Spedizione in abbonamento postale a tariffa intera/Milano

Quaderni di informatica

Rassegna di tecnologia ed applicazioni degli elaboratori elettronici
Anno XIII - Numero 3 - 1987

31

CENTRO STUDI
INFORMATICA E AUTOMAZIONE

Honeywell Bull

FPDM-38

Honeywell Bull

Quaderni di informatica

Rassegna di tecnologia ed applicazioni degli elaboratori elettronici
Anno XIII - Numero 3 - 1987

Sommario

Direttore Responsabile
Franco Filippazzi

Comitato di Redazione
A. Cicu, C. Falcetti, P. Lupo,
M. Nobile, G. Occhini
F. Sala

Redazione
Honeywell Bull
Centro Studi Informatica e Automazione
Via G.M. Vida, 11 - 20127 Milano

Stampa
tipolito Maggioni s.a.s.

Autorizzazione del Tribunale di Milano
n. 93 del 20 Marzo 1974

«Quaderni di Informatica» ospita
articoli e contributi di varia provenienza.
La responsabilità delle opinioni
espresse rimane agli Autori.

Il fenomeno UNIX

Vittorio Cajani
Honeywell Bull Italia
Centro di Ricerca e Progettazione
Pregnana Milanese

1. Introduzione

Quanto segue è una breve analisi dell'evoluzione di UNIX(*), fatta coll'obiettivo di mettere a fuoco il fenomeno UNIX più dal punto di vista del mercato che non delle caratteristiche tecniche del prodotto.

Infatti è convinzione di chi scrive che le ragioni del successo di UNIX siano da ricercare non tanto nell'architettura del prodotto e nelle funzionalità che esso offre (a parte la portabilità del sistema stesso) quanto in altri aspetti. Primo di tutti la situazione del mercato EDP all'inizio degli anni ottanta, dominato da IBM e con un numero notevole di compagnie concorrenti non in grado di sostenere gli investimenti necessari a mantenere la competitività in un mercato in rapida evoluzione e sempre più diversificato. In secondo luogo l'esigenza delle grosse utenze, particolarmente a livello governativo, di non essere legate ad un unico fornitore nel momento in cui l'EDP diventa sempre più una componente essenziale del processo produttivo. Questi ed altri fattori, ben più che l'architettura del sistema, hanno fatto sí che in un ben preciso momento storico UNIX avesse le caratteristiche per affermarsi come sistema standard. Questo senza nulla togliere al fatto che l'architettura e le funzionalità offerte da UNIX sono in genere allo stato dell'arte, salvo eccezioni che verrano discusse più avanti.
In conclusione verrà avanzata un'ipotesi sul possibile futuro di UNIX.

2. UNIX: un sistema che viene da lontano

La prima versione del sistema operativo UNIX è stata sviluppata, alla fine degli anni 60, nei Bell Laboratories (*) da D. Ritchie e K. Tomphson.

Il loro obiettivo era la realizzazione di un software di base (includente, oltre al sistema operativo, un linguaggio di programmazione nuovo, il C language) ed un insieme di programmi applicativi da utilizzarsi internamente ai Bell Laboratories da parte dei gruppi di ricerca. Uso primario del sistema era lo sviluppo di software e la preparazione di documentazione tecnica.

L'architettura originaria del sistema UNIX ripropone molti dei concetti di base del sistema Multics, da cui UNIX si differenzia per una implementazione meno complessa e meno sofisticata. Caratteristiche del sistema UNIX erano, oltre il linguaggio di programmazione del sistema stesso, la sua circoscritta dipendenza dall'hardware per cui era stato sviluppato e la sua flessibilità, soprattutto per quel che riguarda la programmazione e la connessione di periferiche.

Il sistema era stato disegnato per un uso in timesharing, cioè per un uso concorrente da parte di più utenti, ognuno operante in un ambiente di dati e programmi propri e generalmente non condiviso con altri utenti. L'accesso al sistema è tipicamente interattivo, attraverso un videoterminale.

Sin dalle origini il sistema UNIX é stato progettato per essere "portabile", cioé facilmente trasferibile da un hardware ad un altro. Infatti non solo il kernel, cioé la parte di sistema tradizionalmente dipendente dallo hardware da gestire, è stato scritto per la massima parte in un linguaggio ad alto livello e non "machine-dependent", come ad esempio è invece un assembly language, ma anche sono ben isolate tutte le parti di sistema operativo tipicamente dipendenti dalla architettura hardware. Lo stesso compilatore del C language è strutturato in modo tale da essere facilmente modificabile per generare istruzioni di un particolare "interior decor", cioè di uno specifico linguaggio macchina. Questa caratteristica, che sarà poi la premessa per il successo di UNIX, è stata raffinata man mano che all'interno degli stessi Bell Laboratories il sistema UNIX veniva portato su nuovi sistemi hardware.

Questi trasferimenti venivano fatti via via che un gruppo di progetto sceglieva per un prodotto specifico l'uso di uno specifico hardware che eseguisse il software di UNIX. Questo si verificò più volte. Infatti negli anni 70 l'AT&T e le sue consociate non erano produttrici di sistemi ge-

(*) UNIX è un trademark dei Bell Laboratories AT&T.

(*) I Bell Laboratories sono uno dei centri di ricerca più prestigiosi degli Stati Uniti. Vi operano più di quindicimila ricercatori, con una notevole percentuale di PhD e vi hanno lavorato molti premi Nobel.

neral purpose, per cui UNIX veniva portato su sistemi di altri produttori. Inizialmente i sistemi hardware più utilizzati sono stati i minicomputer della DEC, dal PDP-11 al VAX, anche se da anni all'interno dei Bell Labs UNIX è stato portato su mainframe IBM.

Va sottolineato che il parlare di portabilità significa che i costi di adattamento del sistema operativo e del C language ad un hardware differente da quello su cui il sistema è stato sviluppato sono contenuti (non certo nulli) e che il tempo solare richiesto dall'adattamento del sistema ad un nuovo hardware è anche esso molto contenuto. Questo in relazione ai costi di sviluppo ex novo di un sistema operativo che offra funzionalità equivalenti. Dove invece la portabilità ha un significato ben più rigoroso ed importante è a livello applicativo: tutte le applicazioni sviluppate in ambiente UNIX standard, sono eseguibili su qualsiasi altro sistema UNIX (indipendentemente dall'architettura hardware del sistema), non richiedendo altro che una ricompilazione.

Durante gli anni '70 i Bell Laboratories iniziarono una politica di licensing del software UNIX, cioè di distribuzione all'esterno dei sorgenti del sistema operativo e/o della versione oggetto del sistema stesso. I sistemi per cui iniziò la distribuzione di UNIX erano i mini della DEC, sia il PDP11 sia il VAX/11-780.

Date le caratteristiche funzionali del sistema, UNIX ebbe una larga diffusione a livello universitario. In particolare l'Università di Berkeley (CA) fu molto attiva nell'area UNIX: creò delle estensioni al sistema operativo, introdusse nuovi comandi ed utilities ed iniziò essa stessa a distribuire una sua versione di UNIX.

Attività analoga a quella di Berkeley fu iniziata da varie software houses, che offrirono un supporto al trasferimento di UNIX, un servizio di manutenzione del software UNIX (cioè di assistenza in caso di malfunzionamenti software, servizio inizialmente non fornito dalle consociate AT&T) e di estensioni funzionali al sistema stesso. Tipico il caso della MicroSoft, che produsse e commercializzò Xenix, un sistema derivato da UNIX.

Queste attività, mentre da un lato contribuivano in modo determinante alla diffusione di UNIX, d'altro canto crearono un problema. Il problema, derivante dalle estensioni funzionali e quindi anche di interfaccia, era il diffondersi di "dialetti" di UNIX, cioè di sistemi UNIX "look-alike", non rigorosamente compatibili con la versione distribuita da AT&T (e tra di loro). Non essere strettamente compatibili, comportava un rischio mortale per UNIX stesso: il non garantire la portabilità (senza alcuna modifica) di applicazioni da un sistema UNIX ad un altro, compromettendo così la possibilità per UNIX di qualificarsi come sistema standard.

Tuttavia UNIX (o meglio i sistemi UNIX "look-alike", in particolare XENIX e lo UNIX dell'Università di Berkeley) stavano guadagnando una quota di mercato non indifferente, grazie ad un largo numero di piccoli e non piccoli produttori che avevano scelto UNIX come sistema operativo per il loro hardware o per la loro soluzione applicativa. In particolare va sottolineato che sistemi UNIX "look-alike" acquistavano importanza non solo per applicazioni scientifiche o di office automation, ma anche per applicazioni real-time, tipiche del data processing nel manufacturing.

I sistemi su cui UNIX si stava affermando erano i micro ed i mini, allora con architettura a 16 bit.

3. UNIX: da sistema portabile a sistema standard

Fino a qualche anno fa la AT&T non appariva interessata più di tanto alla promozione di UNIX. Invece, subito dopo la "deregulation" di inizio '84, AT&T decise di entrare nel mercato EDP. Fu formata, tra le altre, una compagnia (AT&T Information Systems) coll'obiettivo specifico di produrre e commercializzare sistemi.

La linea di prodotti AT&T, sia i sistemi della linea 3B che il PC (a parte un modello di PC MS-DOS), era basata sul sistema operativo UNIX.

AT&T Technologies, un'altra compagnia sorta dalla deregulation, iniziò la produzione di una serie di chip, inizialmente per uso interno all'AT&T stessa, cioè per i sistemi della famiglia 3B, successivamente offrendoli anche ad altri produttori di sistemi. Tali chip (che includono, oltre l'MPU (Main Processing Unit), una memory management unit, un DMA (Direct Memory Access) controller ed un memory controller) erano stati disegnati espressamente per ottimizzare il supporto "on chip" al sistema operativo UNIX. Si può affermare che dal punto di vista architetturale e di performance, tali chip sono tra i migliori, o forse i migliori, oggi utilizzabili per lo sviluppo di sistemi UNIX.

Ma quel che più interessa è che AT&T perseguì con determinazione l'obiettivo di promozione del software UNIX, indipendentemente dalla vendita dei propri sistemi UNIX. L'obiettivo era di imporre UNIX come il sistema operativo standard per sistemi micro e mini.

Per raggiungere questo obiettivo dovevano realizzarsi più condizioni:

a. prima di tutto doveva esser ben definita l'interfaccia programmatica di UNIX. Questo per impedire l'effetto torre di Babele generato da sistemi UNIX "look-alike", che avrebbe di fatto ucciso le possibilità di UNIX di affermarsi come standard;

b. in secondo luogo bisognava promuovere lo sviluppo di applicazioni nel mondo UNIX. Questo per rendere l'offerta verso l'end-user competitiva nei confronti dei sistemi con un ricco catalogo di applicazioni, come i sistemi IBM, DEC, etc.;

c. infine era necessario un ulteriore stimolo affinché altri produttori di sistemi adottassero UNIX come sistema operativo.

Ma procediamo per punti.

Innanzitutto l'obiettivo di standardizzazione dell'interfaccia UNIX era già perseguito da un gruppo di utenti UNIX, lo User Group, che ad inizio '84 aveva già pubblicato una prima bozza di definizione di interfaccia. AT&T si mosse in parallelo, sviluppando una sua proposta di standard, lo SVID (System V Interface Definitions), che in buona sostanza formalizzava le interfacce programmatiche già presenti nello UNIX (System V Release 2)(*).

Qui nacquero i primi problemi di standardizzazione, dal momento che pur essendo per la maggior parte identiche tuttavia le due definizioni di interfaccia divergevano su un punto nodale: le convenzioni di "locking" dei file (cioè di controllo della concorrenza di accessi da più programmi allo stesso file).

Un altro grosso problema era rappresentato da XENIX, lo UNIX look-alike distribuito dalla Microsoft. L'interfaccia di questo sistema era in molti punti diversa da quella di System V coll'aggravante (per AT&T) che era il sistema UNIX a più larga diffusione sul mercato all'inizio degli anni '80.

AT&T si è mossa in modo molto determinato nei tre anni passati, cercando di far convergere le varie versioni di UNIX verso uno standard comune. Ad oggi si può dire (anche se non tutti i giochi sono stati fatti) che esiste uno standard de facto, che è coerente col nucleo delle System V Interface Definitions. In questo nucleo AT&T ha incluso alcune delle estensioni proposte dallo User Group. In parallelo, la Microsoft ha sviluppato una versione di XENIX che è coerente coll'interfaccia definita dallo standard AT&T (sviluppo congiunto AT&T e Microsoft, a quanto annuciato allo UNIFORUM 85)(*).

Va però rilevato un fatto molto importante, cioè che quello dell'AT&T non è l'unico standard per UNIX oggi esistente. Infatti sia l'X-OPEN (vedi poi) sia IEEE hanno pubblicato un loro standard. Questi in buona sostanza non contengono significative variazioni rispetto allo standard definito dalle SVID, ma più che altro una estensione dello scopo della standardizzazione.

(*) Le varie versioni di UNIX distribuite da AT&T sono identificate da una versione di sistema (III o V) e da un numero di release per tale versione. Di norma le release sono "upward compatible" tra di loro.

(*) Gli UNIFORUM sono mostre di prodotti UNIX tenute annualmente negli Stati Uniti. Sono organizzate dallo User Group.

Infatti le SVID definiscono le interfaccie con:
- il kernel, per quel che riguarda le richieste di servizi di sistema operativo (system calls)
- il C language
- le librerie di subroutine
- alcuni comandi
- il communication, a livello "trasporto" dello standard ISO/OSI.

Lo standard X-OPEN invece estende la definizione della interfaccia ai linguaggi COBOL, Pascal e Fortran, al Data Management di base (C-ISAM), al Data Base (SQL), al supporto per i linguaggi nazionali, etc.

IEEE (Institute of Electrical and Electronic Engineers) sta procedendo alla formalizzazione di uno standard di sistema portatile (chiamato POSIX) con scopi ancora più generali; in particolare si sta occupando della definizione delle estensioni richieste da UNIX in supporto ad applicazioni real-time. Di POSIX si riparlerà nella conclusione.

Come già detto, la chiave del successo di un sistema sta nella ricchezza del catalogo applicativo di cui dispone. Gli investimenti (ed il know-how) richiesti per sviluppare un insieme competitivo di applicazioni (sia orizzontali che verticali) non possono essere sostenuti da un'unica azienda. Per cui il problema può essere affrontato in termini di incentivazione alle software house per lo sviluppo di software applicativo (l'incentivo non è necessario qualora la diffusione del sistema sia tale da garantire alla software house un mercato per le proprie applicazioni; questo è il motivo della ricchezza del catalogo applicativo MS-DOS), oppure in termini di stimolare i produttori di sistemi all'adozione di UNIX come sistema operativo, in modo da creare un parco di sistemi con una diffusione tale da rendere remunerativo per le software house lo sviluppo di applicazioni.

AT&T si è mossa in tutte e due le direzioni (cioè sia verso i produttori di sistemi, sia verso le software house) e con AT&T (o meglio in parallelo ad AT&T) hanno attivamente perseguito la promozione di UNIX come sistema standard altre organizzazioni. Prima tra tutte l'X-OPEN e i gruppi di utenti UNIX.

La strategia AT&T per la promozione di UNIX, pubblicizzata tra l'altro durante gli UNIFORUM, si è articolata in più direzioni:

- consolidamento ed affermazione dello standard di interfaccia di UNIX, come si è già detto. Va aggiunto che non solo AT&T ha formalizzato e pubblicato le SVID (System V Interface Definitions), ma anche ha reso disponibile sul mercato una catena di test (la System V Verification Suite) che verifica l'aderenza alle SVID di un sistema UNIX. Inoltre a partire da UNIX System V Release 3 (resa disponibile da metà '86) AT&T permette il sublicensing di UNIX solo se il trasferimento di UNIX è stato eseguito garantendo la conformità allo standard SVID: questo al fine di forzare ulteriormente l'aderenza allo standard proposto dalle SVID;
- offerta di un servizio di assistenza. Da 1984 è possibile firmare con AT&T un accordo di manutenzione del software UNIX (va sottolineato il basso costo di tale servizio);
- investendo per l'evoluzione del software di UNIX. Tali investimenti si concretizzano in una major release di UNIX ogni 18 mesi circa. I contenuti di tale release sono in genere sia tecnologici (cioè miglioramenti di alcuni aspetti dell'architettura di sistema) sia funzionali. Sono orientati quindi sia a mantenere moderno e competitivo il software UNIX, sia ad estendere lo scopo, cioè le aree applicative di utilizzabilità di UNIX stesso;
- politica di licensing del software molto aggressiva, in particolare per quel che riguarda i costi di distribuzione del codice oggetto di UNIX. Oggi AT&T chiede solo 180 $ per la licenza oggetto di UNIX System V Release 3 su di un mainframe, 60 $ su di un personal computer;
- creazione di società di supporto ad UNIX, tipo AT&T UNIX EUROPE, che cura gli aspetti di distribuzione e di supporto ad UNIX in Europa;

- incentivazioni alle software house per lo sviluppo di pacchetti UNIX (sconti sull'hardware dei 3B, supporto, etc.) In particolare il catalogo del software UNIX per i chips WE 32xxx che nell'84 era di poche decine di applicazioni, oggi supera le 1500 applicazioni (cataloghi analoghi, seppur non tanto ricchi, esistono per altri chip, tipo Motorola MC68xxx ed INTEL iAPXx86).

Analogamente attivo, ma con obiettivi diversi, si è dimostrato l'X-OPEN, un consorzio di produttori di sistemi. Fondato nell'85, era inizialmente costituito solo da produttori europei (Olivetti, Bull, ICL, Nixdorf, Siemens, Philips, Ericsson). Scopo di questo gruppo è di promuovere uno standard (UNIX) che sia una valida alternativa per gli utenti allo standard de facto imposto al mercato da IBM. Nel '86 si è esteso oltre oceano e sono entrare a farne parte DEC, Hewlett-Packard e Sperry. Infine in concomitanza dell'Uniform di gennaio 87 è stata annunciata l'adesione di AT&T all'X-OPEN, fatto questo estremamente significativo dal momento che a questo punto X-OPEN include la grande maggioranza dei produttori di sistemi UNIX.

Per raggiungere l'obiettivo, gli aderenti al gruppo offrono sistemi tra loro rigorosamente compatibili, perché basati sulle stesse interfaccie, quelle definite dall'"X-OPEN portability guide". Come già detto, questo standard è un'estensione coerente delle SVID dell'AT&T, includendo la definizione dell'interfaccia programmatica di linguaggi diversi da C (ad oggi COBOL, Pascal e Fortran), la definizione di una interfaccia di Data Management e Data Base, etc. Ogni compagnia del gruppo contribuisce al finanziamento dell'attività di standardizzazione ed è impegnata allo sviluppo di sistemi che rispettano questo standard. Come risultato, l'offerta di sistemi UNIX diventa estremamente allettante, perché un insieme di compagnie garantisce la disponibilità di una gamma di prodotti tra di loro rigorosamente compatibili e quindi una garanzia agli investimenti fatti in termini di sviluppo applicazioni da software houses.

Gli UNIX User Group si sono creati un poco ovunque, dall'America all'Europa ed al Giappone. Scopo di tali gruppi è essenzialmente informativo, sono rivolti agli utenti UNIX ed in genere sono sponsorizzati da software house e produttori di sistemi UNIX.

4. UNIX: una realtà con ottime prospettive

Volendo oggi dare un quadro dei produttori che offrono sistemi che eseguono UNIX, bisognerebbe fare una lista che elenca tutti i più grossi produttori di sistemi. Nell'elenco si potrebbero fare tre raggruppamenti:

- produttori di soli sistemi UNIX. Sono in generale nuove compagnie, che hanno avuto in questi anni un trend di crescita molto positivo;
- produttori che hanno iniziato la produzione di sistemi UNIX parallelamente alla produzione (preesistente) di sistemi che eseguono software con interfaccia proprietary;
- produttori che offrono sullo stesso sistema sia l'interfaccia proprietary sia l'interfaccia UNIX (cioè lo stesso sistema è in grado di eseguire sia applicazioni sviluppate in base all'interfaccia nativa sia applicazioni UNIX).

I due ultimi casi sono i più interessanti da considerare, perché indicano chiaramente come vi sia una profonda convizione nel futuro di UNIX e nella offerta di applicazioni basate su interfaccia UNIX. Va sottolineato che i due ultimi grossi annunci IBM (il PC/RT e la linea di sistemi 9370) includono l'offerta del sistema UNIX.

I sistemi su cui è offerto UNIX sono in larga maggioranza dei micro e dei mini; tuttavia sia AT&T sia IBM offrono UNIX sui loro PC ed esistono implementazioni di UNIX per mainframe IBM, in cui UNIX è eseguito sotto VM. UNIX su mainframe è oggi commercializzato, oltre che da IBM, da Amdhal. In conclusione si può dire che UNIX è l'unico sistema operativo che copre tutta la gamma di sistemi (dai PC ai mainframe) e che UNIX si è soprattutto affermato come sistema operativo per Work Station e sistemi dipartimentali.

I dati delle tabelle seguenti sono stati ricavati da InfoCorp, un'azienda specializzata in indagini di mercato EDP, ed indicano le quote di mercato (mondiale) ad oggi raggiunte da UNIX e quelle in prospettiva. I dati

TAB. 1 - MICROSISTEMI MULTIUTENTE (fino a 50 K$)

	1985	1986	1990
Numero di sistemi UNIX consegnati	82.280	161.120	510.230
% di sistemi UNIX sul totale dei microsistemi consegnati	17.00	26.46	37.00

TAB. 2 - SISTEMI MULTIUTENTE (da 50 a 350 K$)

	1985	1986	1990
Numero di sistemi UNIX consegnati	9.600	11.200	33.400
% di sistemi UNIX sul totale dei sistemi installati	7.2	8.9	16.9

sono divisi per classi di sistemi; le classi di sistemi sono identificate dal costo del sistema stesso.

Ma quel che è più interessante notare è che, mentre fino a non molti mesi fa le analisi di mercato non indicavano uno spazio rilevante di UNIX nell'area dei sistemi dai 50 ai 350 K$, oggi InfoCorp fornisce i consuntivi e le previsioni riassunte nella tabella 2.

Per quel che riguarda il mercato in cui UNIX è più affermato, va detto che ad oggi è soprattutto richiesto a livello di amministrazioni pubbliche e di grosse aziende. In questi casi in genere la maggior ragione per scegliere sistemi UNIX deriva dalla garanzia che dà un sistema offerto da tutti i maggiori produttori. In particolare nel mercato oggi emergente dell'automazione di fabbrica, la scelta delle maggiori industrie è per UNIX come sistema operativo su cui basare le applicazioni di manufacturing. Questo anche se ad oggi UNIX System V non è adeguato per l'esecuzione di applicazioni in tempo reale, quantunque esistano parecchi sistemi UNIX look-alike che sono stati estesi (rispetto a System V) per dare un supporto adeguato al data processing in tempo reale. Per coprire questa lacuna di UNIX, IEEE ha formato un gruppo incaricato di definire lo standard di interfaccia delle estensioni necessarie ad UNIX per il supporto di applicazioni in tempo reale. A tale gruppo partecipano sia aziende utenti (prima fra tutte la General Motors, che è all'avanguardia negli USA nel processo di automazione della fabbrica) sia aziende produttrici di sistemi.

Va infine sottolineato che il "fenomeno UNIX" coinvolge i produttori di chip oltre che i produttori di sistemi e le software house. Infatti i maggiori produttori di chip di MPU (Motorola, Intel, Fairchild, National, etc) puntano, soprattutto per i chip con architettura a 32 bit, al mercato di sistemi UNIX come mercato primario. Questo significa non solo che definiscono l'architettura del chip in modo tale da ottimizzarne il supporto a UNIX, ma che offrono prodotti software per il mondo UNIX. In particolare sia Intel che Motorola distribuiscono i sorgenti di UNIX ed il C compiler utilizzabili nel trasporto di UNIX su di un sistema basato sui loro chip: questo al fine di minimizzare i costi di trasporto di UNIX per i produttori di sistemi basati su tali chip.

Una delle ragioni del successo di UNIX sta anche nella capacità di tali produttori di progettare chip sempre più veloci, che rendono via via meno rilevanti le naturali disottimizzazioni che ha un sistema portabile rispetto ad un sistema tagliato su misura per un hardware specifico (o viceversa).

5. Applicazioni UNIX

Il mercato applicativo UNIX è, come già detto, in crescita molto veloce. Per cui non è possibile darne una descrizione dettagliata che non sia immediatamente obsoleta. Comunque esistono cataloghi applicativi dei singoli produttori di sistemi ed anche cataloghi generali. Lo User Group ne ha recentemente reso disponibile uno. Quanto sopra è vero soprattutto per applicazioni verticali; invece per applicazioni orizzontali si può fare un quadro abbastanza completo dell'offerta esistente. Raggruppiamo le applicazioni orizzontali in queste categorie:

– communication
– data base management system
– grafica
– spreadsheet, word processor e applicazioni "integrate"
– linguaggi di programmazione

(I raggruppamento è arbitrario, ma semplifica la discussione).

Communication

Le applicazioni di communication oggi offerte per UNIX evidenziano chiaramente il ruolo che oggi UNIX ricopre con maggior successo, cioè quello di sistema dipartimentale.
Come sistema dipartimentale deve offrire sia il communication verso l'host (il mainframe IBM o Honeywell Bull) sia verso i PC e le Work Station.
Per il communication verso l'host IBM vengono offerte applicazioni di emulazione della 3770 batch work station (tipicamente di remote job entry), di emulazione di IBM cluster 3274/3276 (per emulazione di terminali 327x e printer 328x). Inoltre inizia l'offerta di prodotti che permettono l'integrazione del sistema UNIX nel mondo dell'office IBM, tipicamente l'accesso al DISOSS ed il supporto dell'architettura DIA (Document Interchange Architettura) sulla base del LU 6.2 e PU 2.1. Il ruolo del sistema UNIX è di satellite a la connessione all'host è yia linea telefonica o reti specializzate (PSDN).

Per quel che riguarda la connessione dei PC e delle Work Station e le interconnessioni tra sistemi UNIX, le applicazioni di communication offrono tipicamente:

- la possibilità di eseguire da PC applicazioni UNIX.
  Cioè il PC, oltre ad eseguire in locale delle applicazioni MS-DOS, può essere utilizzato come terminale di un sistema UNIX per eseguire applicazioni UNIX.
  Questo offre all'utente la possibilità di accedere da un unico terminale al più vasto mondo applicativo esistente (UNIX + MS-DOS).
- La possibilità di usare un sistema UNIX come "server":
  - disk server: l'applicazione MS-DOS accede (in modo trasparente, cioè come se accedesse a file MS-DOS) al file system di UNIX;
  - printer server: i report prodotti da applicazioni MDS-DOS vengono inviati per la stampa al sottosistema UNIX di spooling.

PC e sistemi UNIX vengono connessi attraverso una local area network (tipo Ethernet, Starlan, etc.) o più semplicemente attraverso una linea (RS232 o RS422).

(Per quel che riguarda la connessione di sistemi UNIX, già nell'offerta del software di base esistono facilities di "peer to peer connection" per remote loging, distributed file system, mailing, etc.).

È interessante sottolineare che questo tipo di funzionalità viene oggi offerto anche per la connessione di Macintosh.

Data base management system

Questi tipi di applicazioni sono molto importanti perché sono il mezzo

per ovviare alle carenze funzionali di UNIX nell'area del transaction processing e dell'information processing. Infatti UNIX, così come è distribuito da AT&T, non offre alcun metodo d'accesso a file che non sia quello della organizzazione a stream, cioè del file visto come sequenza di caratteri non altrimenti strutturati. Inoltre, il kernel di UNIX non offre alcuna primitiva che permetta di definire transazioni, cioè variazioni coerenti della base di dati utente. Queste due lacune funzionali rendono molto difficilmente sviluppabili gran parte delle applicazioni gestionali. Quindi di norma i produttori di sistemi UNIX offrono almeno una applicazione di DBMS. Spesso anzi più d'una, per garantire l'accesso ad un più vasto insieme di applicazioni verticali. Infatti i vari tipi di data base management system offrono facilities differenti e soprattutto non hanno una interfaccia programmatica compatibile verso il data management; in genere invece offrono una interfaccia SQL omogenea, per cui le convenzioni di interrogazione del data base sono compatibili.

Grafica

In questo segmento l'offerta è estremamente vasta. Si va dalla grafica per applicazioni gestionali fino a grafica ad alto livello, per applicazioni CAD. Infatti va ricordato che UNIX si è affermato come il sistema operativo per le work station.

