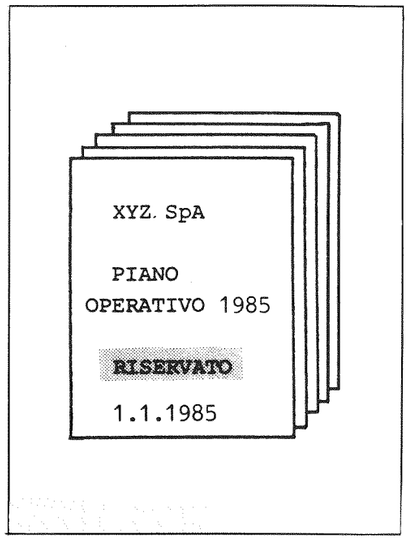


Fig. 1.a

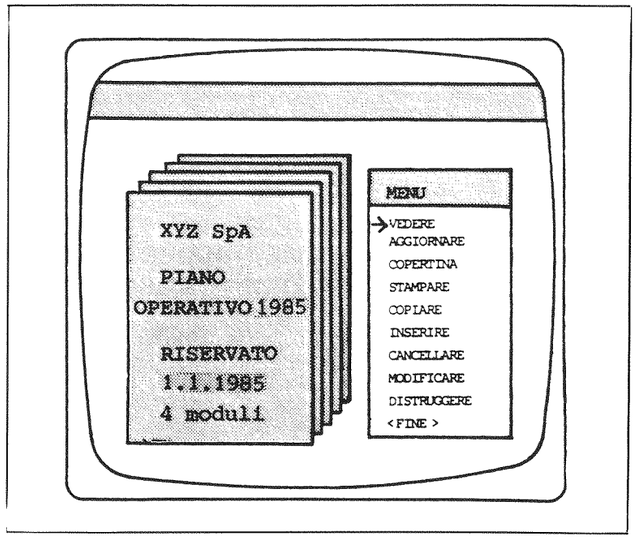


Fig. 1.b

guaggio con cui l'utente può definire le computazioni che dovranno essere effettuate automaticamente su tali moduli.
Il paragrafo 5 contiene alcuni cenni sull'architettura software e, infine, il paragrafo 6 alcune conclusioni.

2. IL DOSSIER ELETTRONICO

L'utente di DOSSIER (°) fa riferimento, nell'uso del software, a un modello concreto e tangibile dell'oggetto "dossier" (un raccoglitore a fogli mobili): il software provvede, utilizzando quando necessario le capacità grafiche della macchina, a presentare sul video, in ogni contesto, una immagine concreta, coerente con tale modello.

Pertanto, nel seguito verrà presentato, di pari passo, il modello concettuale di un "dossier" e la sua visualizzazione sulla macchina.

Un dossier elettronico è un raccoglitore dotato di una *copertina,* sulla quale compaiono alcuni dati identificativi di base: un titolo, il nome dell'autore del dossier, una data. Se il dossier è riservato, ciò è evidenziato sulla copertina con opportuna dicitura. (fig. 1.a).

(°) Indicheremo, per evitare ambiguità, con "DOSSIER" (maiuscolo) il nome del sistema software, e con "dossier" (minuscolo) l'oggetto logico (il dossier elettronico) trattato da DOSSIER.

Ogni dossier è registrato su un singolo floppy-disc (e ogni floppy-disc può contenere un singolo dossier). Quando l'utente inserisce nel drive un floppy/dossier, questo viene immediatamente visualizzato sul video in grafica a media risoluzione, accanto a un opportuno menù di operazioni base (fig. 1.b).

Se il dossier è riservato, il menù compare soltanto dopo che l'utente ha fornito la relativa password.

Subito dopo la copertina, un dossier contiene un *indice* (fig. 2.a). Pertanto l'utente del programma, con opportuna selezione del menù principale (vedi più oltre), può "aprire" il dossier. Compare così sul video la pagina "indice" del dossier stesso, che contiene oltre al titolo del dossier (ripreso dalla copertina), l'elenco (costruito e aggiornato automaticamente) di tutti i "moduli" che costituiscono il dossier stesso (fig. 2.b).

Operando su tali titoli, con semplici movimenti del cursore, l'utente può allora selezionare il "modulo" desiderato.

Ogni *modulo* è costruito da una terna di pagine affiancate: la pagina centrale contiene un "quadro"; la pagina destra contiene un testo libero, redatto dall'utente a commento del quadro (*pagina di commento*); la pagina sinistra è detta *pagina delle istruzioni,* e verrà descritta più oltre.
Le tre pagine di un modulo possiedono un titolo identico: il *titolo del modulo.*

Un *quadro* è, sostanzialmente, una tabella bidimensionale (fig. 3.a), non molto dissimile da un normale "foglio elettronico".

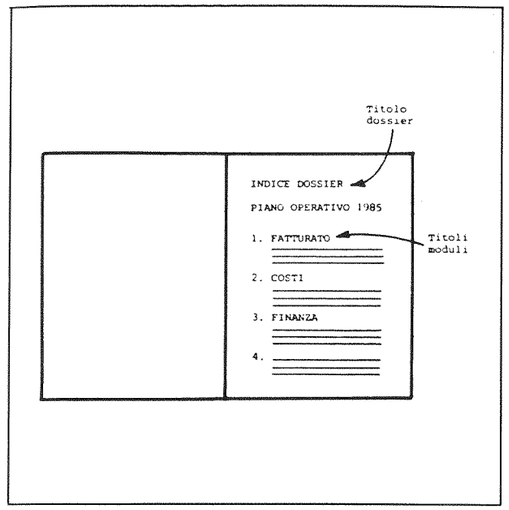


Fig. 2.a

Fig. 2.b

Il formato di un quadro è definito dall'utente stesso, in una fase di definizione della struttura di un dossier (vedi più oltre), in termini di numero di righe e di colonne, titoli di righe e di colonne, larghezza di ogni colonna.

I campi della tabella possono contenere dati qualsiasi (numerici e alfanumerici).

La pagina sinistra e la pagina destra di un modulo compaiono su due schermate differenti; l'utente passerà dall'una all'altra battendo un semplice tasto funzionale, che produrrà il cambio della immagine in modo istantaneo. La struttura delle due pagine è

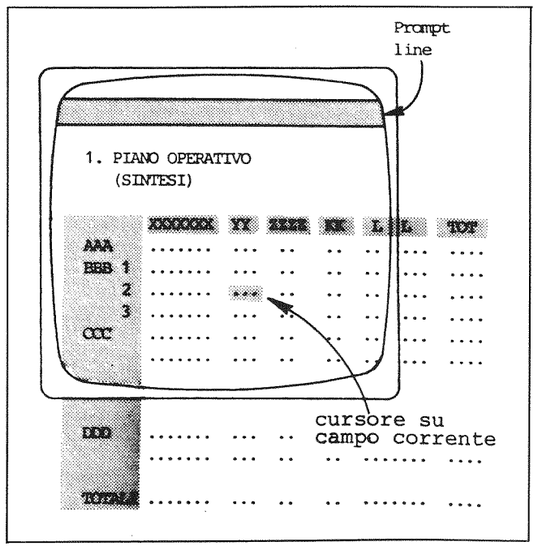


Fig. 3.a

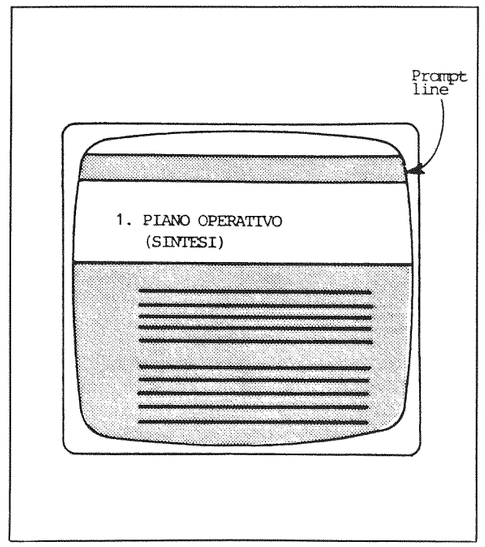


Fig. 3.b

identica (fig. 3.b).

Un dossier elettronico può, inoltre, contenere *pagine grafiche.* Queste permettono la visualizzazione in forma grafica di vario tipo (diagrammi a barre semplici e multiple, diagrammi a linee spezzate) di dati contenuti nel quadro.

3. LE FUNZIONI E L'INTERFACCIA UTENTE

Come si è accennato nei paragrafi precedenti, l'obiettivo principale del progetto è stato quello di fornire all'utente una interfaccia semplice e naturale, che gli permettesse di operare su un dossier elettronico *non* attraverso "comandi" di tipo verbale, riferiti a una struttura di dati "dentro la macchina", ma attraverso "azioni" semplici, effettuate su un oggetto concreto, visibile ad ogni istante sul video, ed "esplorabile" con il cursore (cfr. 2).

Data la molteplicità delle operazioni che devono essere possibili su un dossier elettronico, questa filosofia, se portata alle estreme conseguenze, comporterebbe l'introduzione di numerosi tasti funzione di significato predefinito (si pensi alle molte decine di tasti funzionali spesso gestiti da uno screen editor sofisticato, e alla conseguente difficoltà d'uso).

Pertanto, per ridurre a poche unità il numero di tasti funzione, il sistema fa uso di alcuni menù, visualizzati in finestre sovrapposte all'immagine della porzione di dossier (copertina, moduli, ecc.) che costituisce il contesto corrente.

Le voci di ogni menù sono selezionabili muovendo un secondo tipo di cursore a forma di freccia. La voce correntemente indicata sul menù è sempre, per maggiore chiarezza, spiegata per esteso in una "prompt line" alla sommità dello schermo.

Così, il menù principale compare, inserendo un floppy/dossier nel drive, accanto alla immagine del dossier, come mostrato in fig. 1.b. Si noti che, per dare maggiore concretezza alla immagine del dossier, questo viene visualizzato mostrando, in prospettiva, un numero di pagine uguale al numero di moduli che lo compongono.

Tralasciando, in questa sede, la descrizione dei menù di secondo livello, descriviamo brevemente le singole voci del menù principale (fig. 1.b), la cui composizione riflette alcune scelte di base nella progettazione del sistema:

a) **Vedere**

Permette di "sfogliare" il dossier in sola lettura (i dati contenuti nel dossier sono pertanto protetti, e non possono essere modificati).

Costituisce l'operazione più semplice, e richiede all'utente, in sostanza, la sola conoscenza dei tasti di movimento del cursore (nei vari contesti: per selezionare il modulo desiderato sulla pagina indice e per effettuare gli scroll dei vari quadri) e dei tasti funzionali per passare alle varie pagine (sinistra, centrale, destra) di uno stesso modulo.

b) **Aggiornare**

Permette, oltre che di "sfogliare" il dossier, di aggiornare i valori dei campi dei vari quadri, e di aggiornare il testo delle pagine di commento.

L'utente, per utilizzare questo gruppo di funzioni, dovrà apprendere (oltre a quanto indicato al punto a), soltanto come aggiornare i vari campi di un quadro e come inserire/aggiornare un testo.

Per semplificare ulteriormente l'uso del sistema, l'interfaccia utente per l'aggiornamento dei vari campi è stata modellata su quella degli usuali calcolatori tascabili, così che ogni quadro di un dossier può essere visto come una matrice di calcolatori tascabili (uno per ogni campo), dotati delle usuali operazioni:

CLEAR, +, −, /, *, %, ∧ (esponenziazione), =.

Esiste inoltre, come negli usuali calcolatori tascabili, un "registro di memoria", visibile sullo schermo fuori dal quadro, nel quale è possibile inserire valori volatili (non registrati sul floppy-dossier).

Tale registro di memoria è disponibile sia sulla pagina sinistra che sulla pagina centrale e destra di ogni modulo.

Le modalità di aggiornamento di un testo (pagina dei commenti) sono quelle di un semplice screen editor (sovrascrittura di caratteri, insert e delete di carattere e di linea, rottura di una linea su due linee, fusione di due linee in una). L'utente può, sulle pagine destra e sinistra, scrivere anche in modo "reverse".

c) **Inserire e Modificare**

Questo gruppo di funzioni, che, pur guidate da menù, richiedono da parte dell'utente maggior esperienza, permette di definire e/o di ristrutturare il formato del dossier (titoli e formati dei quadri, relazioni fra i campi di un quadro o di più quadri, definizione dei grafici, e così via).

Le relazioni fra i campi, che rendono "vivo" il dossier, vengono definite, modulo per modulo, sulla pagina sinistra, e sono espresse da semplici istruzioni di un apposito linguaggio, che verrà illustrato più oltre.

d) **Altre operazioni**

Le altre funzioni presenti nel menù principale sono operativamente molto semplici, e non richiedono, sostanzialmente, conoscenze aggiuntive da parte dell'utente:

CANCELLARE: permette di cancellare i *dati* presenti in un dossier, conservandone la struttura;

STAMPARE: permette di stampare il dossier (tutto, o solo parti: la sola copertina, il solo indice, un singolo modulo);

COPIARE: permette di duplicare un floppy/dossier su un altro floppy;

DISTRUGGERE: permette di distruggere un intero dossier (dati e struttura).

Su tutte queste operazioni, l'interfaccia utente è stata definita con l'obiettivo di rinforzare, nell'utente, il modello concreto dell'oggetto dossier.
La tecnica usata, a questo scopo, è stata quella di visualizzare sempre sul video, *dinamicamente,* il processo in corso, con immagini grafiche.

Così, ad esempio:

- la distruzione di un dossier viene visualizzata cancellando via via le pagine della immagine del dossier chiuso (fig. 1.b), man mano che le corrispondenti strutture dati sul floppy vengono distrutte. Alla fine della operazione di distruzione, pertanto, l'immagine del dossier scomparirà completamente dal video;

- analogamente, il processo di duplicazione di un dossier viene visualizzato presentando sul video, accanto al dossier da duplicare e al posto del menù principale, il nuovo dossier (inizialmente costituito dalla sola copertina bianca), al quale vengono via via aggiunte le pagine duplicate;

- eccetera.

È da sottolineare, in quanto esposto sopra, la netta separazione fra le funzioni di *creazione* del dossier (voci INSERIRE e MODIFICARE nel menù principale) e quelle di *aggiornamento* del dossier (voce AGGIORNARE).

Questa scelta è antitetica a quanto viene usualmente fornito dai sistemi per la gestione di fogli elettronici (in cui l'utente può, in ogni istante, variare sia i valori contenuti nel foglio elettronico che gli attributi strutturali del foglio stesso), ed è motivata dalle seguenti considerazioni:

- *impone all'utente un modo di procedere altamente strutturato:* in particolare, impone una separazione netta fra la fase di costruzione del dossier (progettazione della sua struttura, realizzazione e messa a punto) e la fase d'uso effettivo (lettura, aggiornamento e stampa);

- *protegge l'utente dai suoi stessi errori operativi:* in fase di lettura, i dati e la struttura del dossier non possono essere in alcun modo modificati; analogamente, in fase di aggiornamento la struttura del dossier è totalmente protetta;

- *permette l'uso del sistema da parte di diversi livelli di utenza* (solo lettura, solo lettura e aggiornamento, ecc).

È importante aggiungere che il menù principale è associato ad ogni singolo floppy/dossier, e può essere personalizzato (°).

Un utente potrà pertanto vedere (e apprendere) soltanto le operazioni a lui utili (ad esempio, solo VEDERE e STAMPARE, oppure solo VEDERE, AGGIORNARE, STAMPARE, CANCELLARE, e così via).
Ciò, oltre a semplificare ulteriormente l'interfaccia utente, può essere utile anche al fine di proteggere le informazioni contenute in un dossier (ad esempio, sopprimendo dal menù le operazioni di STAMPARE e COPIARE si può evitare il proliferare di copie indesiderate del dossier stesso).

4. LA STRUTTURA DI UN MODULO

4.1 Struttura di un quadro

I quadri del dossier elettronico sono stati progettati per riprodurre, per quanto è possibile, concetti e modalità operative già acquisite nell'uso della modulistica d'ufficio.

Un quadro, in DOSSIER, è essenzialmente una *tabella* costituita da *righe* e da *colonne,* la cui struttura viene definita dall'utente in fase di aggiornamento.

Un esempio tipico di quadro, ancora vuoto di dati, è mostrato in fig. 4.

Ogni quadro viene definito dall'utente con un dialogo interattivo, per successive modifiche di un quadro base, muto, proposto sul video dal programma, se-

(°) Questa funzione, nelle versioni attualmente distribuite, non è resa visibile all'utente.

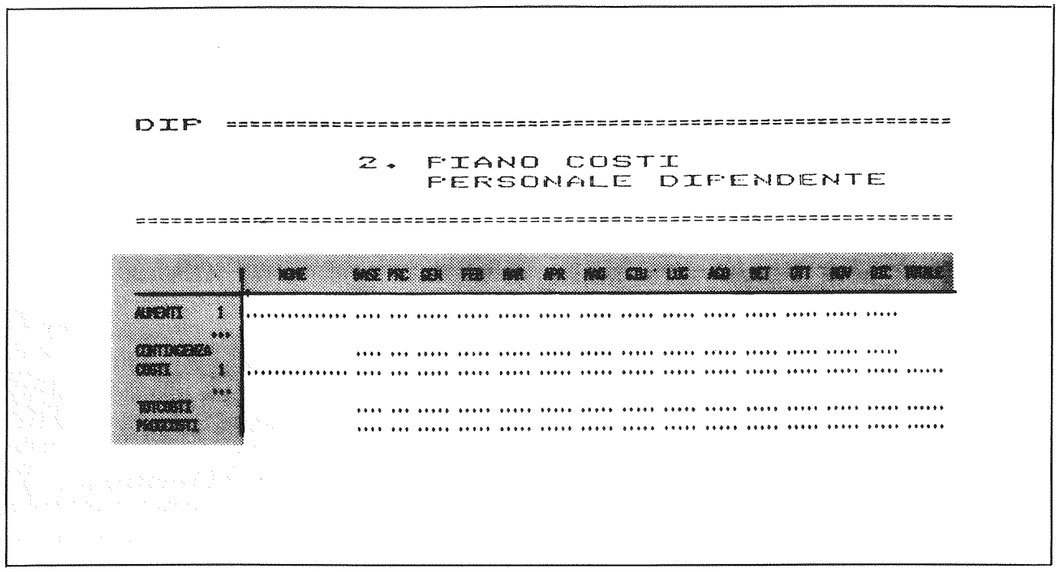


Fig. 4

condo modalità che riproducono quanto verrebbe fatto progettando un modulo su un foglio di carta.

