Liste di tutti i generi

La Finestra di Dialogo è il principale strumento di interazione tra l'utente e tutte le applicazioni Windows. È quindi naturale che anche i suoi componenti, come le Caselle di Testo, i Pulsanti di Opzione, le Liste semplici, le Liste combinate, ecc., il cui funzionamento è tra l'altro assolutamente intuitivo, siano noti a chiunque sappia usare, anche in modo elementare, Windows. Inoltre da una parte queste modalità operative, comuni a tutti gli applicativi, facilitano enormemente all'utente l'approccio verso i nuovi prodotti, dall'altra gli applicativi più evoluti, in cui sia possibile realizzare interfacce personalizzate, permettono di costruire e quindi di sfruttare proprio questi stessi elementi. Lo scopo dell'articolo è quello di approfondire l'argomento Liste, che sono una delle componenti più importanti e significative nelle Finestre di Dialogo, vedendo non tanto come si usano quanto come si costruiscono, con i prodotti che ne permettono la personalizzazione. Con l'occasione studieremo anche alcune problematiche connesse ad un loro uso generalizzato

di Francesco Petroni

Liste corte e liste lunghe, liste fisse e liste dinamiche

Una Finestra di Dialogo molto conosciuta è quella che si richiama quando si voglia scegliere, in qualsiasi applicativo Windows, un carattere (fig.1). Contiene tre Combo Box, delle liste scorrevoli nelle quali si può scegliere il font desiderato, tra quelli disponibili nel sistema, oppure se ne può scrivere il nome, l'attributo desiderato oppure ancora la dimensione.

Ad esempio nella lista non è presente la dimensione 13. Se si vuole proprio questa misura occorre scriverla nella casella di testo in cima alla lista.

Combo sta per «combinata». La Combo Box è infatti la combinazione tra una casella di testo in cui si scrive a mano e una lista semplice in cui si sceglie. In altre parole in una Combo Box si può scegliere l'elemento tra quelli proposti oppure si può digitare, se non c'è nella lista.

Gli oggetti Windows (in termine tecnico si chiamano Controlli) che lavorano su liste sono almeno quattro. La terminologia che li definisce è varia in quanto i vari prodotti usano nomi differenti, ulteriormente differenziati dal fatto che si usano spesso i nomi in inglese.

Per categorizzare usiamo la terminologia proposta da Excel 5.0 in italiano:

- Casella di riepilogo: elenco di stringhe di testo di cui se ne possono selezionare una o più di una;
- Casella a discesa: singola casella di testo non modificabile a cui è affiancata una freccia verso il basso che, se selezionata, mostra la lista delle stringhe;
- Casella di riepilogo combinata: in più rispetto alla semplice c'è la possibilità di digitare nella casella di testo una stringa non presente nella lista;
- Casella a discesa combinata: anche in questo caso il «plus» è dato dalla possibilità di digitare una stringa non presente nella lista.

Quindi il fatto che la casella sia «a discesa» è un puro fatto estetico/operativo. Invece il fatto che la casella sia «combinata» è un fatto sostanziale in quanto viene permessa la digitazione di stringhe non presenti nella lista.

Inoltre è chiaro che la facilità con cui si esegue la scelta di una stringa in una casella combinata dipende anche da altri fattori. Ad esempio dipende dalla lunghezza della lista: una cosa è scegliere tra dieci possibilità, una cosa è scegliere tra cinquecento, oppure dipende dal fatto che la lista sia o meno in ordine alfabetico, oppure che gli elementi della lista siano semplici stringhe o combinazioni di stringhe, ad esempio nome e cognome.

Insomma anche il problema di scegliere un elemento in una lista presenta una serie di varianti operative, che dobbiamo esplorare:

Figura 1 - Una classica List Box per la scelta dei Caratteri. Un qualsiasi utilizzatore di Windows sa usare le Finestre di Dialogo, e all'interno di queste, le Liste. Una delle finestre più usate è quella «comune» che serve per scegliere, da qualsiasi prodotto, il carattere con il quale scrivere... qualsiasi cosa. Il meccanismo è evidente, si sceglie il carattere nella prima lista, poi nella seconda appaiono gli attributi permessi per quel font e nella terza le sue dimensioni, misurate in punti tipografici. In basso il facsimile del risultato ottenuto. Nei nostri esercizi vogliamo approfondire l'argomento lista e sperimentare, con vari prodotti, la loro costruzione.