Applicazioni integrate

Il termine è generico e sta ad indicare l'offerta di un insieme di applicazioni orizzontali elementari tra di loro integrate. Ad esempio l'integrazione tra un data base management system e un'applicazione di word processing, di spreadsheet, di business graphics od altro.

Quella che ne segue è un'offerta omogenea da un punto di vista interfaccia end-user e soprattutto che opera su basi di dati comuni e permette l'integrazione in documenti di tabelle ottenute con spreadsheet, di grafici, etc.

Sono chiaramente le applicazioni di maggior valore ma poche sono ad oggi le software house che sono state in grado di effettuare gli investimenti richiesti per ottenere un'offerta così completa. Tuttavia il trend di integrazioni attraverso joint venture tra software house specializzate in rami diversi è una realtà che si va diffondendo ed è uno dei modi più incisivi scelto da alcune software house per aumentare la competitività in un mercato estremamente aggressivo.

Linguaggi di programmazione

Si può dire che non esiste linguaggio di programmazione non disponibile nel mondo UNIX. Questo è vero sia per i linguaggi tradizionali che per linguaggi di intelligenza artificiale che per linguaggi della quarta generazione.

È una delle aree in cui l'offerta è più ricca e quindi la competitività più spinta.

Per linguaggi tradizionali la competitività si misura sia sull'offerta di un set completo di linguaggi (COBOL, FORTRAN, PASCAL, C, RPG, etc) al fine di garantire di poter richiamare da ogni linguaggio procedure scritte in un altro linguaggio, sia sull'efficienza del codice macchina generato. A tal riguardo va sottolineato che questa competitività ha portato grossi progressi nelle tecniche di compilazione soprattutto nell'area degli ottimizzatori, cioè di quei moduli di un compilatore che si preoccupano di minimizzare il numero di istruzioni generate per eseguire una funzione e di sfruttare al massimo le caratteristiche dell'MPU (tipo e numero di registri, modi di indirizzamento).
Per i linguaggi della quarta generazione, invece, la competitività è orientata all'ottenimento di interfaccie alla programmazione sempre più semplici e potenti, tali comunque da permettere al programmatore di concentrarsi sempre più sul vero problema (cioè quello che l'applicazione deve risolvere) e di dover conoscere sempre meno regole di programmazione e dettagli sul sistema.

6. Conclusione

Da quanto detto, risulta evidente che il maggior interessato e avvantaggiato dal fenomeno UNIX è l'utente. Può scegliere tra più fornitori di sistemi compatibili, cioè funzionalmente equivalenti e che non richiedono modifiche alle applicazioni nel trasferirle da un sistema all'altro. Inoltre la competitività tra i vari produttori, una volta eliminata la variabile "funzionalità offerte dal sistema", si focalizza sul rapporto costo/prestazioni e sui servizi offerti: già fin da oggi è evidente per i sistemi UNIX un trend di aumento delle prestazioni e diminuzione dei costi estremamente veloce. Viceversa l'offerta di funzionalità sempre più valide è garantita dalla competitività tra le molte software house che hanno scelto UNIX come interfaccia per le loro applicazioni.

A questo punto i giochi sembrano fatti:
- esiste un standard de facto UNIX
- i maggiori produttori di sistemi offrono (anche) UNIX
- l'offerta di applicazioni UNIX è già molto vasta ed è in continuo aumento.

Conseguentemente UNIX dovrebbe affermarsi sul mercato mondiale come il sistema operativo standard. Eppure il governo americano ha già annunciato che quando lo standard POSIX della IEEE sarà approvato, questo e non UNIX sarà l'interfaccia di sistema richiesta come prerequisito per ogni offerta di sistema. E c'è da prevedere che lo stesso tipo di richiesta sarà fatto anche dagli altri utenti.

Perché POSIX e non UNIX?

Sembrano funzionalmente la stessa cosa, come già detto. Questo è vero. Ma c'è una differenza fondamentale: UNIX è un prodotto dell'AT&T, cioè è controllato da un'azienda, POSIX ha l'imprimatur ufficiale di un organismo di standardizzazione.

C'è quindi da aspettarsi che in un prossimo futuro il sistema operativo standard abbia nome POSIX e che le applicazioni scritte per UNIX siano eseguibili senza alcuna modifica da POSIX. L'unico punto di domanda riguarda la capacità di IEEE nel seguire con estensioni dello standard l'estendersi delle interfaccie che via via richiedono di essere definite. E resta infine da capire se AT&T darà "source licenses" per uno UNIX "POSIX compatible", o se i singoli produttori di sistemi dovranno acquisire la capacità di estendere l'interfaccia del loro software per mantenere la compatibilità con le evoluzioni dello standard POSIX.

Dal CAI all'ICAI: sistemi intelligenti per insegnare

MARIA TERESA MOLFINO
Istituto per la matematica applicata del CNR
Genova

1. Introduzione

Il processo insegnamento/apprendimento è un processo complesso e ricco di molte sfaccettature. In esso intervengono tre componenti fondamentali: la materia che si vuole insegnare, il docente che deve trasmettere le conoscenze su di essa e lo studente che deve acquisirle.

Durante il processo di insegnamento, l'insegnante deve essere in grado di assumere diversi ruoli, quali, ad esempio, quello di tutore, assistente, consigliere e così via. Deve inoltre saper passare rapidamente dall'un ruolo all'altro, al fine di comunicare la sua conoscenza nel modo più proficuo e più consono possibile al modo di pensare dello studente.

Un buon insegnante deve anche tener conto del fatto che l'apprendimento ha luogo per diverse vie quali, ad esempio, analogia, induzione da esempi, deduzione da regole già note, scoperta, e così via.

Sia i ruoli che l'insegnante deve assumere, sia i cammini di apprendimento, dipendono dalla materia che deve essere insegnata e dalle capacità dell'allievo.

Questi molteplici aspetti rendono particolarmente difficile l'obiettivo di automatizzare il processo di insegnamento/apprendimento; l'apprendimento, inoltre, è un fenomeno la cui natura non è ancora ben nota. Questo fatto, che rende delicato e di difficile gestione il processo didattico di per sè, comporta inevitabilmente delle grossolane semplificazioni quando si tenta di automatizzarlo. Tuttavia, l'utilizzo di tecniche di Intelligenza Artificiale (IA) nella costruzione di programmi per la didattica e lo sviluppo della Scienza Cognitiva hanno dato un notevole contributo: esse infatti hanno permesso l'analisi separata dal punto di vista teorico oltre che implementativo, delle tre componenti fondamentali del processo didattico e hanno consentito di studiare concretamente, attraverso i diversi prototipi che sono stati realizzati, uno o l'altro dei diversi aspetti che in esso intervengono.

2. Dal CAI all'ICAI

Alla possibilità di utilizzare nella didattica sistemi basati sul calcolatore, detti, nel seguito, sistemi CAI (Computer Assisted Instruction), si pensò alla fine degli anni '50. Ci si avvide subito, infatti, che la velocità di calcolo, la possibilità di maneggiare grosse quantità di informazione, la capacità di fornire uscite grafiche, potevano costituire un notevole supporto alla didattica. Relativamente all'educazione, l'avvento del calcolatore venne, da alcuni, paragonato alla invenzione della stampa [Ha180].

Inizialmente, i sistemi per la didattica si basavano su tecniche di istruzione programmata e avevano come obiettivo quello di fornire strumenti di apprendimento flessibili, stimolanti ed individualizzati [Edw75,-Fle72]. Essi utilizzavano quasi esclusivamente una strategia di tipo tutoriale: venivano, cioè, proposti allo studente certi contenuti in un'alternanza di presentazione di unità di informazione e di domande di verifica. La strategia tutoriale si basa, infatti, sull'ipotesi che un processo di apprendimento sia una trasmissione di conoscenza e, di conseguenza, sempre decomponibile in una sequenza di informazioni semplici: è sufficiente che il calcolatore presenti queste informazioni le une successivamente alle altre e verifichi, ogni volta, se l'informazione presentata è correttamente assimilata dal discendente.

Sistemi basati esclusivamente su una strategia tutoriale mostrarono ben presto le loro limitazioni [Osh82]. Il limite principale consiste nella rigidità del dialogo, determinata dal fatto che i cammini didattici sono fissati a priori. Tuttavia, essi si sono rivelati didatticamente utili qualora siano utilizzati in fase di addestramento e di recupero.

Sono stati fatti, successivamente, numerosi sforzi per realizzare software didattico che utilizzi strategie più flessibili, in modo da consentire un'impostazione pedagogica più articolata. Si indirizzò la ricerca verso la realizzazione di sistemi che costituissero un ambiente per risolvere problemi specifici, nel quale lo studente potesse acquisire abilità di base, sviluppando capacità di tipo creativo [Abe80, Pap80].

Gli ambienti di apprendimento sono basati sul convincimento che il ragazzo capisca la natura dei concetti ed impari metodologie di risolu-

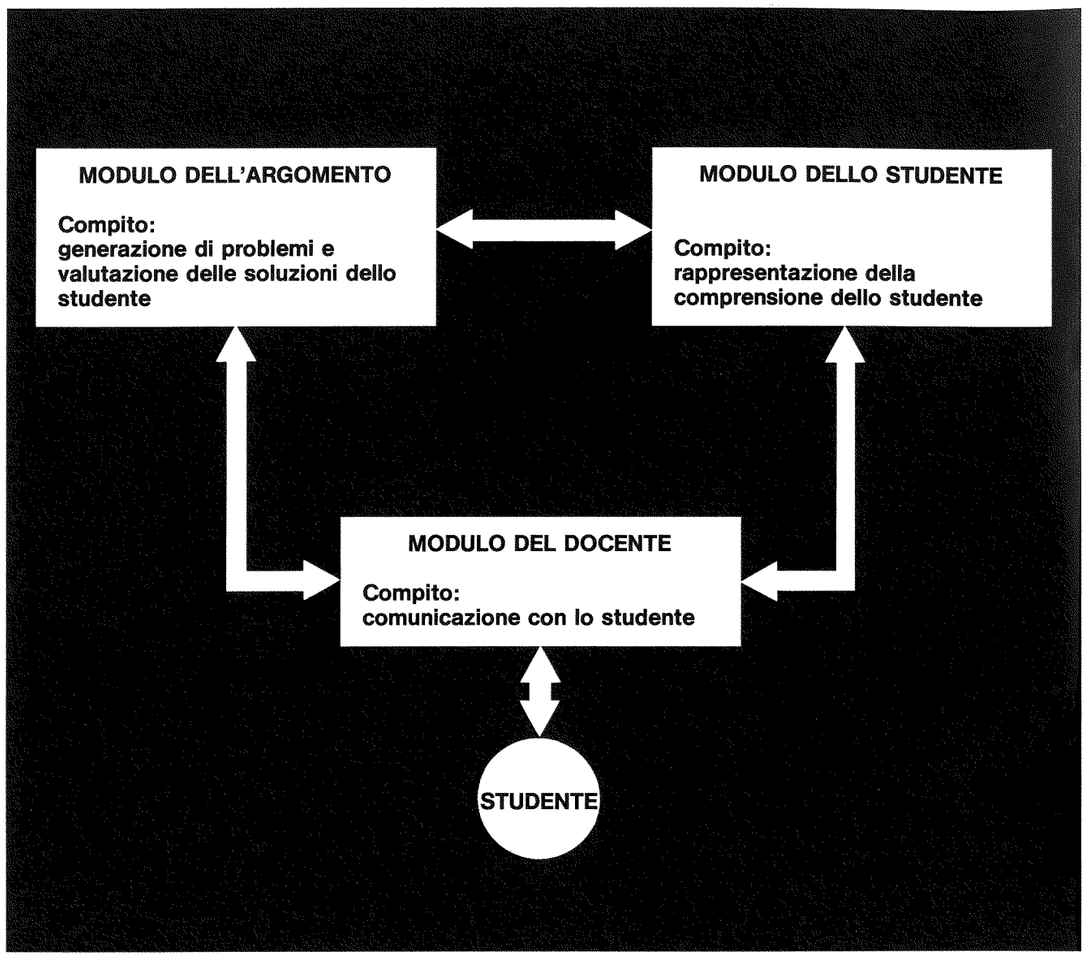


Fig. 1 - Schema generale di un sistema ICAI

zione di problemi se gli si offre la possibilità di lavorare in un ambiente, per sua natura ricco di potenzialità educative, che lo lascia libero di esplorare diverse possibilità e gli chiede di far ricorso alla creatività.

Tali ambienti di apprendimento sono prodotti software adattabili a diverse situazioni didattiche ed inseribili in contesti diversi. Essi mirano ad aiutare gli allievi ad acquisire conoscenze non per memorizzarle ed essere in grado di rispondere alle domande che il calcolatore pone loro, ma piuttosto per raggiungere, anche se in situazioni particolarmente semplici, un grado di autonomia nel trovare le strategie adeguate a risolvere problemi particolari. Tali ambienti presuppongono, ovviamente, la presenza di un insegnante che guida, consiglia ed incoraggia gli studenti nel loro utilizzo.

Notevole impulso a ricerche, finalizzate ad analizzare le limitazioni dei sistemi CAI, siano essi tutoriali o ambienti di apprendimento, ed a sviluppare proposte per superarle, è stato dato sia dallo sviluppo di tecniche IA nella costruzione di programmi, sia da quello della scienza cognitiva [AAVV86, Bar82, Gab80, Har73, Viv85]. In particolare si è rivolta l'attenzione all'analisi separata, dal punto di vista implementativo oltre che teorico, delle componenti fondamentali che intervengono nel processo insegnamento/apprendimento: studente, insegnante, materia, non distinte all'interno dei precedenti sistemi CAI [Bot85]. Tale analisi ha condotto, da un lato, a superare le principali deficienze didattiche dei sistemi tutoriali e, dall'altro, alla realizzazione di ambienti di apprendimento che possano essere utilizzati senza l'intervento dell'insegnante perché egli è, in qualche forma, incorporato nel sistema. Nel processo di apprendimento guidato dall'insegnante, lo sviluppo della lezione è diretto dalla natura degli errori dello studente, dal livello di interesse e gradimento che egli prova per una strategia didattica piuttosto che per un'altra, dalla capacità dello studente di fornire risposte o soluzioni a problemi diverse o più brillanti di quelle pensate dall'insegnante. Obiettivo dei

sistemi ICAI (Intelligent CAI) è quello di studiare metodi e sviluppare proposte finalizzate a risolvere questi problemi.

Oggi, mentre i sistemi CAI sono ampiamente collaudati e si trovano in commercio distribuiti da case di software e hardware e case editrici, gli unici sistemi ICAI disponibili commercialmente sono PROUST [Joh85a,Joh85b] e LISP tutor [And85b]. La maggior parte dei sistemi ICAI, infatti, sono prototipi: attraverso di essi ci si propone di studiare l'uno o l'altro dei problemi che sorgono nell'automazione di un processo didattico e di analizzare anche sperimentalmente la natura dell'apprendimento.

3. Sistemi intelligenti per insegnare

L'idea di utilizzare tecniche di Intelligenza Artificiale nella costruzione di sistemi per la didattica risale al '70. Carbonell in "AI in CAI: An Artificial Intelligent Approach to Computer Assisted Instruction" [Car70], individua nella mancanza di conoscenza la principale causa della scarsa flessibilità dei sistemi CAI e dà l'avvio ad una nuova generazione di sistemi per la didattica basati su calcolatore: i sistemi ICAI o ITS (Intelligent Tutoring Systems). I sistemi ICAI, dal punto di vista dell'IA, sono sistemi esperti, cioè programmi che sono in grado di fare deduzioni analoghe a quelle di un esperto sull'argomento in esame, grazie ad un insieme di meccanismi inferenziali che permettono di dedurre nuove informazioni da quelle note. Ma, oltre alla conoscenza esperta, un sistema ICAI deve anche contenere una componente che sia in grado di capire il comportamento dello studente ed una componente che rappresenti una o più metodologie d'insegnamento. Le tecniche IA, usualmente impiegate nella costruzione di sistemi esperti, devono essere modificate in modo da rendere il processo di ragionamento seguito per fare deduzioni comprensibile per il discente. Questo è avvenuto, ad esempio, quando si è pensato di utilizzare il sistema esperto MYCIN [Buc84] per applicazioni didattiche. A partire dalla conoscenza esperta sulle diagnosi delle terapie antibiotiche contenuta in MYCIN si è realizzato il sistema ICAI GUIDON [Cla82]. In GUIDON le regole contenute in MYCIN sono state modificate ed è stato necessario aggiungere nuove informazioni; infatti, in questo caso, il problema non consisteva solo nell'individuare la terapia adeguata, ma piuttosto nello spiegare in dettaglio allo studente il tipo di ragionamento che consente di individuarla.

3.1 Evoluzione storica

I primi sistemi ICAI sono sistemi che utilizzano una strategia di tipo tutoriale, ma nei quali si tenta di superare la rigidità del dialogo propria dei primi sistemi CAI realizzando un dialogo studente-sistema abbastanza simile a quello studente-insegnante. A tale scopo sono stati indirizzati gli sforzi verso due direzioni: l'organizzazione della materia in una base di conoscenza e lo sviluppo di una interfaccia che permetta il dialogo in linguaggio pseudo-naturale.

La conoscenza esperta, anche se rappresentata in diverse forme, può essere vista come un insieme di fatti e di regole che rappresentano legami tra fatti.

L'insegnante, che è incorporato nel sistema, analizza le risposte dello studente in base alla conoscenza esperta, e lo incoraggia affinché le sue conoscenze si adeguino ad essa. L'insegnante, cioè, interviene in situazioni in cui lo studente ha dimenticato un fatto o una regola che gli permette di raggiungere la risposta corretta.

Per quel che riguarda lo studente, l'ipotesi su cui si basano questi sistemi è quella che uno studente non possieda conoscenza sull'argomento in esame se la sua risposta ad una domanda è diversa da quella che la conoscenza esperta puo dedurre (overlay model) [Carr77, Cla82]. Per un'analisi dettagliata dei limiti dell'overlay model si veda, ad esempio, [For87].

Sistemi di questo tipo sono in grado di rispondere ad un gran numero di situazioni, anche se, essendo le spiegazioni generate in modo automatico, non sempre possono essere fornite spiegazioni diverse, qualora lo studente non sia soddisfatto di quella che gli è già stata proposta. Inoltre, come si è già detto, si adattano ad un numero limitato di situazioni didattiche.

Il più noto sistema ICAI di questo tipo è SCHOLAR [Car70, Car 73].

Successivamente, gli sviluppi della scienza cognitiva e quello delle nuove tecnologie hanno portato a focalizzare l'attenzione più sui processi di apprendimento che sulle metodologie di insegnamento, assegnando all'insegnante il ruolo di coadiutore di tale processo. Di conseguenza, anche i sistemi ICAI sono, successivamente, evoluti nella direzione di costituire ambienti costruttivi per l'apprendimento, cioè ambienti in cui lo studente è libero di formulare ipotesi e cercare soluzioni, ma nei quali la sua azione è indirizzata in modo da permettergli di imparare proprio dagli errori che commette. In tali ambienti la possibilità di operare dello studente è in qualche modo controllata dall'ambiente stesso, in quanto in essi è, in qualche forma, incorporata la figura dell'insegnante che agisce come supervisore [Bro77a].

Nei sistemi ICAI che realizzano ambienti di apprendimento un ruolo fondamentale è quello giocato dagli errori commessi dallo studente, che sono visti come strumento di costruzione della conoscenza. Il compito principale dell'insegnante è, perciò, quello di riconoscere il tipo di errore dello studente, e di intervenire per favorire il processo di apprendimento avvalendosi proprio dell'errore [Fis78].

In un ambiente del tipo descritto non è possibile diagnosticare il comportamento dello studente presentandogli domande o test memorizzati come avveniva nei sistemi di tipo tutoriale. La diagnosi deve essere effettuata principalmente a partire dal comportamento dello studente. In particolare, vengono presi in considerazione i seguenti fattori [Forc87]:

1. la mancanza di conoscenze esibita dall'allievo attraverso il suo comportamento; ad esempio, l'incapacità di applicare una strategia più efficiente di un'altra du-

rante lo svolgimento di un gioco.

2. L'errore commesso nel fornire una risposta ad un problema che richiede l'applicazione di una regola, della quale, ad esempio, lo studente dà una interpretazione errata.
3. I fraintendimenti durante lo sviluppo di un problema che richiede capacità di ragionamento; ad esempio, quelle sull'argomento che conducono a risultati sbagliati perché un ragionamento sensato è stato applicato ad un convincimento errato.

Il tipo di errore che viene analizzato dipende sia dall'ambiente sia dal dominio del problema. L'intervento del sistema varia anch'esso in dipendenza di questi due fattori.

I sistemi intelligenti basati su queste idee sono sostanzialmente di due tipi: ambienti di gioco ed ambienti di lavoro.

Gli ambienti di gioco sono ambienti ricchi e stimolanti, in cui lo studente può svolgere attività di problem solving. La conoscenza viene formalizzata e costruita in modo quasi inconsapevole per lo studente, come side-effect di una attività giocosa [Bur76]. L'insegnante, in questo caso, ha una funzione di "coach" [Bur79, Gol77b, Gol79]. Egli, cioè, deve guidare lo studente alla scoperta, interrompendolo al momento opportuno e dandogli suggerimenti adeguati in relazione agli errori commessi. Lo svolgimento di questa attività richiede la costruzione di un modello diagnostico. Il problema è reso particolarmente delicato dal fatto che, in molti casi, l'errore dello studente non è esplicitamente contenuto nella risposta, ma deve essere dedotto dalla mancanza di un'altra risposta possibile.

Sono, ad esempio, ambienti di questo tipo WEST [Bur82] e WUSOR [Gol82].

Gli ambienti di lavoro sono finalizzati all'acquisizione di abilità di problem solving specifiche, ad esempio, problemi di natura matematica o di programmazione. Lo studente viene invitato a risolvere un problema, proposto da lui stesso o dal sistema, appartenente ad un ambito ristretto, ed il sistema osserva il suo comportamento nella soluzione, assistendolo e fornendogli eventuale aiuto e consiglio. L'insegnante, cioè, agisce come un assistant. Anche questi sistemi incorporano, a diversi gradi di sviluppo, modello studente e modello docente.
Sono esempi di ambienti di questo tipo GUIDON [Buc84, Cla82], PROUST [Joh85a, Joh85b], ADVISOR [Gen82].

3.2 Alcune realizzazioni

I sistemi ICAI oggi realizzati sono numerosi. Essi differiscono l'uno dall'altro per l'argomento che si propongono di insegnare, per il tipo di problemi che indirizzano, e per l'approccio al processo insegnamento/apprendimento.
Per quel che riguarda l'argomento che si vuole insegnare, argomenti privilegiati sono stati quelli di natura matematica e quelli sulle tecniche e sui linguaggi di programmazione.

Per la matematica, ricordiamo: ADVISOR [Gen77, Gen82], un consulente automatico per il sistema MACSYMA; WUSOR [Gol82], finalizzato a sviluppare abilità logiche e probabilistiche nello studente che gioca a WUMPUS [Yob75]; WEST [Bur79], basato sul gioco "How the West Was Won", per l'apprendimento dell'aritmentica; DEBUGGY [Bur82], un sistema per determinare gli errori commessi nell'esecuzione di sottrazioni; LMS [Sle82, Sle83], un sistema che produce un modello diagnostico degli errori commessi risolvendo equazioni algebriche; GEOMETRY Tutor per l'insegnamento della geometria [And85a], QUADRATIC Tutor per insegnare a risolvere equazioni di secondo grado [Osh82], INTEGRATION per l'insegnameno dell'integrazione simbolica [Kim82].
Per quanto riguarda la programmazione, ricordiamo BIP [Bar76], per l'insegnamento del linguaggio BASIC; SPADEO [Mil82], per l'insegnamento del LOGO; PROUST [Joh85a, Joh85b] e MENO [Woo85] per quello del PASCAL; LISP tutor [And85, Rei85] per quello del LISP.
Sono stati inoltre sviluppati sistemi ICAI per l'insegnamento di altre discipline; relativamente all'insegnamento dell'elettronica, ricordiamo il progetto SOPHIE [Bro75, Bro77c, Bro82]. Per la medicina ricordiamo, il già citato progetto GUIDON, per la meteorologia, il sistema WHY [Ste82]; per la geografia, il sistema SCHOLAR. Infine, per la chimica, si vedano [Cab85] e [Bat86].

Per quel che riguarda il tipo di problemi affrontati, alcuni sistemi studiano, in modo particolare, quello di realizzare un dialogo in linguaggio pseudo-naturale. Citiamo, ad esempio, SCHOLAR e SOPHIE. Altri, studiano l'analisi del tipo di errore commesso dallo studente: ad esempio BUGGY [Bro77b, Bro78] e LMS. Infine, alcuni sistemi studiano il problema di adattare all'insegnamento la conoscenza esperta: ad esempio GUIDON.

Per quel che riguarda l'approccio al processo insegnamento/apprendimento, alcuni sistemi sono di tipo tutoriale. Altri sistemi, quale ad esempio WHY, utilizzano una metodologia di tipo socratico [Ste77], in cui l'insegnante risponde all'allievo ponendo altre domande. Ci sono sistemi di tipo coach, quali ad esempio WUSOR e WEST, in cui l'insegnante sorveglia e guida lo svolgimento di un gioco educativo. Altri sistemi, quale, ad esempio, ADVISOR, si propongono di assistere l'allievo durante lo svolgimento del lavoro.

4. Conclusioni

L'obiettivo di automatizzare un processo di insegnamento/apprendimento simile a quello reale è ancora lontano. I diversi modi in cui la conoscenza di concetti, procedure e fatti, anche attraverso percorsi contenenti errori, viene acquisita, sono stati analizzati solo relativamente ad un numero limitato di argomenti. Automatizzare un processo complesso quale è quello dell'apprendimento nella sua generalità, richiede una completa comprensione del processo stesso. Questo comporta uno studio approfondito, non solo su come si acquisiscono le conoscenze specifiche su un determinato argomento, ma anche su come si sviluppano e si acquisiscono, a seconda dell'ambiente educativo e delle condizioni psicologiche dello studente, le diverse strategie che permettono di utilizzare le conoscenze stesse.

Infine, debbono essere sviluppati e approfonditi gli studi relativi ai diversi tipi di ragionamento, siano essi formali, intuitivi od erronei.

Nonostante ciò i sistemi ICAI costituiscono un banco di prova estremamente interessante sia dal punto di vista scientifico sia da quello didattico.

Dal punto di vista scientifico, oggi un problema centrale in IA è quello di studiare come far svolgere alle macchine compiti di tipo cognitivo, in particolare, apprendere. Il CAI intelligente è un campo di sperimentazione eccellente per questi tipi di problemi. Dal punto di vista didattico, notevole è l'interesse per l'uso di sistemi, basati su calcolatore, visti come ambienti di apprendimento. Le numerose esperienze svolte evidenziano i vantaggi derivanti dall'utilizzo di tali sistemi, e fanno ritenere promettenti, dal punto di vista educativo, le applicazioni dell'IA finalizzate a facilitare l'utilizzo di tali sistemi e a sfruttarne al massimo le potenzialità educative.

Le ricerche in corso sono finalizzate a produrre modelli di apprendimento e di insegnamento di tipo nuovo. Questi avranno un inevitabile feedback sull'insegnamento stesso, anche su quello nel quale non si farà uso del calcolatore.

È perciò importante che venga sensibilizzata la classe insegnante su questi temi, in modo che i prodotti realizzati, anche se prototipi, escano dagli ambienti di ricerca e possano essere sperimentati dai loro naturali utenti. Solo una sperimentazione su larga scala e un confronto diretto con altri modi di insegnamento, permetterà di valutare la loro efficacia didattica. L'esperienza sulla sperimentazione dei primi prodotti basati su calcolatore ha già insegnato molto: gli sforzi non sono stati commisurati alle aspettative. È quindi importante una verifica costante dell'adeguatezza delle idee sottostanti i nuovi prodotti, alle situazioni reali, per capire quali strade vale la pena di seguire, quali aspetti debbono essere migliorati, quali ipotesi abbandonare.

Bibliografia

[AAVV86] Reports of International Conference of National Representatives and Experts, Paris 13-15, October, 1986.

[Abe80] ABELSON H., DISESSA A., *Turtle Geometry. The use of computer as a medium of exploring Mathematics,* Cambridge, Mass, The MIT Press, 1980.

[And85b] ANDERSON J.R., BOYLE C.F., YOST G., *The Geometry Tutor,* Proceedings of IJCAI '85, pp. 1-7.

[And85b] ANDERSON J.R., REISER B., *The LISP Tutor,* Byte, Vol. 10 (4), 1985, pp. 159-175.