Così ad ogni colonna vengono associati, in fase di costruzione, soltanto due *attributi:* un *titolo* (nell'esempio: NOME, BASE, PRC, ecc.) e una *larghezza,* cioè il numero massimo di caratteri che ogni campo della colonna può contenere. La larghezza della colonna viene visualizzata, sul video, con dei puntini (ad. es., la colonna NOME ha larghezza 15, le colonne GEN..DIC hanno ciascuna larghezza 5).

Analogamente, ad ogni riga viene associato un *titolo* (es. AUMENTI, CONTINGENZA, ecc.), ed un tipo.

Esistono due tipi di righe: le righe *semplici* (in figura, le righe CONTINGENZA, TOTCOSTI e PROGCOSTI) e le righe *multiple* (la riga AUMENTI e la riga COSTI).

Una riga multipla, pur essendo denotata da un unico titolo, è costituita da un numero *variabile* di righe fisiche (*voci*), numerate progressivamente in modo automatico. Ciò permette di inserire nuove voci o sopprimere voci non più necessarie con semplici comandi accettati *in fase di aggiornamento.* Il quadro, allora, si aprirà o richiuderà "a soffietto", e il sistema rinumererà automaticamente l'elenco così modificato.

Come si vede dalla figura, una riga multipla viene creata, in fase di definizione del quadro, con una sola voce, numerata con 1. I tre puntini, nella zona dei titoli, ricordano all'utente che a questa prima voce potranno essere aggiunte successivamente altre voci, a seconda delle esigenze.

4.2 Aritmetica sui campi

Sebbene ogni campo di un quadro possa contenere stringhe di caratteri qualsiasi, il programma riconosce quattro tipi di dati differenti:

- *numeri*

  Sono rappresentati internamente in virgola mobile, in doppia precisione, ma visualizzati, normalmente, in virgola fissa (per un migliore allineamento in colonna) con valore arrotondato a n cifre decimali (dove n≥0 è definito dall'utente per ogni quadro). Il valore di difetto è due cifre decimali.

- *stringhe*

  Sono sequenze costituite da caratteri qualsiasi, ad eccezione delle sequenze di caratteri che rappresentano i valori dei tipi ERRORE e VUOTO (vedi più oltre).

  Sequenze di caratteri numerici, con o senza segno, sono considerati numeri, e non stringhe.

- *ERRORE*

  Tale tipo è costituito da un unico valore, visualizzato con una sequenza di asterischi della lunghezza del campo. È il valore prodotto da operazioni indefinite (divisione per zero e overflow, operazioni aritmetiche su stringhe, operazioni aritmetiche sul valore ERRORE);

- *VUOTO*

  Anche tale tipo è costituito da un unico valore, visualizzato con una sequenza di puntini della lunghezza del campo. Denota "assenza di valore", ed è trattato come l'elemento "identità" rispetto a tutte le operazioni aritmetiche.

La tabella di fig. 5 mostra il tipo del risultato in funzione dei tipi degli operandi, per le cinque operazioni aritmetiche (+ −∗/% ∧).

operando 1 / operando 2	numero	VUOTO	ERRORE	stringa
numero	numero o ERRORE (!)	numero	ERRORE	ERRORE
VUOTO	numero	VUOTO	ERRORE	ERRORE
ERRORE	ERRORE	ERRORE	ERRORE	ERRORE
stringa	ERRORE	ERRORE	ERRORE	ERRORE

(!) Divisione per zero, overflow di macchina o di campo

Fig. 5

4.3 La pagina delle istruzioni

Si è detto, nel paragrafo 2, che i moduli di un dossier elettronico sono "vivi", nel senso che l'aggiornamento di un campo, da parte dell'utente, produce automaticamente il ricalcolo dei valori di tutti i campi ad esso relazionati, nello stesso quadro o in altri quadri dello stesso dossier.

Le modalità di tale ricalcolo vengono specificate dall'utente, mediante un linguaggio semplice ma potente, che permette di definire le computazioni che il sistema deve effettuare su un certo modulo, tutte le volte che l'utente lo richieda con apposito comando (tasto funzione CALC).

Più in particolare, tali istruzioni vengono annotate dall'utente, *in fase di costruzione del dossier,* sulla pagina delle istruzioni di ogni modulo (fig. 6).

La pagina delle istruzioni è sempre accessibile all'utente, sia in fase di costruzione (quando le istruzioni vengono redatte o modificate) sia in fase di aggiornamento/lettura (quando le istruzioni presenti nella terza pagina possono essere visualizzate sullo schermo, ma non possono essere modificate).

A commento della struttura illustrata in fig. 6, possiamo osservare che, dal punto di vista concettuale, ogni modulo elettronico può essere assimilato a un "programma" per operare su "quadri", espresso in un linguaggio composito (testuale e grafico), e dotato di una struttura tripartita:

- *definizione delle strutture dati:* (di input, di output e, eventualmente temporanei): è espressa in forma grafica dal quadro nella pagina centrale;

- *definizione delle computazioni sui dati:* le istruzioni nella pagina sinistra (vedi paragrafo successivo);

- *commenti:* il testo nella pagina destra. È interes-

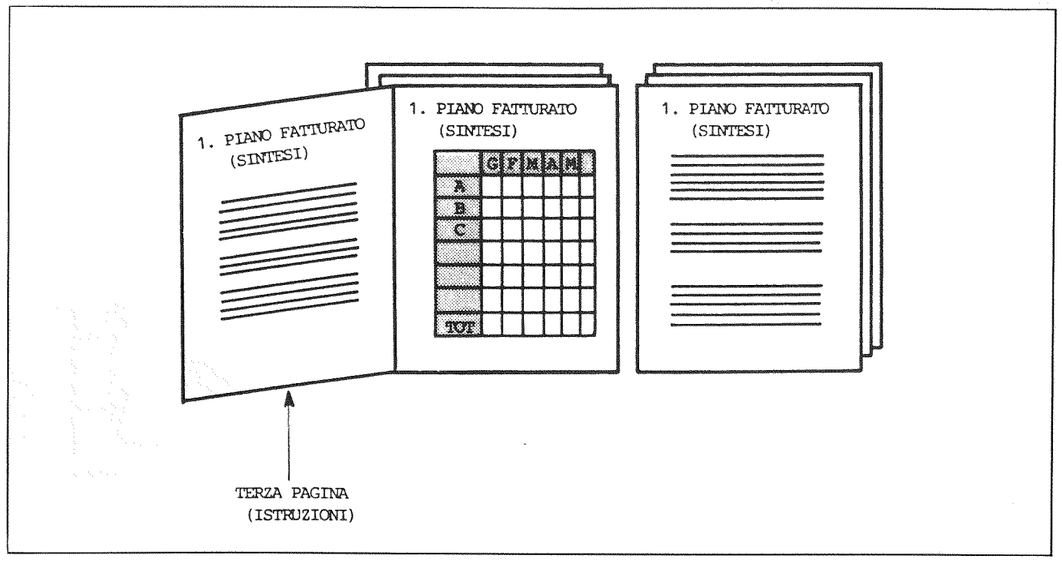


Fig. 6

sante osservare che, nella pratica corrente, tali commenti esprimeranno, di fatto, da un lato proprietà invarianti del quadro e, dall'altro, osservazioni sui dati di input e di output.

Il titolo del modulo (riportato su tutte e tre le pagine) è allora assimilabile al *nome* del programma, e l'intero dossier a una famiglia di programmi tra loro comunicanti.

4.4 Il linguaggio per la specifica delle istruzioni

Si tratta, in sostanza, di un linguaggio di programmazione per operare su quadri, come definiti in 4.1. Tale linguaggio, per facilitarne l'uso da parte di utenti inesperti di programmazione, è stato dotato di due soli tipi di statement, semplici ma potenti: *assegnamenti e richiami di procedure predefinite.*

Gli assegnamenti sono delle forma:

‹sottoquadro› = ‹espressione›;

dove ‹sottoquadro› rappresenta uno dei sottoquadri della pagina centrale *dello stesso modulo,* ed ‹espressione› è una espressione aritmetica comunque complessa costituita da:

- parentesi;
- costanti scalari (numeri o stringhe);
- sottoquadri (dello stesso modulo *o di altri moduli* dello stesso dossier);
- operatori aritmetici (+,−,∗,/,%,∧);
- richiami di funzioni con argomento di tipo sottoquadro (°);
- richiamo di funzioni con argomento di tipo sottoquadro e valore di tipo sottoquadro (°).

Un sottoquadro è espresso mediante la sintassi illustrata in fig. 7.

Ad esempio, alcuni sottoquadri del quadro di fig. 4 sono:

1) AUMENTI..PROGCOSTI (NOME..TOTALE)
2) AUMENTI..PROGCOSTI
3) AUMENTI (TOTALE)
4) CONTINGENZA (MAG)
5) PROGCOSTI (GEN..DIC).

Si noti che 1) e 2) sono sinonimi, e denotano l'intero quadro: 3) denota l'ultima colonna della riga multipla

(°) Le funzioni attualmente fornite sono le seguenti:
- a valore scalare numerico: sommatoria, media, conteggio, massimo e minimo;
- a valore sottoquadro: totali per riga, totali per colonna; progressivo per riga; progressivo per colonna; variazione percentuale per riga; variazione percentuale per colonna; media, massimo, minimo, conteggio per riga e per colonna.

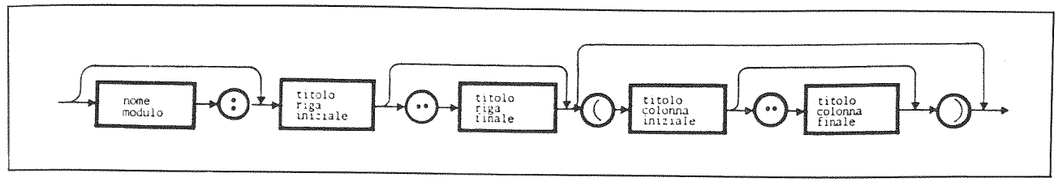
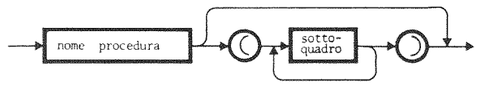


Fig. 7

AUMENTI; 4) denota un singolo valore scalare; 5) denota un segmento della riga PROGCOSTI, e così via.

I richiami di procedure sono della forma:



e permettono ad esempio di definire le modalità di costruzione della pagina grafica associata ad un quadro, di modificare la precisione dei calcoli, ecc.

4.5 Un esempio

Data la natura riassuntiva di questa nota, ci limitiamo a mostrare un esempio d'uso di tale linguaggio, senza esporne la sintassi e la semantica in modo dettagliato.

Considerando il quadro di fig. 4, vogliamo utilizzarlo nel seguente modo:

- *sottoquadri di input* (valori che verranno inseriti dall'utente in fase di aggiornamento):

.. riga multipla AUMENTI: conterrà una voce per ciascuno dei dipendenti di una certa azienda.
Ogni voce riporterà il cognome del dipendente (colonna NOME), il suo stipendio lordo attuale (BASE), la percentuale di tempo lavorativo per eventuali rapporti part-time (PRC), il piano di aumenti di stipendio (lordo) per quel dipendente allocato nel mese di attuazione: colonne GEN..DIC;

.. riga CONTINGENZA: conterrà le previsioni (lordi) dovuti agli scatti di contingenza (colonne FEB, MAG, AGO, NOV):

- *sottoquadri di output* (valori che verranno calcolati automaticamente sulla base dei dati di cui sopra):

.. riga multipla COSTI: conterrà, per ogni dipendente e per ogni mese, il costo aziendale dello stipendio del dipendente stesso, sulla base delle variabili indipendenti riportate in AUMENTI e CONTINGENZA e tenendo conto dei costi dei contributi previdenziali e fiscali, della tredicesima e quattordicesima

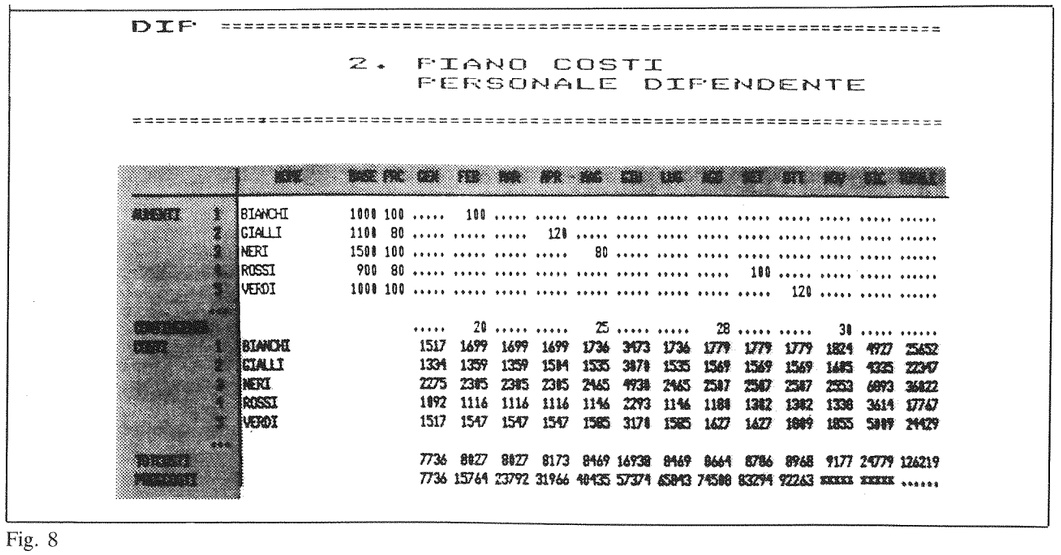


Fig. 8

```
DIP ============================================================================

                    2. PIANO COSTI
                       PERSONALE DIPENDENTE

================================================================================


COSTI(GEN..DIC)=0;

COSTI(FEB)=CONTINGENZA(FEB);
COSTI(MAG)=CONTINGENZA(MAG);
COSTI(AGO)=CONTINGENZA(AGO);
COSTI(NOV)=CONTINGENZA(NOV);

COSTI(NOME)=AUMENTI(NOME);

COSTI(GEN..DIC)=COSTI(GEN..DIC)+AUMENTI(GEN..DIC);
COSTI(GEN..DIC)=PROGR( COSTI(GEN..DIC) );
COSTI(GEN..DIC)=((COSTI(GEN..DIC)+AUMENTI(BASE))*1.517)*AUMENTI(PRC)/100;

COSTI(GIU)=COSTI(GIU)*2;
COSTI(DIC)=COSTI(DIC)*2.7;

TOTCOSTI(GEN..DIC)=TOTC(COSTI(GEN..DIC));
PROGCOSTI(GEN..DIC)=PROGR(TOTCOSTI(GEN..DIC));
COSTI..TOTCOSTI(TOTALE)=TOTR(COSTI..TOTCOSTI(GEN..DIC));
BAR(TOTCOSTI(GEN..DIC));
```

Fig. 9

mensilità, e dell'accantonamento della indennità di liquidazione (allocate nel mese di giugno e dicembre);

.. righe TOTCOSTI e PROGCOSTI: riporteranno, rispettivamente, i totali mensili e progressivi dei COSTI, mentre la colonna TOTALE riporterà il totale dei COSTI annuali relativi a ciascun dipendente.

Un esempio del quadro DIP riportante i dati relativi a cinque dipendenti è mostrato in fig. 8.

La fig. 9 mostra un possibile esempio di programma, contenente le istruzioni per il calcolo automatico dei valori riportati a tratteggio in fig. 8, a partire dagli altri valori indipendenti inseriti nello stesso quadro.

4.6 Pagine grafiche

I moduli contenenti istruzioni grafiche sono dotati di un'ulteriore pagina "logica" per visualizzare il grafico (fig. 10).

Su tale "pagina grafica" l'utente potrà far tracciare automaticamente i grafici desiderati, specificando nel solito modo, sulla pagina delle istruzioni, opportune istruzioni (richiami di procedure predefinite).

5. NOTE REALIZZATIVE

Un sistema come quello descritto nei paragrafi precedenti presenta complessi problemi di architettura software, soprattutto tenendo conto dell'accurato bilanciamento fra occupazione di memoria e tempi di esecuzione imposto delle risorse limitate di un elaboratore personale.

L'architettura attuale è il risultato di numerose versioni (più di una decina). Essa si basa su numerosi livelli di macchine virtuali, realizzate in Pascal UCSD e utilizzando estesamente alcune caratteristiche non standard disponibili in tale linguaggio: il concetto di *unit* per realizzare macchine virtuali, per compilazioni separate e per segmentazione del codice oggetto; primitive di I/O a livello fisico sui floppy; primitive di gestione di stringhe.

Oltre ad alcune macchine virtuali di base (gestione strutture dati su floppy; gestione rappresentazione moduli in memoria centrale; gestione visualizzazione moduli; gestione menù e "help"; primitive grafiche ad alto livello e aritmetica), il sistema contiene un semplice screen editor generalizzato, un compilatore per tradurre il linguaggio di istruzioni sui quadri in un linguaggio interno molto compatto per una macchina virtuale a stack, tipo P-code (detto D-code), e un interprete del D-code.