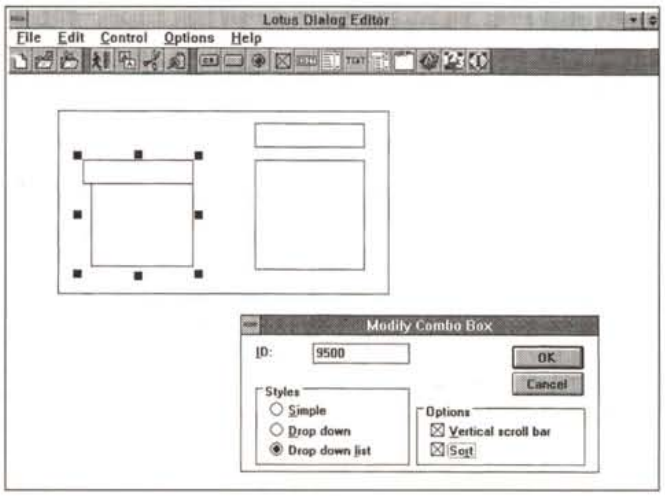


Figura 2 - Lotus Dialog Editor - Caratteristiche della Lista.
In generale possiamo dire che qualsiasi prodotto che permetta la programmazione di oggetti Windows permette la definizione esatta del tipo di Lista. Qui vediamo il Dialog Box Editor della Lotus che suggerisce le varianti: lista semplice, a caduta, con scroll bar, ordinata, ecc. Se si vuole qualcosa di più, e noi la vogliamo, occorre intervenire con un po' di programmazione.

– liste lunghe e liste corte;
– liste fisse o liste cui si possono aggiungere «al volo» elementi;
– liste ordinate o liste non ordinate;
– liste di stringhe complesse ottenute da manipolazione di stringhe semplici;
– liste il cui contenuto dipende al condizioni esterne.

Non analizzeremo la possibilità di selezioni multiple, permesse ad esempio dal File Manager di Windows.

In generale possiamo dire che qualsiasi prodotto che permetta la programmazione di oggetti permette la definizione esatta del tipo di Lista (in figura 2 vediamo una recente versione del Dialog Box Editor della Lotus) ad esempio: semplice, a caduta, con scroll bar, ordinata, ecc. Se si vuole qualcosa di più, e noi la vogliamo, occorre in generale intervenire con la programmazione.

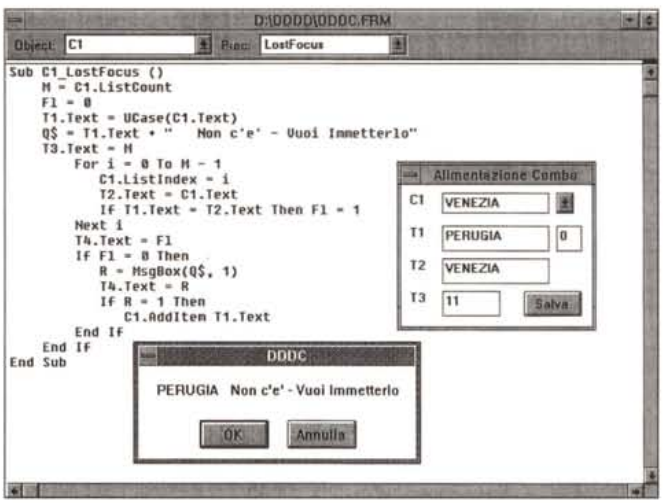


Figura 3 - Visual Basic 3.0 - Alimentazione di una Combo Box.
Una List Box, ovvero una Casella di Riepilogo, ha un contenuto fisso. Una ComboBox, ovvero una Casella Combinata, combina, appunto, le funzioni di una lista, dalla quale scegliere, e una casella di testo, nella quale scrivere un dato non presente nella lista. Il problema che risolviamo in questo esercizio, realizzato con Visual Basic 3.0, è quello di permettere di aggiungere, a richiesta, l'elemento nuovo nella lista, e poi di salvarlo direttamente nel file, dato che la lista si appoggia su un file.