[Bar76] BARR A., BEARD M., ATKINSON R.C., *The Computer as a Tutorial Laboratory: The Stanford BIP Project,* International Journal of Man-Machine Studies, Vol. 8, 1976, pp. 567-596.

[Bar82] The Handbook of Artificial Intelligence, Vol. 2, Cap. IX, BARR A., FEIGENBAUM E.A. (eds.), William Kaufmann Inc., Los Altos (CA), 1982.

[Bat86] BATEMAN D., *An Intelligence Knowledge Based System as an Aid to Chemical Problem Solving,* ESRC, ITE 10/86, May 1986, pp. 6-9.

[Bot86] BOTTINO R.M., MOLFINO M.T., *From CAI to ICAI: an Educational and Technical Evolution,* Education and Computing, North Holland, Vol. 1, 1985, pp. 229-233.

[Bro75] BROWN J.S., BURTON R.R., BELL A., *SOPHIE: A step toward creating a reactive environment,* International Journal of Man-Machines Studies, Vol. 7, 1975, pp. 675-696.

[Bro77a] BROWN J.S., GOLDSTEIN I., *Computer in a Learning Society,* Testimony for House Science and Technology Subcommitte on Domestic and International Planning Analysis on Cooperation, October 1977.

[Bro77b] BROWN J.S., BURTON R., LARKIN K., *Representing and using procedural bugs for educational purposes,* Proceeding of 1977 Annual Conference Association of Computing Machinery, Seattle, October 1977, pp. 247-255.

[Bro77c] BROWN J.S., *Remarks on building expert systems,* in Proceeding on the Fifth International Joint Conference on AI, 1977, pp. 1003-1005.

[Bro78] BROWN J.S., BURTON R.R., *Diagnostic models for procedural bugs in basic mathematical skills,* Cognitive Scienze, Vol. 2, 1978, pp. 155-192.

[Bro82] BROWN J.S., BURTON R.R., DE KLEER J. *Pedagogical, Natural Language and Knowledge Engineering Techniques in Sophie I, II ad III,* in Intelligent Tutoring Systems, Sleeman D. Brown J.S. (eds.), Academic Press, London, 1982, pp. 227-282.

[Buc84] BUCHANAN B.G., SHORTLIFFE E.H. (eds.), *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project,* Addison-Wesley Publ. Comp., Reading, MA, 1984.

[Bur76] BURTON R.R., BROWN J.S., *A Tutoring and Student Modelling Paradigm for Gaming Environment,* SIGCUE Bulletin N. 8, 1976, pp. 236-246.

[Bur79] BURTON R.R., BROWN J.S., *An investigation of computer coaching for informal learning activities,* International Journal of Man-Machine Studies, Vol. 11, 1979, pp. 5-24.

[Bur82] BURTON R.R., Diagnosis bugs in a simple procedural skill, in Intelligent Tutoring Systems, Sleeman D., Brown J.S. (eds.), Academic Press, London, 1982, pp. 157-182.

[Cab85] CABROL D., *Some examples of the use of Artificial Intelligence Techniques in Chemical Education,* 8th International Conference on Chemical Education, Tokyo, Blackwell Scientific.

[Car70] CARBONELL J.R., *AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction,* IEEE Transaction on Man-Machine Systems, Vol. MMS 11, N. 4, December 70, pp. 190-202.

[Car 73] CARBONELL J.R., COLLINS A.M., *Natural Semantics in Artificial Intelligence,* Proceedings of IJCAI '73 pp. 344-351.

[Carr77] CARR B., GOLDSTEIN I., *Overlays: a theory of modelling for Computer Aided Instruction,* AI Memo 406, AI Laboratory, MIT, 1977.

[Cha80] CHAMBERS J.A., SPRECHER J.W., *Computer Assisted Instruction: Current Trends and Critical Issues,* Communications of A.C.M., vol. 23, n. 6, 1980, pp. 332-342.

[Cla82] CLANCEY W.J., *Tutoring rules for guiding a case method dialogue,* in Intelligent Tutoring Systems, Sleeman D., Brown J.S. (eds.), Academic Press, London, 1982, pp. 201-225.

[Edw75] EDWARDS J. NORTON S. TAYLOR S., WEISS M., and VAM DUSSELDORP R., *How effective is CAI? A review of the research,* Education Leadership, November, 1875, 33, pp. 147-153.

[Fis78] FISHER G., BROWN J.S., BURTON R.R., *Aspect of a Theory of Semplification, Debugging and Coaching,* Proceeding of the Second annual conference of the Second annual conference of the Canadian Society for Computational Studies of Intelligence, 1978.

[Fle72] FLETCHER J., SUPPES P., JAMESON D.A., *Note on the effectiveness of computer-aided instruction,* Stanford, Ca 1972 (ERIC Document Reproduction Service No. ED 071 450).

[For87] FORD L., *Teaching strategies and tactics in intelligent computer aided instruction,* Artificial Intelligence Review, 1, 1987, pp. 201-215.

[Forc87] FORCHERI P., MOLFINO M.T., *A historical approach to educational computer-based systems,* Revue d'Intelligence Artificielle, Edition Hermes, (in press).

[Gab80] GABLE A., PAGE C.V., *The use of Artificial Intelligence Techniques in Computer Assisted Instruction: An Overview,* International Journal of Man-Machine Studies, Vol. 12, 1980, pp. 259-282.

[Gen77] GENESERETH M.R., *An Automated Consultant for MACSYMA,* Proceedings of IJCAI '77, pp. 789.

[Gen82] GENESERETH M.R., *The role of Plans in Intelligent Teaching Systems,* in intelligent Tutoring Systems, Sleeman D., Brown J.S. (eds.), Academic Press, London, 1982, pp. 137-155.

[Gol77a] GOLDSTEIN I. PAPERT S., *Artificial Intelligence, Language and the Study of Knowledge,* Cognitive Science, vol. 1, n. 1, 1977, pp. 1-21.

[Gol77b] GOLDSTEIN I., CARR B., *The Computer as Coach: an Athletic Paradigm for Intellectual Education,* Proceeding of 1977 Annual Conference Association of Computing Machinery, Seattle, October 1977, pp. 227-233.

[Gol79] GOLDSTEIN I., *The Genetic Epistemology of Rule systems,* International Jour-

nal of Man-Machine Studies, Vol. 11, 1979, pp. 51-77.

[Gol82] GOLDSTEIN I.P., *The genetic graph: a representation for the evolution of procedural knowledge,* in Intelligent Tutoring Systems, Sleeman D., Brown J.S. (eds.), Academic Press, London, 1982, pp. 51-77.

[Hal80] HALLWORTH H.J., BREBNER A., *Computer Assisted Instruction in Schools: Achievements, present developments and projections for the future,* Faculty of Education, University of Calgary, June, 1980.

[Har73] HARTLEY J.R., SLEEMAN D.H., *Toward more Intelligent Teaching Systems,* International Journal of Man-Machine Studies, vol. 5, 1973, pp. 215-236.

[Kim82] KIMBAL R., *A self-improving tutor for symbolic integration,* in Intelligent Tutoring Systems, Sleeman D., Brown J.S. (eds.), Academic Press, London, pp. 283-307.

[Kof75] KOFFMAN E.B., BLOUNT S.E., *Artificial Intelligence and Automated Programming in CAI,* Artificial Intelligence Journal, Vol. 6, 1975, pp. 215-234.

[Joh85a] JOHNSON W.L., SOLOWAY E., *PROUST: Knowledge-Based Program Understanding,* IEEE Transactions on Software Egineering, vol. SE-11, n. 3, March 1985, pp. 267-275.

[Joh85b] JOHNSON W.L., SOLOWAY E., *PROUST: An Automatic Debugging for Pascal Understanding,* BYTE, Vol. 10, 1985, pp. 179-190.

[Mil82] MILLER M.L., *A Structured Planning and Debugging Environment for Elementary Programming,* in Intelligent Tutoring Systems, Sleeman D., Brown J.S. (eds.), Academic Press, London, pp. 119-133.

[Pap80] PAPERT S., *Mindstorms. Children, Computer and Powerful Ideas,* Basic Books, 1980.

[Osh82] O'SHEA T., *A self-improving quadratic tutor,* in Intelligent Tutoring Systems, Sleeman D., Brown J.S. (eds)., Academic Press, London, 1982, pp. 309-334.

[Rei85] REISER J.B., ANDERSON J.R. FARRELL R.G., *Dynamic Student Modelling in an Intelliget Tutor for LISP programming,* Proceedings of IJCAI'85, pp. 8-14.

[Sel74] SELF J.A., *Student Models in Computer Aided Instruction,* International Journal of Man-Machine Studies, Vol. 6, 1974, pp. 261-276.

[Sle81] SLEEMAN D.H., SMITH M.J., *Modelling Student's Problem Solving,* Artificial Intelligence Journal, Vol. 16, 1981, pp. 171-187.

[SLE82] SLEEMAN D.H., *Assessing Aspects of Competence in Basic Algebra,* in Intelligent Tutoring Systems, Sleeman D., Brown J.S. (eds.), Academic Press, London, 1982, pp. 185-199.

[SLE83] SLEEMAN D.H., *Inferring Student Models for Intellingent Computer-Aided Instruction,* in Machine Learning, Michalski R.S., Carbonell J.G., Mitchell T. M., (eds.) Springer-Verlag, Tioga Press, 1983, pp. 483-510.

[Ste77] STEVENS A., COLLINS A., *The Goal Structure of a Socratic Tutor,* Proceeding of 1977 Annual Conference Association of Computing Machinery, Seattle, October 1977 pp. 256-263.

[Ste82] STEVENS A., COLLINS A., GOLDIN S.E., *Misconceptions in Students' Understanding,* in Intelligent Tutoring Systems, SLEEMAN D., Brown J.S. (eds.), Academic Press, London, 1982, opp. 13-24.

[Viv85] VIVET M., *Intelligence Artificielle et Recherche en Education,* Colloque Recherche en education, Paris 23-24/9/1986.

[Yaz86] YAZDANI M., *Intelligent tutoring systems survey,* Artificial Intelligence Review, 1, 1986, pp. 43-52.

[Yob75] YOB G., *Hunt the Wumpus,* Creative Computer, September-October, 1975.

I sistemi dipartimentali nell'automazione d'ufficio

GABRIELLA CAMPI

Honeywell Bull Italia
Networking & Office Systems
Roma

1 - Introduzione

Un'azienda può essere assimilata ad una serie di reticoli informativi, gli uffici, che mettono in connessione i fabbisogni di informazione per ciascun processo aziendale, con le fonti dei dati e che si estendono a tutti i processi aziendali ivi comprese le azioni degli utilizzatori.

Quando gli uffici vengono automatizzati si innescano una serie di peculiari fenomeni indotti, legati non solo all'organizzazione del lavoro ma anche alle modalità lavorative del singolo individuo, che esprime la sua personalità e le sue potenzialità attraverso il lavoro stesso.

Pertanto la pianificazione di una soluzione per l'automazione d'ufficio non può essere avulsa dal contesto in cui l'esigenza nasce.

In tale ottica l'approccio meno traumatico e più funzionale deve essere articolato e prevedere la progettazione simultanea di "tecnologie, organizzazione e figure professionali"; fornire un'adeguata flessibilità organizzativa che permetta di mantenere le strutture invarianti al mutare degli uomini e degli eventi.

Gli obiettivi da raggiungere sono: *efficacia* ed *efficienza;* essi danno luogo ad una cooperazione tra tutte le componenti aziendali che può essere sintetizzata nei seguenti punti:

- coerenza fra tipo di lavoro, ruolo e responsabilità
- appropriata distribuzione dei compiti rispetto ai ruoli
- scambio continuo di informazioni per il raggiungimento degli obiettivi aziendali, il superamento delle divergenze e di eventuali conflitti.

2 - Dall'ufficio industriale verso l'ufficio informatico

Le nuove tecnologie modificano necessariamente l'organizzazione del lavoro. Si possono definire tre stadi nell'evoluzione dell'organizzazione del lavoro:

a) preindustriale;

b) industriale;

c) informatico;

Ogni stadio è caratterizzato oltre che dalle tecnologie impiegate, da uno stile di gestione, di politica del personale, di struttura gerarchica degli staff dirigenziali, di efficienza e di relazioni interpersonali tra i lavoratori dell'ufficio.

a) La gestione di un ufficio preindustriale si basa in gran parte sull'impiego dei singoli, senza fare particolare affidamento su ausili elettronici e su di un'organizzazione sistematica del lavoro. Lo stile personale di ciascuno influisce sul taglio generale dell'attività e contribuisce al suo successo.

b) Lo schema di organizzazione industriale del lavoro si basa su di uno sforzo esplicito e consapevole per massimizzare l'efficienza e la produttività. Nascono esigenze di standardizzazione per le attività dei singoli, le procedure, le tecnologie e persino per le relazioni interpersonali; la ripartizione delle responsabilità vanno di pari passo con la crescita della struttura burocratica e quindi della proliferazione della documentazione cartacea.

L'ufficio così organizzato presenta gravi carenze poiché in esso il lavoro viene svolto secondo le modalità del procedimento a catena. Tale tecnica, utilizzata per la gestione di compiti non procedurali, cioè che non implicano necessariamente operazioni sempre uguali e ripetitive, ha come diretta conseguenza una forte tendenza a commettere errori a causa della suddivisione del lavoro; per cui aumentano i costi di revisione e di supervisione facendo calare l'efficienza generale.

c) L'ufficio informatico, invece, sfrutta le nuove tecnologie per conservare le migliori caratteristiche dei precedenti modelli di organizzazione evitandone le carenze; il suo obiettivo è la massima efficienza unita ad un ritorno al lavoro centrato sull'uomo anziché sulla macchina, dando spazio così alla creatività e all'espressione della personalità dell'individuo. Tale ufficio può accrescere, oltre la produttività, anche l'efficienza e la gratificazione del lavoro fornendo ad ogni individuo gli strumenti adatti al proprio ruolo e all'ambiente in cui è inserito.

Per cui vengono coinvolti in questo tipo di organizzazione anche gli uffici direttivi e di amministrazione, che precedentemente non venivano considerati, data la grossa discrezionalità presente in essi.

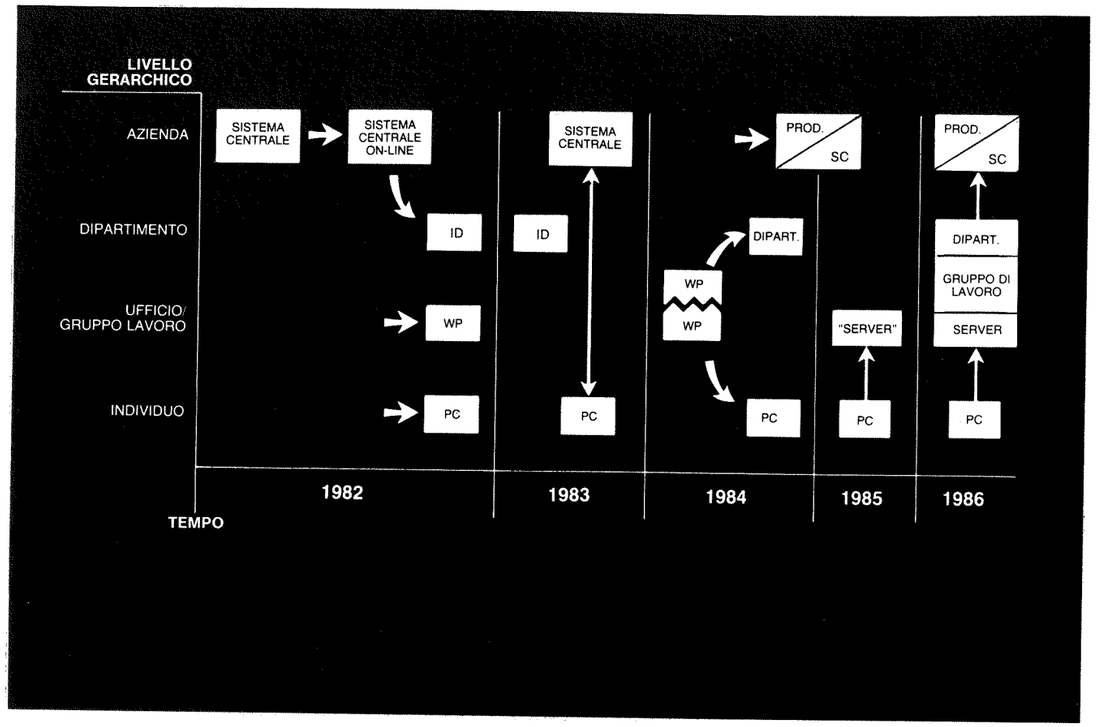


Fig. 1 - Evoluzione delle strutture informatiche rispetto ai cambiamenti organizzativi dell'azienda (ID = Informatica Distribuita; WP = Word Processing; SC = Solution Center; PROD = Sistema di produzione).

La trasformazione degli uffici è possibile grazie a tecnologie centrate sull'individuo, come le stazioni di lavoro intelligenti e sulla possibilità di "scambiare informazioni" nel formato tipico degli uffici, ad esempio documenti quali: lettere, circolari, normative, ecc.

Infatti è la possibilità di fare uno scambio efficace di informazioni che lega in sistema i moderni mezzi di elaborazione che altrimenti sarebbero entità chiuse in se stesse, capaci di comunicare solo con l'uomo.

Altro aspetto importante in tale tipo di organizzazione è quello economico.

Le comunicazioni per via elettronica diventano sempre meno costose rispetto a quelle su carta, per es. la posta elettronica può ridurre del 70% o più il costo dell'invio di una comunicazione; ulteriori economie derivano dalla elaborazione elettronica dei documenti, ma queste riduzioni piuttosto ovvie dei costi passano in secondo piano rispetto al risparmio di tempo da parte dei dirigenti e funzionari, di gran lunga la voce più pesante nel bilancio di un ufficio.

Infatti è stato appurato che l'80% del lavoro d'ufficio consiste nel comunicare. Delle due attività che si svolgono in un ufficio, elaborazione e comunicazione, la seconda non è soltanto quantitativamene prevalente, ma è anche il nucleo essenziale in senso qualitativo del lavoro organizzativo.

L'ufficio è il luogo in cui si prendono le decisioni fondamentali che determinano l'efficienza dell'intera organizzazione; esso è inoltre il luogo dove la tempestività di una decisione o di una risposta può avere le più grandi conseguenze: se l'ufficio è inefficiente lo è necessariamente anche l'organizzazione a cui esso appartiene.

L'infomatizzazione degli uffici si è evoluta nel tempo parallelamente agli schemi di organizzazione aziendale. A partire dall'82, periodo in cui imperava il sistema informativo aziendale come nucleo di quella che viene chiamata "informatica procedurale", cioè di quella parte dell'informatica che si occupa di tutte le procedure per la gestione amministrativa di un'azienda, e nell'ufficio si utilizzava solo il word processing per la gestione dei testi, l'utilizzo del PC non più in modo stand-alone ha permesso gradualmente all'utente inesperto di fare sia elaborazioni locali che collegarsi in modo radiale all'*host*.

Nasce così il concetto dell'*info-center*.

A questo punto comincia a delinearsi un'approccio all'informatica tale da permettere la gestione delle esigenze di gruppi omogenei che svolgono lo stesso tipo di lavoro e che hanno necessità di condividere risorse sia HW che SW (dischi, stampanti di qualità, banche dati, archivi ecc.) e di avere una maggiore aderenza alla struttura aziendale, che nel frattempo stava evolvendo dal modello gerarchico a quello interfunzionale.

Tale approccio si concretizza nel sistema dipartimentale che gioca così un ruolo primario come architettura che sottende tale nuova filosofia organizzativa.

3 - Relazione tra modelli aziendali e informatizzazione degli uffici

3.1 - L'azienda come sistema

Generalmente un'azienda può essere considerata come un "sistema", in quanto essa rappresenta una entità dinamica e pertanto ha una sua evoluzione dipendente non solo da fattori endogeni, quanto piuttosto dall'interazione di questi ultimi con il contesto in cui è inserita. Con queste premesse, a seconda dell'ambiente in cui opera, delle strategie di risposta al mercato concorrente, degli obiettivi e del tipo di impostazione manageriale, possono essere individuate, per grandi linee, quattro classi in cui rientrano le varie tipologie aziendali (fig. 2) e ad ognuna di queste classi può essere associata una modalità di informatizzazione,

Fig. 2 - Tipologie aziendali.

1. ENTE DI GESTIONE

Ente che opera in un mercato tranquillo, con politica manageriale di controllo e risposte strategiche difensive.

AUTOMAZIONE

- introduzione funzionale
- top-down
- EDP gestionale

2. ENTE DI AMMINISTRAZIONE

Ente che opera in un mercato disturbato, con politica manageriale di tipo amministrativo e risposte strategiche reattive.

AUTOMAZIONE

- introduzione funzionale
- top-down, con studi locali
- EDP + meccanizzazione posto di lavoro

3. ENTE DI INNOVAZIONE

Ente che opera in un mercato imprevedibile, con risposte strategiche creative e politica manageriale innovativa.

AUTOMAZIONE

- diffusione
- EDP + meccanizzazione + "computer aids"

4. IMPRESA EVOLVENTE

Impresa che opera in un mercato turbolento, con risposte strategiche propositive e politica manageriale imprenditiva.

AUTOMAZIONE

- sviluppo automazione d'ufficio come parte del processo di pianificazione strategica
- (1+2+3) + sistemi di supporto alla decisione

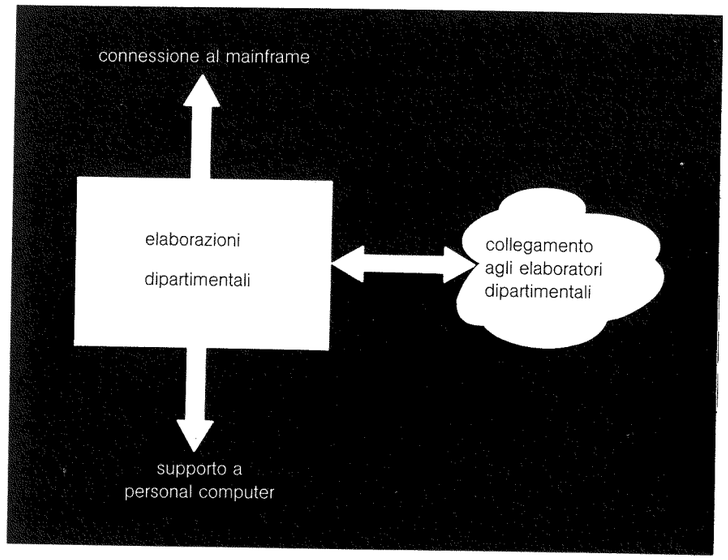
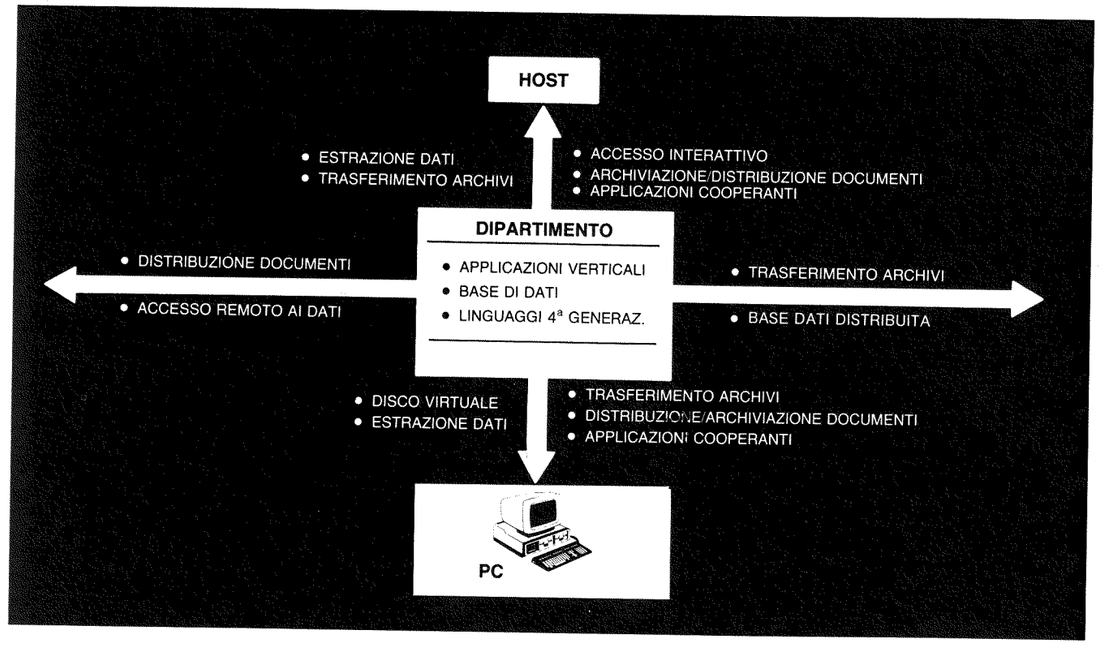


Fig. 3 - Principali macrofunzioni realizzate da un sistema dipartimentale.

Fig. 4 - Dettaglio delle macrofunzioni dipartimentali.



in modo da massimizzare la ricettività degli individui e minimizzare la possibilità di fallimento degli obiettivi.

3.2 - Sistemi dipartimentali

Per quanto detto precedentemente un'azienda moderna ed efficiente deve far sì che il flusso delle informazioni nei suoi uffici sia continuo e razionalizzato, in modo da permettere il rispetto e l'interazione delle principali stratificazioni dell'organizzazione:

- *individui*
- *gruppi o dipartimenti*
- *azienda* (identificata con il suo sistema informativo)

L'organizzazione informatica che meglio rispetta tale tipo di stratificazione è quella dipartimentale, poiché è definita da alcune macrofunzioni generali che possono essere ritagliate ad hoc per le realtà in cui bisogna operare.

Fondamentalmente queste macro funzioni sono (fig. 3):

- connessione al mainframe
- elaborazioni dipartimentali
- connessioni reticolari
- gestione delle stazioni di lavoro intelligenti.

Ogni macrofunzione a sua volta è costituita da una serie di sottofunzioni che sono specifiche del livello che realizzano (fig. 4).

A seconda delle necessità più o meno sofisticate di elaborazione, comunicazione e gestione dell'informatica individuale dell'azienda in oggetto, il fulcro di tale sistema (elaboratore dipartimentale) che chiameremo "server", può essere un elaboratore della fascia dei mini-medi oppure un PC (fig. 5).

3.3 - Sistema dipartimentale di primo livello

Un sistema dipartimentale di primo livello è costituito da una rete di personal computer, in cui anche il "server" è un personal computer. Tale configurazione è valida essenzialmente come soluzione iniziale oppure per automazioni non molto sofisticate, in quanto a livello dipartimentale le applicazioni verticali, cioè quelle applicazioni che risolvono problematiche specifiche di un segmento di mercato o di una fascia di utenti, che possono essere condivise sono particolarmente limitate. Inoltre in questo caso anche le funzionalità di comunicazione non sono molto sofisticate; per cui l'informatica individuale è predominante rispetto a quella dipartimentale.

Il reale vantaggio offerto da tale soluzione è rappresentato dal suo basso costo e dalla facilità di implementazione; così possono essere realizzati dei prototipi per la sperimentazione dell'automazione. La naturale evoluzione di tale configurazione tende ad una automazione più articolata che utilizzi strumenti più sofisticati; il sistema che soddisfa tali requisiti è detto appunto: "sistema dipartimentale evoluto".

3.4 - Sistema dipartimentale evoluto.

In un sistema dipartimentale evoluto, essendo il server un elaboratore appartenente alla fascia dei minimedi, le funzionalità offerte sono molto spinte verso la "multifunzionalità e le comunicazioni". Ciò significa che oltre ad avere una gestione dipartimentale degli archivi, con possibilità di scambio dati con l'*host,* con la periferia (PC) e gli altri dipartimenti, c'è una grossa possibilità di poter utilizzare programmi applicativi verticali.