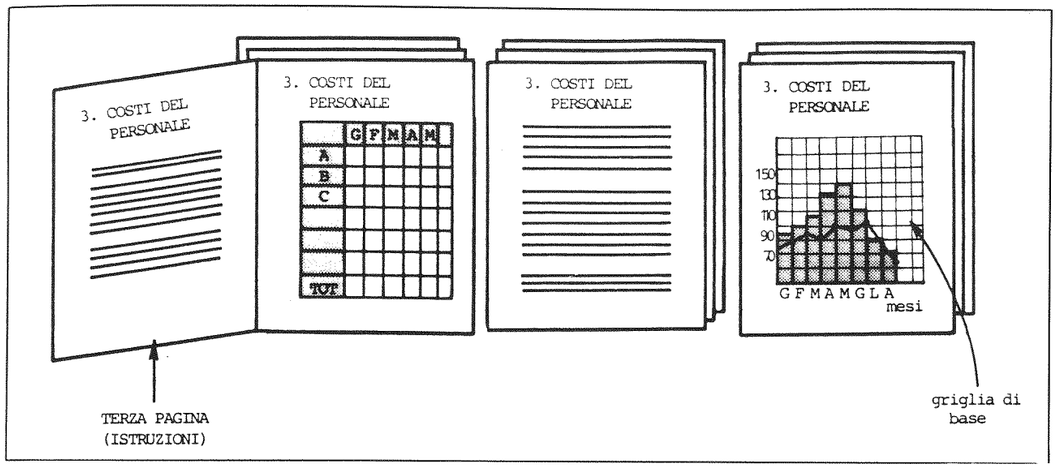


Fig. 10

Tale tecnica di gestione degli aggiornamenti automatici dei quadri (che avrebbero potuto essere trattati più efficientemente, a costo di un maggior spreco di memoria centrale, rappresentando le relazioni tra campi con strutture a puntatori) è stata imposta dall'obbiettivo di non superare il limita massimo di 128 K bytes di memoria centrale.

6. CONCLUSIONI

La presente nota ha descritto brevemente gli aspetti salienti di DOSSIER, un sistema software che implementa il concetto di dossier elettronico su elaboratori personali.

Nonostante che il concetto di dossier elettronico sia suscettibile di svariate applicazioni, esso è stato concepito principalmente come uno strumento per l'ufficio, orientato a utenti con funzioni direttive e di controllo.

Si pensi, ad esempio, all'uso dello strumento per tenere sotto controllo in modo ordinato e strutturato il budget di un settore aziendale nelle sue varie voci, gli indicatori economici fondamentali di un progetto, le previsioni di costi del personale di una piccola azienda, eccetera.

È da sottolineare, a questo proposito, che i vantaggi dello strumento sono molteplici:

- permette di organizzare a priori, in modo ordinato e fortemente strutturato, l'insieme dei quadri-base che riassumono (eventualmente su più livelli di dettaglio) i dati fondamentali del settore che si desidera tenere sotto controllo (fase di costruzione del dossier);
- permette di aggiornare con semplicità tali quadri, man mano che nuovi dati di base risultino disponibili, effettuando automaticamente, ad ogni aggiornamento, il ricalcolo di tutti i dati correlati ai nuovi input (fase di aggiornamento);
- fornisce in modo automatico una visualizzazione grafica degli andamenti più rilevanti, e mantiene aggiornata automaticamente tale visualizzazione ad ogni nuovo aggiornamento);
- fornisce immediatamente, su richiesta, un "report" su carta, completo e già organizzato (l'intero dossier, con la copertina, l'indice, i vari moduli) oppure report parziali (ad esempio, i soli grafici);
- può essere appreso in modo incrementale (ad es., solo comandi di aggiornamento) e con rapidità (fornisce all'utente un modello concettuale che fa esclusivamente riferimento a nozioni abituali nel modo dell'ufficio).

Ciò è ottenibile facendo uso di tecnologie oggi disponibili a basso costo: personal computers di tipo professionale, dotato di video a 80 colonne e con capacità grafiche a media risuluzione, tastiera dotata, preferibilmente, di qualche tasto funzione, capacità di memoria di almeno 128 K bytes.

7. RINGRAZIAMENTI

Numerose persone hanno contribuito, con osservazioni e commenti, alla messa a punto della versione attuale del sistema, evolutasi da svariate versioni prototipali. Il progetto software è stato realizzato da un

gruppo costituito da Alessandro Mazzetti, Riccardo Paulin, Roberto Polillo, Luisa Reina e Piero Schiavo Campo.
Gli autori ringraziano G. Degli Antoni per aver fornito, nel corso di svariate discussioni, numerosi spunti sulle tecniche di prototyping, e sugli strumenti software per l'automazione d'ufficio.
La struttura del dossier elettronico costituisce un'applicazione della metodologia di documentazione PSPN, sviluppata e largamente sperimentata nell'ambito dell'Istituto di Cibernetica dell'Università di Milano, dove sono stati altresì sviluppati numerosi prototipi di strumenti per l'automazione dell'ufficio.

8. RIFERIMENTI

[1] VISICALC User Guide, Personal Software Inc., Sunnyvale, CA.

[2] D.C. Smith, C. Irby, R. Kimball, B. Verplank, E. Harslem, DESIGNING THE STAR USER INTERFACE, BYTE, April 1982, pp. 242-282.

[3] G. Attardi, G. Barber, M. Simi, VERSO UNA STAZIONE DI LAVORO INTEGRATA PER L'UFFICIO, Automazione e Strumentazione, 3 (1989).

[4] G. Degli Antoni, M. Maiocchi, R. Polillo, PSPN: UN APPROCCIO AI PROBLEMI DI COMUNICAZIONE DELL'INFORMAZIONE TECNICA, Note di Software, n. 6/7, Aprile-Settembre 1978, pp. 32-35.

[5] G. Degli Antoni, R. Polillo, M. Pozzi, B. Zonta, DALL'UFFICIO DEL FUTURO ALL'UFFICIO DI OGGI, Convegno AICA 1980, Bologna, pp. 1170-1182.

GUIDELINES TO CHECK COHERENCE IN EXPLANATORY TEXTS

Graziella Tonfoni - Bertram Bruce (*)

RIASSUNTO

Questo articolo intende presentare un modello per il controllo e la verifica dei livelli di coerenza nei testi di carattere esplificativo. Attraverso la identificazione di diversi elementi e fattori quali determinatori della coerenza, tale modello fornisce uno schema per la identificazione dei problemi e le relative strategie di ristabilimento della coerenza.

ABSTRACT

This papers deals with the problem of coherence in Explanatory Texts. The model which has been presented gives a set of evaluation criteria in order to establish the level of coherence which has been reached by a certain text, to check wether such level is adequate and to increase the level of coherence if such is not the case by using the guidelines proposed by the computational model.

1. INTRODUTION

A substantial part of recent research on reading has focused on the importance of well-structured texts in both comprehension of a text and subsequent recall of the information the text contains (see Anderson, Osborn & Tierney, 1981). This research has produced a variety of methods for analyzing texts of various types: stories, non-fictional narratives, persuasive essays, and explanatory texts. Of all these, texts whose purpose is to explain (or inform) are perhaps the most important for learning in school. Yet, while there is a rich literature describing methods for story analysis (Bruce & Newman, 1978; Mandler & Johnson, 1977; Rumelhart, 1975; Stein-Glenn, 1978; Wilensky, 1978), the methods for analysis of explanatory texts are either partial or overly general. Important contributes to resolving this problem have been made by (Anderson, Spiro & Andersonb, 1978; Goetz & Armbruster, 1980; Tierney, Mosenthal & Kantor, 1980). Armbruster, 1980; Tierney, Rosenthal & Kantor, 1980).

There are many reasons to want to have better methods of analysis for explanatory texts. Tonfoni (1982), for example, has shown that coherent explanatory texts are easier to comprehend, summarize and remember. Problems that children have in learning from texts may be attributable to the incoherence of the texts they are expected to master. It would help in teaching if one could determine wheter and in what ways a particular text lacked coherence. Moreover, knowing just how a text failed to be coherent could be useful in teaching strategies to a reader for coping with the "inconsiderateness" (Anderson, in press) of the text. Publishers, who produce texts, could well use guidelines that would ensure coherence in the context of satisfying the numerous demands placed upon them in producing school texts.
Those who select texts for children to read could likewise use help in determining what makes a text coherent.

This paper draws from research on coherence in explanatory texts (Tonfoni, 1982). It presents a set of principles for examining texts that can be used to determine relative coherence and also to point to areas in which a particular text might be improved. The principles, or guidelines, define a method of analysis that is consistent with, but more precise than, our intuitive notion of coherence. We hope that these guidelines will be improved as the result of application of the guidelines to school texts and further research on text analysis.

2. WHAT DO WE MEAN BY EXPLANATORY TEXT?

We can define an Explanatory Text as a text whose main goal is to explain something, to provide information, or to teach something about a specific topic.

(*) G. Tonfoni: Bolt Beranek and Newman, Cambridge, MA, USA - Università di Bologna, Italia.
B. Bruce: Bolt Beranek and Newuman, Cambridge, MA, USA.

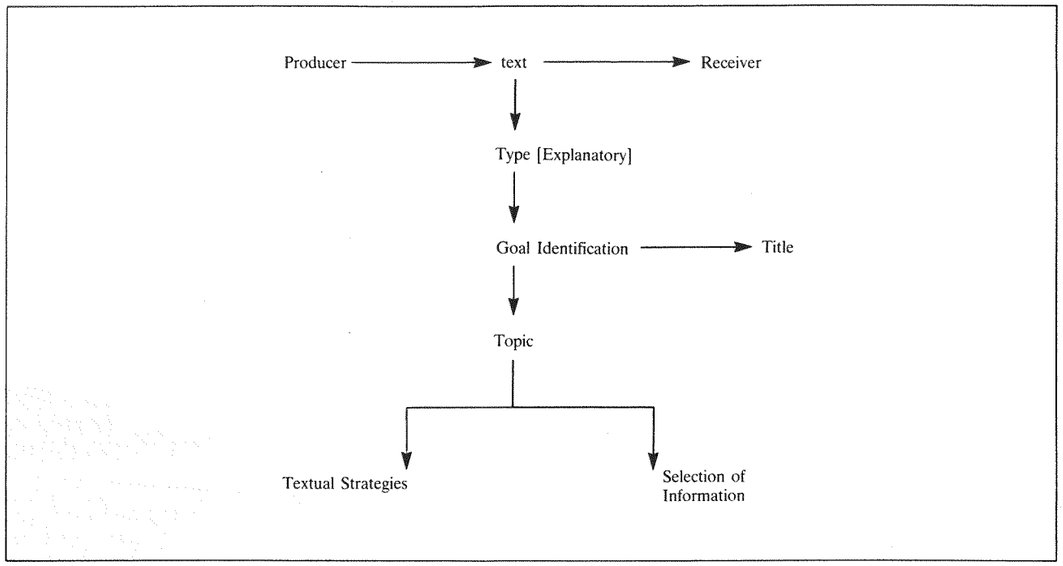


Fig. 1

Writing a text that has explanatory goals implies the use of specific constraints and rules on Text-production relative to those goals. The author must be aware of which information he or she will select to have the reader understand from reading the text.

Explanatory texts can have different levels of specificity and different amounts of technical details; different topics determine a different choice of writing strategies.
An Explanatory Text can also have different levels of redundancy.
Redundancy can help the reader remember important facts given in the text; too much redundancy can be misleading and can create problems in text-understanding.

Writing an Explanatory Text requires a selection of the most relevant facts or pieces of information which must be given about a specific topic. The identification of the relevant information in turn requires the use of what we call Focus-Strategies in order to have the reader understand and select such information as opposed to other redundant or not strictly relevant information which might also be in the text.

Consider figure 1:

In order to produce a satisfactorily explanatory text, which can be understood by a text-receiver, the text-producer must identify one or more goals by choosing a topic. After having chosen a topic he or she will select the information which is most relevant.

The goal of an Explanatory Text is typically represented by its title. Any selection of information necessarily implies a set of specific textual strategies. Some important strategies in Explanatory Text Producing are those which can improve Text-Understanding. Focalization strategies, as well as topic-maintenance strategies are critical, because they force the Receiver/Learner to receive and understand the Text in a specific way.

3. WHAT IS COHERENCE IN AN EXPLANATORY TEXT?

We might say that an Explanatory Text is coherent when it is logically consistent and when its parts fit together into a unified whole. But, this definitions is insufficient. A text that satisfies the definition may still be incoherent if the reader cannot determine exactly what the author wanted to say.
Thus we should add that an Explanatory Text is coherent only when the Reader (Text-Receiver) can identify the Author's (Text-Producer) goal which means he or she can identify the topic and understand the text in the manner that the Text Producer desired (see Adams & Bruce, 1982).

But the definition is still incomplete. An Explanatory Text is coherent only when all its parts are causally or temporally related and when there are no gaps

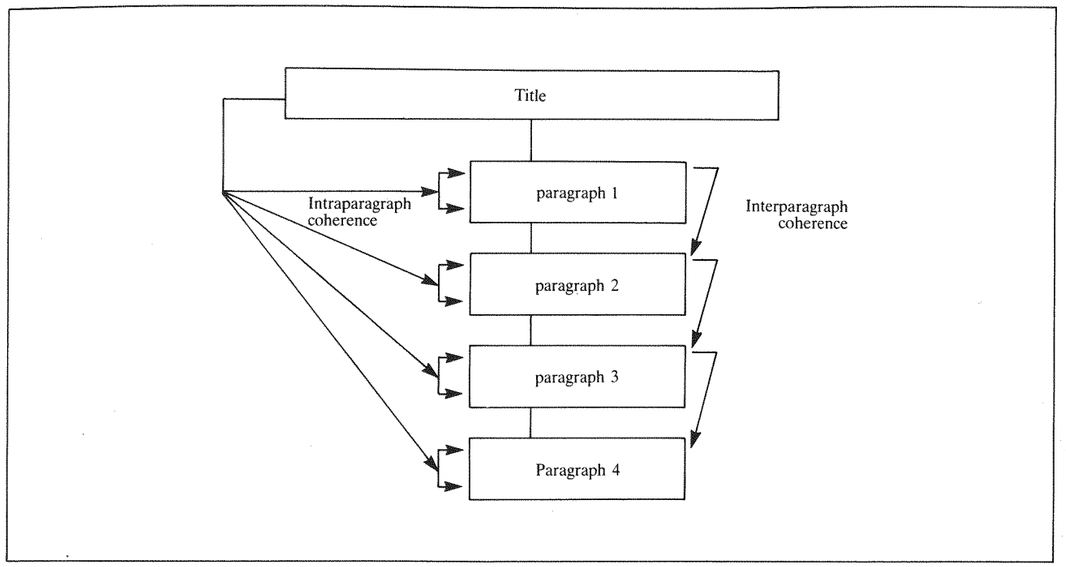


Fig. 2

which cannot be filled by the use of information given previously in the text or likely to be known by the "ideal reader" (Fillmore, 1980). Furthermore, the information selection must be compatible with the Text Receiver's presuppositions and expectations. Thus, coherence is the complex result of a set of variables which determine and affect any text production.

Coherence in explanatory texts can be checked at various levels. For example, we can distinguish between local coherence (the coherence of a sentence or of a short set of sentences) and global coherence (the coherence of a paragraph or a whole text with respect to the topic given by the title). The coherence of an explanatory text can be better defined depending on the nature of its topic; for example, a history text will base its coherence on logical-chronological relations, a descriptive text will base its coherence on logically ordered sequences.

By distinguishing different levels and degrees of coherence within the same text, we can also consider different levels of understanding of a text. Which are the most relevant features of an explanatory text? An explanatory text has to make evident the topic and the information which has been selected and given. Whenever a condition on coherence is violated, confusion is likely to result.

An explanatory text might be incoherent at some level, although it is coherent at some others. For example, an explanatory text could be coherent with respect to its topic, but syntactically disconnected, or it could be syntactically connected but not coherent with respect to the given topic, or its parts might not be logically - chronologically ordered, or textual coherence could be compromised by a set of abrupt topic-shifts (1). All those aspects create troubles in understanding and can all be considered as a result of an incorrect use of textual strategies.

To summarize, we can say that any attemp to define textual coherence has to be specified by identifying different degrees of coherence in the different levels of a text.

4. DIFFERENT LEVELS OF A TEXT

A text can be defined as a set of sentences which are syntactically and semantically related (Halliday-Hasan 1976).
Syntatic relations include such things as anaphora (e.g., pronominalization) or the use of connectors. Connectors are linguistic elements that explicitly link sentences together.
They can represent logical or temporal relations (e.g., "since", "in spite of", "after that"). Semantic relations include such things as topic maintenance or

(1) Topic-shift is the change of the topic from one paragraph into another or from a sentence into another.

topic domain maintenance. This means that, once a topic has been given by the title of the text, the text must keep it as the focus of attention. If the topic shifs, the new topic has to be part of the main topic-domain and has to be relevant to the main topic. By referring back to the syntatic coherence, we can easily identify different levels within an explanatory text:

1. Sentence level (correctness of a sentence);

2. Paragraph level (coherence in relating and chaining a set of sentences by a correct use of connectors and connecting strategies);

3. Textual level (coherence in relating and chaining a set of paragraphs by a correct use of connecting strategies).

Referring to semantic coherence we can again distinguish different levels in explanatory text:

1. Sentence level
2. Paragraph level
3. Textual level
4. Main Topic-Title level

We can represent a text as in figure 2. According to this schema we can distinguish different levels of coherence:

1. Intraparagraph coherence: how sentences of a paragraph are related by the use of connectors and connettive strategies.

2. Interparagraph coherence: how different paragraphs of a text are related by the use of connective strategies.

3. Textual coherence: how the whole text is related to the topic given in the title.

Texts can be ranked with respect to their level of coherence, for example, a text coherent at level (1), but not at levels (2) or (3) would be less coherent than one which was coherent at all three levels. We can represent different levels of coherence in Figure 3.

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
Intraparagraph Coherent	X	X	X	X				
Interparagraph Coherent	X	X			X		X	
Textual Coherent	X			X	X			X

Fig. 3

The highest coherent text is T_1 the lowest is T_6. We can have intermediate levels of coherence (2 degrees - 1 degree).

5. PRINCIPLES AND RULES FOR COHERENCE IN EXPLANATORY TEXTS

In an explanatory text "what is difficult to understand" or the new information which has to be learned by the Text-Receiver can be defined as the topic of the text. The topic of an explanatory text is relevant in the following rules:

(a) **Topic identification:** the topic has to be given in the title and has to be identifiable within a paragraph, and within the texts.