Alimentazione di una Lista

Cominciamo con un esercizio realizzato in Visual Basic, un po' meno «scemo» (l'esercizio) di quello che appare vedendo la relativa figura 3. Nella figura c'è un collage in cui si vedono la Form «Alimentazione Combo», una Message Box che avverte che la stringa digitata non è presente nella lista, e il listato relativo all'evento Lost Focus (che si verifica quando si esce dalla Combo).

La Combo (identificata con C1) presenta una serie di città (supposte già caricate in sede di lancio dell'applicazione, ad esempio in conseguenza dell'evento Form Load). Trattandosi di

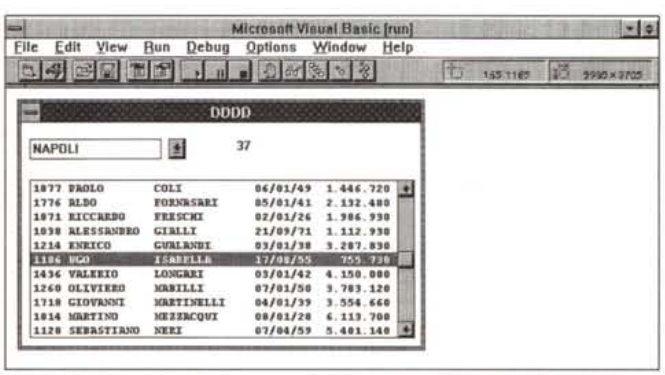


Figura 4 - Visual Basic 3.0 - Lista pluricampo in risposta alla Combo-Box.
Si sceglie la Sede desiderata nella ComboBox in alto. In conseguenza di questa scelta viene generata una Lista, in cui vengono assemblati più campi, dei soli record appartenenti a quella Sede. Tutto qui. A differenza dell'esempio che vedremo immediatamente dopo, qui i dati sono già in memoria, parcheggiati in una serie di vettori. Parte di questi dati vengono poi scaricati nella Lista.

una Combo Box si può scegliere una città della lista oppure si può digitare, nella parte superiore della Combo, una nuova città. A questo punto occorre decidere se la nuova città digitata deve o meno entrare a far parte anche lei della Lista. A tale scopo viene visualizzata la Message Box, e in caso di risposta positiva, viene eseguito il comando Additem che è quello che aggiunge l'elemento alla lista.

Nella Form vengono visualizzate altre tre caselle di testo, la T1, in cui viene visualizzata la stringa digitata o scelta, la T2, di cui parleremo dopo, e la T3 che contiene in numero di elementi della lista (caratteristica ListCount del Controllo Lista).

Ebbene supponiamo di aver scritto nella Combo una città. Il programma confronta questa nuova stringa con tutte le precedenti, in pratica eseguendo un ciclo da 1 al numero di elementi e posizionandosi così su ciascun elemento della lista (caratteristiche ListIndex per posizionarsi, Text per leggere) e confrontando il singolo elemento con la stringa digitata. Se la stringa già c'è, evenienza che accade anche quando si sceglie una città direttamente nella lista, non succede nulla. Viceversa se la stringa non c'è viene eseguita la Message Box e quindi inoltrata la richiesta all'utente. Se la risposta è positiva viene eseguita la istruzione Additem. Una serie di considerazioni:

– il fatto che la lista appaia in ordine alfabetico è una caratteristica della lista che può essere impostata all'inizio, direttamente in fase di disegno della Form e su cui non è più necessario intervenire;

– il programma non si occupa di memorizzare la lista. È abbastanza semplice per chi programma in VB inserire istruzioni per leggere la lista iniziale da un semplicissimo File Sequenziale e per riscriverla alla fine, in quanto potrebbe essere stata modificata;

– gli oggetti T1,T2,T3 non servono a nulla. Servono in fase di test per vedere se la procedura sta funzionando a dovere. Avendoli comunque inseriti possiamo notare come sia possibile usare delle espressioni del tipo:

T2.Text=T1.Text

che fanno riferimento direttamente alle caratteristiche degli oggetti, senza doversi appoggiare su variabili o quant'altro.