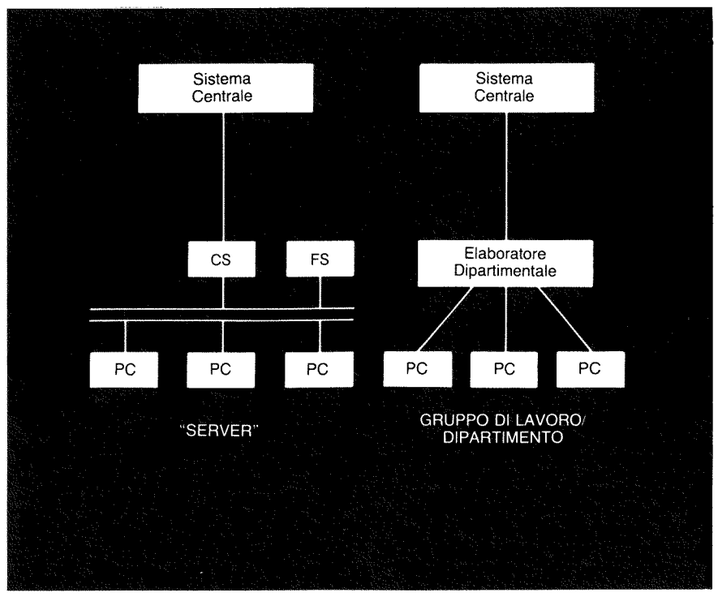
Inoltre, essendo più evolute anche le funzionalità di comunicazione, vi è la possibilità che l'utente del dipartimento possa scambiare "documenti" non solo con altri utenti della sua azienda, mediante un servizio di posta elettronica, ma anche con utenti di altre aziende che utilizzano il DISOSS, attraverso una rete con architettura SNA. Infatti esso, attualmente, è l'unico insieme di programmi che permette di immettere, formalizzare, distribuire, visualizzare e stampare i "documenti" in modo trasparente rispetto agli strumenti fisici che possono essere disomogenei.

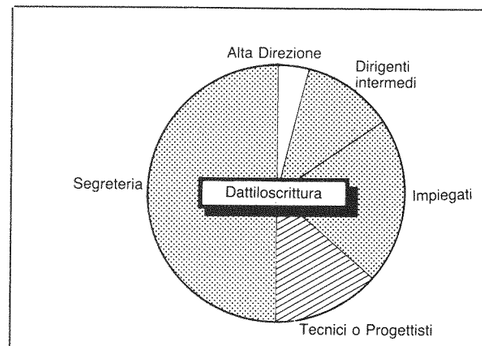
Tali caratteristiche fanno si che i sistemi dipartimentali evoluti abbiano un ruolo chiave nell'automazione degli uffici di una azienda, perché come si può dedurre dai grafici di fig. (6/7) l'insieme dato da: informatica individuale, gestione, composizione e scambio documenti rappresenta il denominatore comune di tutti gli uffici, sia dal punto di vista funzionale che per figure professionali.

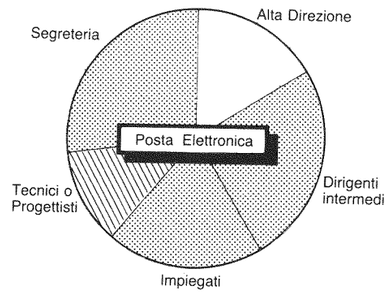
4 - Conclusioni

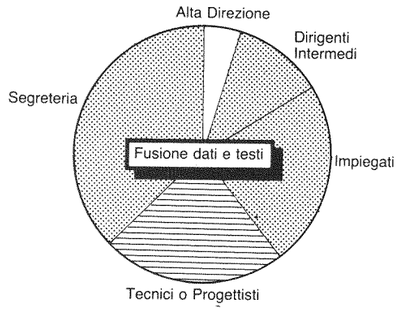
Secondo il modello di Nolan, con il quale vengono messi in relazione l'evoluzione nel tempo del budget informatico di un'azienda e numerose condizioni al contorno (es.: tipo di applicazioni, di pianificazione, controllo e organizzazione dell'informatica aziendale, figure professionali informatiche e ruolo della

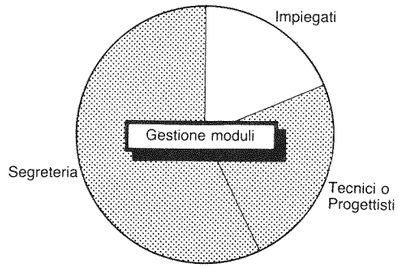
Fig. 5 - Possibili configurazioni di un sistema dipartimentale (CS = server per comunicazioni; FS = server per l'archiviazione; PC = personal computer).

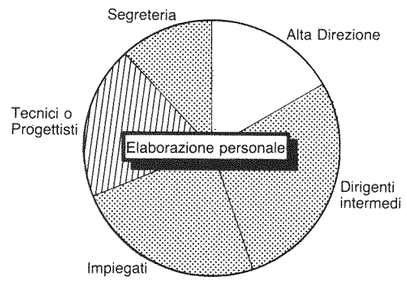










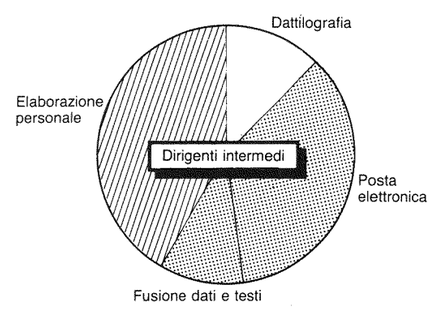


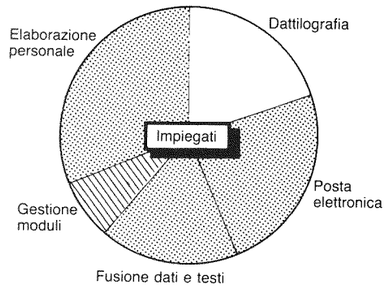
FUNZIONI	FIGURE PROFESSIONALI					
	Alta Dir.	Dir. Int.	Impieg.	Tecn. Prog.	Segret.	Tot.
DATTILOSCRITT.	3.8	11.5	21.2	13.5	50.0	100
POSTA ELETTRONICA	16.4	25.4	19.4	11.9	26.9	100
FUSIONE DATI E TESTI	4.7	11.6	23.2	23.3	37.2	100
GESTIONE MODULI	0.0	0.0	19.0	23.8	57.1	100
ELABORAZIONE PERSONALE	16.9	28.2	23.9	19.7	11.3	100

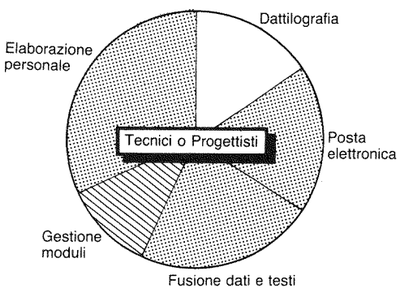
Fig. 6 - Analisi per funzione degli utenti potenziali di automazione d'ufficio (Fonte: Nomos).

utenza), si arriva alla conclusione che il budget informatico si è evoluto e dovrebbe continuare ad evolvere nel tempo secondo una curva (fig. 8) corrispondente ai cicli evolutivi di apprendimento e assimilazione, da parte dell'azienda, della "tecnologia dell'elaboratore e delle risorse dati".

Ognuno di questi cicli passa attraverso quattro stadi: iniziale, di proliferazione, di controllo e di maturità. Ai benefici che derivano inizialmente dalla riduzione dei costi di produzione (anche amministrativi),







FUNZIONI	FIGURE PROFESSIONALI				
	Alta Dir.	Dir. Int.	Impieg.	Tecn. Pr.	Segret.
DATTILOSCRITT.	7.4	12.5	20.0	15.9	32.5
POSTA ELETTRONICA	40.7	35.4	23.6	18.2	22.5
FUSIONE DATI E TESTI	7.4	10.4	18.2	22.7	20.0
GESTIONE MODULI	0.0	0.0	7.3	11.4	15.0
ELABORAZIONE PERSONALE	44.4	41.7	30.9	31.8	10.0
TOT.	100	100	100	100	100

Fig. 7 - Analisi per figure professionali degli utenti potenziali di automazione d'ufficio (Fonte: Nomos).

dovuti soprattutto ad economie di lavoro, si aggiungono quelli derivanti dalla riduzione dei costi di "transazione"(*).

Sintetizzando, dall'analisi fatta emerge che, oltre all'espansione dei sistemi di informatica tradizionale, sono in corso e sono previsti grossi investimenti in sistemi d'ufficio e in telematica poiché ci troviamo nel

(*) Per transazione si intende ogni azione che ha per oggetto l'ingresso o l'uscita di un elemento dei flussi delle risorse che caratterizzano un'azienda: *flusso prodotti / flusso struttura / flusso personale / flusso monetario.*

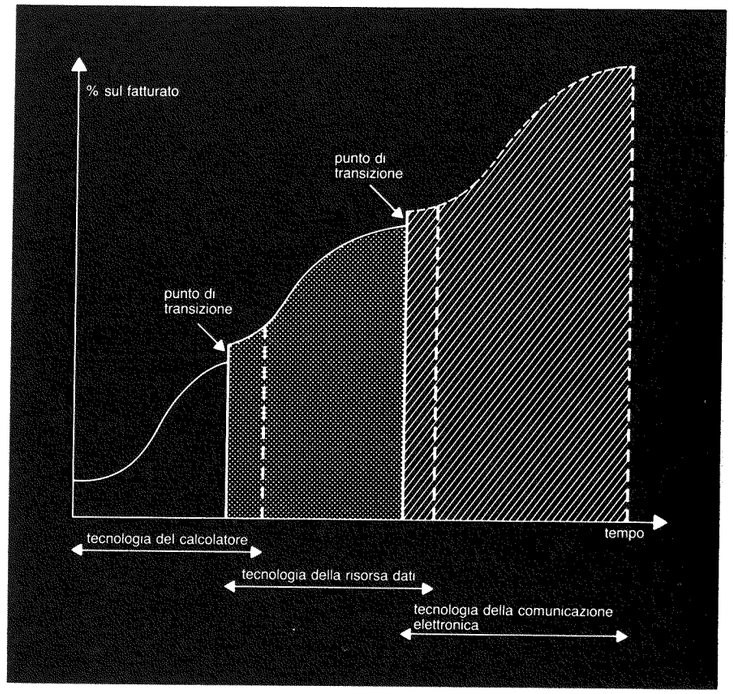


Fig. 8 - La curva di Nolan.

terzo ciclo della curva di Nolan, che corrisponde alla possibilità di gestire le transazioni, specie interaziendali, riducendone i costi e permettendone la moltiplicazione.

La situazione in Italia, secondo i dati riportati in un recente studio [9], è in crescita per quanto riguarda il software e i servizi, soprattutto quelli "transazionali", che riguardano cioè la tecnologia della comunicazione e dell'integrazione. Inoltre data la diversa composizione dell'utenza di sistemi general-purpose si nota che è possibile un incremento notevole nel settore terziario, non ancora molto automatizzato e nel quale l'automazione coinvolgerà soprattutto il lavoro d'ufficio.

Prendendo in considerazione anche l'incremento degli investimenti in percentuale tra il 1984 e il 1985 il dato che emerge è che oltre all'espansione dei sistemi di informatica tradizionale sono in corso e sono previsti grossi investimenti in sistemi d'ufficio e in telematica.

Per concludere si può affermare che il settore dell'automazione d'ufficio dà spazio alla combinazione di tutte le tecnologie più avanzate, ma le soluzioni vincenti sono quelle sottese da un'analisi accurata dell'organizzazione delle funzioni da automatizzare, in modo da realizzare una reale coerenza tra l'utilizzo di strumenti elettronici e l'esercizio dei ruoli funzionali all'interno di un'azienda.

5 - Bibliografia

[1] STRASSMAN P.A., *Overview of strategic aspects of information management,* Office, technology and people, marzo 1982.

[2] NOLAN R.L., *Managing the crises in data processing,* Harvard Business Review, dic. 1979.

[3] BELL D., *The coming of post industrial society,* Basic Books, N.Y., 1973.

[4] BARRAS D., BUTERA F., CAPRANICO S., *I modelli d'impresa,* 1986.

[5] GARTNER GROUP, *Small computer systems,* 1985.

[6] GALBRAITH J., *Designing complex organization,* Addison-Wesley, Reading Mass, 1973.

[7] MAGGIOLINI, *Costi e benefici di un sistema informativo,* Etass Libri, 1981.

[8] CONRATH D.W. et al, *The electronic office and organizational behavior-measuring office activities,* Computer networks, 1981.

[9] *Italia Informatica,* a cura della Honeywell Bull, Ediz. Sole/24 Ore, 1986.

Dai dati ai documenti multimedia

LEONARDO FELICIAN
TIZIANA TIZIANEL
Università degli Studi
Dip. di Matematica e Informatica
Udine

1. Introduzione

La storia dell'informatica può essere vista come la sua progressiva estensione verso il trattamento di categorie sempre nuove di informazioni. Inizialmente, negli anni '50, solo dati puramente numerici venivano considerati adatti all'elaborazione automatica e la macchina era realmente un "calcolatore", nel senso che le operazioni effettuate erano quasi esclusivamente di tipo aritmetico.

Con le prime elaborazioni in campo gestionale, negli anni '60, ci si rivolge al trattamento dei dati non numerici, codificati o a struttura fissa, e gli aspetti di manipolazione non aritmetica (memorizzazione, ricerca, riformattazione, comunicazione) divengono importanti e spesso predominanti.

A metà degli anni '70 si iniziano a trattare anche informazioni testuali, cioè stringhe di caratteri prive di struttura fissa, con elaboratori più o meno specializzati, in forma interattiva. Applicazioni di questo tipo vanno sotto il nome di word-processing.

Negli anni '80 si assiste al tentativo di trattare mediante elaboratore una nuova categoria di informazioni, dette "immagini", la cui rappresentazione interna, o codifica, non dipende dal "significato", ma dal "significante", che consiste nei segni contenuti su un foglio di carta.

Un'altra capacità recentemente acquisita da alcuni elaboratori è quella di trattare informazioni di tipo vocale, con possibilità di produrre in modo automatico messaggi vocali e di riconoscere frasi, o più semplicemente di memorizzare in forma digitale e di riprodurre successivamente non solo voci, ma anche musica.

Si può dunque affermare che gli odierni elaboratori sono in grado di ricevere, produrre, elaborare, memorizzare e trasmettere informazioni che riflettono ogni tipo di comunicazione umana, dal testo scritto all'immagine e al messaggio vocale.

Questa progressiva estensione verso il trattamento di categorie sempre nuove di informazioni, che riflette a ritroso l'evolversi dei mezzi di comunicazione umana dalla preistoria ad oggi (dalla comunicazione puramente verbale alla comunicazione per immagini - pitture, ideogrammi - fino alla comunicazione a mezzo stampa), è stata originata innanzitutto dalla percezione di bisogni applicativi sempre più sofisticati.

Ad esempio la creazione dei primi strumenti di word processing è dovuta al problema di produrre testi lunghi a volte migliaia di pagine (documentazione tecnica, manuali, etc.), sottoposti a varie correzioni in sede di redazione iniziale e a successive riedizioni periodiche con stringenti esigenze di tempestività. Da qui il loro impiego si è esteso anche in ambienti caratterizzati da esigenze meno esasperate, man mano che il costo diveniva più accessibile.

Lo studio dei primi lettori ottici e dei sistemi in grado di gestire le immagini è stato originato dalle necessità di due ambienti distinti: le biblioteche, per le esigenze di conservazione e accessibilità del materiale librario che si possono soddisfare con l'impiego dell'elaboratore, e gli uffici di aziende del terziario avanzato, interessate non solo ad una efficiente gestione degli archivi (finora puramente cartacei), ma soprattutto al trattamento automatico ed integrato di documenti misti, composti da testi, immagini e dati.

Anche l'impiego dell'elaboratore e delle sue risorse per la memorizzazione e la gestione di immagini audio è stato determinato dalle esigenze degli ambienti d'ufficio (trasmissione automatica di messaggi vocali, in interazione con il telefono, oppure per la memorizzazione di parti vocali in documenti dalla struttura complessa) ed anche dalla necessità di archiviazione definitiva di brani musicali (sinfonie, concerti, etc.).

2. Categorie di informazioni

Come è noto, tutte le informazioni memorizzate e trasmesse all'interno degli elaboratori sono codificate come stringhe di zeri e uni; pertanto ogni informazione che si vuole trattare mediante elaboratore deve subire un processo di codifica che ne produce la rappresentazione interna (stringa di 0 e 1) comprensibile all'elaboratore. Da questa è possibile risalire all'informazione originale attuando un processo di decodifica *(fig. 1)*.

Le categorie di informazioni finora presentate (dati, testi, immagini e

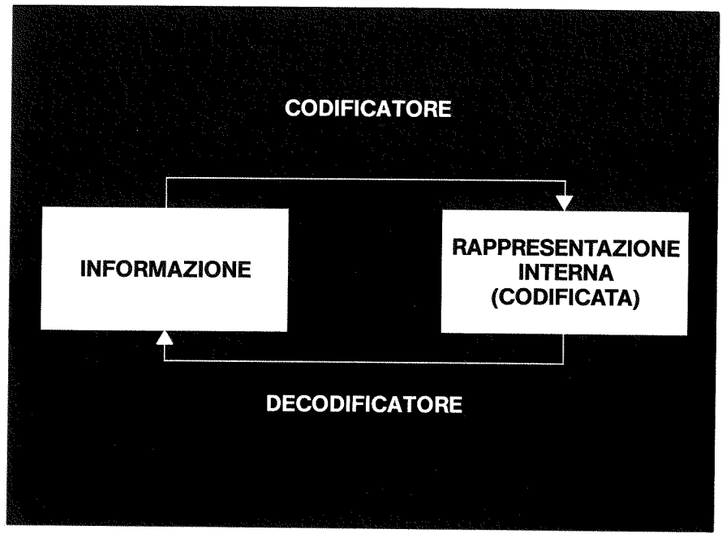


Fig. 1 - Processo di codifica-decodifica di ogni categoria di informazioni.

voce) si differenziano tra loro non tanto per il loro significato (è evidente che un'immagine è diversa da un messaggio vocale) quanto per il tipo di codificatore che ne produce la rappresentazione interna.

Quindi definire una categoria di informazioni significa definire le caratteristiche della rappresentazione interna, la classe degli algoritmi che genera tale rappresentazione (codificatori) e quella degli algoritmi in grado di interpretare la codifica (decodificatori).

DEFINIZIONE 1. Si definisce *dato* una sequenza di bit di lunghezza fissa tale che ad ogni configurazione ammissibile dei bit corrisponde uno dei valori assumibili dal dato.

La lunghezza della sequenza dipende dal tipo di dati a dai valori che questo può assumere e deve essere minimizzata.

I processi di codifica e decodifica possono avvenire mediante algoritmi oppure mediante tabelle. Ogni tipo di dato richiede un apposito algoritmo (o tabella) di codifica/decodifica.

DEFINIZIONE 2. Si definisce *testo* una stringa di byte di lunghezza variabile. Ogni byte della stringa contiene la codifica di un carattere alfanumerico (una lettera dell'alfabeto, una cifra decimale, un simbolo di punteggiatura oppure un carattere speciale).

La definizione mette in luce che ogni carattere alfanumerico viene codificato come dato (lungo 8 bit) e che la sequenza di byte che compone il testo non ha lunghezza prevedibile. Per quanto riguarda la codifica dei singoli caratteri alfanumerici, a seconda dell'elaborazione sono adottati usualmente il codice ASCII (American Standard Code for Information Interchange) o il codice EBCDIC (Extended Binary Coded Decimal Interchange Code). Con gli 8 bit che costituiscono un byte sono possibili 256 configurazioni e altrettanti caratteri alfanumerici diversi: pertanto i segni che non rientrano nelle configurazioni possibili non possono essere memorizzati in un testo.

DEFINIZIONE 3. Si definisce *immagine bit-map* la matrice M di bit (concatenamento di m sequenze di n bit ciascuna) avente m righe e n colonne tale che:
$M_{ij} = 1$ se il pixel in posizione i, j dell'immagine contenuta nel foglio di carta da memorizzare ha un tono di grigio che supera una soglia fissata (nero)
$M_{ij} = 0$ se altrimenti (bianco)

I valori della matrice, cioè la codifica bit-map dell'immagine sul foglio, sono ottenuti mediante la scansione operata da un lettore ottico (scanner). Lo scanner attua sia un processo di campionatura, suddividendo l'immagine in tanti piccoli rettangoli, detti pixel (contrazione di "picture element"), che di quantizzazione, stabilendo cioè se la luce riflessa dal pixel supera o no un valore soglia, ottenendo così quella che viene anche detta *forma canonica* dell'immagine bit-map.

È evidente che questo tipo di codifica permette di memorizzare qualsiasi segno, poichè l'informazione contenuta sul foglio di carta è trattata a livello di "significante" e non di "significato". È possibile pertanto codificare e memorizzare disegni, grafici, firme, illustrazioni, fotografie, manoscritti e naturalmente tutte le informazioni memorizzabili come testi che non si ritiene opportuno - tipicamente per motivi di tempo - introdurre nel sistema sotto tale forma codificata tramite digitazione.

Valori standard di risoluzione degli scanner, cioè di dimensione, orizzontale e verticale, del pixel sono 8 pixel/mm per la risoluzione orizzontale e 7.7 oppure 3.85 linee/mm per quella verticale. In base a questi valori una pagina di formato A4 (21x26 cm^2) viene divisa in 2002 oppure 1001 (dipende dalla risoluzione verticale scelta) righe ognuna delle quali contiene 1728 pixel. Dunque l'immagine bit-map determinata con questi valori richiede 3.459.456 oppure 1.729.728 bit, uno per ogni pixel, cioè circa 1.7 oppure 3.5 milioni di bit a seconda della risoluzione adottata.

Con la risoluzione più alta si avrebbe dunque un'occupazione per pagina di circa 420 Kbyte: si preferisce a volte parlare di K-ottetti, evidenziando in questo modo il fatto che il byte stesso non ha in questo caso un significato particolare. Per ridurre l'occupazione di memoria si può sfruttare la circostanza che solitamente le immagini hanno molti spazi completamente bianchi o neri, la cui rappresentazione può avvenire in forma abbreviata. Esistono diversi algoritmi di compattamento che lavorano in modo monodimensionale (per righe di punti, algoritmi di tipo Huffman modificato) o bidimensionale (per aree, ad esempio quelli che usano i "quadtree" [14]) capaci di raggiungere fattori di compressione di 10÷30 volte, secondo il tipo di documento e la sua densità di segni: si arriva quindi a occupazioni medie di 15÷30 Kbyte per pagina.

DEFINIZIONE 4. Si definisce *immagine compressa,* o semplicemente *immagine,* la stringa di bit ottenuta applicando un algoritmo di compressione all'immagine bit-map ottenuta dallo scanner.

Pertanto se le immagini bit-map (matrici di bit) possono considerarsi dati, secondo la definizione data, le immagini compresse, o immagini, a lunghezza variabile e ottenute nel modo illustrato, sono una nuova categoria di informazioni.

Infine si presenta l'ultima, ma non meno importante categoria di informazioni: i messaggi vocali. Come accennato nell'introduzione, ci sono due modi distinti di considerare la voce e i messaggi vocali: da un lato la voce è un'informazione da memorizzare e gestire in modo analogo alle altre categorie di informazioni, dall'altro è un mezzo, molto naturale, di comunicazione tra uomo e macchina. Si fa riferimento per ora a semplici messaggi vocali che è necessario archiviare all'interno di un documento.

DEFINIZIONE 5. Si definisce *messaggio vocale* la stringa di byte ottenuta mediante un convertitore analogico-digitale in grado di codificare la voce.

I convertitori, che trasformano il segnale analogico vocale in una stringa di byte, adottano un meccanismo di campionatura e quantizzazione del segnale acustico simile a quello attuato dagli scanner per le immagini. Tipicamente il processo di campionatura adotta 20.000 campioni al secondo e l'informazione contenuta in un campione viene codificata con 2 byte [9]; pertanto la memorizzazione di 1 secondo di voce richiede circa 40 Kbyte e la quantità di memoria richiesta da un breve messaggio vocale di durata 1 minuto si aggira sui 2 Mbyte.

3. I sistemi di gestione specializzati

Sinora sono stati sviluppati sistemi specializzati nella gestione delle singole categorie di informazioni presentate: i Data Base Management System per i dati, gli Word-Processor per i testi, i sistemi di archiviazione per le immagini e i sistemi di gestione audio per le voci. I primi tre tipi di sistemi sono di seguito brevemente descritti.

3.1 Dati

È noto che il funzionamento di un'azienda, di un ente pubblico o, più in generale, di un'organizzazione, è condizionato dall'accuratezza delle informazioni di cui dispone e dalla tempestività con cui vengono elaborate. Si riconosce altresì che le informazioni - ridotte a dati mediante un processo di quantizzazione e formalizzazione - costituiscono una risorsa patrimoniale dell'azienda, con un costo ed un valore, da memorizzare e gestire nel modo più efficiente possibile.

I sistemi informatici sviluppati qualche anno fa si basavano sull'unico strumento generalmente offerto dai sistemi di elaborazione sin dalla metà degli anni '60: un linguaggio di programmazione per la definizione e l'uso di archivi permanenti di dati. Il problema principale di questo approccio sta nella completa responsabilità da parte del progettista nell'organizzare e gestire i dati.

In particolare è necessario sviluppare tutto il software per la raccolta, verifica, archiviazione e recupero dei dati, e cio può rappresentare anche il 40÷60% di un programma applicativo [2]. Sistemi informatici sviluppati in questo modo creano solitamente molti problemi: difficoltà nello sviluppo efficiente del software, ridondanza dei dati e quindi - molto spesso - inconsistenza, mancanza di sicurezza, etc.

Per superare questi limiti, sin dalla fine degli anni '60 è andata affermandosi la tecnologia dei Data Base (DB, Basi di Dati) e dei Data Base Management System (DBMS, Sistemi per la Gestione delle Basi di Dati).

DEFINIZIONE 6. Un *Data Base* è un insieme di dati strutturati e permanenti, raggruppati in insiemi omogenei in relazione tra loro, organizzati con la minima ridondanza per essere usati da applicazioni diverse in modo controllato [2].

DEFINIZIONE 7. Un *Data Base Management System* è un insieme strutturato di programmi che forniscono un metodo comune e controllato di accesso e modifica al DB, in modo da proteggerne la privatezza e l'integrità.

I DBMS garantiscono alcune proprietà di indipendenza fisica e logica:

- Indipendenza delle applicazioni dall'organizzazione fisica dei dati: non occorre modificare i programmi applicativi se cambia parte dell'organizzazione fisica dei dati, ad esempio se si vogliono variare le strutture dati - alberi, etc. - che agevolano il reperimento dei dati, oppure il modo in cui essi sono memorizzati, oppure i supporti di memorizzazione, le tecniche di compattamento, etc.
- Indipendenza delle applicazioni dall'organizzazione logica dei dati: non occorre modificare i programmi applicativi in seguito a variazioni dello schema logico.

Per questo i DBMS offrono tre livelli distinti di descrizione dei dati: *schema logico, interno* ed *esterno.* Lo schema logico, o concettuale, è la descrizione della struttura del DB senza nessun riferimento alla sua organizzazione fisica o al modo in cui vengono memorizzati i dati nelle memorie di massa. Lo schema interno è la descrizione di come sono or-

ganizzati i dati nelle memorie secondarie, e delle strutture dati ausiliarie previste per facilitarne l'uso. Lo schema esterno, infine, è una descrizione di come appare la struttura del DB ad una certa applicazione (o ad un certo utente). Uno schema esterno viene anche detto "vista utente", per evidenziare il fatto che esso si riferisce a ciò che l'utente immagina che sia il DB.

3.2 Testi

Per quanto riguarda i testi, la meccanizzazione del loro trattamento prende l'avvio dai programmi di editing realizzati sugli elaboratori interattivi, che permettono all'utente di correggere errori ed apportare modifiche al programma che si sta immettendo nella macchina. La funzione di editing, introdotta dapprima come supporto alla realizzazione di programmi ha poi gradualmente introdotto una certa automazione anche in altri campi, come quello della produzione di testi, fino ad allora realizzata mediante macchine per scrivere.

Attualmente i sistemi chiamati Word-Processor (WP) sono dei micro o minielaboratori che controllano una stampante, il video, la tastiera e un'unità di memoria e permettono la preparazione, la correzione, la riproduzione, l'archiviazione, il reperimento e la trasmissione dei testi.

Si distinguono due tipi di configurazioni: quelle autonome, in cui ogni macchina è completa di tutte le sue parti (ad esempio i sistemi basati su personal computer), e quelle condivise, che prevedono la condivisione di elaboratore e supporti di memoria. I sistemi di trattamento dei testi consentono in modo sempre più agevole le operazioni di editing e di riedizione dei testi, permettendo la preparazione e la stampa di testi dalle caratteristiche tipografiche sempre più sofisticate, comprendenti caratteri appartenenti a alfabeti diversi, simboli speciali - matematici, etc. (TEX, ETRUDE, etc.).