(b) **Topic maintenance:** the topic given by the title has to be kept in the paragraphs and in the text.

(c) **Topic-maintenance:** if the identified topic of a text is not kept within the whole text, those paragraphs which don't keep it have to have another topic which is nevertheless part of the main topic-domain and which is strictly related (logically-temporally) to it.

(d) **Topic-shifting:** the main topic can be changed from a paragraph to another; the new topic is not part of the main topic-domain only if it respects: (e) Adherence principle, (f) Reestablishment principle.

(e) **Topic-adherence:** if the main topic is changed between paragraphs and the new topic is not part of the main-topic-domain, it has to be related to (logically temporally) and adherent to the main one.

(f) **Topic-reestablishment:** if the main-topic is changed between paragraphs and the new topic is not part of the main topic-domain we have a set of informations, which we can define as "digression". At the end of the digression, the main topic has to be reestablished.

We can represent topicalness and the set of principles in the diagram of figure 4.

In order to maintain coherence, an ideal explanatory text must use correct connectors and connective strategies by following a logical or temporal ordering.

(g) **Logical ordering:** sequences and paragraphs must follow by respecting causal logical relations (especially expository texts).

(h) **Temporal ordering:** sequences and paragraphs must follow by respecting temporal sequencing (specifically expository narrative texts).

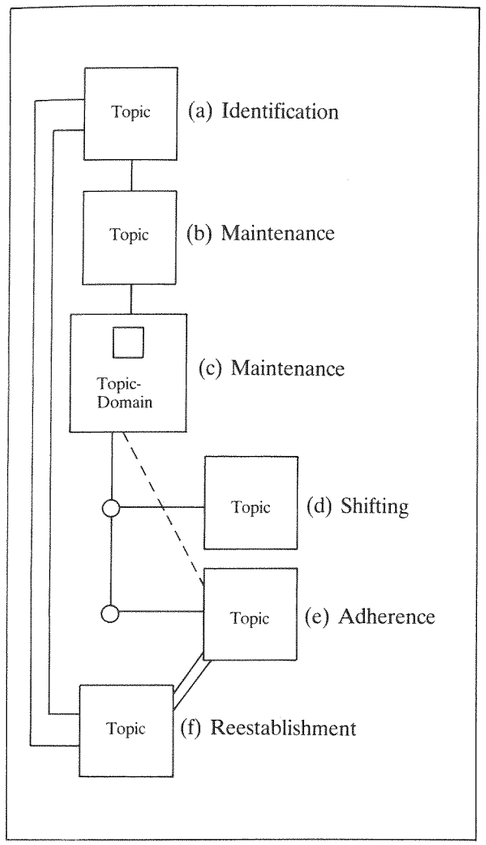


Fig. 4

The ideal coherent text would satisfy all of the following principles:

(i) **Specificity:** the information given within each paragraph and more generally within the entire text has to be specifically related to the topic or topic-domain, which means to the title.

(l) **Pertinence:** any kind of information given within the paragraph of the text which is not specifically related to the topic must be pertinent and be related at least to the topic-domain. Any digression, has, however, to be justified and cann t be inserted randomly.

(m) **Focus Evidence:** the topic of the paragraph or the text must be the focus and has to be recognizable.

(n) **Style Continuity:** the style used in the different paragraphs and in the whole text has to be homogeneous.

(o) **Explicit Coreference:** any anaphoric pronoun or noun must have an explicit coreferential antecedent and must be totally unambiguous.

(p) **No Gapping:** paragraphs as well as sentences cannot violate (g) and (h) which means that information has to be given explicitly and by respecting logical-chronological sequencing.

We can represent an explanatory text with respect to the whole set of principles which have been stated about coherence in the schema of figure 5.

Any Expository text can be evaluated by considering this whole set of principles if we want to state the degree of coherence according to the schema 9 fig. 4.

To summarize, principles (a), (b), (c), (d), (e), (f), determine intraparagraph coherence, (g), (h), (l), (m), (n), (o), (p) determine interparagraph coherence, (i) determines textual coherence.

6. COHERENCE AND UNDERSTANDING, COHERENCE AND LEARNING

The coherence of an explanatory text is strictly related to the way information about the topic has been selected and ordered. In other words, since the main goal of an Explanatory Text is to give a set of facts and to teach something about a specific topic, the text has to be produced in the best possible way in order to facilitate the reader's understanding.

According to the distinction made between intraparagraph, interparagraph and textual coherence we can identify "local coherence", "extended local coherence" and "global coherence", which corresponds to the previous categorization.

Local coherence determines a good understanding of a part of a text but does not imply a global understanding of a text.

Extended coherence can help in memorizing and storing more facts, but still is not sufficient to reach a global understanding of a text.

Global Text-Understanding can be reached only by texts which have a 3 degree coherence. Studies (Tonfoni, 1982) have shown that only 3 degree coherent texts can be really understood and remembered. This hypothesis has been supported by a set of experiments done with high school students in Italy (Bologna) by giving them texts having different levels of coherence and by checking their understanding of the texts through summarization and question asking. The most coherent texts turned out to be the best understood, summarized, and remembered.

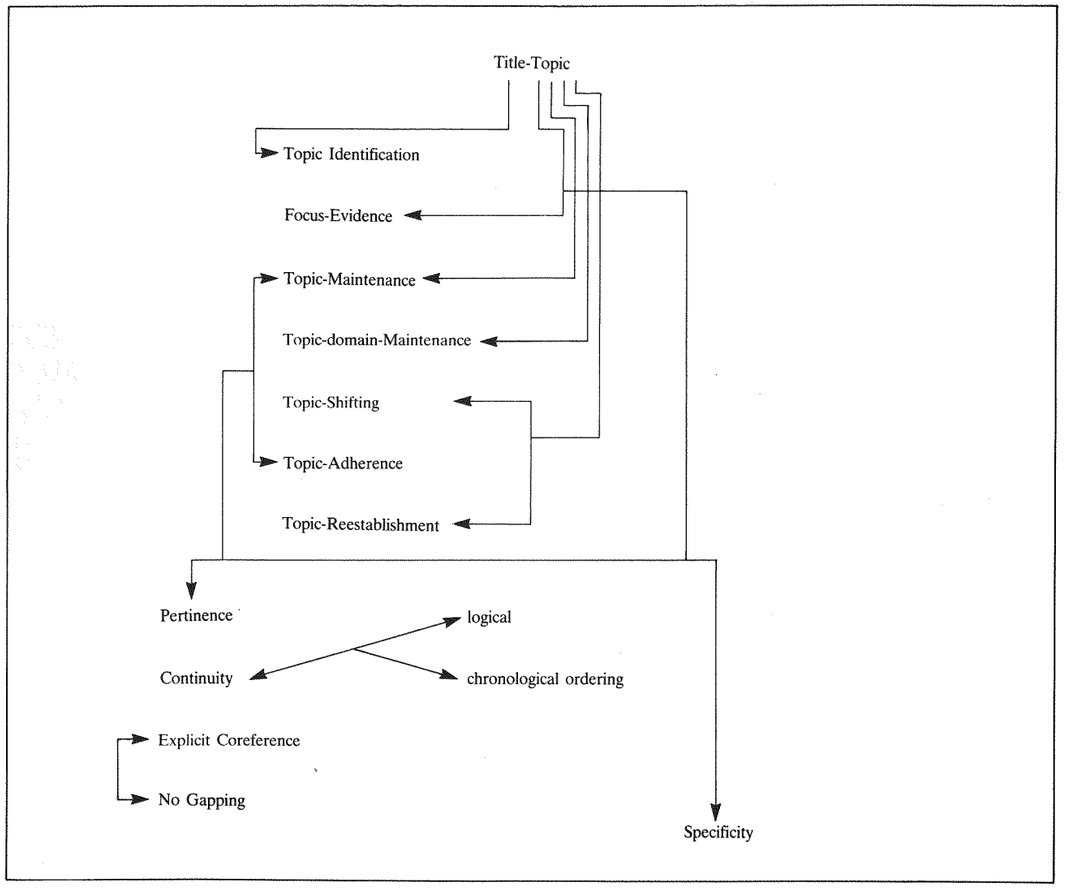


Fig. 5

By understanding, we mean being able to identify the topic and to reorganize the information by selecting and keeping the most relevant elements according to the choice previously made by the Author (Text-Producer).

Text-Understanding can be checked by asking the Text-Receiver (Learner) to summarize it. Summarization shows the level of understanding which has been reached by the Learner.
Text Understanding is a precondition for learning from a text.

Figure 6 summarize what has been stated:

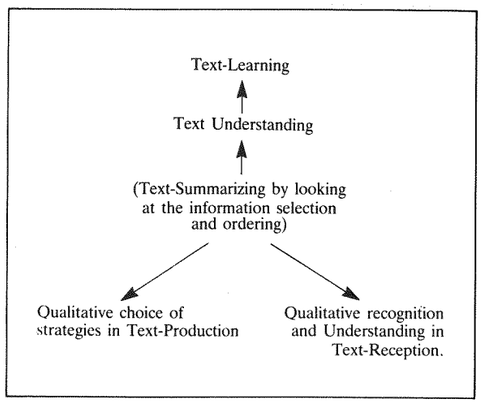


Fig. 6

7. HOW TO EVALUATE A TEXT BY LOOKING AT ITS COHERENCE

A text can be evaluated by analyzing its coherence degree.
Referring back to the schema given in figure 3 it is possible to identify 8 possibilities just by looking at the way texts have been produced.

Let's consider the following text:

Which People Came from Asia? (2)

Many Chinese came to America in the 1840's when gold was discovered in California. Some Chinese looked for gold. Others found that they could make more money working as cooks and in other service jobs. The Chinese were not always welcome in the new land. Many people thought they were different. Some white workers were afraid the Chinese would take away their jobs.

Chicago has a Chinatown. It is a community near the Stevenson Expressway, a short distance from downtown.
The main business street; Wentworth Avenue, is lined with restaurants and shops which tourists like to visit.

Many young Chinese are moving from Chinatown to North Side neighborhoods. Young Chinese women and men work in business and professionals jobs. But everyone likes to return to Chinatown on the Chinese New Year. The dragon dances and fireworks make this and exciting event.

The text which has been presented as an example shows a very low level of coherence:

(a)	Topic-identification (People coming from Asia)	
(b)	Topic-maintenance	NO
(c)	Topic-domain-maint.enance	YES
(d)	Topic-shifting	YES
(e)	Topic-adherence	NO
(f)	Topic-reestablishment	NO
(g)	Logical ordering	NO
(h)	Temporal ordering	NO
(l)	Pertinence	YES
(m)	Focus-Evidence	NO
(n)	Styl-Continuity	YES
(o)	Explicit Coference	NO
(p)	No Gapping	NO
(i)	Specificity	NO

(2) This text appears in "Chicago - The City and Its People". M. Staned, Benedict Press, Chicago, 1981 p. 51.

The text doesn't actually give information about the topic (Asian People) but only about some of them (Chinese People). The topic is shifted several times wihout being reestablished.
There is no topic adherence. Paragraphs are neither logically nor temporally ordered. We do have pertinence about the shifted topic (Chinese People) but there is no Focus Evidence. We have Style Continuity but no Explicit Coreference due to the continuous shifting. We do have Gapping in information because of the lack of the main topic. Finally, it follows that the text doesn't have any specificity.

It is possible to revise the text so that it conforms to all of the coherence principles. The basic information can be reorganized in order to produce a more appropriate and coherent text as follows:

Which People Came From Asia?

Many people came from Asia to the States: Chinese Koreans, and Japanese. In particular, Chinese people came to America in the 1840's when gold was discovered in California. Some Chinese looked for gold, others found other jobs. The Chinese were not always welcome in the new land because many of the other people already living in the States thought that Chinese were different and some white workers were also afraid the Chinese would take away their jobs.

Chinese people used to live in a part of the city called Chinatown. Chicago has a Chinatown which is a community near the Stevenson Expressway, a short distance from downtown. The main business street, Wentworth Avenue, is lined with restaurants and shops that tourists like to visit.

Many young Chinese are moving from Chinatown to Northside neighborhoods. Young Chinese women and men work in business and professional jobs but everyone likes to return to Chinatown for the Chinese New Year when the dragon dances and the fireworks made this an exciting event.

According to our ranking scale we can produce different text having various degrees of adherence to the set of identified principles.

8. CONSISTENCY OF THE GUIDELINES WITH INTUITIVE JUDGMENTS

The guidelines given above are far more specific than our intuitive notion of "coherence" and give a finer grained measure of relative coherence of texts. The question arises: Is the resulting measure consistent with our intuitive notion?

To answer this question, we asked ten (non-researcher) adults to rank eight texts in order of coherence.

The texts were all short selections from the Chicago text. Each text represented one of the eight categories shown in Figure 3. The results of the subjects' rankings are shown in Table 1.

Examination of the table shows that the text rated most coherent by the subjects was coherent at all three levels; the text rated least coherent was not coherent at any level. In general, textual and intraparagraph coherence seem more significant than interparagraph coherence.

9. CONCLUSION

Any Text-Producer must select the information according to the stated principles in the following order:

Identify Topic (Topic-Identification) Specificity;

Organize Information which is pertinent (Logical-Ordering, No Gapping, Pertinence);

Identify Priorities (Focus Evidence) in Information Giving and the Relative Hierarchical Structure;

Choose Correct Style (Continuity Strategies for In/Explicit Coreference Formation; Topic Maintenance, Topic Domain/Topic Shifting,Topic Adherence).

By respecting this set of principles Explanatory Text will be well formed and ready to be read and understood by the Text Receiver/Learner. Learning from a text implies being able to select that information which is the most relevant and retain it in memory. There is no a priori way of recognizing the most relevant information: it has to be made evident by a correct Text-Production. This is why we wanted to underline the importance of a correct Text-Production which will improve teaching methods and learning techniques.

Rank	Text Coherent	Inter	Intra
1 (best)	YES	YES	YES
2	YES	NO	YES
3	YES	NO	YES
4	NO	YES	YES
5	YES	NO	NO
6	NO	NO	YES
7	NO	YES	NO
8 (worst)	NO	NO	NO

Table I: Comparison of Rankings in Terms of Coherence with Prior Text Analysis

10. REFERENCES

[1] Adams, M.J., Bruce, B.C. Background knowledge and reading comprehension for J. Langer and M.T. Smith-Burke (eds); Reader Meets Author-Bridging the Gap: A psycholinghistic and sociolinguistic perspective, Dover Del, Internat'l Reading Asoc., 1982. Also as Reading Education Report n. 13, Urbana: Univ. of Illinois, Center for the Study of Reading, January, 1980.

[2] Anderson, R.C., Spiro, R., Anderson, M.C., SCHEMATA AS A SCAFFOLDING FOR THE REPRESENTATION OF INFORMATION IN CONNECTED DISCOURSE, American Educational Research Journal, 1978, 15, p. 433-440.

[3] Anderson, R.C., Osborn J., Tierney R.J., LEARNING TO READ IN AMERICAN SCHOOLS: BASAL READERS AND CONTENT TEXTS, University of Illinois, The Center for the Study of Reading, 1981.

[4] Anderson, T.H., Armbruster, B.B., Kantor, R.N., HOW CLEARLY WRITTEN ARE CHILDREN'S TEXTBOOKS? OF BLADDERWORTS AND ALFA, Center for the Study of Reading, University of Illinois at Urbana-Champaign, 1980.

[5] Fillmore, C.J., IDEAL READERS AND REAL READERS, in "Gerorgetown Univ. Round table on Languages and Linguistics", Tannen B, Georgetown Univ. Press, 1981, pp. 248-270.

[6] Goetz, E.T., Armbruster, B.B., PSYCHOLOGICAL CORRELATES OF TEXTSTRUCTURE, in "Theorical Issues in Reading Comprehension", 1980, Lawrence, Erlbaum, Ass, Hillsdale, N.J. p. 201-216.

[7] Bruce, B.C., PLANS AND SOCIAL ACTIONS; Spiro, R.J., Bruce, B.C., Brewer, W.F. THEORETICAL ISSUES IN READING COMPREHENSION, 1980, Lawrence Erlbaum Ass., Hillsdale, N.J. p. 367-382.

[8] Halliday, M. AK., Hasan, R., COHESION IN ENGLISH; Longman, London 1976.

[9] Mandler, J.M. Johnson, N.S., REMEMBRANCE OF THINGS PASSED: STORY STRUCTURE AND RECALL, Cognitive Psychology, 9, 111-115, 1977.

[10] Rumelhart, D.E.; NOTES ON A SCHEMA FOR STORIES, in D.G. Brown and A. Collins (eds) "Representation and Understanding Studies in Cognitive Science", Academic Press. New York 1975.

[11] Stein, N.L., Glenn, C.G., AN ANALYSIS OF STORY COMPREHENSION IN ELEMENTARY SCHOOL CHILDREN, in R. Freedle ed. "Discourse Processing" Multi-disciplinary perspective", Hillsdale, N.J. Ablex, 1978.

[12] Tierney, R.J., Rosenthal, J., Kantor, R.N., SOME CLASSROOM APPLICATIONS OF TEXT ANALYSIS TOWARD IMPROVING TEXT SELECTIOON AND USE, Center for the Study of Reading, University of Illinois at Urbana-Champaign, 1980.

[13] Tonfoni, G., PER UN'IPOTESI DIDATTICA TESTUALE, Bo, Modena, 1982.

[14] Tonfoni, G. DALLA LINGUISTICA DEL TESTO ALLA TEORIA TESTUALE, Milano, Unicopli, 1982.

[15] Wilensky, R. UNDERSTANDING GOAL-BASED STORIES, Ph. D. thesis, Yale University, 1978.

[16] Rumelhart, D.E.; NOTES ON A SCHEMA FOR STORIES, in D.G. Brown and A. Collins (eds) "REPRESENTATION AND UNDERSTANDING STUDIES IN COGNITIVE SCIENCE", Academic Press. New York 1975.

Note:
This paper is the result of a research activity done by Graziella Tonfoni and Bertran Bruce at Bolt Beranek and Newman, Cambridge Mass. U.S.A.