In definitiva anche quando si deve intervenire con programmi l'uso delle Liste è assolutamente facilitato.

```
Dim C$(307), N$(307), G$(307), S$(307), L$(307), D$(307), Q$(307), I$(307)

Sub Combo1_LostFocus ()
    B$ = "                ": LIST1.Clear
    For K = 1 To 307
      If S$(K) = COMBO1.Text Then
        Y$ = C$(K) + " " + Mid(N$(K) + B$, 1, 12) + Mid(G$(K) + B$, 1, 14)
        Y$ = Y$ + D$(K) + " " + I$(K)
        LIST1.AddItem Y$
      End If
    Next K
    KF = LIST1.ListCount: TEXT1.Text = LIST1.ListCount
End Sub

Function DS (X)
    D1 = Val(Mid$(X, 1, 4)): D2 = Val(Mid$(X, 5, 2)): D3 = Val(Mid$(X, 7, 2))
    DD = DateSerial(D1, D2, D3): DS = Format(DD, "DD/MM/YY")
End Function

Sub Form_Load ()
    COMBO1.AddItem "BARI": COMBO1.AddItem "FIRENZE"
    COMBO1.AddItem "GENOVA": COMBO1.AddItem "MILANO"
    COMBO1.AddItem "NAPOLI": COMBO1.AddItem "PALERMO"
    COMBO1.AddItem "ROMA": COMBO1.AddItem "TORINO"
    K = 1: Open "D:\DDDD\DDDD.TXT" For Input As 1
    Do While Not EOF(1)
      Input #1, C$(K), N$(K), G$(K), S$(K), L$(K), D0, Q$(K), I0, P0
      D$(K) = DS(D0): I0 = I0 * (1 - P0)
      I$(K) = Right("     " + Format(I0, "###,###,###"), 10)
      K = K + 1
    Loop: Close 1
End Sub
```

Figura 5 - Visual Basic 3.0 - Lista pluricampo.
Il Sub Form Load si occupa del caricamento dei dati, provenienti da un file sequenziale con i campi delimitati. Alcuni vengono trattati prima di essere scaricati nei vettori. Viene anche usata una Function personalizzata, che serve per ricostruire il formato della data, che arriva nel formato «AAAAMMGG». La lista viene costruita al verificarsi dell'evento LostFocus della Combo, ovvero quando si esce dalla Combo.

Liste lunghe

Se la lista di stringhe da mostrare nella List Box o nella Combo Box è troppo lunga diventa più difficile usarla. Una cosa è scegliere tra dieci elementi, una cosa è scegliere tra mille. Il problema, detto più in generale, è quello di trovare dei sistemi per limitare la lista usando dei controlli che agiscano a monte.

Come prima usiamo VB3 per realizzare il nostro esercizietto, di cui vediamo il risultato in figura 4, che mostra una lista «multicampo» il cui contenuto dipende dalla scelta eseguita sulla Combo superiore. In figura 5 vediamo i listati relativi agli eventi principali e che commentiamo.