3.3 Immagini

Si trovano da qualche anno sul mercato sistemi specializzati in grado di leggere e archiviare le immagini. La configurazione hardware tipica di questi sistemi comprende:

- una stazione di ingresso delle immagini, composta da uno, o più, scanner e terminali grafici per il controllo delle immagini inserite
- un minicomputer e unità di memoria a disco (solitamente con dischi ottici)
- una stazione di uscita delle immagini, sia in forma "soft", cioè su video, mediante schermo grafico ad alta risoluzione o work-station, che "hard", cioè un supporto cartaceo, mediante stampante laser.

Il software di gestione permette di inserire, archiviare, ricercare, stampare e cancellare unità base di elaborazione dette documenti [6]. Questi risultano spesso composti da contenuto e descrittore: il contenuto consiste di una sequenza ordinata di immagini mentre l'intestazione consiste di un insieme di attributi di ricerca (cfr. fig. 2).

Questi ultimi riassumono le caratteristiche del documento utili in fase di ricerca, quali il nome dell'autore, la data di creazione, etc., e vengono definiti all'atto dell'inserimento del contenuto.

Le aree di applicazione dei sistemi di gestione delle immagini sono molteplici. Si pensi prima di tutto alle applicazioni in ambito bibliotecario. Fino a poco tempo fa erano considerate efficienti le biblioteche in grado di fornire agli utenti un servizio informatizzato di information retrieval. Un opportuno DB contiene le descrizioni, più o meno dettagliate, dei libri e delle riviste (nei casi migliori dei singoli articoli componenti le riviste) compresi nella biblioteca. L'utente, tramite il linguaggio di manipolazione dati supportato dal DBMS precisa le caratteristiche del materiale richiesto. Il risultato del processo di interrogazione e ricerca nel DB consiste in un elenco di indicazioni sulla presenza e collocazione dei libri e delle riviste richiesti [11, 15].

In questo modo il DBMS è un sistema che gestisce i descrittori dei dati che sono oggetto di archiviazione e ricerca in una biblioteca, e cioè libri e riviste, e non i dati stessi [1]. Il vantaggio fornito da questo metodo di gestione è essenzialmente l'agevolazione delle operazioni di ricerca; gli svantaggi sono tutti collegati al fatto che gli oggetti da gestire restano fuori dal DB.

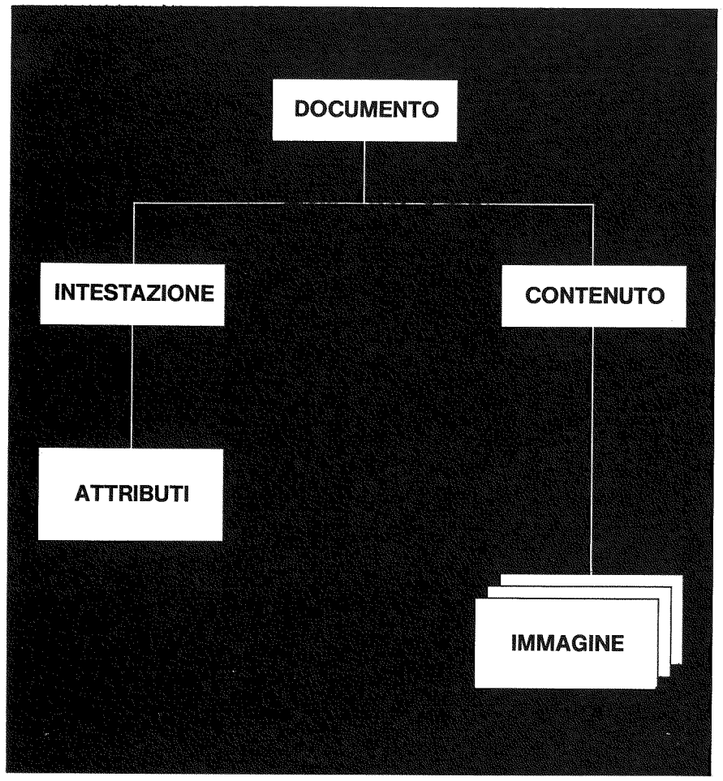
Un primo problema è la non disponibilità in ogni momento di riviste e libri di cui viene fatta richiesta molto spesso. Inoltre libri antichi e preziosi non vengono mai messi a disposizione dei ricercatori perchè potrebbero venir danneggiati (problema di disponibilità e di conservazione del materiale librario). Un secondo problema è l'efficienza del servizio: non è comodo nè veloce, sia per il personale della biblioteca sia per l'utente, eseguire le operazioni di consultazione e di prestito.

Ad entrambi i problemi si rimedia mantenendo nel sistema non solo i descrittori dei testi, ma i testi stessi, inserendoli pagina per pagina mediante uno scanner; in altre parole, utilizzando un sistema che consenta di gestire documenti dalla struttura appena illustrata. Il servizio di consultazione si traduce in una prima fase di selezione del materiale e in un successivo esame dello stesso tramite terminale grafico, in grado di visualizzare la pagina così come letta dallo scanner. L'utente dopo aver "sfogliato" gli articoli scelti può ottenere una stampa, tramite stampante grafica, delle pagine che desidera consultare più a fondo [7, 8, 12].

Un'altra importante area applicativa è costituita dagli uffici delle aziende del terziario avanzato, interessate ad automatizzare il flusso cartaceo dei documenti, essenzialmente composti da immagini, prodotti al di fuori dell'ufficio. Sono indispensabili infatti, per il normale svolgimento del lavoro d'ufficio, l'archiviazione e la consultazione di lettere, circolari, ordini, fatture, ricevute ed altri documenti cartacei prodotti all'esterno dell'ufficio.

La grande quantità di documenti di questo tipo - si arriva facilmente alle centinaia di migliaia di fogli - ne rallenta e complica la gestione con i tradizionali archivi cartacei. Si pensi solamente ai problemi di utilizzo: i tempi di reperimento dei documenti sono a volte dell'ordine delle ore, ed è indispensabile la corretta riarchiviazione del documento accedu-

Fig. 2 - Struttura dei documenti trattati da molti sistemi di gestione delle immagini. Il documento è composto da una sequenza ordinata di immagini, spesso in corrispondenza biunivoca con le pagine componenti il documento cartaceo da memorizzare, e da un insieme di attributi che ne riassume le caratteristiche.



to. Non è inoltre possibile la lettura da parte di più persone contemporaneamente, può succedere che l'ultimo utilizzatore dimentichi di riarchiviare il documento, etc.

L'importanza della consultazione dei documenti per lo svolgimento dell'attività lavorativa spiega dunque il successo dei sistemi di gestione delle immagini attualmente in commercio. Tali sistemi si rivelano utilissimi anche nelle applicazioni dette "di archiviazione definitiva", in cui cioè l'esigenza principale è quella di memorizzare informazioni che non devono essere modificate. Tipicamente si tratta di documenti di tipo legale, militare, bancario, assicurativo, etc.

4. I documenti multimedia

Sino ad ora sono stati proposti sistemi volti alla gestione specifica delle categorie di informazione presentate: si sono illustrati i DBMS per i dati, i WP per i testi e i sistemi per l'archiviazione di immagini, e sono già in commercio sistemi di gestione audio. La frontiera odierna consiste nel progetto e nella realizzazione di sistemi più complessi, in grado di gestire in modo integrato oggetti di tipo misto, cioè composti da dati, testi, immagini e messaggi vocali, mescolati tra loro in vario modo.

I termini correntemente usati nella letteratura per indicare oggetti di questo tipo sono *multimedia document* oppure *multimedia object* [13, 16, 17].

> DEFINIZIONE 8. Si definisce ***documento multimedia*** l'unità elementare contenente informazione costituita dalle componenti dati, testo, immagine e voce.

La definizione mette in luce che l'oggetto da gestire è un'unità di per sè indivisibile, ma composta da più componenti profondamente diverse tra loro. I documenti multimedia utilizzati e in fase di studio si differenziano tra loro per la struttura, definita come l'insieme delle componenti del documento e delle loro interrelazioni.

Generalmente si propone la gestione di documenti multimedia dalla struttura articolata e complessa, composti da dati, testi, immagini e messaggi vocali, mescolati tra loro in modo quasi arbitrario.

Il progetto e lo sviluppo di sistemi di gestione di documenti multimedia presenta vari problemi: è necessario innanzitutto stabilire come deve essere l'architettura software del sistema, quali metodi di accesso e quali strategie occorre adottare per ottenere tempi accettabili nella trattazione dei documenti, quali supporti di memorizzazione conviene utilizzare, e infine come definire l'interfaccia utente per l'uso degli attuali sofisticati dispositivi di I/O [4]. Comunque, le molteplici applicazioni sono tali da giustificare l'impegno di ricercatori e produttori di sistemi commerciali in questo campo.

Le applicazioni possibili costituiscono un sovrainsieme di quelle dei si-

stemi specializzati sinora presentati. Infatti gli oggetti gestiti da tali sistemi sono solitamente *semplificazioni* dei documenti multimedia e questi ultimi hanno una struttura abbastanza complessa da suggerire e consentire nuove ed interessanti applicazioni.

Per fornire un esempio delle potenzialità e del funzionamento dei sistemi di gestione di documenti multimedia si descrive nelle sue linee essenziali il MINOS, un sistema informativo multimedia attualmente in fase di implementazione [5]. L'obiettivo dei progettisti è di fornire un sistema che consenta la creazione e l'interrogazione di documenti multimedia per mezzo di una potente work-station, dotata di quanto può allargare la banda di comunicazione tra uomo e macchina (schermo grafico, dispositivi di I/O della voce, mouse, etc.).

I documenti gestiti dal MINOS, detti multimedia objects, sono logicamente composti da un identificatore unico e da una sequenza di pagine. Queste sono o tutte pagine visuali o tutte pagine audio. Per pagina visuale si intende ciò che appare sullo schermo della workstation nello stesso istante; per pagina audio si intende una parte della componente vocale del documento di durata temporale definita. È prevista la possibilità di esplicitare le relazioni logicamente intercorrenti tra documenti, in modo da agevolare la lettura dei documenti relativi ad un certo argomento.

La pagina visuale comprende una mescolanza di testi e di immagini (cfr. fig. 3). Possono comparire simboli speciali che rivelano la presenza di messaggi vocali, ascoltabili a discrezione dell'utente (mediante posizionamento del mouse). Tra i tipi di pagine visuali speciali permesse, si ricordano le trasparenze e le sovrascritte. Le trasparenze vengono mostrate sullo schermo senza che le loro parti bianche coprano quanto già presente sul video (cfr. fig. 4 per un esempio). Le sovrascritte sono pagine contenenti immagini aventi la proprietà che quando devono lasciare il posto, sullo schermo, alla successiva pagina visuale, le loro immagini "restano sul video", cioè non vengono sostituite da quanto eventualmente contenuto, in loro corrispondenza, alla pagina successiva.

Durante l'ascolto delle pagine audio l'utente può vedere comparire sullo schermo alcuni messaggi visivi (lunghi al massimo una pagina). Si stanno attualmente studiando primitive che permettano all'utente di muoversi all'interno delle pagine audio in modo analogo a ciò che si può fare con le pagine visuali (andare indietro o avanti di una parola, di un paragrafo, di una pagina, etc.)

Il MINOS adotta, per la ricerca dei documenti, l'originale metodo denominato *filtering and browsing* [16, 5]. Presupposto fondamentale per l'applicazione di questo metodo è che da ogni documento si possano estrarre in modo automatico delle informazioni facilmente riconoscibili dall'utente: ad esempio le microimmagini (*miniatures*) e i riassunti vocali (*fast talks*). Le microimmagini sono astrazioni visuali delle pagine del documento ottenute riducendo in scala la pagina e rappresentandone in scala il contenuto. Ogni parola di testo viene visualizzata come una linea più o meno lunga e spessa posizionata in modo da riflettere la collocazione della parola nella pagina. Per le immagini si applica un procedimento analogo a quello adottato per i testi.

I riassunti sonori sono invece brevi messaggi (1÷2 minuti) che riassumono il contenuto della parte vocale del documento. Attualmente non sono ottenuti in modo automatico. Le informazioni estratte sono utilizzate durante la fase di ricerca, che prevede l'esecuzione di più cicli di "filtering and browsing". L'utente

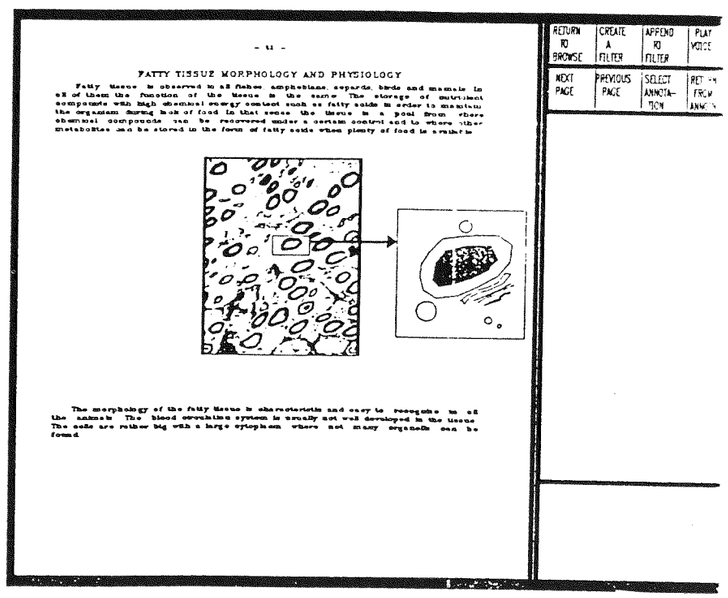
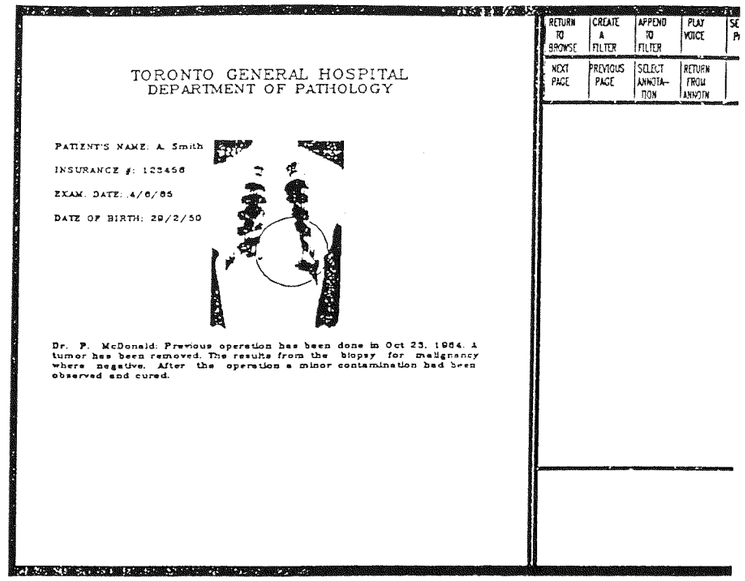
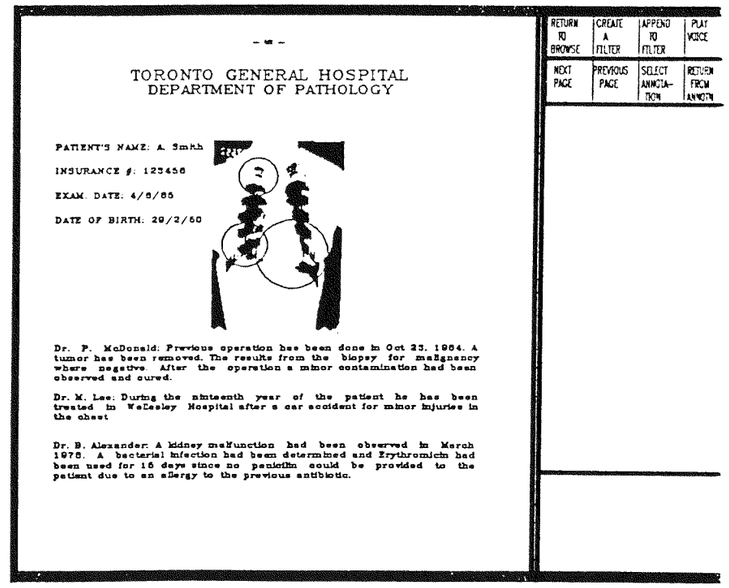


Fig. 3 - Un esempio di pagina visuale facente parte di un documento gestito dal Minos, un sistema informativo multimedia attualmente in fase di realizzazione. La pagina visuale contiene informazioni di tipo diverso: testi, immagini bit-map e grafici (da [5]).

Fig. 4 - Una applicazione dell'uso delle trasparenze in un documento Minos di tipo medico. Le trasparenze sono pagine visuali che vengono mostrate sullo schermo senza che le loro parti bianche coprano quanto già presente sul video: in questo caso le trasparenze contengono uno o più cerchi, che evidenziano zone della radiografia (immagine bit-map) sottostante, e una parte testo (da [5]).





specifica delle condizioni logiche che devono essere soddisfatte dai documenti voluti (ad esempio, condizioni su certi attributi dei documenti oppure più generiche, come: "il documento contiene delle parti vocali"). Tali condizioni costituiscono il filtro in base a cui avviene la prima fase della selezione (fase "filtering"). Tutti i documenti soddisfacenti le condizioni vengono mostrati all'utente nella loro forma di microimmagine e riassunto vocale. In base a ciò che vede e sente l'utente decide se il messaggio lo interessa, e quindi lo seleziona (fase di "browsing") per farselo mostrare. L'utente può poi riprendere il ciclo modificando il filtro in modo da selezionare più o meno documenti, in base alle informazioni ricavate precedentemente. In questo modo si arriva gradualmente alla selezione dei documenti voluti. L'idea che sta alla base di questo metodo di ricerca è efficacemente riassumibile con un paragone che illustra il significato del termine "browsing" [10]. Si immagini un cliente che entra in un grande magazzino alla ricerca di un determinato oggetto. Il modo più efficiente di trovare l'oggetto è di chiederne la collocazione ad un commesso e

quindi di dirigersi allo scaffale giusto. Può darsi tuttavia che il cliente non sia in grado di descrivere l'oggetto, oppure non ne sappia con precisione il nome, pur avendo chiaro di che cosa si tratta, oppure può darsi che non sia presente il commesso.

In questi casi il cliente deve applicare una tecnica di ricerca differente, usualmente detta "browsing". Questa si traduce nel camminare tra gli scaffali correggendo la propria direzione e velocità in base agli oggetti che si vedono e alla loro somiglianza con quello cercato (*"navigation"*), oppure nel fare dei tentativi dirigendosi direttamente verso uno scaffale dove si spera di trovare l'oggetto (*"probing"*). La ricerca con l'aiuto del commesso, quando applicabile, è sicuramente più veloce, tuttavia in alcune situazioni il "browsing" è la sola soluzione possibile. Il "browsing" viene accoppiato al "filtering", che permette di ridurre, sin dal principio oppure dopo una prima visita (browsing), le zone del grande magazzino in cui va effettuata la ricerca. Più cicli di "filtering e browsing" portano al ritrovamento dell'oggetto voluto.

Gli ambienti applicativi in cui si può inserire il MINOS sono i più vari. Il MINOS può infatti gestire tutti i documenti usualmente utilizzati negli uffici, documenti di tipo medico, immagini create con l'ausilio dell'elaboratore (CAD/CAM), informazioni turistiche, storiche, geografiche, permettendone - grazie alle numerose facilities di gestione - un facile accesso ed una gradevole interrogazione. Si pensi, ad esempio, all'utilizzo del MINOS, o di un sistema dalle analoghe caratteristiche, in ambiente d'ufficio. Non solo si possono compiere le operazioni di inserimento, archiviazione e ricerca di documenti composti da sole immagini, ma si possono creare ed utilizzare documenti multimedia molto più sofisticati.

L'uso di trasparenze e messaggi vocali consente, ad esempio, la creazione di documenti la cui presentazione agli utenti riproduce quanto questi vedrebbero e sentirebbero in presenza di una persona che espone oralmente un certo argomento, avvalendosi di trasparenze al fine di correlare le informazioni. Sul video compaiono le immagini, che eventualmente si sovrappongono l'una all'altra, e contemporaneamente, mediante dispositivi di output della voce, si ascoltano i messaggi vocali relativi, etc.

È indiscutibile l'efficacia nel trasmettere e presentare le informazioni di documenti multimedia di questo tipo; ed è proprio a tali documenti e a tali tecniche di comunicazione (ascolto di un messaggio e contemporanea visione di un'immagine, sovrapposizione di immagini, selezione di opzioni per mezzo del mouse, etc.) che si fa riferimento quando si parla di ampliamento della banda di comunicazione tra uomo e macchina.

Bibliografia

[1] Agosti M., *"Nuove architetture di sistemi per la documentazione automatica"*, Rivista di Informatica, Vol. 14, N° 4, Ottobre-Dicembre 1984.

[2] Albano A. e Orsini R., *"Basi di dati"*, Boringhieri, Torino, 1985.

[3] Bracchi G., *"Automazione dell'ufficio - Aspetti tecnici e organizzativi"*, Supplemento a Sistemi e Automazione, N° 226 Maggio 1982.

[4] Christodoulakis S., *Multimedia Data Base Management: Applications and Problems"*, Proceedings of ACM-SIGMOD International Conference on Management of Data, Austin, Texas, may 1985.

[5] Christodoulakis S., Ho F. e Theodoridou M., *"The Multimedia Object Presentation Manager of MINOS: a Symmetric Approach"*, Proceedings of ACM-SIGMOD International Conference on Management of Data, May 1986, Washington D.C.

[6] Felician L., *"Valutazioni hardware e software per la gestione di archivi immagini di grandi dimensioni"*, Rivista di Informatica, Vol. 16, N° 4, 1986.

[7] Harding J.R. e Nuget W.R., *"Library Application of Optical Storage"*, Encyclopedia of Library and Information Science, Vol. 38, Suppl. 3, 1985, pp. 242-266.

[8] Manns B. e Swora T., *"Books to bits: Digital Imaging at the Library of Congress"*, Journal of Information & Image Management, October 1986.

[9] Martelli A. e Volpi G., *"Comunicare a voce con le macchine"*, Note di Informatica, N° 13, Settembre 1986.

[10] Motro A., *"Browsing in a Loosely Structured Data Base"*, Proceedings of ACM-SIGMOD Annual Meeting, Boston, MA, June 1984.

[11] Nugent W.R. e Harding J.R., *"Optical Storage of Page Images and Pictorial Data - Opportunities and Needed Advances in Information Retrieval"*, Proceedings of the ACM Annual Meeting, October 1983, Washington D.C.

[12] Nugent W., *"Optical Disc Technology and Libraries"*, IFLA Journal, Vol. 12, N° 3, 1986.

[13] Papa M. P., Christodoulakis S. e Theodoridou M., *"Multimedia Document Presentation and Communication"*, Atti del Congresso Annuale dell'A.I.C.A., 1985.

[14] Samet H., *"Region Representation: Quadtrees from Binary Arrays"*, Computer Graphics and Image Processing, Vol. 13, 1980.

[15] Thoma G.R., Suthasinekul S., Walker F.L., Cookson J. e Rashidian M., *"A. Prototype System for the Electronic Storage and Retrieval of Document Images"*, ACM Transactions on Office Information Systems Vol. 3, N° 3 July 1985.

[16] Tsichritzis D., Christodoulakis S., Economopoulos P., Faloutsos C., Lee A., Lee D., Vandenbroek J. e Woo C., *"A Multimedia Office Filing System"*, Proceeding of 9th Conference on Very Large Data Bases, Florence, Oct-Nov. 1983.

[17] Woelk D., Kim W. e Luther W., *"An Object-Oriented Approach to Multimedia Databases"*, Proceedings of ACM-SIGMOD International Conference on Management of Data, May 1986, Washington D.C.

Verso un linguaggio per la descrizione dei movimenti umani

THOMAS W. CALVERT
Fraser University
Vancouver (Canada)

1. Introduzione

L'obiettivo che ci prefiggiamo con il nostro lavoro è un linguaggio capace di specificare ed animare i movimenti umani, che possa essere interpretato sia dal computer che dall'uomo e che, oltre ad applicarsi ai movimenti in generale, risponda ai particolari requisiti della danza, del cinema e del teatro. Intrinsecamente di tipo interdisciplinare, il nostro lavoro attinge ampiamente dalla linguistica, computer science e fisica, oltre che dalla danza, cinema e teatro.

In questo articolo ci si limiterà alla presentazione dei fondamenti del linguaggio ed all'esame di alcuni primi risultati. Il raggiungimento degli obiettivi finali presuppone un ulteriore ampio sviluppo teorico e sperimentale.

Non esiste attualmente un linguaggio generale per descrivere i movimenti e per molte persone può non essere neppure abbastanza evidente la sua utilità. La nostra cultura è fondamentalmente analfabeta in fatto di movimenti; siamo abituati a descrizioni piuttosto vaghe dei movimenti umani, in particolar modo quelli riferiti alle attività abitudinarie, e da un punto di vista generale non disponiamo del vocabolario o della disciplina per descriverli con precisione. La necessità di termini più precisi è emersa recentemente nella produzione cinematografica, in cui le sceneggiature tradizionali si sono rivelate inadeguate.

Già molti anni fa, nel campo della danza, si era ravvisata la necessità di descrivere la coreografia per poterla tramandare, ma tutti i sistemi di notazione all'uopo sviluppati sono riusciti a rappresentare il movimento soltanto ad un livello molto basso. Magari sofisticati nei dettagli, ma carenti in termini strutturali (ovvero sintattici) tali sistemi sono di difficile apprendimento e pertanto di uso limitato. L'esperienza acquisita nel lavoro notazionale ha contribuito tuttavia ad una maggiore consapevolezza della necessità di un linguaggio di livello più elevato.

2. La specifica dei movimenti: notazione e copioni.

Sistemi notazionali. Un approccio alla descrizione dei movimenti caratteristici di certi processi psicologici è stato sviluppato da Scheflen [24], mentre Birdwhistell ha proposto [6] un sistema complesso per la descrizione dei movimenti usati nella comunicazione interpersonale.

Questi ed altri sistemi specializzati sono tuttavia inadatti per applicazioni generali. Tra i molti sistemi notazionali di quest'ultimo tipo, solo tre sono comunemente usati, e cioè: quelli di Benesh [3], quello di Eshkol-Wachmann [14], e la Labanotation di Hutchinson [17], tutti nati per registrare la danza.

Essi sono stati anche usati in altre applicazioni: la Labanotation negli studi dei tempi e movimenti in attività industriali, il metodo Benesh e la Labanotation per la rappresentazione dei movimenti nella sfera clinica [22], il sistema Eshkol-Wachmann per registrare il comportamento (behavorial patterns) degli animali [15].

La Labanotation, descritta diffusamente nella letteratura (ad es. [17]) costituisce la base del sistema attuale.

In figura 1 è illustrato un esempio di Labanotation. La partitura viene riportata verticalmente su uno spartito le cui colonne centrali rappresentano il supporto del movimento (normalmente il piede, destro o sinistro). Le colonne che si incontrano mano a mano che ci si allontana dal centro rappresentano i gesti delle gambe, del corpo, delle braccia e delle mani, mentre la testa è posta arbitrariamente a destra. Su appropriate colonne sono riportati dei simboli, che indicano il livello e la direzione dei movimenti di un arto e, con la loro lunghezza in verticale, la durata di tali movimenti.

I comandi della Labanotation sono in massima parte semplici istruzioni di spostamento di una parte d'arto (piede, avambraccio, ecc.) tra due orientamenti diversi nello spazio, con la specifica del momento d'inizio e della durata dello spostamento stesso.

Alcuni comandi Laban implicano una specie di stenografia, che deve essere interpretata in modo intelligente dal ballerino (ad esempio un salto).