This paper circulates as BBN Technical Reports - Bolt Beranck and Newman. Cambridge Mass.

UN ESEMPIO DI AUTOISTRUZIONE GUIDATA CON IL CALCOLATORE

Federico Monzani
Atropo - Milano

ABSTRACT

We are going to present a "CAI Course" which will allow the training of an end user with no acknowledge of EDP, to the use of a minicomputer.
The different problems in constructing a course of auto-instruction aided by computer and the reasons of some choices taken are explained throughout the realisation itself.

RIASSUNTO

Presentiamo un "Corso CAI" che permette l'addestramento di un utente finale con nessuna conoscenza EDP, all'uso di un minicomputer.
I differenti problemi nella realizzazione di un corso in autoistruzione guidato dal calcolatore e le ragioni di alcune scelte prese sono spiegate direttamente nella nota.

1. INTRODUZIONE

Da più di vent'anni ormai vengono presentate sulle riviste di settore realizzazioni teoriche e pratiche, nell'ambito della istruzione assistita con il calcolatore (AED).

Si passa via via che arrivano nuove tecnologie informatiche o nuove linee di tendenze didattiche, da corsi realizzati su grossi calcolatori dedicati all'istruzione, a sistemi di istruzione che funzionano in multielaborazione con altre procedure su grossi calcolatori, all'avvento dei microelaboratori con software dedicato di limitato costo ma anche di limitate capacità di memoria e sistemi globali di education su microcomputer evoluti con più terminali e con le necessarie potenti capacità di memoria, proposte a un prezzo competitivo.

Sul fatto che l'elaboratore è ormai, con buon diritto, uno degli strumenti che vengono in aiuto all'insegnante, sono stati scritti innumerevoli articoli su riviste di informatica e di didattica.

È però indubbio il fatto che uno strumento quale l'elaboratore, che permette corsi in autoistruzione con altissime possibilità di interazione con l'allievo, non è stato ancora sfruttato nelle sue piene capacità.

Le passate esperienze, tolta qualche eccezione mai completamente decollata per gli alti costi, hanno insistito troppo su tentativi di sostituzione "ex abrupto" dell'insegnante e della carta stampata, con il video terminale, con il risultato di costringere l'insegnamento entro schemi rigidi, obbligando chi seguiva il corso a lunghe e noiose letture di testi al terminale. È soltanto da poco che compaiono corsi agili e poco noiosi con frequenti riferimenti ad altri media, grazie alle potenzialità grafiche dei nuovi elaboratori che vengono proposte a costi contenuti.

Questa nota non vuole però essere una ennesima rievocazione della storia, dei fallimenti o dei successi ottenuti nelle applicazioni dell'elaboratore nella didattica, ma vuole solo portare il risultato di una esperienza di realizzazione di corsi.

2. IL COURSEWARE

Come risulta chiaro a chiunque esamini un corso AED rispetto ad un normale corso di istruzione programmata, la differenza sta nel prodotto "applicazione didattica su elaboratore" in altre parole "courswaре".

Al momento attuale la realizzazione di un buon courseware non può prescindere dall'hardware sul quale lo si andrà a realizzare e dal software di base a disposizione.

Un programma potrà essere infatti realizzato utilizzando un semplice interprete Basic oppure uno dei "linguaggi autore" a disposizione sull'hardware prescelto (PILOT, MASTER, ecc.) o addirittura dei veri e propri sistemi didattici (CAN-8, ecc.).
I linguaggi autore sono infatti dei linguaggi di programmazione costruiti per le esigenze di un realizzatore di corseware.

Rimane però scontato che un hardware di buone prestazioni a un prezzo di mercato economicamente conveniente, supportato da un buon software, nulla può se la realizzazione del corso, da parte dell'autore o del gruppo di lavoro che lo realizza, non riesce a convogliare l'ettenzione dei discenti per un massimo trasferimento di informazioni.

2.1 Le due tendenze nella produzione del courseware

Esattamente come per una parte del software si stanno delineando, nella produzione del courseware, due tendenze. Una di tipo ingegneristico, dove il prodotto corso è pensato e realizzato da un gruppo di lavoro composto da persone di diverse conoscenze che fondono le loro specializzazioni; un'altra invece che tende a dare a una persona sola la possibilità di realizzare interamente un corso, supportata da interventi esterni, strumenti di lavoro, ecc...

Proprio l'aumento degli strumenti tecnici a diposizione farà scegliere per la realizzazione di corsi a larga diffusione la seconda possibilità, da alcuni autori comparata con il "prototyping" nel campo della produzione di software.

È certo però che una realizzazione di questo tipo avvicina l'autore di un corso all'autore in campo editoriale dove esiste il binomio autore-editore, mentre la prima più si avvicina al normale modello delle software-house.

2.2 Competenze necessarie in un gruppo di realizzaziene di courseware

Approfondiremo in questa sede il primo tipo di realizzazione, quello "ingegneristico", piuttosto che il secondo, quello "creativo": non perchè non se ne vede la validità, ma perchè si ritengono al momento attuale quantitativamente più rilevanti le realizzazioni del primo tipo.

Al momento attuale le competenze necessarie per le realizzazioni di un corso sono difficilmente individuabili in una sola persona, vuoi per la mancanza di supporto tecnico (mancanza che sta però terminando), vuoi per la ancora relativa mancanza sul mercato di figure professionali quali gli autori di courseware con esperienza di programmazione, se non a livello universitario.

Generalmente in un gruppo di produzione di courseware sono necessarie le competenze di diverso tipo che mostriamo di seguito, non necessariamente individuabili ciascuna con un componente del gruppo.

- **Esperienza didattica**

  Coincide di solito con l'esperienza dell'Autore che decide la struttura del corso nelle sue fasi, determinando l'ampiezza e la quantità dei paragrafi di auto-istruzione.

  L'autore è anche, in mancanza di un coordinatore, chi fornisce ad altre persone del gruppo le specifiche del loro lavoro e garantisce il loro coordinamento. Una figura di questo tipo deve sicuramente avere anche delle compentenze di natura metodologica per un proficuo passaggio delle informazioni dal corso agli allievi.

- **Esperienza della materia**

  Sono esperienze che solo casualmente appartengono all'autore: possono essere portate da una o più persone totalmente dedicate all'interno del gruppo di lavoro o periodicamente richieste dell'autore a uno o più esperti esterni al gruppo.

  L'esperto della materia decide con l'autore le soluzioni didattiche e la verifica per una loro effettiva ed efficace trasmissione agli allievi.

- **Esperienza di programmazione**

  Anche in presenza di sofisticati linguaggi autore, è necessaria, al momento attuale, la presenza di una persona con conoscenza della costruzione di software, che traduca gli schemi e le lezioni proposte dall'autore in un linguaggio eseguibile dall'elaboratore.

- **Esperienze nel campo della comunicazione visiva**

  Anche se l'attenzione alla comunicazione visiva è stata un poco trascurata, si tratta di un tipo di esperienza assolutamente necessaria in un gruppo di produzione di courseware.

  In seguito alle nuove possibilità offerte agli autori dalla grafica diventerà sicuramente necessario integrare i corsi AED con la migliore competenza di presentazione grafica.

Come è stato già accennato in precedenza, comunque, a volte una o più di queste competenze apparterranno ad una persona sola, di solito l'autore.
Sarà solo il volume dei progetti a tenere separate, in persone diverse, le predette competenze.

3. IL "FAI"

Nei paragrafi precedenti è stata introdotta la problematica della didattica assistita con l'elaboratore e la produzione del courseware.

Presenteremo adesso le linee guida della realizzazione di un corso AED mostrando uno di questi corsi: il FAI (Formazione Autodidattica Interattiva).

Il corso ha una durata complessiva di circa due giorni. Diciamo circa perchè nell'istruzione programmata, così come nell'AED, la durata temporale del corso, dipende in buona parte dall'allievo.

Le finalità di questo corso sono di educare l'utente dei nuovi microSystem 6 della Honeywell ad un uso autonomo dell'elaboratore.

Al termine del corso infatti l'utente, di solito di prima meccanizzazione, e quindi con i timori tipici di un neofita verso i computer, deve essere in grado di utilizzare correttamente l'elaboratore e di conoscerlo nelle sue parti fondamentali.

3.1. Scopo del corso.

Lo scopo fondamentale di un corso AED è sostanzialmente quello di rendere il rapporto prezzo/prestazione il più favorevole possibile alla formazione di un numero elevato di utenti.

Le esperienze passate insegnano che i campi di applicazioni favorevoli alle tecniche AED sono sostanzialmente due:

a. Applicazioni di tipo monografico, dove, all'interno del corso viene insegnato un determinato argomento più o meno complesso (elementi di statistica, tecniche di vendita, ecc...).

b. Applicazioni di tipo procedurale dove viene insegnato all'utente l'utilizzo di una procedura ben precisa (controllo della produzione, terminalista di sportello bancario, ecc.).

Un corso didattico sull'utilizzo del microcomputer rientra quindi chiaramente nell'ipotesi b.

L'indice del corso comprende tutte le singole parti del microcomputer utilizzate dall'utente.

In più, visto che l'utente tipico è di prima meccanizzazione, viene introdotto nel corso anche un capitolo sui concetti fondamentali dell'EDP (archivio, programma, ecc.).

Attraverso questo capitolo vengono fornite alcune informazioni che, pur nella loro semplicità, permettono all'utente di chiarire i collegamenti tra computer e la propria normale attività aziendale.

Una scelta prioritaria è stata effettuata per stabilire la quantità di informazioni trasmesse attraverso il canale tradizionale del manuale e attraverso il nuovo mezzo interattivo.

L'esperienza dice infatti che l'impiego massiccio dell'elaboratore nel corso non garantisce una buona assimilazione degli argomenti.

Esistono casi di corsi AED per i quali la noia da parte dell'allievo cresce subito dai primi capitoli, garantendo un tasso percentuale di completamento del corso veramente molto basso.

Esistono d'altra parte corsi dove la media di impiego del calcolatore è del 15-20% sul tempo totale di apprendimento, essendo il resto utilizzato nella lettura di testi o esercizi diversi o, in alcuni casi, incontri con docenti o lezioni tradizionali.

Siamo totalmente d'accordo con chi sostiene che l'autore deve tener ben presente che il calcolatore non va utilizzato per sostituire in maniera generalizzata libri e manuali.

Le informazioni possono venir lette molto più tranquillamente e agevolmente su un manuale che con ore passate al video terminale.

Il calcolatore è infatti essenzialmente un mezzo interattivo e come tale deve essere utilizzato.

L'allievo deve verificare le proprie conoscenze in mancanza di un docente, e un buon programma lo può fare.

Il calcolatore inoltre "tiene memoria" del percorso dell'allievo attraverso le varie possibilità garantite dal programma. È possibile conoscere quindi quante domande l'allievo ha sbagliato e se ha ripetuto alcune fasi di ripasso o altro.

Di questa fase "molto importante" che chiameremo "Gestione amministrativa" parleremo più in dettaglio in seguito.

3.2. Corso su carta e corso su video.

Il corso FAI è organizzato attraverso un manualetto integrativo che accompagna l'allievo durante lo studio della parti su video e delle parti su carta (fig. 1). Questo manualetto consiglia all'allievo di eseguire parte del corso al terminale, o di leggere i manuali della macchina, o di eseguire altri esercizi molto semplici ma estremamente pratici intercalati alle altre fasi, anche con lo scopo di spezzare la continuità dello studio su uno stesso mezzo (video, manuale, ecc.).

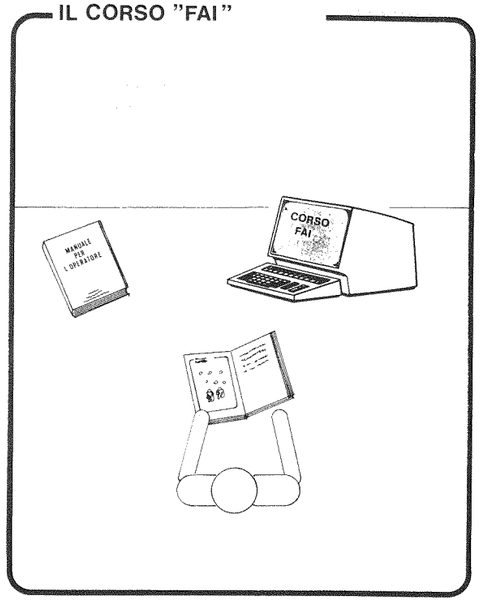


Fig. 1

"Formattare" una diskette o inserire la carta nella stampante non rientra necessariamente nella spiegazione del computer ma sono sicuramente delle fasi pratiche attive che spiegano meglio di cento videate il concetto di "floppy-disk", "disk-driver" e "istruzione del sistema operativo".

Le parti del corso proposte tramite video hanno le seguenti caratteristiche:

- Spiegazioni brevi e chiare (frame didattici) che per nessuna ragione devono richiamare alla mente le pagine dense di parole di un libro di testo.

- Esercizi il più possibile piacevoli che invoglino lo studente a "giocare" con il calcolatore non per misurarsi con esso, ma per apprendere, tramite attività apparentemente non impegnative, conoscenze che, per analogia, verranno trasferite in ambito lavorativo (fig. 2).

- Test volti a misurare il grado di apprendimento raggiunto (frame valutativi).
  La semplicità nella presentazione del test è una caratteristica essenziale perchè il fruitore del corso capisca con altrettanta chiarezza ciò che gli viene richiesto e risponda con altrettanta lucidità.

Le parti del corso proposte tramite supporti cartacei, utilizzando le tecniche di documentazione PS/PN, hanno invece le seguenti caratteristiche:

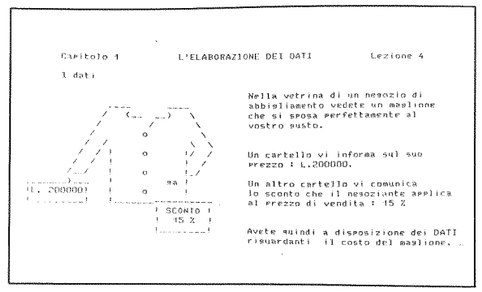


Fig. 2

- formato pocket, estremamente maneggevole;
- aspetto "invitante" che, non compromettendo la serietà e il rigore del contenuto, invogli lo studente alla consultazione (fig. 3).

L'uso del testo viene visto soprattutto, come è già stato detto, per le spiegazioni più tecniche che, risulterebbero troppo pesanti se proposte tramite video.

Inoltre, una parte iconografica ricca e articolata supporta gli argomenti trattati in un modo che non sarebbe possibile con l'elaboratore a meno di non usare estesamente la grafica a video (il che può comportare in alcuni casi un aumento molto forte dei costi di realizzazione).

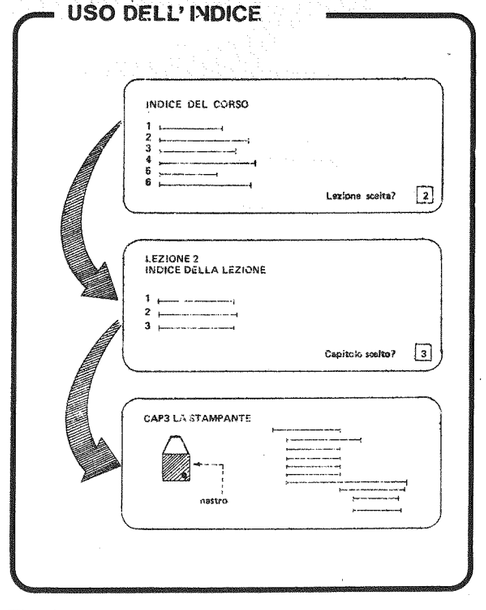


Fig. 3

3.3. La valutazione

In ogni caso, sia dopo la lettura dei manuali che dopo una serie di frame didattici, l'apprendimento viene controllato attraverso dei frame valutativi.

Didatticamente è sbagliato trattare semplicisticamente una risposta dell'allievo attraverso l'alternativa:

- Esatto, vai avanti
- Sbagliato, rifare

I tipi di errore possono essere diversi, ed in ogni caso anche un rinforzo positivo, proposto insieme alla parola "Esatto", può guidare l'allievo nelle risposte successive.

Nel caso di risposta esatta, bisogna convalidare infatti più il processo logico che ha portato alla risposta che la risposta stessa.

Per quanto riguarda le risposte errate, presenteremo ora i vari tipi di errore che bisogna tenere presente e valutare diversamente l'uno dall'altro.

a. Non si è capita la spiegazione.

   Il calcolatore deve allora utilizzare l'errore dell'allievo per riproporre il concetto o l'informazione; poi gli pone un'altra domanda.

   È stato utilizzato questo schema in alcuni casi di risposte errate facendo rileggere obbligatoriamente all'allievo più frame didattici per poi riproporgli la stessa domanda.

b. Si è capita la spiegazione ma si è data una risposta errata.

   Una delle cause di questo errore è la non comprensione della domanda.

   Nel caso di ulteriore errore dopo aver proposto più frame didattici, già visti dall'allievo, viene riproposta una domanda sugli stessi concetti ma formulata in maniera diversa.

c. Errore di battitura.

   Viene gestito in alcuni casi direttamente dalla "facilities " del sistema

   Nel caso di utilizzo di un linguaggio autore, questo tipo di errore viene addirittura gestito automaticamente all'interno del linguaggio di programmazione.

   Viene esaminata ad esempio, la domanda: "Qual'è la capitale d'Italia?", la coincidenza della risposta "ROMA" con "RONA", "ROOMA", ecc. evidenziando così i possibili errori di battitura.

   Nel caso del corso descritto in questo articolo realizzato in BASIC, sono state realizzate delle routine che eseguono, ad ogni risposta, alcuni controlli di questo tipo.

d. Errore del quarto tipo.

   Avviene quando l'allievo decide deliberamente di ignorare la domanda.

   Ad esempio se alla domanda:

   Qual'è la capitale d'Italia?
   A - MILANO
   B - ROMA
   C - PALERMO

   l'allievo risponde per tre o quattro volte consecutive F, G o L (chiaramente fuori dal range delle risposte previste), è importante richiedere all'allievo una maggiore attenzione alla domanda o, in alcuni casi, anche decidere la sospensione obbligatoria del corso.