I Dati sono in pratica costituiti da un file sequenziale, 8 campi e 307 record,

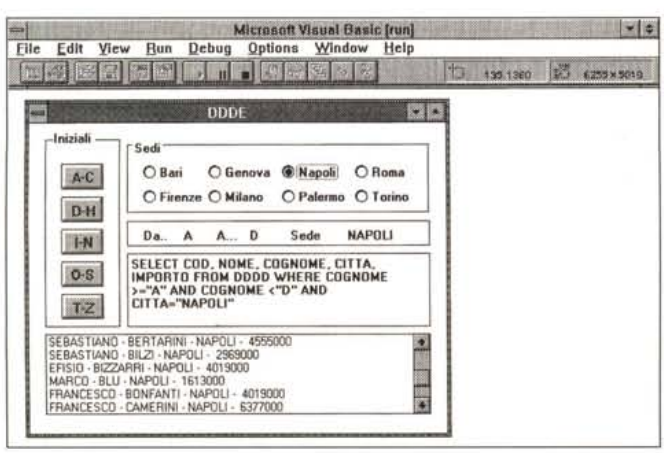


Figura 6 - Visual Basic 3.0 - Lista più complessa.
In questo caso la lista viene caricata al volo nel senso che vengono caricati, dal file esterno, solo i dati che superano il filtro costruito impostando due iniziali (esempio da A a C) per selezionare i Cognomi e una Sede. Viene costruita «a pezzi» un'istruzione SQL e poi eseguita. La differenza rispetto all'esempio precedente consiste nel fatto che qui vengono letti dal file esterno (in formato DBF) solo i dati desiderati.

```
Dim Db As Database, SN As SNAPSHOT
Dim in1$, in2$, ct$

Sub ESEGUI ()
  L2.Caption = in1$: L4.Caption = in2$: V$ = Chr(34): L$ = " - "
  s$ = "SELECT COD, NOME, COGNOME, CITTA, IMPORTO FROM DDDD"
  s$ = s$ + " WHERE COGNOME >=" + V$ + L2.Caption + V$
  s$ = s$ + " AND COGNOME <" + V$ + L4.Caption + V$
  s$ = s$ + " AND CITTA=" + V$ + L6.Caption + V$
  text1.Text = s$
  Set SN = Db.CreateSnapshot(s$)
  LIST1.Clear
  Do While Not SN.EOF
    LIST1.AddItem SN("NOME")+L$+SN("COGNOME")+L$+SN("CITTA")+L$+Str(SN("IMPORTO"))
    SN.MoveNext
  Loop
End Sub

Sub Form_Load ()
  Set Db = OpenDatabase("D:\DDDD", False, False, "dBASE III;")
End Sub

Sub GroupPush3D1_Click (Value As Integer)   ' ed altri
    in1$ = "A": in2$ = "D"
    ESEGUI
End Sub

Sub Option3D1_Click (Value As Integer)      ' ed altri
   L6.Caption = "BARI"
   ESEGUI
End Sub
```

Figura 7 - Visual Basic 3.0 - Lista più complessa.
La caratteristica fondamentale della parte «programma» sta nel fatto che la Sub Esegui viene richiamata da qualsiasi pulsante o option box che serve per impostare il criterio di selezione. Nella Sub Load si definisce quale è il database attivo. Mentre la Sub Esegui crea un'Istantanea sui dati e la riversa nella Lista.

che viene tutto caricato in 8 vettori di 307 elementi. Questo avviene al verificarsi dell'evento Form Load, ovvero all'inizio del programma. Immediatamente prima viene caricata la Combo Box con una serie di Item, che corrispondono a 8 sedi, cui i dati fanno riferimento.

È chiaro che una soluzione più completa, ed elegante, dovrebbe prevedere una serie di implementazioni. Ad esempio la dinamicità della dimensione del file dati, il caricamento automatico della Combo, con dati letti direttamente nel campo Sede, e così via.

Volendo realizzare qualcosa di elegante comunque abbiamo definito una Function, la DS, alla quale si passa la data, memorizzata nel file dati nel formato AAAAMMGG, e che la restituisce nel formato GG/MM/AA.

La Form, il cui titolo è DDDD, contiene una Combo, che mostra le otto sedi caricate al verificarsi dell'evento Sub Form Load, e una Lista, dotata di Scroll Bar, che mostra i soli dati di quella sede.

Il caricamento della Lista è conseguente all'evento Lost Focus della Combo, evento che corrisponde al momento logico in cui si sta uscendo dalla Combo. La prima cosa è ripulire la Lista (List.Clear) poi, scorrendo i 307 dati, si individuano i record il cui campo S$ corrisponde alla sede scelta (IF S$(k)=Combo1.Text..). Vengono composti in un'unica stringa i campi dei record che soddisfano la condizione e la stringa viene sottoposta ad un'istruzione Additem.

Abbiamo fino ad ora già visto le numerose istruzioni con le quali gli elementi delle liste vengono caricati, contati, cancellati, usati, individuati tramite un indice numerico. Insomma tutto quello che serve per una loro corretta e produttiva gestione.

Nel caso dell'esercizio ora visto viene caricato in vettori l'elenco completo dei dati, mentre viene alimentata dinamicamente la lista con i soli dati che interessano al momento, scartando gli altri.

Una soluzione più generale, e sicuramente più elegante, è quella che consiste nel caricare, in altre parole nel leggere dall'archivio, solo i dati che soddisfano certe condizioni impostate a monte, ignorando quelli che non interessano. L'esercizio relativo è quello mostrato in figura 6 e il cui listato più significativo è in figura 7.

L'archivio con i dati è sempre lo stesso, ma immaginiamo che sia molto più lungo. Gli obiettivi che ci prefiggiamo sono sia quello di eseguire una ricerca più esatta, indicando non solo la SEDE, e stavolta le otto sedi sono individuate da una serie di otto OPTION BOX, ma anche l'iniziale del cognome, sia quella di eseguire un'interrogazione SQL che selezioni e legga soli i dati desiderati.