Benchè sia teoricamente possibile specificare la velocità e l'accelerazione dei movimenti (equivalenti a

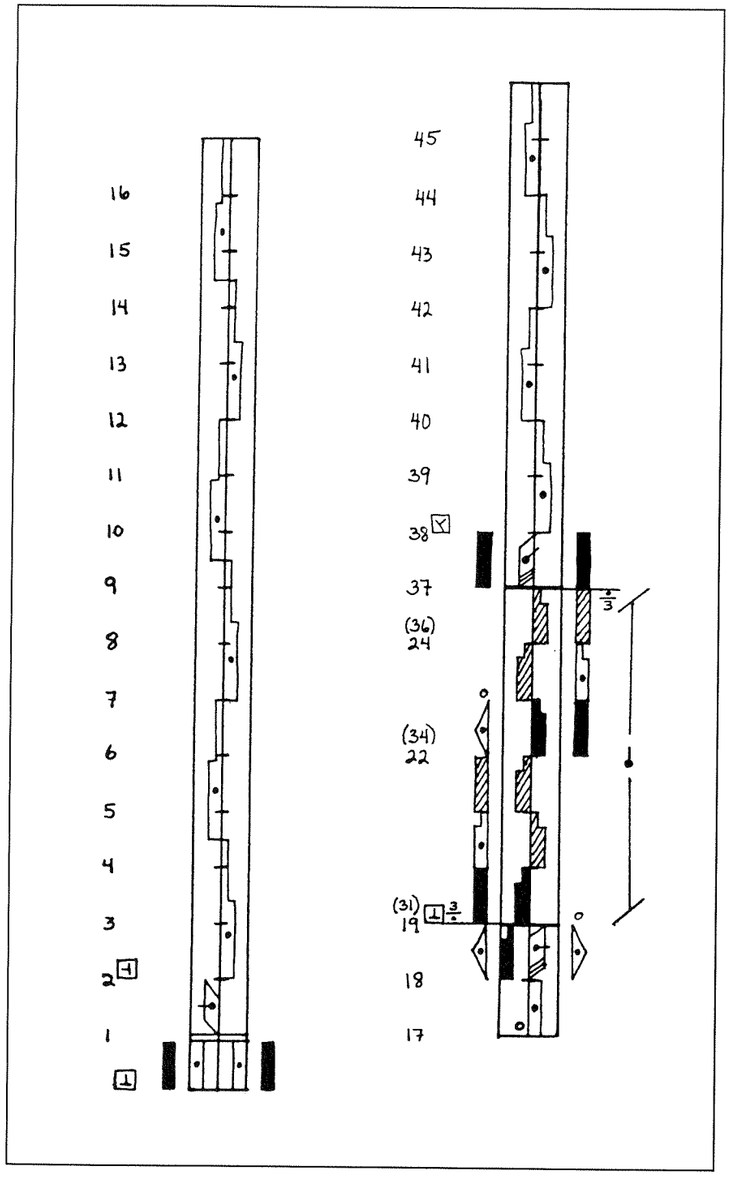


Fig. 1 - Una semplice partitura in Labanotation

rapidità e potenza, ovvero la qualità del movimento), ciò richiede la suddivisione di ogni movimento in parti sempre più piccole, rendendo il compito estremamente pesante.
La Labanotation lascia perciò in molti casi al ballerino una libera interpretazione della qualità del movimento e a tal fine è stata affiancata da un sistema notazionale parallelo ('Effort-Shape').
Interpretazione computerizzata della notazione. Come si è precedentemente osservato, i vantaggi dei sistemi notazionali sono parzialmente compromessi dalla loro intrinseca complessità e dalla difficoltà di apprendimento. Si è lavorato perciò per ottenere sistemi in grado di interpretare automaticamente le notazioni. Il computer è stato impiegato in vario modo, sia come ausilio alla composizione ed editazione dello spartito dei movimenti [22, 26], che come interprete in grado di produrre animazioni dello spartito e quindi agevolarne l'apprendimento [2, 11, 25].

Un editor computerizzato per la notazione Benesh è in fase di sviluppo presso l'Università di Waterloo, [21], dopo che un analogo editor per la Labanotation è già stato sviluppato all'Università di Pennsylvania [25].
In Australia, con un approccio diverso al problema, è stato sviluppato su computer un linguaggio secondo il codice Benesh presso l'Università di Sydney [16].
Essendo un sistema tra i più generalizzati, la Labanotation è stata adottata in almeno tre progetti basati su computer: il progetto Weber, Smoliar e Balder [26] all'Università di Pennsylvania, il progetto Savage e Officer [23] all'Università di Waterloo, ed il sistema da noi progettato presso la Simon Fraser University. [8, 9, 11].

Benchè i simboli Labanotation possano essere introdotti direttamente nei terminali grafici, si è scoperto, nel corso del nostro progetto, che era più conveniente utilizzare un codice alfanumerico. Il sistema interpretativo del computer analizza i comandi Labanotation e li traduce in una sequenza di angoli che rappresentano l'orientamento spaziale delle varie parti del corpo, dati che sono a loro volta tradotti nelle coordinate tridimensionali di ogni punto ad ogni istante. L'utente definisce l'angolazione e la distanza di osservazione della scena e una velocità appropriata di visualizzazione dei fotogrammi.

Il programma interpretativo per Labanotation è scritto in PASCAL ed è stato implementato su un terminale grafico IRIS.

Questo sistema grafico di elevata qualità genera figure filiformi o stilizzate, con un potere risolutivo dello schermo di 1204x768 pixel. È un sistema trasportabile che può funzionare su molti sistemi, di grandi e piccole dimensioni, anche su personal computer.

Alcuni esempi di output sono riportati nelle Figure 2, 3 e 4. La rappresentazione più semplice e veloce del corpo umano è quella filiforme, come in figura 2, in cui una figura isolata appare sulla trama del pavimento.

Questa rappresentazione filiforme, pur dando ogni informazione disponibile sui movimenti del corpo, presenta tuttavia qualcosa di innaturale agli occhi di molti osservatori; perciò si può disporre anche di una versione meno stilizzata, come quella illustrata in figura 3, che utilizza la tecnica sviluppata da Badler et al. nel 1979 per un display semplice di tipo raster. Una figura a colori di maggior qualità, prodotta sullo sfondo di un display 512x512 pixel, è riportata in figura 4.
Uso di "macro" per semplificare la partitura. Con la notazione un osservatore è in grado di registrare l'analisi di ogni movimento; tuttavia an-

Fig. 2 - Figura filiforme su terminale grafico.

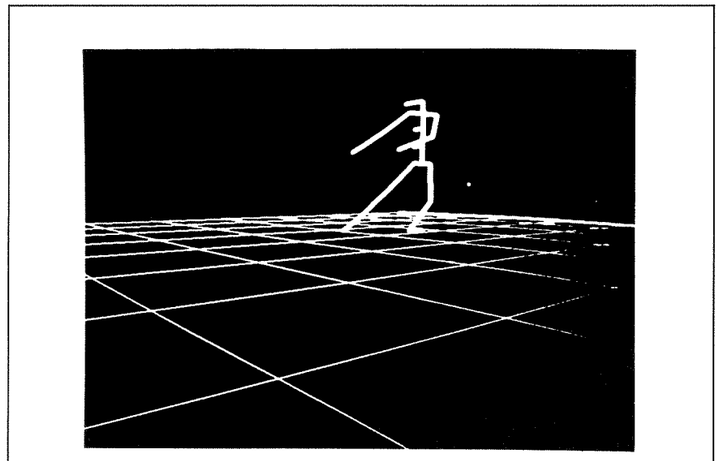
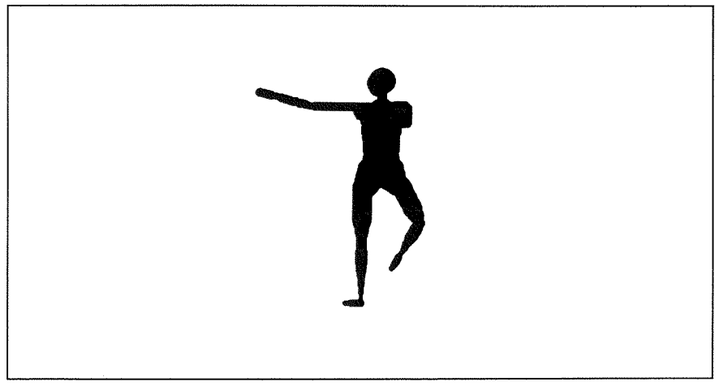


Fig. 3 - Figura stilizzata generata con sfere multiple.



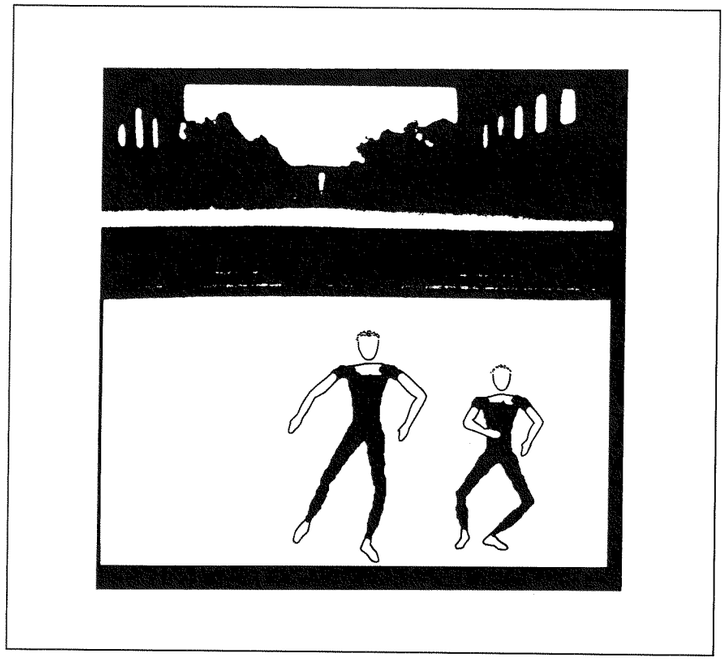


Fig. 4 - Animazione di una partitura di danza. (L'originale è a colori).

che con l'ausilio del computer questa operazione può essere alquanto noiosa e ripetitiva. Usando l'analogia dei linguaggi dei computer, la notazione può essere paragonata ai programmi in linguaggio macchina o al codice in linguaggio assemblatore. Seguendo questa analogia, abbiamo sviluppato un insieme di macro potenti per aiutare la stesura delle notazioni.

Nella loro forma più semplice, i macrocomandi sono sequenze di istruzioni, con un nome. Il loro uso è reso ancora più vantaggioso dalla possibilità di specificare parametri che consentono di modificare la sequenza in modo molto semplice: o meglio, non di modificare la sequenza di istruzioni ma alcune caratteristiche importanti, come la direzione e la velocità, che possono essere appunto parametri delle macro. In macro più complesse, i parametri servono per l'assemblaggio condizionato delle istruzioni, così da ottenere sequenze anche molto diverse, con il verificarsi di determinate condizioni. Ad esempio, la macro WALK (CAMMINARE) potrebbe essere espressa come segue:

WALK {distanza; numero di passi; tempo; lunghezza del passo; direzione; piede di partenza; stile}

Tutti i parametri, eccetto il primo, potrebbero anche essere impliciti, essendo di tipo naturale. Questa macro si presenta comunque come un insieme di sottomacro, con la sintassi riportata in figura 5, che rappresenta uno schema di Wirth per la macro WALK [18].

Si vede in figura che WALK comporta un passo iniziale (destro o sinistro), seguito da un numero arbitrario di passi sulle due gambe alternativamente, e termina con l'ultimo passo sulla gamba appropriata. Disponiamo già di una libreria di numerose macro, comprendenti gran parte delle posizioni standard del balletto.

È importante notare che le macro possono utilmente contenere sequenze predisposte di comandi Labanotation, per una loro esecuzione ripetuta. L'uso di parametri semplici consente di assemblare la sequenza per un numero diverso di ripetizioni, in base al verificarsi di certe condizioni, ad esempio per la direzione e lo stile dei movimenti. Con tali macro si può abbreviare sensibilmente la scrittura di una sequenza di movimenti.

In effetti le macro costituiscono un metodo intermedio tra la notazione di base ed un linguaggio naturale di livello più elevato.

Un linguaggio dei movimenti per la cinematografia. Quando il direttore artistico programma la produzione di un film, in collaborazione con il regista e lo sceneggiatore, sviluppa una serie di schizzi sul copione, mediante i quali l'artista visualizza gli elementi importanti di ciascuna scena. Sistemi di raffigurazione, noti anche come "copioni elettronici", sono attualmente utilizzati da Robert Abel della Digital Productions, e da altri, per conferire una sia pur grossolana animazione ad una scena nel momento della sua concezione. Le apparecchiature tipiche usate a questo scopo consistono di un display con grafica vettoriale e del computer associato. Gli oggetti presenti nella scena vengono introdotti con tutti i dettagli occorrenti a produrre disegni costituiti da linee, che possono essere manipolati uno rispetto all'altro in uno spazio tridimensionale.

Il regista può predisporre manualmente i fotogrammi-chiave usando manopole, joystick, mouse o tablet. Con una semplice animazione si visualizzano poi, in tempo reale, i movimenti da un fotogramma all'altro.

Quando il regista ha ottenuto una composizione soddisfacente dei fotogrammi-chiave e dei movimenti intermedi, si possono rilevare le coordinate e gli orientamenti degli oggetti, nonchè l'orientamento ed il movimento delle macchine da presa, allo scopo di girare scene dal vivo, oppure per introdurre tali dati in un database per produrre animazioni computerizzate.

Sistemi di controllo di questo tipo offrono al regista la possibilità di provare molte volte una scena prima di decidere come filmarla effettivamente. Consentono inoltre risparmi considerevoli in quanto le riprese e gli allestimenti possono essere programmati nei minimi dettagli, riducendo così il tempo di ripresa fino al 50%. Abel ha stimato che il costo del controllo preventivo di un lungometraggio è all'incirca pari a quello di una ripresa dal vivo di un giorno [1]. I sistemi finora sviluppati hanno tuttavia trattato il corpo umano come un oggetto solido singolo, poichè l'animazione corporea è molto complessa e difficile da specificare, e ciò limita ovviamente i loro vantaggi per il regista.

L'applicazione di un linguaggio gestuale all'animazione della produzione cinematografica appare quindi ovvia. Tuttavia, anche per i cartoni animati, i movimenti sono ancora in gran parte specificati con procedimento manuale. Finora l'animazione computerizzata nel cinema si limita essenzialmente agli effetti speciali; un'animazione realistica presuppone, infatti, la produzione di strutture gestuali naturali per ogni soggetto o personaggio da simulare.

Esistono già diversi sistemi parzialmente efficaci in quest'area, soprattutto per oggetti non articolati, ma nessuno è veramente capace di animazione umana. Il nostro lavoro rappresenta un punto di partenza per l'animazione umana. Un ente commerciale ha sperimentato il nostro sistema basato su Labanotation, ma, anche usando le macro, le partiture sono riuscite molto lunghe e pesanti. Benchè non sia consentita ancora un'animazione veramente realistica degli attori (è particolarmente difficile produrre l'animazione dei primi piani del viso), ciò è probabilmente fattibile per personaggi che stanno sullo sfondo o in scene affollate.

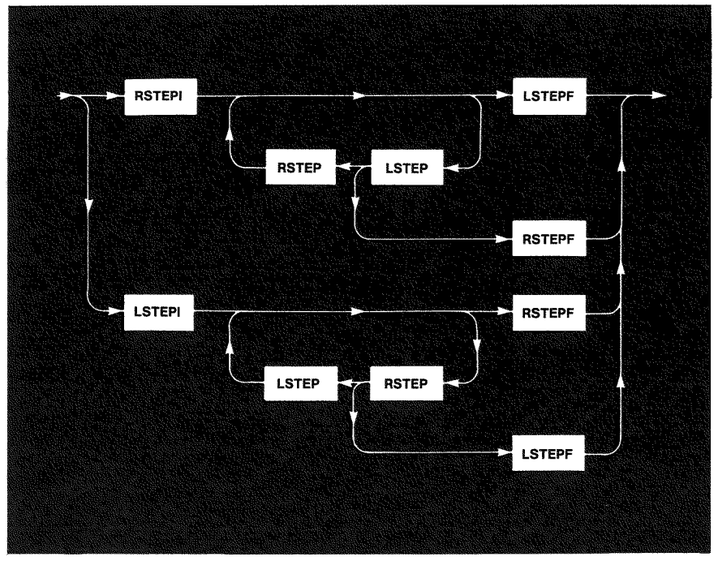
L'animazione computerizzata trova già invece un'applicazione abbastanza estesa nella produzione pubblicitaria televisiva, e almeno due organizzazioni importanti, la Lucasfilm e la Digital Productions, stanno sviluppando tecniche per lungometraggi [13], di cui un certo numero già terminati (ad es. *Tron* e *The last starfighter*), con sequenze significative di animazione computerizzata.

Un linguaggio dei movimenti per il teatro? Non ci risulta siano stati fatti tentativi in campo teatrale per applicare sistemi di controllo analoghi al copione elettronico, benchè anche per il teatro tali sistemi abbiano sicuramente un potenziale altissimo. La dinamica della produzione è tuttavia molto diversa rispetto alla produzione cinematografica. Mentre al cineasta può bastare un sistema semplice di animazione per oggetti solidi, al regista teatrale occorrerà sicuramente un metodo per specificare il movimento umano. Inoltre il direttore tecnico potrebbe avere interesse a combinare insieme una sceneggiatura computerizzata con altri elementi, come il controllo delle luci.

Riepilogo dei requisiti. Dall'insieme di tutti questi studi si deduce la necessità di un metodo naturale e flessibile per specificare e/o animare i movimenti umani nel campo della danza, del cinema e del teatro; questo sistema potrebbe presumibilmente applicarsi anche in altri campi, ed in particolare all'ergonomica.

Fig. 5 - Organigramma di Wirth della "macro" WALK.

Legenda:
RSTEP1 = passo iniz. destro
LSTEP1 = passo iniz. sinistro
RSTEP = passi successivi d.
LSTEP = passi successivi s.
RSTEPF = passo finale d.
LSTEPF = passo finale s.



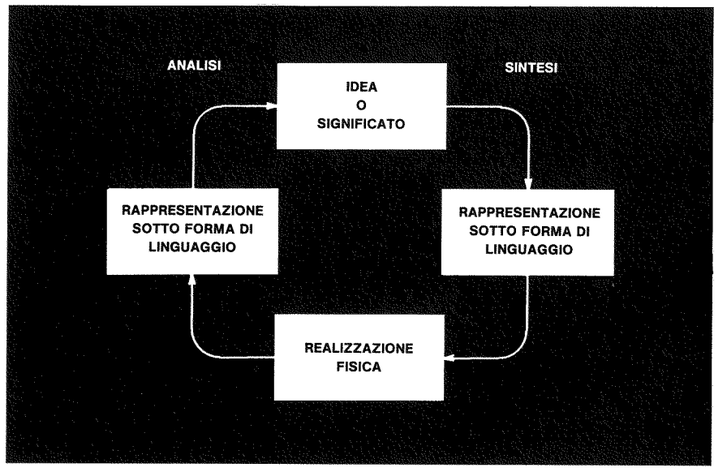


Fig. 6 - Sintesi ed analisi nella rappresentazione di idee mediante un linguaggio.

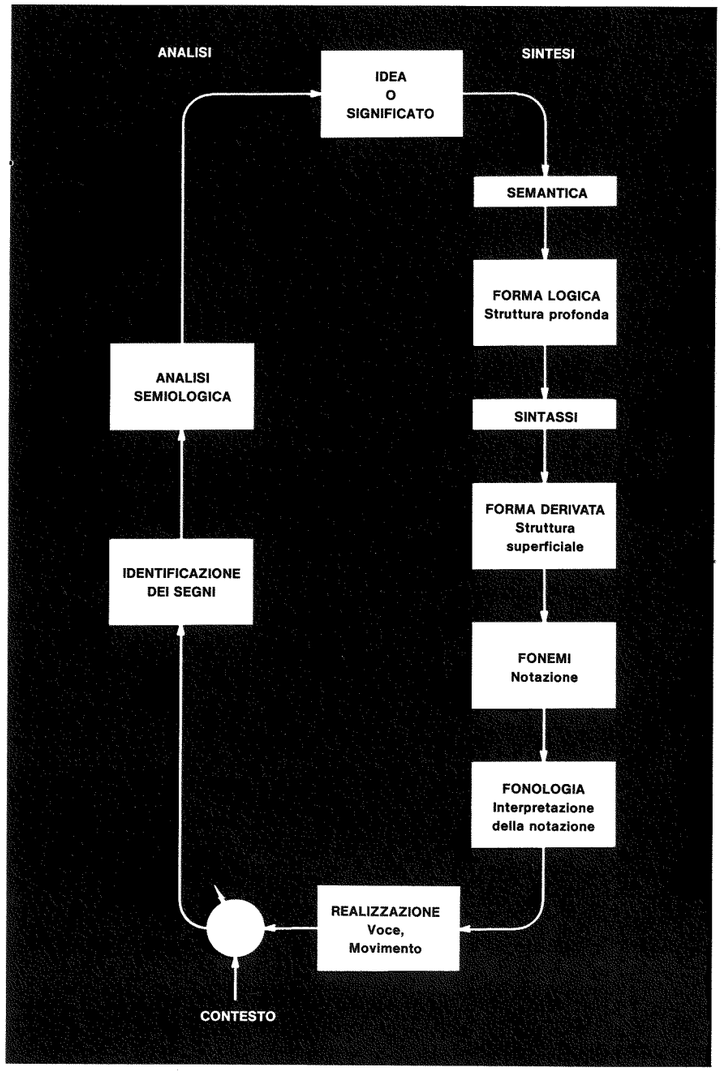


Fig. 7 - Il ruolo del linguaggio nella comunicazione vocale o gestuale.

L'approccio sinora usato è tuttavia ancora embrionale. Gli attuali sistemi notazionali, tipicamente elementari e strutturalmente carenti, non possono sostituirsi nè sopperire all'assenza di un esauriente linguaggio per i movimenti. Tutte le indicazioni finora emerse convergono verso un approccio di fondo più concettuale per la rappresentazione e la specifica dei movimenti.

3. Una base teorica per un linguaggio dei movimenti.

Analogie con la comunicazione vocale delle idee. L'approccio teorico qui seguito è basato su un modello semplice del ruolo di un linguaggio. Come illustrato in figura 6, la caratteristica essenziale del modello è che un'idea da comunicare viene anzitutto rappresentata per mezzo di un linguaggio (ad esempio un linguaggio naturale, come l'inglese), e poi trasmessa mediante un mezzo fisico (come la voce). Si tratta quindi di un processo di sintesi. Si ha il processo inverso quando un osservatore analizza il fenomeno fisico e lo rappresenta sotto forma di linguaggio, da cui deduce il significato.

Questo modello è ulteriormente sviluppato nello schema di fig. 7.

L'autore di un'idea, sia essa una semplice comunicazione od una narrazione complessa, come una commedia, un film o la coreografia di un ballo, rappresenta prima la sua idea od insieme di idee sotto forma di una sceneggiatura scritta, arricchita possibilmente da grafici come quelli usati per un copione cinematografico. Qualunque sia il mezzo trasmissivo, si deve poter disporre di un dizionario. Quando le idee vengono rappresentate in un linguaggio naturale, la padronanza da parte dell'autore nella grammatica del linguaggio e la sua conoscenza della semantica, guideranno la traduzione delle sue idee verso una particolare forma logica, ovvero struttura profonda (Chomsky, [12]). A partire da questa struttura profonda, lo stile proprio dell'autore determinerà poi la forma derivata, ovvero la struttura superficiale. Le relazioni tra queste due strutture sono governate dalle regole della sintassi.

Se la comunicazione è invece di tipo verbale, la conoscenza fonologica di chi legge traduce il testo in una sequenza di fonemi, che sono poi convertiti in suoni da un appropriato sistema motorio che controlla i muscoli della laringe, del petto e della bocca. Il lettore usa la sua conoscenza di un insieme piuttosto complesso di regole per scegliere i fonemi e l'intonazione appropriati a rappresentare il testo.

La sintesi della voce eseguita dal computer può essere veramente molto efficace quando è pilotata dai fonemi (cioè quando l'uomo sceglie i fonemi per la rappresentazione del testo), ma non riesce altrettanto bene quando il programma del computer deve attuare le regole per la scelta dei fonemi e dell'intonazione.

Rappresentazione delle idee di movimento. Esiste un'analogia abbastanza evidente tra la materializzazione di un'idea come voce e la sua materializzazione sotto forma di comunicazione gestuale o linguaggio del corpo [6]. La rappresentazione dell'idea nella danza è spesso dominata da considerazioni estetiche, ma è, sotto gli altri aspetti, analoga alla realizzazione della narrazione in una commedia teatrale od in un film. Tuttavia, benchè il processo sia simile nelle linee generali, la traduzione delle idee in schemi motori consiste di fasi che non sono altrettanto chiare.

La rappresentazione del significato in una "struttura profonda" per la danza [27] è assai controversa per quanto riguarda gli elementi in gioco [28]. Salvo una trattazione introduttiva [9], non si è fatto alcun tentativo di trovare una sintesi per un linguaggio gestuale e manca perciò un linguaggio completo dei movimenti. Tuttavia, a differenza del rapporto tra testo e voce, è stato sviluppato un certo numero di schemi notazionali per rappresentare i movimenti elementari, schemi più direttamente paragonabili a notazioni per la specifica di fonemi che a un linguaggio naturale.

Pertanto la notazione è piuttosto esplicita e di interpretazione abbastanza immediata sia da parte dell'uomo che del computer. La difficoltà principale insita negli schemi notazionali dipende dalla loro natura necessariamente elementare ed estremamente dettagliata; di conseguenza è difficile impararli ed il loro utilizzo è piuttosto pesante. Dovrebbe risultare evidente che notazioni a questo livello non costituiscono certo un surrogato di un linguaggio naturale esteso.

Un linguaggio naturale dei movimenti. Si impone quindi la necessità di sviluppare un linguaggio naturale per i movimenti umani. Poichè la potenza di un linguaggio è legata ai vantaggi derivanti dalla struttura delle idee da esprimere e degli schemi motori che realizzano tali idee, è necessario formalizzare la sematica e la sintassi del movimento. Ci dovrà essere una gerarchia di grammatiche poichè un certo numero di forze diverse impongono una struttura al movimento umano.

A livello fondamentale, ci sono vincoli di tipo anatomico, fisiologico e biomedico, che limitano la gamma dei movimenti possibili, oltre a determinare quali movimenti sono più naturali e facili da eseguire.

Tali vincoli determinano le strutture dei movimenti funzionali, come il camminare, il nutrirsi e l'abbigliarsi, che svolgono generalmente un ruolo minore nella comunicazione ma possono averne uno più importante sotto l'aspetto estetico. Ad un livello più elevato, si hanno schemi gestuali di derivazione culturale: i gesti comunicativi, ad esempio, sono generalmente legati ad una particolare cultura. Infine, al massimo livello si hanno gli schemi gestuali, essenzialmente estetici, della danza.

Tutti i livelli sopra esaminati sono fortemente strutturati e si dovrebbe poterli ricondurre a descrizioni sintattiche formali, ma non ci risulta sia mai stato fatto un serio tentativo a questo proposito.

Si deve osservare che non c'è una completa analogia con la comunicazione in generale e con la voce in particolare. Alcuni movimenti si prefiggono uno scopo ma non hanno un vero significato; sono solo funzionali. Ad esempio, l'istruzione di *attraversare la strada* oppure *bere da una tazzina* non trasmette necessariamente un significato.

Dobbiamo pertanto ampliare il ruolo del significato, includendo le attività orientate ad uno scopo e quelle funzionali.

Analisi semiologica. Tutti i processi sopra descritti comportano una sintesi. Partendo dalle idee di un autore, di un coreografo o di un altro attore della comunicazione, si deriva una descrizione in linguaggio formale od informale, che viene a sua volta tradotta dallo speaker, attore, o ballerino, o anche dal computer, nell'attuazione fisica delle idee originarie. Si può essere aiutati a determinare un linguaggio formale per la descrizione degli schemi gestuali, dallo studio del processo di analisi, inverso del precedente.

La sintassi di un linguaggio è un mezzo utile per determinare il significato di un testo o discorso, ma l'analisi semiologica è ancora uno strumento grossolano per l'analisi del significato globale di un testo narrativo, una commedia, un film o un numero di danza [5].

L'analisi semiologica comporta l'identificazione di segni e relazioni e la corrispondente deduzione di significati. Chi interpreta uno spettacolo utilizza un insieme di regole personali per individuarvi dei segni, anche se è spesso inconsapevole di queste regole. Tali segni svolgono un ruolo analogo alle parole o anche alle frasi di un testo e le regole per le relazioni reciproche sono analoghe alla sintassi del testo.

Mentre la semiologia si trova ancora in una fase embrionale, il lavoro di sviluppo di un linguaggio cinematografico [20] è certamente un esempio di attività decisamente orientata allo sviluppo di un linguaggio generale del movimento. Alcune idee generali per una semiologia della danza sono state trattate da Zelinger [28].

4. Progetto funzionale di un linguaggio dei movimenti.

Il linguaggio dovrebbe idealmente essere disponibile all'utente in tre livelli, illustrati in figura 8, e che saranno trattati in relazione ai loro input, funzioni e base di conoscenza. Occorre sottolineare che l'input normale per ogni livello di questo linguaggio, comprende non solo un testo ma anche dati numerici e grafici; l'input sarà generalmente introdotto interattivamente attraverso una stazione di lavoro grafica e conterrà effiancati testo e scene grafiche.