3.4. Gestione amministrativa

Tutti gli errori o le risposte vengono memorizzate all'interno di quella è stata chiamata "Gestione amministrativa".

Nei sistemi dedicati alla didattica permette di gestire globalmente una classe o un insieme di classi che frequentano uno o più corsi, anche nei termini di un controllo del tempo utilizzato.

Più semplicemente, invece, ed è quello che è stato utilizzato nel corso FAI, ad ogni risposta errata viene incrementato un opportuno contatore.

Si hanno quindi a disposizione gli elementi per seguire, durante il corso o a posteriori, l'apprendimento dell'allievo.

Questo permette eventuali azioni del tipo:

- Memorizzare delle risposte errate, in funzione del tipo di errore, per eventuali valutazioni successive.
- Memorizzare delle risposte corrette per eventuali valutazioni successive (saltare paragrafi riassuntivi, domande troppo semplici, ecc.).
- Espulsione dal corso nel caso di ripetizioni di errori del tipo d.
- Eventuale stampa degli indici degli argomenti di cui si consiglia il ripasso.

- Valutazione non solo dell'allievo, ma anche del corso (ad esempio, se ad una domanda sbaglia un gran numero di allievi, c'è qualcosa che non va, o nella formulazione della domanda o nella spiegazione precedente).
- Statistiche generali.

3.5. Il sistema utilizzato

La parte interattiva del corso è realizzata sugli elaboratori microSystem 6/10 e 6/20 della Honeywell.

Come abbiamo già detto in precedenza, il coursewaare, come tutti i prodotti realizzati con l'elaboratore, ha delle esigenze precise per quanto riguarda la sua programmazione.

Fare delle domande, gestire delle risposte, controllare una classe comporta, nella realizzazione di un corso, la disponibilità di un linguaggio di programmazione con caratteristiche ben diverse da un linguaggio concepito per realizzare un programma che stampa una fattura o che esegue in memoria un calcolo matriciale.

Da un po' di anni a questa parte sono emerse, da parte di chi scrive corsi, necessità specifiche che solo un linguaggio specifico può risolvere.

Sono stati realizzati per queste esigenze vari linguaggi autore che in funzione della macchina per la quale sono stati costruiti facilitano all'autore, o al programmatore, la scrittura di un corso.

Esistono poi dei sistemi educativi completi che oltre a un linguaggio autore propongono anche diverse altre funzioni (gestione automatica della classe, biblioteca integrata di corsi, librerie di lavoro comuni, ecc.).

Gli elaboratori oggetto del corso sono, come abbiamo già detto, il microSystem 6/10 (nelle due versioni "table top" con due floppy-disks da 650 Kb e "floor model" con un floppy da 650 Kb e un disco fisso da 20 Mb e il microSystem 6/20 (la cui configurazione minima comprende, per quanto riguarda le memorie di massa, 1 floppy da 650 Kb e un disco Lark da 20 Mb fissi più 20 Mb mobili). Ogni configurazione è accompagnata da video terminali e stampanti di diverso tipo.

Non si avevano quindi particolari limitazioni di memoria di massa.

Al momento della realizzazione del corso il software a disposizione rientrava all'interno del normale corredo di software di base.

Non era ancora stato annunciato infatti, in Italia, il CAN-8, un sistema globale di AED realizzato in Canada su elaboratori Honeywell della linea 6.

La scelta obbligata fu quindi di utilizzare il sistema operativo standard GCOS 6 Mod. 400 nella release 3.0.

Il linguaggio di programmazione prescelto, per le notevoli caratteristiche della versione implementata sul GCOS 6 fu il BASIC. La scelta di un linguaggio generalizzato quale il BASIC permette, oltre alla facile scrittura del courseware, di autoinstallare il prodotto FAI da parte dell'utente finale su qualsiasi configurazione della macchina prescelta.

3.6. Composizione del gruppo di lavoro

Tra la scelta di una realizzazione di tipo individuale e una di gruppo fu scelta la seconda impostazione.

La mancanza di un linguaggio autore implica infatti la necessità di avere all'interno del gruppo competenze di programmazione; la quantità di lavoro da realizzare in tempi limitati portò a costituire un gruppo di lavoro composto da un autore, un esperto didattico e un programmatore.

Il continuo lavoro interattivo all'interno del gruppo portò poi a far sì che le esperienze dei singoli componenti passassero agli altri, e che alcune scelte fossero prese in riunioni collettive.

Di fatto, poi, anche l'esperto didattico rivestì il ruolo di autore, realizzando anche praticamente la maggior parte dei "frame didattici" tutti di elevata qualità.

Come tutti i lavori di gruppo, e specialmente in quelli dove il lavoro passa attraverso figure professionali definite, anche in un gruppo di lavoro AED occorrono degli standard di comunicazione precisi. Presenteremo nel prossimo paragrafo la modulistica realizzata per il corso FAI.

3.7. Standard di documentazione

Come già visto, il corso viene inizialmente definito a tavolino nei suoi obiettivi; viene suddiviso in lezioni e per ogni lezione vengono evidenziate le parti da realizzare su modulo cartaceo e quelle invece da realizzare sull'elaboratore (fig. 4).

Come per un qualsiasi corso di istruzione programmata, per ogni lezione e per ogni capitolo viene redatto un documento che indica i prerequisiti , gli obiettivi di ogni capitolo e un indice di massima fino ad arrivare al singolo frame o modulo elementare di visualizzazione (fig. 5).

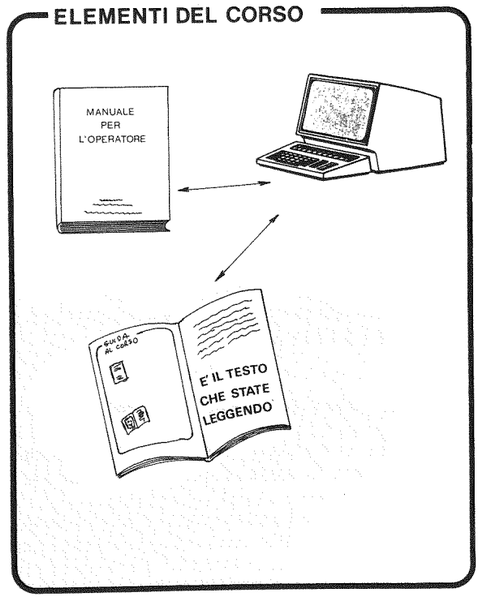


Fig. 4

I singoli frame vengono collegati tra loro attraverso un linguaggio grafico, più chiaro e modificabile di altri tipi di linguaggi. La nostra scelta è banalmente caduta nei flow-chart, il linguaggio grafico più comune e quindi più leggibile da possibili altri utilizzatori della documentazione (manutentori, traduttori del corso in altre lingue, ecc.).

A questo punto avviene una prima verifica sul corso così impostato.

Lezione per lezione, il corso viene "letto" da una persona diversa da quella che lo ha realizzato (di solito il responsabile della realizzazione di corsi all'interno della struttura di produzione o da altri autori o/e esperti didattici della materia) e vengono fornite all'autore indicazioni, consigli o suggerimenti sull'impostazione dei singoli capitoli, su una eventuale esplosione di alcune parti, sul ridimensionamento di altre o su un diverso mezzo realizzativo (carta, elaboratore, videotape, altro).

È questa una fase molto delicata perchè in essa viene definito, globalmente e nei dettagli, il corso, e un buon "controllo" (o "brain storming" a seconda delle impostazioni del gruppo) riesce a garantire un corso di buona qualità.

Si passa quindi, capitolo per capitolo, alla definizione dei singoli frame.

Per passare le informazioni dal realizzatore dei frame al programmatore sono stati prodotti degli standard di documentazione ad hoc.

Ci si rende conto che, nel caso di presenza di grafica in movimento o di colore o di un linguaggio autore potente, questa documentazione non è esaustiva: la si propone qui, in ogni caso, come spunto di lavoro.

a. Schemi di flusso

Lo schema di flusso di fig. 6 semplifica un semplice colloquio.

Dopo aver proposto all'allievo alcuni frame didattici, viene posta una domanda molto semplice, a scelte multiple, che rappresentiamo con la fig. 7.

L'allievo digita forzatamente una delle tre risposte dirette, poichè il caso di digitazione diversa da qualle previste viene trattato come errore di battitura e viene subito riproposta la maschera con una segnalazione di errore molto semplice.

Nel caso di una risposta prevista, il programma porta a frame specifici, che danno stimoli e indicazioni precise all'allievo (figg. 8,9,10).

Nell'esempio mostrato, la scelta è stata di non far ripetere la domanda per la ovvia semplicità della situazione.

Risposte errate a domande diverse possono però dare origine a semplici riproposizioni delle domande, o a ripassi obbligatori di frame già visti o addirittura, nel caso di perseveranza nell'errore, all'interruzione forzata del corso.

b. Il "Frame"

Nel paragrafo precedente abbiamo visto che il flow-chart guida il programmatore e l'autore al collegamento di frame diversi.

La modulistica qui proposta è stata realizzata per garantire all'autore di poter trasmettere tutte le informazioni necessarie al programmatore, organizzate al meglio.

Ogni frame è completamente riassunto in due moduli distinti.

Il primo (fig. 11) non è altro che un prospetto video rappresente le 24 righe per 80 colonne a disposizione dell'autore. La testata del modulo propone il nome della videata utilizzata all'interno dal programma e il numero progressivo del capitolo. All'interno di questo modulo l'autore disegna la videata che verrà proposta dal programma, con tutte le scelte grafiche che ritiene utile usare.

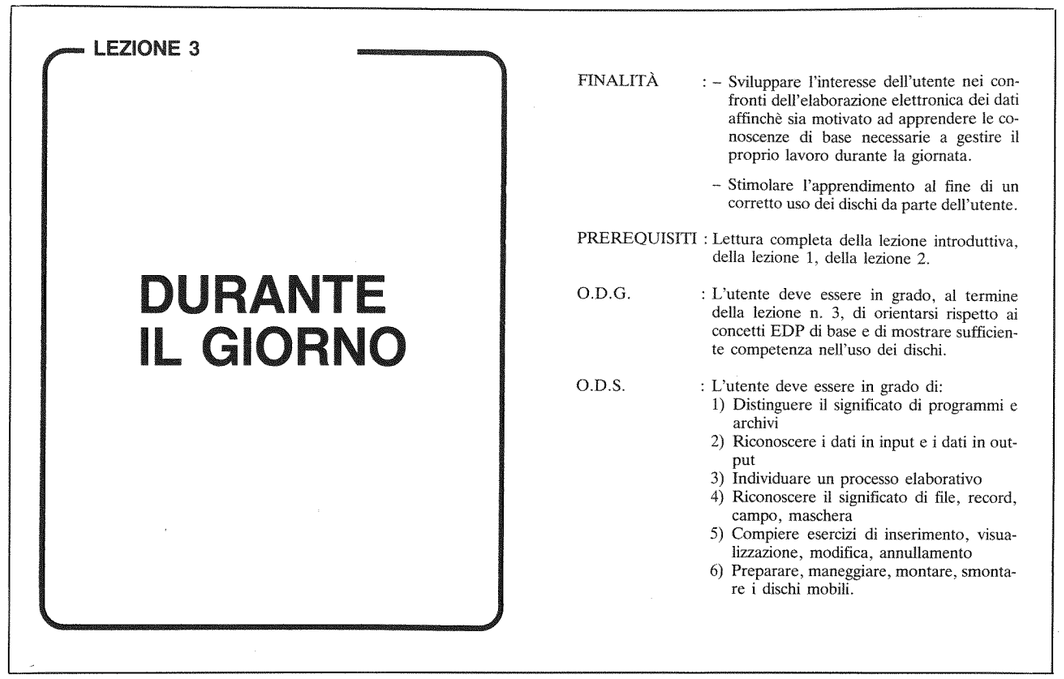


Fig. 5

Il secondo modulo (fig. 12), invece, può essere suddiviso in tre parti.

La prima parte riporta tutte le differenze grafiche dalla luminosità standard che sono utilizzate per disegnare la schermata (vale a dire attributi visivi come sottolineature, blinkins, reverse video, ecc.).

La seconda parte riporta invece tutte le variabili di programma gestite in questo frame valutativo.

Nel nostro semplice esempio è la variabile C1 che accoglie la risposta e la costante NRISP che riporta le possibili risposte gestite dal frame.

La terza parte riporta invece tutte le strade possibili che può seguire il programma.

Alcune di queste sono presenti in quasi tutti i frame e sono definite da tasti funzione (torno indietro, vado avanti, smetto il corso e torno all'indice). Altri invece sono variabili in funzione del frame. Ad esempio, se la risposta è B, vado alla maschera 0401150 a pag. 22, se è C alla maschera 0401160, ecc.

Per qualche risposta vengono incrementate delle variabili che permettono di gestire la funzione amministrativa, come ad esempio il numero di risposte errate ad una domanda molto semplice, che può dare origine ad una segnalazione diversa o all'impedimento ad andare avanti se non si è risposto esattamente ad una particolare domanda precedente.

3.8 Realizzazione sul calcolatore

Una volta che tutti i frame di un capitolo sono definiti, vengono passati al programmatore che li tradurrà nel linguaggio a disposizione.

Nel caso che prendiamo in esame, di scrittura in BASIC, data l'alta ripetitività di operazioni elementari, sono state realizzate delle macro che permettono al programmatore di seguire la codifica con un notevole risparmio del tempo di scrittura.

Ad esempio è stata realizzata una macro che permette la stampa su video di una spiegazione (frame didattico); una che permette la stampa su video di una spiegazione o di un messaggio e che richiede una risposta, restituendo al programma il controllo per la continuazione nel ramo di programma individuato; una che gestisce automaticamente tutte le informazioni di un archivio STUDENTI (Gestione amministrativa) ecc.

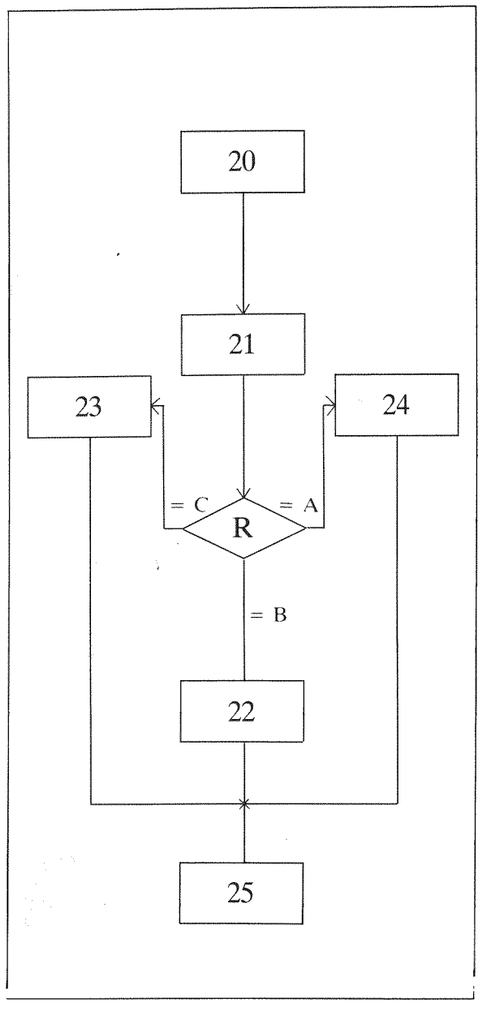


Fig. 6

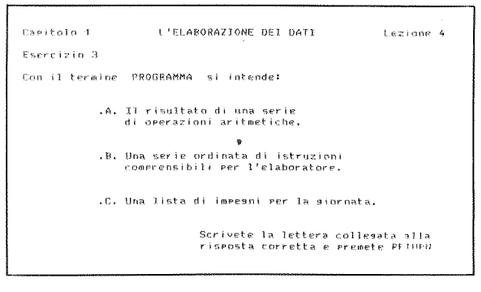


Fig. 7

Il programma passa poi attraverso la fase di controllo e testing che deve essere effettuata in prima battuta dal programmatore per valutare l'aderenza con i frame proposti su carta e, in seconda battuta, per valutare non solo l'effetto grafico, ma anche i "tempi" del corso, i suoi ritmi e la sua efficacia didattica.