Nella Form vediamo a sinistra 5 pulsanti, in gruppo e quindi mutuamente escusivi, nei quali si scelgono le iniziali, in alto le 8 option box, anche queste mutuamente esclusive, per loro natura. Immediatamente sotto una stringa,

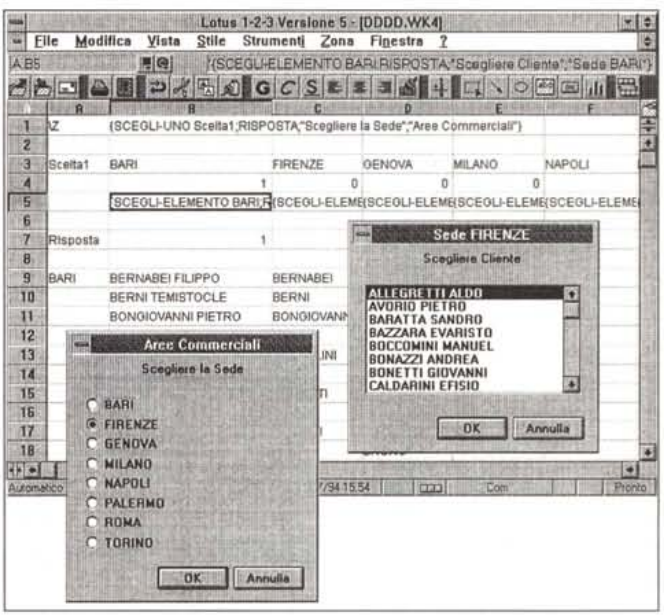


Figura 8 - Lotus 123 per Windows versione 5.0 - Istruzioni compatte.
Una strada alternativa a quella della costruzione di una Dialog Box con più campi è quella che comporta l'utilizzazione di due finestre legate, tra di loro, in serie. Nella prima viene richiesta la sede desiderata. Poi, a seconda della scelta effettuata, viene presentata la lista dei clienti connessi a quella sede. La costruzione della Macro è semplicissima data la compattezza delle istruzioni.

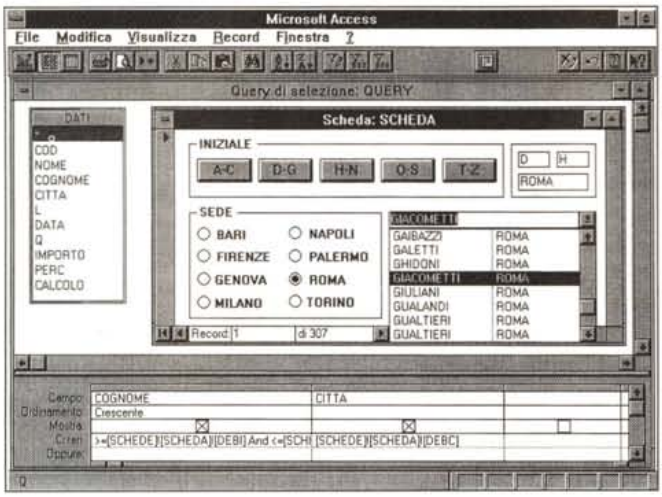


Figura 9 - Access 2.0 - Una Scheda che lancia una Query.
Il meccanismo attraverso il quale viene alimentata la Lista fa uso dei cinque pulsanti con i quali si scelgono le Iniziali che limitano la scelta dei cognomi, degli otto pulsanti di opzione (sia i cinque che gli otto sono tra di loro mutuamente esclusivi). Le due scelte operate vengono passate a due caselle di testo che a loro volta alimentano una Query. Questa viene rieseguita per alimentare, in una specie di circolo vizioso, la Lista presente sulla scheda.