Livello fondamentale. A questo livello le specifiche sono espresse in funzione degli scopi da conseguire, delle funzioni da eseguire e dei significati da trasmettere.

Pertanto gli input tipici potrebbero consistere di specifiche di questo genere:

- *Bevi dal bicchiere sul tavolo.*
- *Attraversa la stanza ed apri la porta.*
- *Alzati e cammina verso la figura B; stringile la mano.*
- *"Ronde de jambe à terre en dehors" con la gamba destra nella quarta posizione "devant-demi pliè" rivolta verso l'angolo 1.*
- *Il gruppo si siede sul pavimento. Parlano. Distendono le mani, si afferrano reciprocamente le braccia. Poi cercano di alzarsi. Becky rivolge la schiena a tutti; tutti sono in piedi eccetto Becky.*

Tutti gli input a questo livello hanno in comune la caratteristica di essere privi, in maggiore o minore grado, di specificità; essi descrivono un'attività finalizzata ad uno scopo, un movimento o sequenze di movimenti in un ballo, ma in tutti i casi manca la vera informazione. Ciò che necessita per l'interpretazione di queste istruzioni e la loro traduzione in un completo e specifico insieme di istruzioni, è una base di conoscenza che includa informazioni su tutti gli aspetti importanti dell'ambiente (dimensioni della stanza, palcoscenico o set cinematografico; dimensioni e ubicazione dei mobili e di altri oggetti), come pure sulle figure presenti (si usa il termine "figura" invece di attore, ballerino, ecc.). Questo livello di input si può ottenere direttamente dalla sceneggiatura.

Struttura profonda. Utilizzando le regole semantiche si traduce il "livello fondamentale" in qualcosa di analogo alla struttura profonda del linguaggio naturale. Questa trasformazione secondo le regole semantiche comporta il completamento di tutte le informazioni mancanti relative al movimento generale delle figure, lasciando tuttavia indefiniti i dettagli a basso livello. Ad esempio, l'istante della partenza e dell'arrivo e il percorso da fare verrebbero specificati, ma non lo sarebbe il numero di passi e lo stile personale (che dipendono entrambi dalle caratteristiche della figura particolare). Sarebbe inoltre importante per la trasformazione assumere dei valori ragionevoli per quei parametri indefiniti il cui valore non può dedursi dalla base di conoscenza. La "struttura profonda", in quanto rappresentazione interna, non è però prevista per accettare input a questo livello.

La "struttura profonda" avrà una forma simile alla "rappresentazione superficiale", ma ne differirà per il fatto che molte rappresentazioni superficiali diverse possono derivare da un'unica struttura profonda.

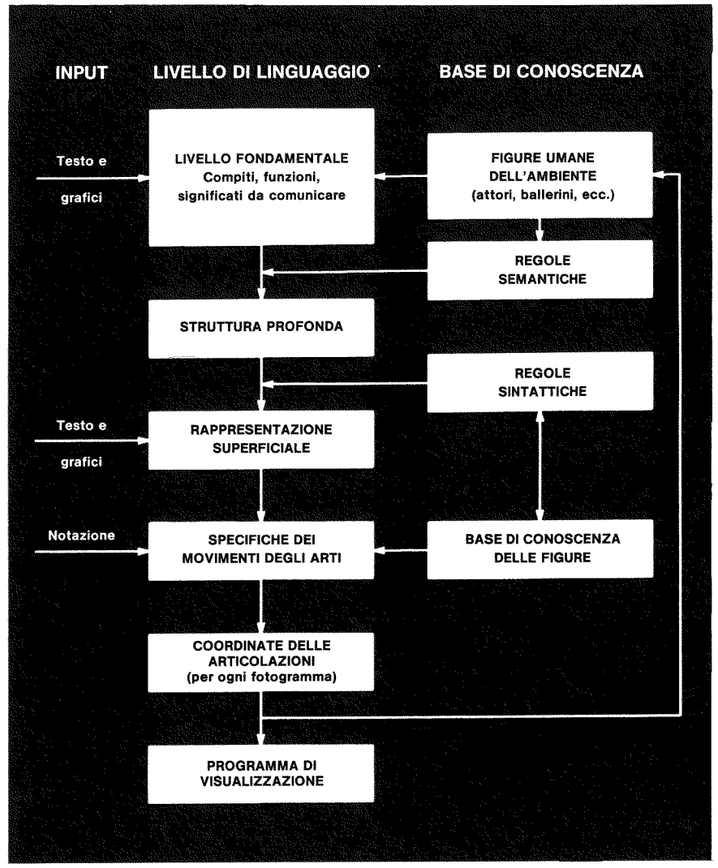
Questa varietà è da attribuire alla diversità delle dimensioni corporee, delle caratteristiche di movimento ed anche dello stato d'animo delle diverse figure. Tali differenze vengono sistemate mediante la "base di conoscenza delle figure" e le regole sintattiche valide per ogni figura.

Rappresentazione superficiale. L'input per questo livello proviene dalla struttura profonda, ma può essere anche un input esterno indipendente. Un input di questo tipo potrebbe ad esempio assumere la seguente forma:

La figura A protende il gomito destro fino a portare orizzontali l'avambraccio e la mano. Successivamente, il braccio si stende, mantenendo orizzontali l'avambraccio e la mano, fino a che la mano aperta si trova alle coordinate X=31, Y=65, Z=96. La mano si richiude fino a che la distanza tra il pollice e le altre dita è uguale al diametro del bicchiere. Piegare il braccio e muovere la mano in orizzontale fino alla distanza 2.0 dalla bocca.

(Questo esempio corrisponde a quello già visto: *"Bevi dal bicchiere posto sul tavolo„).*

Fig. 8 - Le componenti funzionali del linguaggio gestuale.



Si può osservare che un input di questo genere comprende un insieme molto dettagliato di istruzioni su come prendere un bicchiere e prepararsi a bere; si noterà tuttavia che rimangono ancora in massima parte indefinite le angolazioni dell'arto. In complesso però questa descrizione risulterà molto più dettagliata rispetto a quelle delle sceneggiature più ricche di annotazioni.

Specifiche del movimento degli arti. Dalla rappresentazione superficiale si deriva il livello minimo di linguaggio, applicando le regole sintattiche per la figura in questione. Così si terrà conto dei dettagli delle dimensioni corporee come pure delle restrizioni, di tipo anatomico e biomedico, al movimento. Uno sviluppo molto più difficile sarà quello basato su un metodo che privilegi i movimenti più naturali e anche le strutture e gli stili motori abituali degli individui.

Le "specifiche del movimento degli arti" possono essere espresse mediante una versione estesa di Labanotation e potenziate dalle macro sopra descritte, consentendo così all'autore un input diretto a questo livello. Sarà anche possibile una notazione di editing e di aggiustamento, derivandola dai livelli superiori del sistema, anche se ciò non comporterà necessariamente la possibilità inversa, cioè di portare ritocchi di livello basso nella scrittura ad alto livello. Si può anche avere un input diretto da soggetti animati muniti di goniometri ed altri strumenti.

5. Realizzazione.

Il sistema che abbiamo delineato è globale ed ambizioso. Per la sua realizzazione saranno necessari molti anni-uomo di lavoro per sviluppare le basi di conoscenza per i vari sistemi esperti che dovranno catturare la terminologia del regista/sceneggiatore/artista, le caratteristiche fisiche del set/ambiente, la rappresentazione e la forma degli oggetti e delle figure umane da animare, ed il movimento di tali figure ed oggetti. Questi studi rivestono tuttavia un particolare interesse per la loro caratteristica di poter essere sviluppati secondo uno schema gerarchico. Si può così sviluppare in tempi veramente brevi un sistema di linguaggio semplice, in quanto si dispone già di gran parte dei suoi elementi. Tale sistema avrà una interfaccia semplice uomo-macchina ed una limitata base di conoscenza per i sistemi esperti. Con il suo ulteriore sviluppo, il sistema si arricchirà di nuove funzionalità, tendendo verso il sistema ideale.

L'implementazione di un sistema di linguaggio siffatto si gioverà notevolmente dell'impiego delle più recenti tecnologie hardware e software. I grandi progressi nel campo dei

microprocessori di basso costo ma elevata potenza, e dei display grafici bit-mapped con alto potere risolutivo, sono stati determinanti per lo sviluppo di stazioni di lavoro sofisticate, che arricchiscono l'interfaccia-utente con grafica interattiva, un ambiente software evoluto, un processor molto potente (a 32 bit) ed il collegamento a larga banda su rete locale con altri processori, ecc. Queste stazioni di lavoro sono ideali per lo sviluppo del sistema di linguaggio in discussione. Grazie alle ricerche sull'Intelligenza Artificiale si è definita la metodologia di sviluppo dei sistemi esperti e delle basi di conoscenza ad essi associate; tali sistemi esperti conferiscono coerenza e sistematicità al processo di formalizzazione delle regole sintattiche e sematiche del linguaggio proposto.

Sono in piano le seguenti fasi principali di implementazione:

- *Sviluppo di un sistema esperto in linguaggio.*

Per determinare il vocabolario appropriato e la base di conoscenza, sarà necessaria un'estesa interazione con i registi, i tecnici e gli attori impegnati nella produzione cinematografica o teatrale e nell'animazione. Il sistema esperto è in fase di sviluppo su una stazione di lavoro Sun con linguaggio C e C-Prolog. Il prototipo, denominato l'"Apprendista regista", si concentra su quegli aspetti del linguaggio che sono particolarmente significativi nella produzione cinematografica.

- *Sviluppo di un sistema esperto in animazione umana.*

Questa fase comporta la costruzione della base di conoscenza per i parametri fisici, i vincoli anatomici e fisiologici e gli schemi gestuali naturali degli attori. È già stato concluso il lavoro preliminare per la produzione in modo interattivo di sequenze di animazione.

- *Sviluppo di un programmatore di movimento in ambiente disordinato.*

Per questo studio ci si avvale largamente del lavoro di programmazione dei movimenti dei robot in ambiente disordinato (cluttered environment). Questo è soltanto un esempio della capacità autonoma di soluzione dei problemi, richiesta per implementare gli input della struttura profonda.

6. Conclusioni.

Benchè lo sviluppo teorico presentato in questo articolo sia piuttosto generale, si prevede una implementazione iniziale specifica in determinate aree applicative. Questa specificità non solo renderà più gestibile il problema nel suo complesso, ma faciliterà anche l'impostazione degli ardui problemi da affrontare.

È interessante notare che nello sviluppo del linguaggio da uno schema teorico astratto a un progetto dettagliato adatto alla realizzazione pratica, si è andato gradatamente modificando anche il nostro modo di vedere il sistema in sviluppo. Esso viene ora visto come un sistema a componenti multipli, costituito da una interfaccia molto sofisticata per interazioni con l'utente, da un insieme di sistemi esperti e basi di conoscenza organizzati secondo criteri gerarchici e da un display di qualità, per generare l'animazione.

Non ci si dovrebbe stupire di questi cambiamenti nel modo di vedere il sistema in discussione, in quanto i suoi componenti sono proprio quelli che l'uomo ha sviluppato nell'evoluzione del linguaggio.

Bibliografia

[1] *American Cinematographer,* July 1982 (Special Issue on Computer Assisted Filmmaking).

[2] Badler, N.I. and S.W. Smoliar (1979). Digital representation of human movement, *Computing Surveys,* 11, 19-38.

[3] Benesh, R., and J. Benesh (1956). *An Introduction to Benesh Dance Notation* (A.C. Black, London).

[4] Benesh, R., and J. McGuinnes (1974). Benesh movement notation and medicine, *Physiotherapy,* 60, 176-178.

[5] Berger, A.A. (1982), *Media Analysis Techniques,* Sage COMTEXT Series (Sage Publications, Beverly Hills), vol. 10.

[6] Birdwhistell, R.L. (1970). *Kinesics and Context: Essays on Body Movement Communication* (Univ. of Pennsylvania Press, Pennsylvania).

[7] Calvert, T.W. (1983). Computer Assisted Filmmaking, in *Proc. Graphics Interface 83 Conference,* Edmonton, Alberta.

[8] Calvert, T.W., J. Chapman and A. Patla (1982). Aspects of the kinematic simulation of human movement, *IEEE Computer Graphics and Applications,* 2, 41-50.

[9] Calvert, T.W., J. Chapman and A. Patla (1980). The integration of subjective and objective data in animation of human movement, *Computer Graphics,* 14, 198-203.

[10] Calvert, T.W., J. Chapman an J. Landis (1979). Notation of dance with computer assistance, in *New Directions in Dance,* D.T. Taplin (ed.) (Pergamon Press, Toronto), 169-178.

[11] Calvert, T.W. and J. Chapman (1978). Notation of movement with computer assistance, in *Proc. ACM Annual Conf.,* 2, 731-736.

[12] Chomsky, N. (1965). Aspects of the Theory of Syntax (MIT Press, Cambridge).

[13] Demos, G., M. Brown and R. Weinberg (1984). Digital scene simulation: The synergy of computer technology and human activity, *Proc. IEEE,* 72, 22-31.

[14] Eshkol, N. and A. Wachmann (1958). *Movement Notation* (Weidenfeld and Nicholson, London).

[15] Golani, I (1976). Homeostatic motor processes in mammalian interactions: A choreography of display, ch. 2 in *Perspectives in Ethology,* 2, P.P.G. Bateson and P.H. Klopfer, editors (Plenum, New York).

[16] Herbison-Evans, D. A human movement language for computer animation, in *Language Design and Programming methodology,* J. Tobias ed. (Springer-Verlag, New York).

[17] Hutchinson, A. (1960). *Labanotation* (Theatre Art Books, New York).

[18] Jensen, K. and N. Wirth (1975). *Pascal-User Manual and Report* (Springer-Verlag, New York).

[19] Korein, J.U. and N.I. Badler (1982). Techniques for generating the goal directed motion of articulated structures, *IEEE Computer Graphics and Applications,* 2, 71-81.

[20] Metz, C. (1974). *Film Language* (Oxford University Press, new York).

[21] Ryman, R., J.C. Beatty, K.S. Booth and A. Singh (1983). A computerized editor for Benesh movement notation, *CORD Dance Research Journal.*

[22] Ryman, R., A. Patla and T. Calvert (1983). Use of Labanotation for clinical analysis of movement, *Proc. Int. Congress Kinetography Laban.*

[23] Savage, G.J. and J. Officer (1977). Choreo: An interactive computer model for choreography, in Proc. 5th Man-Machine Communication Conf,Calgary, Alberta.

[24] Scheflen, A.E. (1974). *How Behavior Means* (Anchor Books, Garden City).

[25] Smoliar, S.W. and L. Weber (1977). Using the computer for a semantic representation of Labanotation, in *Computing and the Humanities,* S. Lusignan and J.S. North, eds. 253-261 (University of Waterloo Press, Waterloo, Ont.).

[26] Weber, L., S.W. Smoliar and N.I. Badler (1978). An architecture for the simulation of human movement, in *Proc. ACM Annual Conf.,* 2, 737-745.

[27] Williams, D. Deep structures of the dance, *Journal of Human Movement Studies,* 2, 123-144 and 155-181.

[28] Zellinger, J. (1979). Semiotics and theatre dance, in *New Directions in Dance,* D.T. Taplin ed. (Pergamon Press, Toronto).

[29] Zeltzer, D. (1982). Representation of complex animated figures, *Proc. Graphics Interface 82,* 205-212, Toronto.

[30] Zeltzer, D. (1982). Motor control techniques for figure animation, *IEEE Computer Graphics and Applications,* 2, 53-59.

Il software SDI potrà mai essere a prova d'errore?

Ware Myers

Contributing Editor
IEEE Computer

1. Premessa

La distruzione dei missili avversari al momento del lancio, o durante il loro tragitto nello spazio, o al rientro nell'atmosfera, comporta la necessità di un sistema su scala mondiale di sensori, piattaforme di attacco e telecomunicazioni. Secondo le prime valutazioni, il sistema di comando, controllo e comunicazioni (C3) che dovrà legare insieme tutti questi elementi, si baserà sul progetto di software più imponente mai affrontato finora.

"A causa dell'estrema criticità dei compiti che è chiamato a svolgere e dell'impossibilità di collaudarlo (in condizioni operative reali), non saremo mai assolutamente sicuri della riuscita", ha affermato David L. Parnas, uno dei più noti esperti mondiali di software, dimissionario il 28 giugno 1985 dalla Commissione per "Computing in Support of Battle Management" dell'SDI (Strategic Defense Initiative).

Secondo Parnas, professore di computer science all'Università di Victoria nella British Columbia, "le armi nucleari rimarranno una potente minaccia" [1].

"È troppo presto per cedere al pessimismo", ribatte Frederick P. Brooks, autore di *The Mythical Man-Month* e professore di computer science alla Università della North Carolina, in una audizione ad un comitato del Congresso [2]. "Non vedo i motivi per cui io stesso o qualsiasi altro software manager competente ed esperto, come ce ne sono tanti, non possa impegnarsi nella costruzione di tale sistema".

L'House Appropriations Committee (Comitato Stanziamenti del Congresso), nell'esame dei fondi da destinare all'SDI, ha puntualizzato "l'esistenza di problemi consistenti nella progettazione e collaudo del software per il sistema SDI" [3]. Il comitato ha sottoposto in proposito nove domande alla Organizzazione SDI, o SDIO. In risposta, il generale James A. Abrahamson, Direttore SDIO, ha ammesso che nessun sistema complesso mai costruito ha raggiunto la "perfezione" [4]. Tuttavia, gran parte dei sistemi sviluppati per aree critiche come operazioni spaziali, voli militari e commerciali, controllo del traffico aereo, telecomunicazioni, banche e medicina, si possono considerare sistemi "corretti" nel senso che essi sono adeguatamente efficaci e affidabili.

Nel dicembre 1985 la commissione di cui faceva parte Parnas prima delle dimissioni, concluse che "le risorse computazionali ed il software per la gestione di un sistema di difesa strategico sono alla portata delle tecnologie hardware e software potenzialmente sviluppabili in un arco pluriennale" [5].

Vi sono, dunque, sostanziali divergenze interpretative e di giudizio. C'è comunque un fattore che il dibattito ha portato alla luce ed è il ruolo critico del software nel quadro dell'SDI. A giudizio della commissione, "la prevista complessità del software e la necessità di provare, modificare e far evolvere il sistema, rendono la gestione del sistema C3 il problema dominante della difesa strategica".

L'interrogativo più inquietante cui dobbiamo rispondere, nei termini in cui è stato posto da Brooks, durante i lavori della sua commissione all'ottava Conferenza Internazionale sul Software Engineering, a Londra, il 30 Agosto 1985, è: "Quando possiamo dire che un sistema è sufficientemente buono?" (*How good is good enough?*).

Il problema della "bontà sufficiente" si può meglio capire, nel caso di un attacco di missili nucleari, con un esempio: supponendo che il 99,9% costituisca un valore efficace, qualora venissero dirette contro di noi 10.000 bombe nucleari, ne giungerebbero a destinazione "solo" 10. Ovviamente non è questa una prospettiva piacevole.

Errori del software non equivalgono, tuttavia, automaticamente a crepe nello scudo protettivo. Un errore di sofware può, ad esempio, comportare soltanto l'errata collocazione di un elemento su uno schermo video. Potrebbe invece risultare ben più grave se il sistema assegnasse due armi di difesa contro una sola minaccia, quando ne dovrebbe assegnare solo una, ha detto Brooks al comitato del Senato, aggiungendo

---

Questo articolo è ripreso da "Computer", Vol. 19, No. 11, November 1986, per gentile concessione dell'Institute of Electrical and Electronics Engineers.

però "che non si tratta ancora di un problema che tolga il sonno".

Sarebbe infatti ancora più grave un errore che facesse fallire la distruzione di un bersaglio durante la fase di lancio. Un software perfettamente a punto ai livelli successivi potrebbe ancora eliminare il missile, purchè l'errore nella fase di lancio non sovraccarichi i livelli a monte, nel qual caso l'errore iniziale di software diverrebbe veramente drammatico. Ciò che preoccupa è la possibilità che alcune migliaia di errori nel software provochino alcune centinaia di fallimenti significativi che risultino in 10 crepe nello scudo difensivo.

2. Il problema degli errori

Nei sistemi di software di grandi dimensioni e complessità, si commettono errori in fase di formulazione degli obiettivi, scrittura delle specifiche, disegno e codifica dei programmi. Tali errori si eliminano con passate ed ispezioni, riesami del progetto, collaudo dei moduli, prima separati e poi integrati. Alti errori possono intervenire durante la riprogrammazione effettuata per eliminare errori precedenti. Un sistema di software diventa infatti operativo con gli errori rimasti; questi sono scoperti gradualmente ad ogni malfunzionamento del sistema e vengono allora corretti, con l'introduzione però di nuovi errori. Inoltre, la prima release di un sistema software di grandi dimensioni e complessità non si attaglia, di norma, perfettamente ai problemi che il sistema era chiamato a risolvere.

Esso deve essere allora modificato per coniugarlo meglio con i problemi, ma nel processo di modifica si generano altri errori, di cui solo una parte vengono eliminati. Successivamente, occorre prendere atto che i problemi stessi stanno cambiando, portando ad una revisione del sistema. Non è affatto sorprendente che in tutto questo processo si creino errori, di cui solo una parte possa essere corretta. Rimane sempre nascosto un certo numero, probabilmente molto piccolo, di errori che emergono al verificarsi di una combinazione insolita di circostanze operative. In definitiva, l'utilizzatore del sistema non può mai essere assolutamente certo che tutti gli errori siano stati trovati ed eliminati.

Numero di errori. Il numero di errori generati non è mai irrilevante. Dai numerosi studi effettuati è risultato che gli errori vanno da 30 a 85 ogni mille istruzioni sorgente [7].

La variabilità nel numero di errori dipende dal tipo di software, dalle dimensioni del sistema, dalla definizione stessa del termine "errore", dalle procedure utilizzate nelle varie organizzazioni per raccogliere gli errori, e da altri fattori. Ci sono in ogni caso molti errori, ecco ciò che conta.

Per fortuna, essi vengono in massima parte trovati e corretti nel corso dello sviluppo e del collaudo, prima della distribuzione.

Ad esempio, secondo Dennis W. Fife, del National Bureau of Standards, il software distribuito per i grandi sistemi contiene tipicamente un errore ogni 300 statement di programma, ossia si ha un'incidenza del 3,3 per mille.

Un sistema di 2 milioni di righe di programma sviluppato e gestito dal Bell Communications Research ha sperimentato in field una variabilità tra 0,8 e 1,3 difetti ogni mille righe di programma sorgente, nuovo o modificato, su tre release all'anno per un periodo di 4 anni [8]. Una indagine dei Bell Laboratories sui difetti del software ha trovato un'incidenza tipica di errori tra 0,5 e 3,0 ogni mille righe di programma, secondo i dati forniti da T.R. Thomsen, presidente della AT&T Technology Systems [9].

Dimensione del sistema. Ad aggravare il problema degli errori nel caso di un sistema della mole del SDI, contribuisce il fatto che l'incidenza degli errori aumenta esponenzialmente con la dimensione del sistema software [10]. Questa scoperta è il risultato dell'analisi dei dati d'errore di vari progetti software, ma è anche abbastanza intuitiva.

La quantità di software necessaria per l'SDI è stata definita "molto grande" (dell'ordine di 10 milioni di righe di programma) dal Defensive Technologies Study Team (Commissione Fletcher), nominato dal Ministro della Difesa USA nel 1983 per studiare la fattibilità della concezione SDI [11]. Questa stima è naturalmente poco più di una semplice congettura poiché è stata fatta prima del progetto del sistema. Si sono ipotizzati anche valori fino a 30 milioni e persino 100 milioni di righe. Applicando alla stima inferiore di 10 milioni di righe il tasso ridotto di 0,5 errori ogni 1000 righe, citato da Thomsen, l'SDI conterrebbe circa 5000 errori al momento di diventare operativo, o anche di più se si tiene conto dell'aumento esponenziale della quantità di errori.

L'esperienza dei Bell Laboratories sull'unico sistema anti-balistico di grandi dimensioni del passato, il Safeguard, testimonia un notevole successo nella riduzione del tasso di errori. Questo sistema, sviluppato tra il 1969 ed il 1975, comprendeva 2.261.000 istruzioni in linguaggio assembly, di cui 789.000 relative a software per tempo reale. Si individuarono circa 5000 difetti di progettazione software, abbastanza gravi da influenzare "l'obiettivo primario di prestazioni del sistema", che si riuscì ad eliminare prima di attivare il sistema [12]. A titolo indicativo dell'impegno richiesto per la ricerca e la correzione degli errori, il collaudo dei singoli moduli ed il collaudo integrato del software per tempo reale assorbì il 60% delle risorse umane dedicate a questa sezione del progetto.

Dopo lo spiegamento del sistema attorno ai silos dei Minuteman nel North Dakota nel 1975, l'insieme di test eseguiti misero in luce solo un altro errore grave. Non è dato sapere se altri errori avrebbero potuto essere rivelati da un utilizzo operativo di lunga durata, poiché il sistema fu decommissionato nel 1976, apparentemente a causa della possibilità che un eventuale aggressore potesse facilmente metterlo in crisi aumentando il numero dei missili di attacco al di là della potenzialità difensiva prevista.

Il Safeguard era un sistema di difesa anti-balistico relativo alla fase terminale del tragitto dei missili ed era limitato dal trattato ABM del 1972 a solo 100 intercettori. Era pertanto molto più piccolo del sistema SDI, che opererà invece su tre livelli, cioè fase di lancio, tragitto intermedio e fase terminale, e sarà presumibilmente in grado di affrontare alcune migliaia di missili balistici intercontinentali. Sembra così ragionevole supporre che il software SDI sarà di gran lunga più esteso del software Safeguard, richiedendo un lavoro proporzionalmente più pesante per la rimozione degli errori.

3. Perché il software è inaffidabile

"I metodi convenzionali di sviluppo del software, applicati ampiamente nell'industria e nel settore militare, non sono adeguati alla costruzione di grandi sistemi in tempo reale che debbano essere affidabili sin dall'inizio del loro utilizzo", ha affermato Parnas. Il metodo convenzionale di sviluppo comincia con il programmatore che cerca di "pensare come un computer". Questo metodo funziona bene con problemi di piccole dimensioni, che non comportano programmi con un gran numero di diramazioni e di loop".

"Arrivati ad un punto del ragionamento in cui l'azione del computer deve dipendere da condizioni che non sono note fino a quando il programma venga eseguito, siamo costretti a scostarci dal metodo, etichettando una o più delle azioni e ricordandoci in che modo arrivare a quel punto", ha spiegato Parnas [1].

"Appena introduciamo dei loop nel programma, ad alcuni punti di esso si potrà accedere in molti modi, che dovremo ricordare tutti. Procedendo ancora più a fondo nell'algoritmo, ci rendiamo conto della necessità di informazioni su eventi precedenti ed aggiungiamo variabili alla nostra struttura di dati. Dobbiamo ora cominciare a ricordarci che cosa significa il dato ed in quali circostanze è significativo".

"Proseguendo nel tentativo di "pensare come un computer", la quantità di informazioni da ricordare cresce continuamente. Le regole semplici che definiscono i modi per arrivare a certi punti di un programma tendono a complicarsi quando si accede ad essi attraverso diramazioni da altri punti. Le regole semplici che definiscono il significato dei dati si complicano quando troviamo altri usi per variabili già esistenti ed aggiungiamo nuove variabili. È inevitabile cadere in qualche errore.".

In un sistema come l'SDI, ci saranno altre due cause che complicheranno ed appesantiranno il lavoro mentale del programmatore, e cioè la "concorrenza" di programmi e la multielaborazione, ha osservato ancora Parnas.

In conclusione, "scrivere e comprendere programmi molto ampi in tempo reale, "pensando come il computer", sarà un compito al di là delle nostre capacità intellettive".

Specifiche difficoltà dell'SDI. Tutti i sistemi di software di grandi dimensioni e complessità, sono di costruzione molto difficile, dice Parnas, ma l'SDI ha ulteriori elementi di difficoltà. Anzitutto, i suoi algoritmi devono essere adattati alle caratteristiche dei bersagli e dei mezzi per trarre in inganno usati dal nemico. Non si potrà mai essere sicuri di aver centrato in pieno l'algoritmo giusto, poiché il potenziale aggressore tiene sotto controllo le suddette caratteristiche e le può cambiare di tanto in tanto.