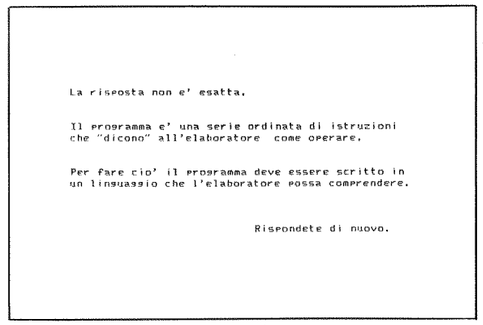


Fig. 8

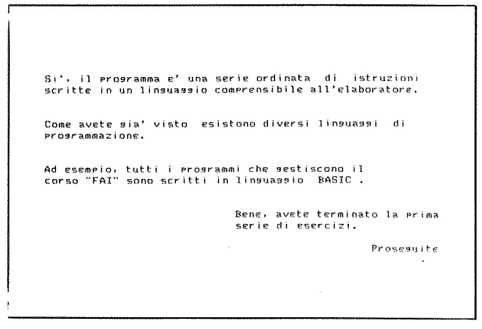


Fig. 9

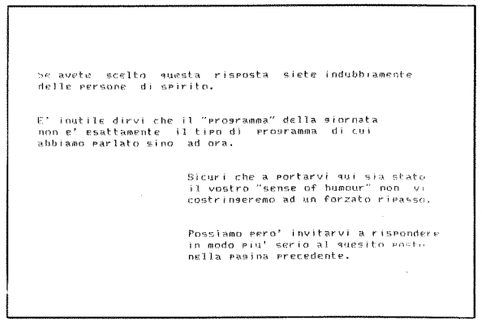


Fig. 10

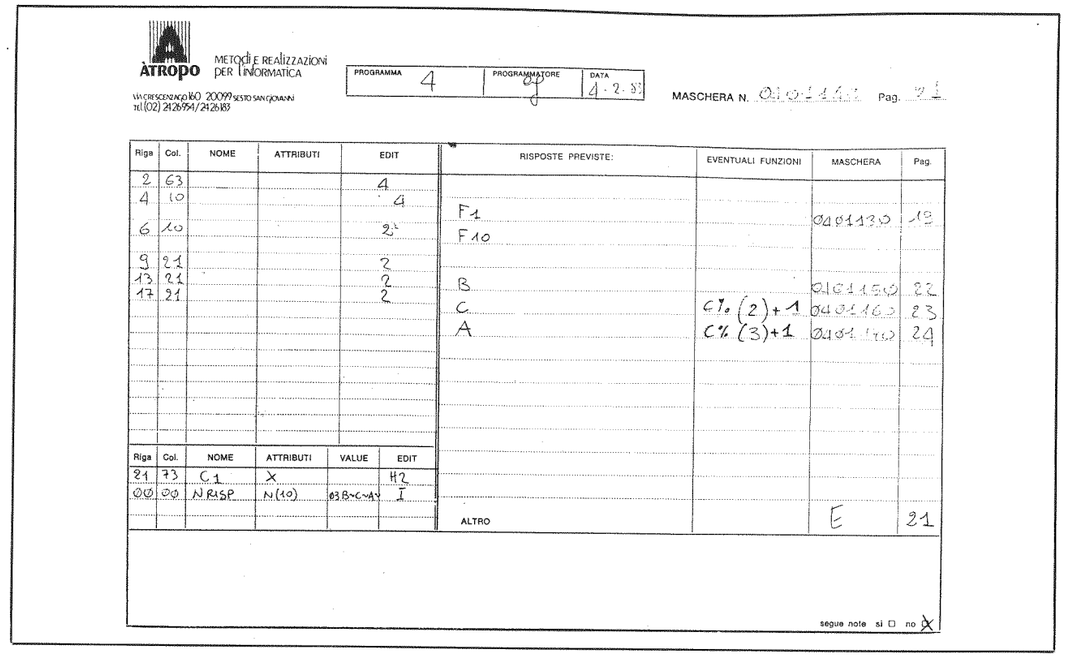


Fig. 11

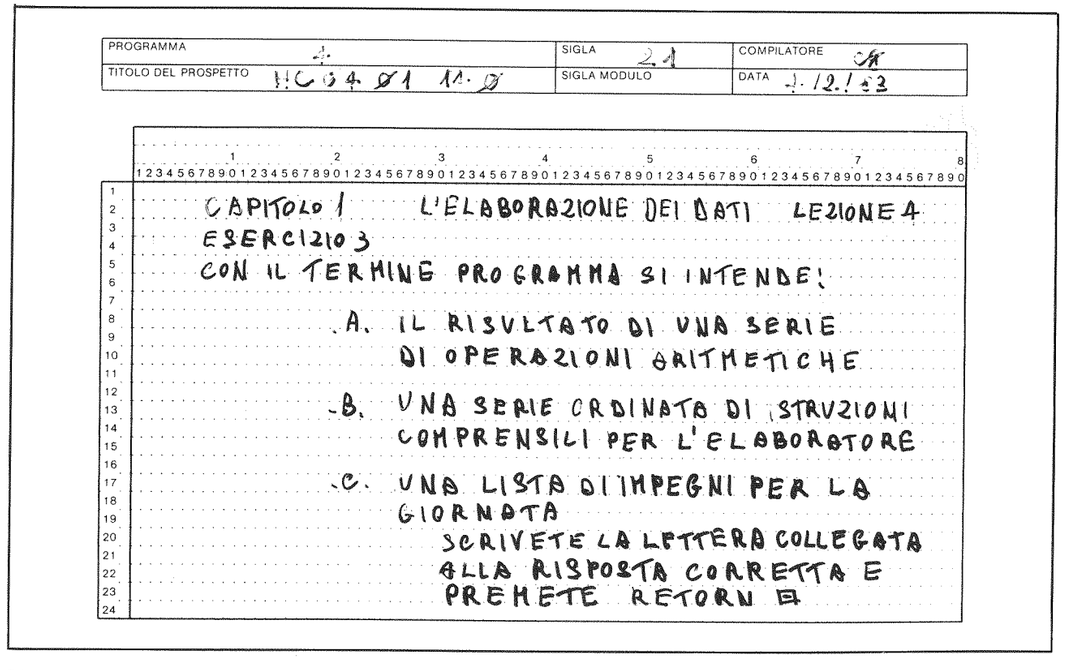


Fig. 12

IL LINGUAGGIO SGML (STANDARD GENERALIZED MARKUP LANGUAGE) PROPOSTO DA ISO.

Le van Huu

Istituto di Cibernetica - Università degli Studi di Milano

RIASSUNTO

Fra le principali attività riguardanti l'elaborazione di testi, quali definizione della struttura logica e fisica del testo, creazione e modifica, formattamento e stampa, archiviazione e ricerca e trasporto, la prima forse può definirsi più importante.
L'argomento del presente lavoro riguarda appunto tale attività, cioè la definizione della struttura del testo. Il lavoro presenta SGML (Standard Generalized Markup Language), una componente dello standard multiparti, riguardante le attività di trattamenti di testi, chiamato STPL) Standard Text Processing Languages) e proposto da ISO.
SGML standardizza le tecniche di markup del tipo "generic coding" e "generalized markup". Fornisce una sintassi coerente e non ambigua che permette all'utente di descrivere gli elementi logici che intende mettere in evidenza nel documento.

ABSTRACT

Processing a text comprehends principal activities as: defining the logical and physical text structure, editing, formatting, storing and retrieving, interchanging.
The topic of this paper is related to the first activity, i.e the text structure definition. It presents SGML (Standard Generalized Markup Language), a component of the multipart standard for Text Interchange and Processing called STPL (Standard Text Processing Languages), proposed by ISO. SGML standardized the application of the generic coding and generalized concepts. It provided a coherent and unambiguous syntax for describing whatever a user chooses to identify within a document.

INTRODUZIONE

Negli ultimi anni gli strumenti per il trattamento automatico di documenti, che vanno dalla generazione alla modifica, dal formattamento alla stampa dei testi, stanno assumendo una importanza tale da diventare parte integrante di quasi tutti i sistemi di calcolo.

D'altra parte è altrettanto vero che quasi tutti i sistemi di elaborazione di testi nascono in modo autonomo, in seguito a precise richieste, tendenti a coprire particolari aree di esigenze dell'utente finale. Di conseguenza l'ambiente di funzionamento, l'interfaccia utente, il formato del testo sorgente elaborabile non sono sempre generalizzabili e ciò provoca considerevoli difficoltà nel trasporto sia dei sistemi stessi che dei documenti da elaborare.

Nasce quindi l'esigenza di una normalizzazione delle attività che riguardano l'elaborazione di testi, e ciò fa parte di una proposta, iniziata nel 1979, del comitato EGCLPT (Expert Group Computer Languages Processing Text) appartenente all'ISO (International Standards Organization).

2. GENERALITÀ

Nell'ambito ISO un testo è costituito da una struttura logica che definisce le funzioni di elaborazioni e una struttura fisica che rappresenta il vero contenuto del testo stesso.
L'elaborazione di un testo consiste quindi in una serie di funzionalità volta al completamento del suo ciclo di vita, che si suddivide in più fasi, non necessariamente in sequenza, fra cui troviamo:

. la definizione del testo sorgente. Probabilmente può essere considerata fra le fasi più importanti. Consiste nella descrizione, da parte dell'autore, della struttura logica e/o delle elaborazioni da associare a varie porzioni di testo, secondo una sintassi particolare comprensibile dal "formatter".

. la generazione e la redazione, considerate comunemente come attività di "editing", del testo sorgente per renderlo elaborabile nelle fasi successive.

. la memorizzazione e la ricerca, che si inseriscono fra le attività di gestione, del testo creato.

. l'interpretazione del testo sorgente e l'attivazione delle azioni associate; la visualizzazione e la stampa del testo finale, attività che possiamo definire di "formatting".

Infine possiamo aggiungere l'attività di "programming", considerata come quella di implementazioni di sistemi capaci di svolgere una o più attività sopra elencate.

La proposta ISO quindi si orienta verso la definizione degli standards relativi ai linguaggi e alle funzionalità per quanto riguarda l'elaborazione di testi, precisamente di

1) un linguaggio formale (sarà chiamato SGML, Standard Generalized Mark-up Language) per la descrizione del testo che permetta al documento di essere processato da più applicazioni e scambiato fra utenti di diversi sistemi.

2) un linguaggio di programmazione (sarà chiamato TPPL, Text Processing Programming Language) che comprenda tutte le funzioni necessarie per l'implementazione di applicazioni orientate al trattamento di testi, atte a svolgere le attività descritte sopra.

3) un interprete di comandi capace di interfacciare l'utente con il sistema operativo.

Gli standards, raggruppati sotto il nome di STPL (Standard Text Processing Languages) sono stati definiti con la precisa intenzione di aumentare e facilitare l'intercomunicazione fra le aree riguardanti l'elaborazione di testi.

Il presente lavoro si propone di fornire una descrizione della prima componente dello STPL, ossia il linguaggio di markup SGML per la definizione formale del testo. Essendo il linguaggio ancora nella fase di completamento, ricordiamo che i concetti che verranno esposti si riferiscono alla versione 6 del linguaggio, definita dall'ISO in data 15 giugno 1983.

3. DEFINIZIONE DEL TESTO.

Un testo sorgente, per poter essere soggetto di trattamenti finora considerati, oltre al suo contenuto effettivo, è costituito anche da un insieme di informazioni aggiuntive che definiscono il tipo di elaborazione da associare al testo stesso. Tali informazioni costituiscono i markup del documento.

Nei primi sistemi, verso gli anni 60, i markup comprendevano

. i comandi di formattamento, quali salto di pagina, indentazione, spazio bianco...

. gli attributi di presentazione dei diversi elementi (sottolineatura, grassetto...),

e sono estraibili dal testo perchè definiti da:

. un carattere particolare nella prima colonna della riga di comando. (RUNOFF, sviluppato presso M.I.T. verso il 1966), oppure

. i caratteri delimitatori che specificano l'inizio e la fine della stringa comando (FORMAT, realizzato dall'IBM verso la fine degli anni 60).

In questi primi tipi di formattatori la descrizione del documento consiste solamente nello specificare rigidamente la disposizione e l'apparenza che le varie porzioni del testo sorgente devono presentarsi nella pagine formattate, perdendo così le informazioni riguardanti gli attributi del documento stesso. Infatti se, per esempio, l'utente decide di sottolineare sia il nome dei paragrafi che quello delle figure, la marcatura della funzione di sottolineatura non è in grado di specificare su quale componente logica del testo sta operando.

Verso la fine degli anni 60, nel tentativo di ovviare tali inconvenienti, è stato introdotto il concetto di "oggetto logico". Un documento allora è visto come un raggruppamento di tali oggetti, organizzati in determinata gerarchia, ciascuno qualifica il contenuto di una porzione di testo, come per esempio il paragrafo, l'intestazione, le note... La struttura logica del documento allora non si riferisce più al tipo di elaborazione, che può variare da un formattatore all'altro, ma identifica solamente la classe delle varie parti del testo.

Fra i sistemi più noti basati su questa tecnica troviamo PUB (Stanford Artificial Intelligence Laboratory), SCRIBE (Carnegie-Mellon University) e forse, in un modo meno appropriato, anche NROFF (Bell Laboratories).
Questi sistemi, salvo forse lo SCRIBE, pur tendente all'astrazione della struttura logica dei documenti sorgenti, riservano ancora alcune limitazioni comuni, caratterizzate dalla loro dipendenza dalla macchina e dal fatto che determinano una corrispondenza biunivoca fra il documento sorgente e quello finale.
L'introduzione di un altro tipo di markup, il "generalized markup", ha lo scopo di limitare questi incovenienti.

È basato su due fondamentali caratteristiche:

. permette all'utente di descrivere la struttura logica del documento e gli altri suoi attributi senza specificare l'elaborazione associata.

. gli elementi di markup devono essere definiti rigorosamente in modo da permettere a qualsiasi sistema una loro corretta interpretazione.

I sistemi che maggiormente si avvicinano a questa tecnica possono essere il TEX dell'Honeywell Information Systems Inc., lo SCRIPT/GML (General Markup Language) dell'IBM, il GenCode diGCCA (Graphic Communicatinos Computer Association), e ancora lo SCRIBE della Carnegiek-Mellon University.
Ancora una volta però, mentre si concentra lo sforzo nel tentativo di scindere i comandi procedurali dal testo sorgente, questi sistemi non hanno focalizzato l'attenzione sulla sua trasportabilità. Ciò significa che ogni sistema vincola l'autore ad una diversa sintassi di descrizione, necessitando quindi di diversi compilatori di comandi.

4. SGML.

SGML (Standard Generalized Mark-up Language), una componente del più generale STPL, è stato definito con l'intento di delineare una forma di descrizione più generale e più completa del documento.
È visto quindi come un linguaggio di markup,in grado di descrivere in modo coherente e non ambiguo la struttura logica e gli attributi dei documenti. Inoltre è corredato di regole che permettono all'utente di defininire i propri elementi di markup e che determinano le funzionalità necessarie per i sistemi che intendono interpretare ed elaborare i documenti preparati con il linguaggio SGML.
In SGML un documento è costituito da un insieme di costrutti logici chiamati "elementi", organizzato in un albero i cui noti terminali costituiscono il contenuto fisico di tali elementi. Il documento "libro" per esempio può contenere l'elemento "capitolo" che a sua volta contiene gli elementi "paragrafo" e "figura", ciascuno di essi contiene infine una sequenza di caratteri che costituiscono il "contenuto" dell'elemento stesso.
Ogni "elemento" della struttura logica descritta è identificato da un nome simbolico, chiamato generic identifier (indicheremo d'ora in poi con GI). Una porzione di testo, affinchè possa essere considerata appartenente ad un "elemento" bisogna che il suo contenuto corrisponda alle caratteristiche sull'elemento stesso (per esempio trovarsi in un punto preciso del documento, avere tutti i caratteri alfabetici e non numerici...)

Per l'utente allora descrivere un documento significa

a) localizzare i suoi elementi più significativi, identificarli con dei generic indentifiers,

b) definire le regole di ogni "element" formando la struttura logica del documento (per esempio dove e per quante volte l'elemento può occorrere). Si stabilisce così il "tipo" del documento. Due documenti sono dello stesso tipo se le loro strutture logiche soddisfano lo stesso insieme di regole stabilite dai GIs,

c) infine collocare gli elementi nella loro giusta locazione contrassegnandoli con i relativi GIs.

SGML mette a disposizione dell'utente, per effettuare dette operazioni, un insieme di elementi di markup. Tali elementi, separati dai contenuti effettivi del testo mediante i caratteri "‹" e "›" (chiamati tags), si suddividono in 3 categorie principali:

. Markup declaration

. Markup descriptive

. Processing instruction

Vediamoli da vicino nei paragrafi che seguono.

4.1 Markup declaration:

Come accennato la fase iniziale della descrizione di un documento consiste nella definizione della sua struttura logica, cioè il "tipo di documento". Per questo tipo di descrizione l'utente dispone di un insieme di elementi di markup, classificati come "markup declaration". Tali elementi si suddividono in tre tipi, per

a) dichiarazione di struttura:

Mediante i markup di questo tipo, riconoscibili dalla parola chiave STRUC, l'utente specifica l'ordine di presentazione di ogni elemento che deve entrare a far parte della struttura logica del documento.
Per esempio le seguenti dichiarazioni

‹!STRUC article (title, author, body)›

‹!STRUC body (abstract, paragraph)›

stabiliscono che un elemento logico del documento associato al GI di nome "article" è composto da altri elementi chiamati "title", "author", "body"; quest'ultimo è composto a sua volta, come si vede dalla seconda frase da "abstract", "paragraph", i quali a loro volta possono contenere altri elementi e così via... Di conseguenza lungo il testo dovremo trovare per prima la porzione rappresentante il titolo ("title"), quindi il nome d'autore ("author") e infine il paragrafo ("paragraph").

Una dichiarazione consiste in 2 parti:

. il nome del GI che identifica l'elemento logico

. il modello delle sue caratteristiche

Il modello di un elemento specifica normalmente i nomi (GIs) degli altri elementi in esso contenuti, inoltre indica anche l'ordine con cui tali elementi si presentano lungo il testo. A questo scopo è stato previsto un insieme di delimitatori, rappresentati dai vari caratteri speciali come "!", "&"... I delimitatori, interposti fra i GIs che fanno parte del modello di valori, si suddividono in 3 tipi:

. sequenza, rappresentato dalla virgola. Indica che tutti gli elementi connessi devono essere collocati nel documento secondo l'ordine con cui sono stati elencati. Per esempio la dichiarazione

‹!STRUC article (title, author, abstract)›

indica che l'elemento logico "title" del testo deve essere seguito senz'altro dall'elemento "author" e quindi dall'elemento "abstract".

. OR, rappresentato dal carattere "!". Indica la mutua esclusione fra due elementi, in altre parole la presenza nel testo di un elemento esclude quella dell'altro.
Così la dichiarazione

‹!STRUC article (title, author, abstract ! keywords)

specifica che dopo il nome dell'autore possiamo collocare nel testo o il riassunto ("abstract") del documento o la lista delle sue parole chiave ("keywords") ma non tutti i due elementi nello stesso tempo.

AND, rappresentato dal carattere "&". Gli elementi connessi con questo delimitatore devono essere comunque presenti, in qualsivoglia ordine.