In secondo luogo, l'aggressore può fare in modo di sovraccaricare il sistema. I programmi SDI, per poter operare in tempo reale, devono avere piani precalcolati sulla base di assunzioni aprioristiche sulle caratteristiche dell'attacco. Cambiando queste caratteristiche, ad esempio dirigendo verso un solo obbiettivo un numero inaspettatamente alto di missili, il nemico potrebbe sovraccaricare i computer relativi, provocandone la paralisi o quanto meno rendendoli incapaci di rintracciare e gestire tutti i missili in arrivo.

La terza considerazione da fare, secondo Parnas, riguarda il costo estremamente elevato del collocamento di computer e mezzi trasmissivi nello spazio, che risultano, oltretutto, estremamente vulnerabili. Queste difficoltà sono ingigantite dalla necessità di ridondanze, per ragioni di affidabilità, e di disporre di un numero sufficientemente elevato di unità per ovviare a contromisure dell'aggressore.

"Un'alta affidabilità è ottenibile solo se i guasti dei singoli componenti del sistema sono statisticamente indipendenti; ma non è questo il caso per un sistema sottoposto ad attacchi coordinati".

Verifica del prodotto. L'apparente contraddizione, rappresentata dalla esistenza di programmi di grandi dimensioni, ma tuttavia di affidabilità sufficiente per consentirne l'utilizzo, viene spiegata da Parnas con l'ottimizzazione per approssimazioni successive. I programmi non nascono esenti da errori; i primi test mirano ad un certo livello di affidabilità, che viene successivamente via via elevato, eliminando gli errori durante l'utilizzo effettivo.

I progettisti rendono affidabili i prodotti con gli strumenti dell'analisi matematica, con l'analisi esaustiva di casi specifici e con test di durata, tutti opportunamente combinati; questi metodi si rivelano però inadeguati nel caso del software. Infatti gli strumenti dell'analisi matematica lavorano bene su funzioni continue, spiega Parnas, ma il software è un sistema a stati discreti, non descrivibile con tali funzioni.

Un altro approccio analitico consiste nella prova dei programmi, ma "i migliori strumenti di verifica matematica del software servono solo per programmi relativamente piccoli ed il loro intrinseco grado di approssimazione può nascondere errori gravi".

Parnas non prevede che questi strumenti possano perfezionarsi sensibilmente nell'arco di tempo dell'SDI.

"La verifica dei prodotti per mezzo di una analisi esauriente di casi specifici è fattibile solo quando si ha un numero limitato di casi da analizzare, oppure quando il prodotto ha una struttura altamente ripetitiva. Il software invece presenta un numero enorme di stati e non ha alcuna regolarità. È impossibile eseguire un collaudo esauriente di un sistema di software di grandi dimensioni e complessità perché esso ha un numero troppo elevato di condizioni di input così come di transizioni tra stati interni".

Sistemi meno critici dell'SDI sembra entrino in servizio ad un livello di affidabilità del 95% circa, lasciando quindi ancora in essere il 5% degli errori totali [10]. Questi errori re-

sidui sono presi in carico durante la fase di "manutenzione" che può durare mesi ed anni.

L'SDI dovrà presumibilmente entrare in servizio con un certo numero di errori residui, di cui una parte sarà trovata ed eliminata nel corso di test, in linea o simulati, durante la fase operativa.

Il suo pieno utilizzo operativo significa purtroppo l'impiego in un conflitto nucleare. "Non sarà possibile metterlo a punto se fallisce", ha detto Parnas, "poiché in quel momento avremo da pensare ad altro, ammesso che potremo ancora pensare".

4. Il problema computazionale

La commissione SDIO dedicata al "Computing in Support of Battle Management", nota anche sotto il nome di Gruppo di studio Easport, è stata realistica nel prendere atto che "tutti i sistemi complessi possano contenere errori di software" [5].

Il problema che essa si è posto nella sua lunga relazione (30.000 parole) è su cosa fare in proposito. Le risposte al problema riguardano una serie di temi: l'architettura del sistema, gli sviluppi esplorativi, la potenza di calcolo, i collaudi, la tolleranza all'errore, la ricerca sul software.

L'architettura. La commissione ha raccomandato lo sviluppo da parte di SDIO di una architettura aperta e distribuita. "Un esercito non può nè deve coordinare le singole azioni di ogni soldato tramite il comandante in capo. Al contrario, le responsabilità e l'autorità sono delegate secondo una catena gerarchica; analogamente, un sistema di difesa strategico non necessita nè deve essere coordinato in modo rigido".

La commissione ha esemplificato questo concetto simulando l'assegnazione di armi di difesa ai singoli missili durante la fase di lancio, nei due casi di architettura centralizzata e decentralizzata. Si è così appurato che l'approccio decentralizzato richiederebbe, per distruggere un certo numero di missili, un numero di colpi di circa il 20% superiore rispetto ad un sistema perfettamente coordinato. "Il bilancio complessivo segna quindi un certo risparmio nel costo dell'hardware a favore di un sistema perfettamente coordinato; in cambio, però, esso presenta una maggiore complessità del software e una minore collaudabilità".

Appare ovvio che l'SDI non sarà mai un sistema statico, in quanto appariranno periodicamente nuove versioni, con nuovi modelli di armi, sensori, collegamenti trasmissivi e computer. Si faranno anche modifiche per rispondere efficacemente alle contromisure dell'avversario. Pertanto, secondo il parere della commissione, l'architettura deve essere formulata come un "sistema aperto", che consenta il rapido inserimento di elementi non previsti o modificati.

Ciò che importa, ha sottolineato il Gruppo Eastport, è che negli studi sui compromessi architetturali, vengano considerati i problemi del software: l'affidabilità e la collaudabilità.

Sviluppi esplorativi. La commissione ha definito una deformazione della realtà il paradigma del "diagramma a cascata" dello sviluppo software, cioè la concezione di un progetto che procede unidirezionalmente dalla definizione dei requisiti alle specifiche di progetto, dalla codifica del programma ai test ed alla consegna del prodotto. In pratica avviene invece che ognuna delle suddette fasi può, come spesso succede, riciclare su una delle precedenti.

La commissione ritiene molto ampia la latitudine delle architetture possibili e che non ci sono metodi analitici per poter valutare la fattibilità di ciascuna di esse. Per un'indagine approfondita di questi problemi ha raccomandato pertanto la costruzione di diversi sistemi prototipali di software da parte di diverse organizzazioni. L'uso della "prototipizzazione veloce" consente di vedere il comportamento del sistema prima della sua effettiva e completa realizzazione.

Questi progetti dovrebbero, secondo la commissione, essere inizialmente abbastanza modesti, a livello di circa 25 anni - uomo, e dovrebbero lavorare su ipotesi o simulazioni approssimate delle caratteristiche dei sensori e delle armi. Ogni progetto dovrà essere suscettibile di essere ampliato ad uno sviluppo esplorativo su scala maggiore. Al crescere delle dimensioni e della potenza dei sistemi prototipali, si eleverà anche il livello di realismo e di dettaglio delle simulazioni".

I programmi prototipali dovrebbero poter stabilire una gamma di software fattibile, portando in definitiva alla definizione di specifiche soddisfacenti. Da tali progetti dovrebbero, in particolare, emergere precise definizioni della formulazione del sistema in termini di moduli e dei relativi protocolli di interfaccia.

Potenza di calcolo. Il progresso della tecnologia hardware fa prevedere un continuo aumento della velocità di calcolo, insieme alla riduzione di ingombro, peso e consumo di energia dei computer. La raccomandazione generale della commissione è di impiegare la maggior potenza disponibile per semplificare altri aspetti, in particolare il software, diventando sempre meno necessario complicare quest'ultimo per ovviare a mancanza di potenza di calcolo.

Nell'area dello sviluppo software, la maggior potenza di calcolo potrebbe, ad esempio, consentire ai progettisti di creare nuovi strumenti e tecniche, come ambienti diagnostici molto avanzati, o di supportare tecniche di controllo semantico a livello molto più dettagliato delle attuali. Potrebbe inoltre, secondo la Commissione, diventare fattibile l'animazione di algoritmi in tempo reale.

Prove e collaudi. Concorrono in misura determinante all'affidabilità dell'SDI, e la commissione ha indicato gli approcci più promettenti al problema:

1. Sviluppo di architetture di sistema con un fattore relativamente modesto di dipendenza dal coordinamento, o in cui le informazioni di coordinamento vengono utilizzate solo se disponibili. Queste architetture presentano una maggiore semplicità di collaudo delle loro varie componenti.
2. Un vasto impiego di simulazioni su computer ad altissima velocità e/o programmabili con concorrenzialità elevata, per collaudare le istruzioni e gli algoritmi operativi in condizioni di estrema variabilità dei parametri della situazione.
3. Collaudo dei componenti e dell'intero sistema nell'effettivo ambiente operativo, con lo scambio di dati simulati tra sensori e piattaforme di lancio, nell'impossibilità di operare in modo reale.
4. Continuare le prove ed i test per l'intera vita del sistema installato.

I sistemi organizzati a livelli gerarchici si prestano particolarmente a questo tipo di collaudo, ha osservato la commissione. Inoltre con sistemi suddivisibili si può ridurre il numero di stati da provare. D'altra parte, disponendo di maggior potenza di calcolo è possibile aumentare tale numero.

Tolleranza all'errore. La progettazione di hardware per la presa in carico di guasti e per le procedure di ripristino è una pratica già consolidata, ma la progettazione di software a tolleranza d'errore si trova ad uno stadio meno progredito. "SDIO deve concentrarsi a fondo nella ricerca, affinamento e valutazione di tecniche di progettazione software, che rendano possibile l'esecuzione delle funzioni computazionali malgrado guasti di hardware ed errori di software", è la raccomandazione della commissione.

Una delle tecniche proposte consiste nell'incaricare diversi gruppi di programmazione di scrivere indipendentemente un programma basato sulle stesse specifiche. Poi i vari programmi vengono eseguiti insieme con una procedura decisionale che ne confronta i risultati.

Per una valutazione di questa tecnica, la commissione ha raccomandato esperimenti più ampi di quanto fosse possibile nel passato, mettendo peraltro sull'avviso circa la possibilità di un elevato grado di correlazione degli errori di programmazione nei diversi programmi.

Tra le altre tecniche si possono citare i processi di sorveglianza (watchdog), le revisioni (audit) della struttura dati, i blocchi di ripristino, i controlli "a caldo" (runtime checking) e i processi decisionali fondati su distribuzioni probabilistiche.

Il runtime checking viene già applicato, ad esempio, per imporre limiti alle variabili, ma secondo la commissione, potrebbe trovare un impiego più vasto. Le decisioni basate su distribuzioni probabilistiche tendono a contenere il fenomeno della propagazione di un errore; ogni dato critico verrebbe, infatti, rappresentato non da un valore singolo, ma da una funzione di densità probabilistica.

La ricerca sul software. Altre aree di ricerca sono state delineate dalla commissione e sono qui di seguito indicate.

Tecniche matematiche di verifica.
Benché i metodi di verifica formali presentino diverse limitazioni e sia probabilmente lento un loro progresso, essi potrebbero tuttavia servire almeno per qualche modulo di piccole dimensioni ma di importanza fondamentale.

Linguaggi di specifica.
Esiste una classe di linguaggi che danno al programmatore solide garanzie che, aggiungendo dettagli a livelli successivi, la semantica della rappresentazione più dettagliata rispetti quella dell'iterazione precedente.
Una seconda classe di tali linguaggi, i linguaggi di specifica esecutiva, induce invece ad una maggiore precisione del programmatore, e quindi ad una minore ambiguità, riducendo così gli errori.

Elaborazione parallela, concorrente, o distribuita.
Si diffondono sempre più sistemi con queste funzionalità, cui non si accompagna però una crescita proporzionale delle capacità di sfruttamento. In proposito, la commissione ha raccomandato a SDIO di "costituire centri di studio sui problemi algoritmici, di compilazione e dei sistemi operativi, nell'ambito di sistemi a programmazione concorrente potenzialmente adatti a contesti applicativi generalizzati".

Gruppi di sviluppo.
L'avvento di reti con stazioni di lavoro computerizzate, di linguaggi di specifica precisi e di altri strumenti avanzati di sviluppo, ha aperto nuove possibilità di strutturazione dei gruppi di sviluppo software. La commissione suggerisce in proposito di sperimentare gruppi di sviluppo opportunamente strutturati, e inoltre di condurre uno studio storico sui precedenti grandi progetti di software militare.

Ambienti di software.
Gli ambienti dotati di strumentazione con velocità molto spinte sembrano creare un nuovo stile di sviluppo dei programmi, con metodi nuovi e più efficienti. Il programmatore può, ad esempio, spezzare un programma e rimetterlo insieme in una nuova struttura, più facilmente che un tempo. Si deve però studiare come struttare al meglio il nuovo stile di programmazione e la maggiore potenzialità dell'ambiente operativo.

Manutenzione del software.
La commissione si aspetta un sostanziale impulso nelle attività di manutenzione del software, facendo rilevare la scarsa ricerca svolta sinora sugli strumenti di modifica. Allo scopo, la commissione propone, ad esempio, lo sviluppo di metodi per la realizzazione ed il monitoraggio di moduli di programma che riflettano i requisiti del sistema, facilitando il compito di adattare eventuali cambiamenti a tali requisiti, anche se la parte di programma interessata si trova dispersa anziché concentrata.

5. Ci saranno comunque errori

L'intera storia dello sviluppo di sistemi di software di grandi dimensioni e complessità esclude la possibilità di eliminare completamente gli errori. Il Gruppo di studio Eastport va segnalato per l'approfondita analisi del problema e per aver evidenziato modalità atte a ridurre o tollerare gli errori. Non siamo in grado di sapere ora, prima ancora di aver definito l'architettura del sistema, di aver effettuato la ricerca, e di aver strutturato l'organizzazione di sviluppo del software, in quale misura verranno attuate le raccomandazioni del gruppo di studio.

Ammesso che l'attività nel software venga portata avanti con impegno paragonabile alle attività di analisi sopracitate, rimarrebbero comunque almeno 0,5 errori ogni mille istruzioni sorgente al momento

Per raggiungere il 99,9% di affidabilità

I risultati ottenuti dagli studi sulla teoria della affidabilità del software consentono di prevedere il numero approssimato di errori in un programma e l'entità del lavoro richiesto per ridurli [10].

Il numero di errori commessi e successivamente trovati e corretti è riportato sulla curva di Rayleigh di figura 1, in cui l'asse delle y rappresenta gli errori al mese e l'asse delle x il tempo espresso in mesi. La curva è stata ipotizzata per un sistema di comando e controllo costituito da 1 milione di righe di codice sorgente, indicativamente pari ad uno dei sottosistemi dell'SDI.

Le linee verticali indicano le pietre miliari dello sviluppo e della fase operativa del software:

(1) Progetto preliminare
(2) Revisione critica
(3) Prima stesura del codice
(4) Test del sistema integrato
(5) Collaudo del sistema con l'utente
(6) Capacità operativa iniziale
(7) Piena capacità operativa (affidabilità del 95%)
(8) Affidabilità del 99%
(9) Affidabilità del 99,9%

Per un sistema di queste dimensioni e complessità, raggiungere un livello di affidabilità del 95%, cioè depurato dal 95% degli errori, richiede circa cinque anni. Altri due anni e mezzo di esercizio e manutenzione sono necessari per raggiungere un livello di affidabilità del 99,9%.

Gli altri sottosistemi SDI, se concepiti in modo indipendente da questo e tra loro, possono essere sviluppati in parallelo, ma si deve tener conto di un tempo ulteriore per la loro integrazione. Nel caso di una certa interdipendenza tra i sottosistemi, il loro sviluppo dovrebbe avvenire, almeno parzialmente, in serie, con la conseguente probabile estensione a 10 anni o più del tempo occorrente per raggiungere un'affidabilità del 99,9%.

La Fig. 2 mostra come si siano totalizzati 6.022 errori per arrivare alla piena operatività (affidabilità del 95%), e 6.333 errori per arrivare ad una affidabilità del 99,9%. In piena operatività restano da trovare 316 errori e con un'affidabilità del 99,9% rimangono ancora probabilmente sei errori.

Supponendo che l'intero sistema di software SDI sia di dimensioni dieci volte superiore, il numero totale di errori sarebbe all'incirca di 60.000, con circa 60 errori ancora da trovare al 99,9% di affidabilità.

Il calcolo di questa curva d'errore è basato sul consuntivo degli errori rilevati in numero limitato di sistemi; inoltre tali dati non sono stati sempre definiti in modo uniforme. Non possiamo perciò pretendere che la nostra proiezione abbia un alto grado di precisione. Con questa curva si intende piuttosto illustrare l'andamento del fenomeno, ossia il modo con cui gli errori si manifestano e sono eliminati fino al 99,9% del totale.

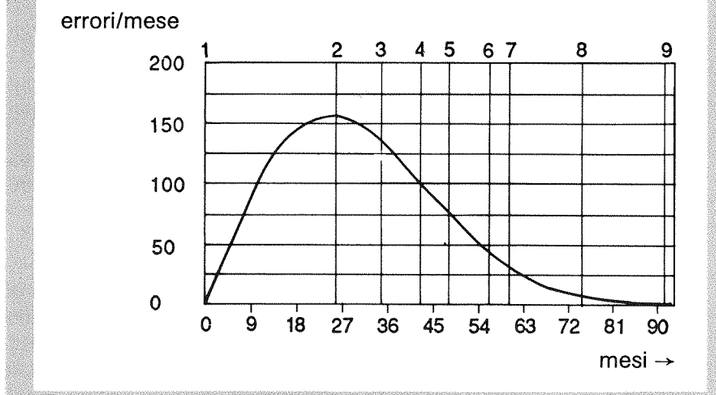


Fig. 1 - Tasso di errori atteso per un sistema di software in tempo reale di 1 milione di righe di programma sorgente. Le linee verticali (1-9) indicano gli stadi fondamentali dello sviluppo. Livelli previsti di affidabilità: 95% allo stadio 7; 99% allo stadio 8; 99,9% allo stadio 9.

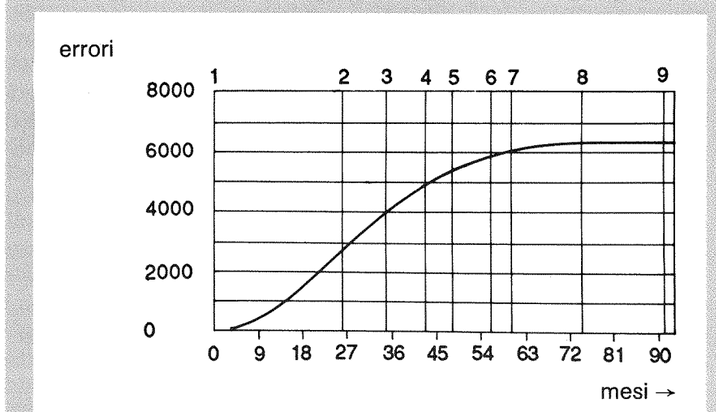


Fig. 2 - Numero di errori attesi per lo stesso sistema software di fig. 1.

dell'installazione. Nell'ipotesi che il software relativo alla fase terminale della traiettoria dei missili consistesse di circa 800.000 istruzioni, come il software per tempo reale Safeguard, e che le due fasi precedenti di lancio e di volo intermedio avessero una dimensione equivalente, le tre sezioni di software avrebbero complessivamente 2,4 milioni di istruzioni. Al tasso d'errore ipotizzato, questo insieme di software conterrebbe circa 1200 errori al momento di entrare in servizio.

Scendere a tale livello di errori comporta una efficienza organizzativa particolarmente elevata. Per raggiungerlo, non è ovviamente sufficiente, secondo il Gruppo Eastport, che SDIO sfrutti le tecniche software più progredite attualmente disponibili, ma SDIO deve anche imparare ad usare nuove tecniche, frutto della ricerca promossa dalla stessa SDIO.

Il miglioramento qualitativo che potrebbe derivare al sistema da un progresso sostanziale della tecnologia software, risulta, però, alquanto problematico se si considera che - sempre secondo il Gruppo Eastport - parecchi dei maggiori contraenti degli sviluppi software del Ministero della Difesa USA sono attualmente molto in ritardo sullo stato dell'arte.

A questo proposito il Gruppo ha dedicato un capitolo del suo rapporto al program management, con una serie di raccomandazioni intese a colmare questa lacuna.

6. Conclusioni

Mi pare di dover concludere che, pur ammettendo il massimo impegno di tutti gli enti e le persone coinvolte, permarranno errori di software e che falle dello scudo protettivo saranno inevitabili, anche se per poche testate nucleari. Siamo così ricondotti all'obbiettivo originario della Strategic Defense Initiative. Se era quello di rendere "impotenti e obsolete" le armi nucleari, come affermato dal Presidente Reagan nel discorso di presentazione del concetto SDI nel marzo 1983, allora queste armi costituiranno ancora una minaccia molto potente. Se, invece, l'obbiettivo era quello di una difesa parziale, per la protezione dei silos dei missili di rappresaglia e per assicurare la sopravvivenza di almeno una parte della popolazione, ossia per evitare una catastrofe totale, allora questi obbiettivi sono alla portata dell'SDI, malgrado gli errori che il sistema potrà avere.

7. Bibliografia

[1] David Lorge Parnas, "Software Aspects of Strategic Defense Systems," *American Scientist,* Sept.-Oct. 1985, pp. 432-440.

[2] Frederick P. Brooks, *Testimony before the Hearing of the Senate Armed Services Committee on Strategic and Nuclear Forces,* Oct. 30, 1985.

[3] Letter dated Oct. 18, 1985 from Congressman Robert J. Mrazek of the Committee of Appropriations to Lt. Gen. James A. Abrahamson, USAF, SDIO Director.

[4] Letter (undated but apparently in late October, 1985) from Lt. Gen. James A. Abrahamson to Congressman Robert J. Mrazek.

[5] Danny Cohen, chairman, Eastport Study Group, Summer Study 1985, Dec. 1985, 70 pp.

[6] Harold Brown, "Is SDI Technically Feasible?," *Foreign Affairs,* Vol. 64, No. 3, 1986, pp. 435-454.

[7] Barry W. Boehm, *Software Engineering Economics,* 1981, Prentice-Hall, Inc., Englewood Cliffs, N.J., p. 383.

[8] Nathan H. Petschenik, "Pratical Priorities in System Testing", *Software,* Sept. 1985, pp. 18-23.

[9] T.R. Thomsen, letter of Nov. 1, 1985 to Lt. Gen. James A. Abrahamson.

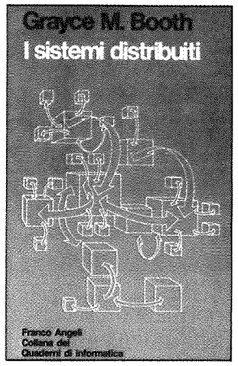
[10] Lawrence H. Putnam, Douglas T. Putnam, and Lauren P. Thayer, "Quality in software is Free (Almost)", *Proc. Int'l Soc. Parametric Analysts,* Seventh Ann. Conf. May 1985, 37 pp.

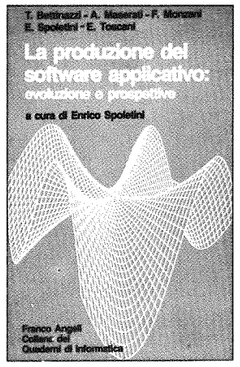
[11] James C. Fletcher, study chairman, *Report of the Study on Eliminating the Threat Posed by Nuclear Ballistic Missiles,* SDI Organization, 1983.

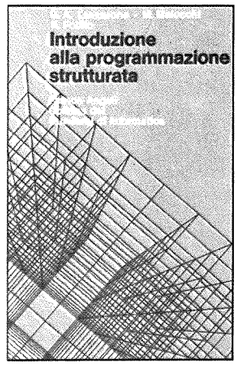
[12] W.E. Stephenson, "An Analysis of the Resources Used in the Safeguard System Software Development," *Proc. 2nd Int'l Conf. Software Engineering,* 1976, pp. 312-321.

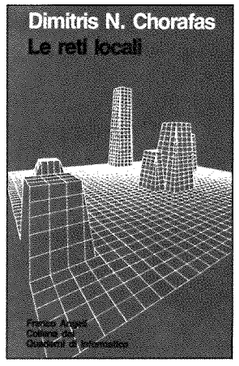
Collana dei Quaderni di Informatica

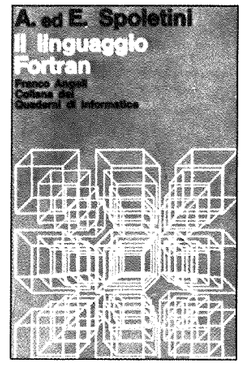
La collana di testi "Quaderni di Informatica" rappresenta la proiezione sul piano librario della presente rivista. La direzione scientifica della collana (che è edita da F. Angeli) è del "Centro Studi Informatica e Automazione" della Honeywell Bull Italia. A titolo di rassegna sintetica, riproduciamo qui di seguito le copertine dei volumi finora usciti, che coprono alcune delle più importanti aree dell'informatica.

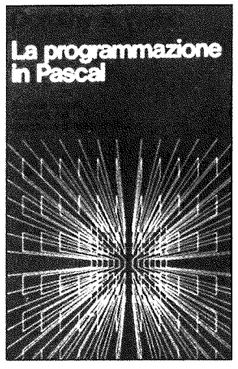






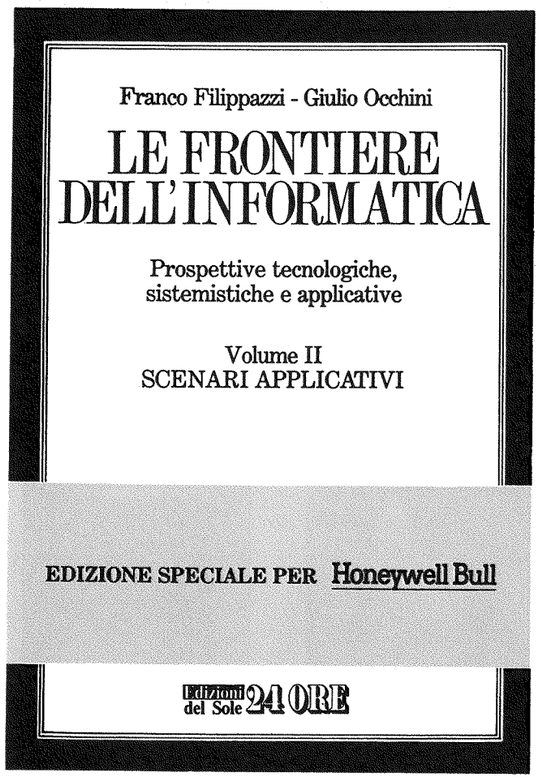






BIBLIOTECA
NOVITÀ

Franco Filippazzi, Giulio Occhini: Le frontiere dell'informatica - Vol. II - Edizioni del Sole/ 24 Ore, 1987 (L. 28.000).



È questo il secondo volume de *Le frontiere dell'informatica,* dedicato agli scenari applicativi. Esso fa seguito al volume pubblicato nel 1986, dedicato invece alle prospettive tecnologiche e architettoniche del computer.
Le applicazioni del computer ormai non si contano più. Non esiste settore delle attività umane in cui i metodi, le tecniche e gli strumenti informatici non vengano impiegati già oggi o non possano esserlo in futuro.
Cercare di rendere "leggibile" questo quadro complesso ed estremamente diversificato è l'obiettivo del secondo volume de *Le frontiere dell'informatica,* dedicato appunto agli scenari applicativi. Rendere leggibile significa sintetizzare in un numero limitato di pagine una mole immane di materiale, individuando uno schema semplice ma organico di strutturazione ed esposizione della materia.
In queste pagine si è cercato non solo di enucleare le macroaree applicative ma, per ciascuna di esse, di evidenziare il valore innovativo intrinseco all'uso del computer. In altri termini, di sottolineare il contributo di valore aggiunto che l'informatica può dare allo sviluppo delle attività economiche e sociali della nostra epoca postindustriale.
Anche questo volume è rivolto più al futuro che al presente, delineando le linee di tendenza e i potenziali sviluppi applicativi. L'esposizione, pur rigorosa, rimane accessibile anche ai non specialisti di informatica: opinion maker, manager, professionisti, studiosi e, in generale, uomini di cultura.
Come il precedente, anche questo volume è stato sviluppato nell'ambito del *Centro Studi Informatica e Automazione,* una struttura creata dalla Honeywell Bull Italia per condurre ricerche sugli orientamenti di fondo di tali settori.

LE FRONTIERE DELL'INFORMATICA - Volume II

SOMMARIO