Quindi se gli esempi di un determinato documento sono strutturati in modo che il loro numero d'ordine (chiamiamo "numexample") possa essere collocato indipendentemente prima o dopo il loro contenuto (bodyexample), allora una possibile specificazione della struttura degli esempi può essere la seguente:

‹!STRUC paragraph (text, bodyexample & numexample)›

Un esempio d'uso dei delimitatori descritti è riportato nelle figure seguenti, dove a partire da una stessa struttura logica del documento, determinata dalle due frasi di dichiarazione STRUC (figura 1a), seguono due documenti con strutture fisiche differenti. In effetti nel primo documento (figura 1b) è presente l'elemento "abstract" piuttosto che "keywords", il quale invece è presente secondo documento (figura 1c). Ciò è permesso dalla presenza del delimitatore OR nella dichiarazione.

Ancora l'elemento "example" del primo documento inizia con il contenuto ("bodyexample") per proseguire poi con la numerazione ("numexample"), mentre nel secondo documento accade esattamente il contrario. Questa possibilità è indicata appunto dal delimitatore AND relativamente ai due elementi "bodyexample" e "numexample".

. ripetizione. Un elemento può essere presente più di una volta nella posizione riservatagli dalla struttura logica del documento. Esistono 3 tipi di delimitatori che denotano il numero di ripetizione di ogni elemento:

"+" : ripetizione dell'elemento di una o più volte
. . . : ripetizione dell'elemento di zero o più volte
"?" : opzionalmente l'elemento può essere presente o meno

Un utilizzo di questi tipi di delimitatori è riportato in figura 2a dove si fa riferimento ancora alle dichiarazioni presenti nell'esempio precedente. Due possibili documenti fra quelli che soddisfano la struttura dichiarata sono rappresentati dalle figure 2b e 2c.
L'interpretazione delle dichiarazioni presenti nell'esempio risulta abbastanza semplice se prestiamo particolare attenzione ai delimitatori che seguono i nomi dei GIs presenti nei modelli di valori:

. almeno un elemento "author" deve essere presente, però possono esserci anche più nomi d'autori (ciò è indicato dal carattere "+" dopo il GI "author"), come avviene nel primo documento,

. gli elementi "abstract" e "keyword" sono opzionali (carattere "?"), infatti sono assenti nel primo documento,

. gli elementi costituenti il "paragraph" possono occorrere più volte. Per ogni ripetizione gli elementi "head" e "text" sono obbligatori, mentre "example" può essere omesso, come succede nella prima parte del documento in figura 2b, dove vediamo che immediatamente dopo gli elementi "head" e "text" seguono nuovamente altri ele-

‹!STRUC article (title, author, abstract | keywords, paragraph)›

‹!STRUC paragraph (text, example)›

‹!STRUC example (bodyexample ⟲ numexample)›

Fig. 1a

‹article›
‹title›
An Unix Text Editor.

‹author›
B. Kernighan

‹abstract›
Almost all text input...

‹paragraph›
‹text›
Ed is a "text editor"...
... We can see in this example:

‹example›
‹bodyexample›
ed file
w
q
‹numexample›
ex. 1.2

Fig. 1b

‹article›
‹title›
The C language.

‹author›
D. Ritchie

‹keywords›
language, Unix

‹paragraph›
‹text›
Programming in C is ...
The following program:

‹example›
‹numexample›
Program 2.3

‹bodyexample›
```
main ( )
{     print f ("begin");
}
```

Fig. 1c

menti "head" e "text". La prima parte del documento rappresentato in figura 2c invece corrisponde alla struttura più completa dell'elemento "paragraph", ossia "head" seguito da "text" e quindi da "example".

b) Dichiarazione di attributi:

La dichiarazione di struttura di un GI non è sufficiente per descrivere il contenuto dell'elemento associato. Infatti diversi elementi del testo possono seguire le stesse regole strutturali, quindi identificati dallo stesso GI, ma nello stesso tempo devono essere in grado di distinguersi l'uno dall'altro. È necessario allora arricchire ulteriormente le informazioni relative ai GIs introducendo nelle dichiarazioni anche gli "attributi", i quali rappresentano le caratteristiche da associare a singoli elementi dello stesso GI, in modo che

. diversi tipi di attributi, oppure
. valori diversi di uno stesso attributo

‹!STRUC article (title, author +, (abstract | keywords)?, paragraph*)›

‹!STRUC paragraph (head, text, example)›

‹!STRUC example (bodyexample ⟲ numexample)›

Fig. 2a

‹article›
‹title›
An Unix Text Editor.

‹author›
B. Kernighan

‹author›
William Joy

‹abstract›
Almost all text input...

‹paragraph›
‹head›
1. Introdution.

‹text›
Ed is a "text editor"...

‹head›
2. Commands.

‹text›
a) insert.
... We can see in this example:

‹example›
‹bodyexample›
ed file
w
q
‹numexample›
ex. 1.2

Fig. 2b

‹article›
‹title›
The C language

‹author›
D. Ritchie

‹keywords›
language, Unix

‹paragraph›
‹head›
1. Overview

‹text›
Programming in C is ...

‹example›
‹numexample›
Program 1.1

‹bodyexample›
main ( ) ...

‹head›
2. Functions.

‹text›
The I/O functions...

‹head›
3. Data

‹text›
We have seen...

Fig. 2c

possono identificare elementi del documento che sono fisicamente distinti, pur appartenenti alla stessa categoria delle regole strutturali.

Così due diversi elementi del documento identificati dallo stesso GI "figura" possono distinguersi mediante il contenuto di un loro attributo, che chiamiamo "tipo-figura". infatti se il primo elemento si riferisce ad un grafico allora il suo attributo "tipo-figura" può assumere per un esempio il valore "grafico" mentre se il secondo elemento è una figura del tipo flow-chart allora il suo attributo potrà avere il valore "flow". In questo modo due porzioni di testo che corrispondono alle stesse caratteristiche dell'elemento "figura" ma nello stesso tempo hanno contenuti rispettivamente del tipo "grafico" e "flow-char", saranno contrassegnati dalle seguenti frasi di markup descriptive (il cui significato sarà specificato fra breve)

‹figura tipo-figura = grafico›

[contenuto del grafico]

‹figura tipo-figura = flow›

[contenuto del flow-chart]

La dichiarazione di un attributo di un GI è effettuata mediante l'indicazione del nome di GI, seguito dal nome dell'attributo e quindi dai possibili valori (separati dal carattere "!") che esso può assumere. Per esempio la seguente frase

‹!ATT figura tipo-figura (grafico ! flow)›

rappresenta la dichiarazione dell'attributo "tipo-figura" dell'elemento "figura". I valori che l'attributo "tipo-figura" può assumere sono "grafico e flow".

c) Dichiarazione di entity:

Porzioni di testi, stringhe di caratteri di varie lunghezze, testi contenuti in files esterni... possono essere considerati all'interno di un documento come degli oggeti distinti, chiamati "entity", identificabili mediante dei nomi simbolici.
L'associazione è effettata mediante una frase di dichiarazione di entity, così per esempio la dichiarazione

‹!ENTITY sgml "standard generalized markup language"›

stabilisce che la stringa "standard generalized markup language" è sostituita dall'entità di nome "sgml". Allora lungo il testo ogni qualvolta che l'utente desidera riferirsi alla stringa "standard generalized markup language" può semplicemente specificare il nome dell'entità preceduto dal carattere "&" (questa notazione rientra in un tipo di markup chiamato "entity reference").
Possiamo allora affermare che la seguente porzione di testo

"In questo articolo l'attenzione è posta sullo &sgml., definito da ISO..."

corrisponde esattamente a

"In questo articolo l'attenzione è posta sullo standard generalized markup language, definito da ISO..."

dove l'entità di nome "sgml" è stata sostituita con il suo valore "standard generalized markup language". Vediamo comunque più in avanti che l'uso di "entity" non si limita solamente alla semplice funzione di sostituzione di stringe ma si estende anche all'inclusione di dati variabili nel documento.

4.2 Markup descriptive

Una volta stabilita la struttura logica del documento e le caratteristiche dei suoi elementi mediante i markup declaration sopra riportati, l'utente può effettuare la descrizione collocando fisicamente i vari elementi del testo nelle varie posizioni stabilite dalle regole della struttura logica per formare in modo completo tutto il documento. Per far questo si serve delle frasi chiamate "markup descriptive".
Ogni porzione di testo, il cui contenuto corrisponde ad una particolare componente della struttura logica definita dalle frasi di dichiarazione finora descritte, è delimitata da due markup descriptive (chiamati rispettivamente "start-element" e "end element"), i quali contrassegano l'inizio e la fine della porzione interessata con il relativo GI, distinguendola dal resto del documento.
Riportiamo un esempio

```
1  ‹!STRUCT article (title, author, body)›
2  ‹article›
3  ‹title›
4  L'evoluzione di text processing.
5  ‹/title›
6  ‹author›
7  Le van Huu
8  ‹/author›
9  ‹abstract›
10 Il presente lavoro intende riportare...
```

dove la linea 1 è una frase di markup declaration, secondo la quale la struttura logica del documento deve essere tale che la parte rappresentante il titolo ("title") del testo deve essere seguita dal nome autore

("author") e quindi dalla parte di riassunto ("abstract").

Le linee rimanenti invece demarcano quelle porzioni di testo che si trovano nelle posizioni stabilite dalla dichiarazione riportata nella linea 1. Infatti le linee 3 e 5 sono dei markup descriptive che delimitano la il titolo (linea 4) del documento, mentre le linee 6 e 8 racchiudono l'elemento del testo identificato da "author", cioè la linea 8.

In diversi casi alcuni markup descriptive possono essere omessi, se ciò non comporti delle errate interpretazioni della struttura del documento descritto.

Infatti le linee 5 e 8 dell'esempio precedente possono non essere presenti perchè la struttura del documento in esame specifica chiaramente che l'inizio dell'elemento "author" determina senz'altro la frase dell'elemento "title"; così accade anche per gli elementi "author" e "abstract". Allora il documento dell'esempio precedente potrebbe essere ridotto in

```
1  ‹!STRUCT article (title, author, body)›
2  ‹article›
3  ‹title›
4  L'evoluzione di text processing.
5  ‹author›
6  Le van Huu
7  ‹abstract›
8  Il presente lavoro intende riportare...
```

Ricordiamo infine, per quanto riguarda le frasi di markup descriptive, che in ognuna di esse, oltre al nome del GI interessato, è possibile inserire anche la specificazione dei suoi attributi. Come è già stato spiegato, gli attributi, a secondo del loro valore, distinguono fra di loro gli elementi appartenenti alla stessa categoria (quindi identificati dallo stesso GI). Il riferimento all'attributo consiste nell'indicazione del suo nome, seguito dal segno "=" e quindi dal valore dell'attributo stesso. Così la frase

‹figura tipo-figura = grafico›

assegna all'attributo di nome "tipo-figura" del GI "figura" il valore "grafico".

4.3 Processing instruction

Indica le azioni da intraprendere nel punto preciso del testo in cui il markup è specificato. Normalmente corrisponde al nome di una procedura presente in libreria. Al contrario degli altri tipi di markup, i comandi dipendono dal sistema di calcolo e dall'ambiente in cui il documento è introdotto.
I markup del tipo "processing instruction" sono specificati con la seguente notazione:

‹? nome-procedura›

I "processing instruction" riescono, se utilizzati insieme ai markup del tipo "entity declaration", a far sì che questi ultimi vadano oltre alla semplice sostituzione di porzioni di testi con dei nomi simbolici.

Infatti specificando opportunamente i parametri della dichiarazione di entity è possibile includere nel testo il risultato dell'elaborazione di una procedura.

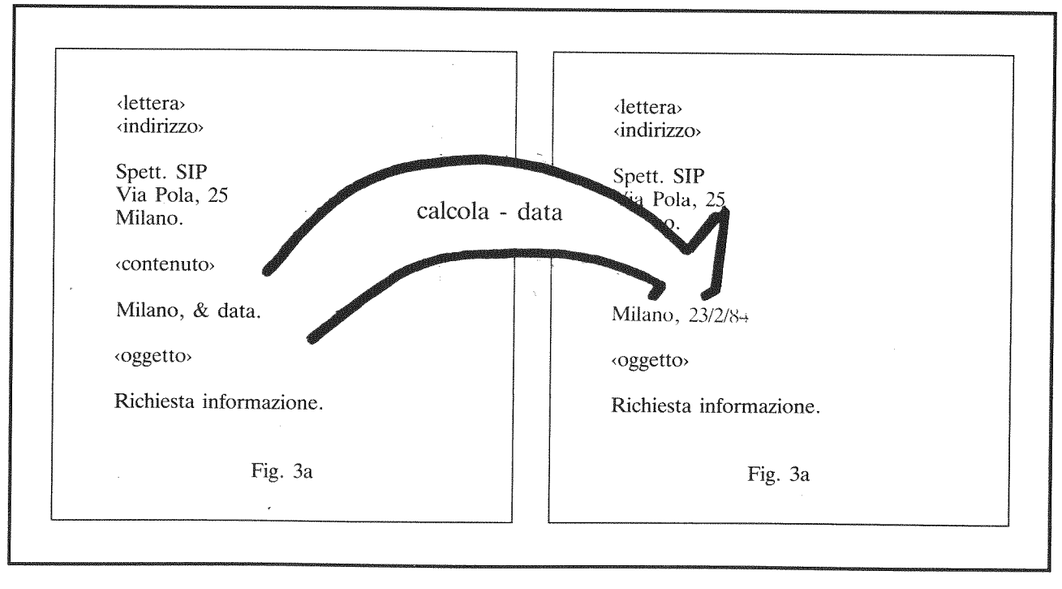


Fig. 3a Fig. 3a

```
‹!STRUC
        -- Element        Value model --

        article           (title, body)
        title             (&#CHAR.)
        body              (paragraph*)
        paragraph         (& # CHAR | list)*
        list              (item*)
        item              (paragraph | list)*
```

Fig 4a

```
‹article›
‹title›
    Il lavoro
‹/title›

‹body›
‹paragraph›
    Un lavoro può essere:

‹list›
‹item›
    piacevole
‹item›

‹item›
    noioso
‹/item›

‹/list›

‹/paragraph›

‹paragraph›
Il lavoro nobilita l'uomo.
‹/paragraph›

‹/body›

‹/article›
```

Fig. 4b

```
‹article›
‹title›
    Il lavoro


‹body›
‹paragraph›
    Un lavoro può essere:

‹list›
‹item›
    piacevole


‹item›
    noioso

‹/list›


‹paragraph›
Il lavoro nobilita l'uomo.


‹/article›
```

Fig. 4c

Il parametro da usare è PI, seguito da una frase di markup del tipo Processing Instruction, in cui si specifica il nome della procedura da cui si vuole ottenere il risultato. Per esempio dalle dichiarazioni

```
‹!ENTITY data PI›
‹?calcola-data›
```

è possibile ottenere valori sempre diversi, derivanti

dall'elaborazione della procedura "calcola-data" nelle varie parti del testo, semplicemente riferendosi all'entità "data" dichiarata. In particolare un testo descritto come in figura 3a potrà produrre un documento finale rappresentato dalla figura 3b se si suppone che la procedura "calcola-data" produce il risultato "23/2/84".

Concludiamo l'esposizione delle generalità di SGML riportando un esempio completo che evidenzia i concetti finora trattati. La struttura del documento è rappresentata dalla fig. 4a. L'entità è = , e il carattere CHAR" a cui si riferiscono i valori dei GIs "title" e "paragraph" corrisponde alla classe dei caratteri alfanumerici. La struttura dichiarata è abbastanza complessa, dove vediamo che l'elemento "paragraph" può contenere un altro "paragraph". Quest'ultimo è considerato però come elemento di "item" che a sua volta è un elemento di "list", il quale fa parte ancora di "paragraph".

Una completa descrizione di un documento che corrisponde alla struttura in esame è riportata in figura 4b, mentre lo stesso documento può essere descritto in un altro modo come in figura 4c, dove si notano delle omissioni di alcuni elementi di markup, precisamente gli end-elements di "title", "body", "item" e "paragraph". Al contrario il fatto che l'end-element di "list" sia presente o meno comporta due interpretazioni totalmente diverse del documento descritto, dato che la struttura dichiarata è tale che l'elemento "paragraph" può far parte sia di "body" che di "item". Infatti se l'end-element di "list" è presente un nuovo "paragraph" che si trova immediatamente dopo "item" sarà una componente di "body", altrimenti sarà considerato appartenente all'elemento "item".

Inoltre, come si vede in figura 4c, l'end-element di "article" determina la fine di "body" e l'ultimo "paragraph" senza che i loro end-element siano presenti.

5. CONCLUSIONE

Quanto abbiamo esposto finora è soltanto un insieme di elementi basilari del linguaggio SGML, sufficienti per valutare il suo significato. Facciamo presente però che una delle caratteristiche principali del linguaggio è la sua "granularità": gli elementi del linguaggio stesso si suddividono in vari livelli autonomi, ciascuno di essi può essere implementato autonomamente. Le funzionalità di ogni livello sono indipendenti da quelle degli altri livelli, nello stesso tempo però ogni livello può essere integrato agli altri senza alcun problema.

È in corso presso l'Istituto di Cibernetica di Milano l'implementazione di un parser del primo livello (quello basilare) di SGML. Il progetto, coordinato dal Prof. G. Degli Antoni, è nell'ambito di CP (Programming and Comunication) Project, in collaborazione con la Honeywell Information System Italia, ed è supportato da un gruppo di laureandi e studenti della facoltà di Scienza dell'Informazione.

6. BIBLIOGRAFIA

[1] INFORMATION PROCESSING SYSTEMS - SIX WORKING DRAFT, International Standard 1983 June ISOTC 97/SC5/EG GLPT/N 177-3

[2] A. H. Bigman - DRAFT OF GENCODE TUTORIAL. Graphic Communications Association - Arlington, 1982

[3] Brian K. Reid - SCRIBE: A DOCUMENT SPECIFICATION LANGUAGE AND ITS COMPILER - Department of Computer Science - Carnegie-Mellon University - 1980

[4] IBM SCRIPT/370 USER'S GUIDE - Manual SH20-1857-D. IBM Data Processing Division, White Plains, NY, 1976.

[5] J. F. Ossanna - NROFF / TROFF USER'S MANUAL. - Computing Science Technical Report N. 541 - Bell Laboratories 1976

[6] ACM COMPUTING SURVEYS - Vol. 14 N3 Sett 82.