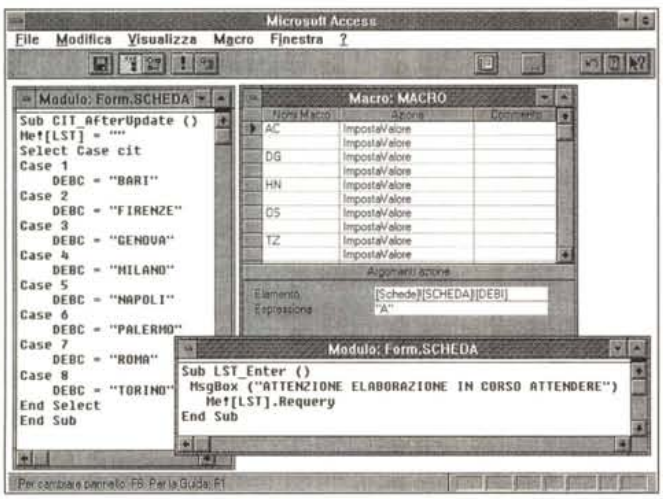


Figura 10 - MS Access 2.0 - Macro e Procedure. Il «momento logico» che richiede l'utilizzo di Macro e/o Procedure è quello che consiste nel passaggio di dati da un elemento all'altro della scheda. Per fare ciò si utilizza l'istruzione Macro ImpostaValore e la si lega all'evento click sul pulsante. Un po' più complesso è il problema del passaggio del valore dell'option box che non è una stringa, ma un numero. Se ne può occupare una microprocedura legata all'evento modifica della option box stessa. Il rilancio della Query è legato al verificarsi dell'evento attivazione della Lista.

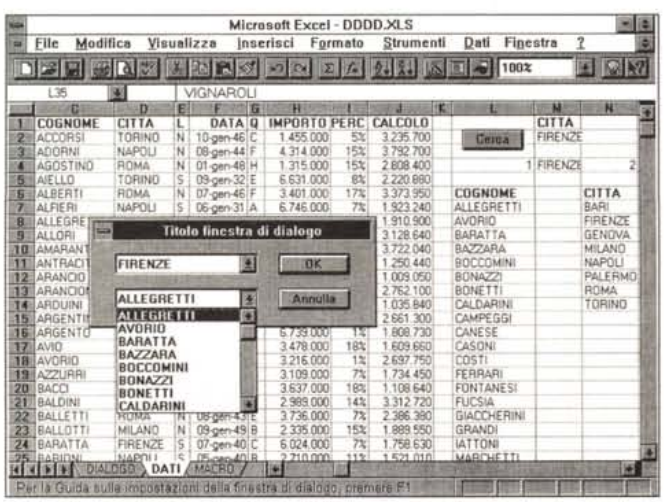


Figura 11 - Excel 5.0 - Alle prese con il Visual Basic for Application - Dati.
In questo ulteriore esperimento, del quale vi mostriamo tre figure, sfruttiamo alcune delle novità presenti in Excel 5.0 e facenti riferimento al VBA, Visual Basic for Application. Qui vediamo il foglio Dati dal quale, usando un pulsante, si richiama una mini Dialog Box che chiede Sede, e nominativo. Scelta la Sede viene eseguita una Macro che esegue un'estrazione di dati, limitandoli a quelli della sede impostata. La zona di estrazione viene utilizzata per costruire la seconda Lista.

composta come output, che ripete le condizioni impostate su iniziali e sede. In basso la istruzione SQL, composta dal programma sulla base dei tre criteri impostati.

Ancora più in basso la lista risultato dell'esecuzione dell'istruzione SQL.

Passando ad esaminare il listato lo troviamo oltremodo sintetico.

Il Sub Form Load si occupa della sola apertura del file dBASE con i Dati.

Poi esistono una serie di pulsanti GroupPush3Dx ognuno dei quali individua un range di iniziali, ad esempio da A compreso fino a D escluso, e quindi da A a C.

Poi ancora esistono una serie di pulsanti Option3Dx ognuno dei quali individua la Sede.

I valori delle due iniziali e della sede vanno a finire in tre Label, ed esattamente la L2, L4 e L6, che entrano nella grossa stringa che ripete le scelte eseguite.

Sia quando si pigia un pulsante che quando di sceglie un'opzione viene eseguita la Sub Routine ESEGUI che fa una serie di cose in sequenza:

– sistema la stringa di esplicitazione;

– confeziona la stringa con il comando SQL;

– la visualizza nella casellona di testo in mezzo alla Form;

– esegue, come SnapShot, vista logica non aggiornabile, l'istruzione SQL che agisce sul database DDDD (in pratica è il nome della subdirectory in cui risiedono i file DBF coinvolti nell'istruzione SQL);

– pulisce la lista risultati;

– riempie la lista con i dati ricavati dalla nuova selezione.

È evidente che l'aspetto più interessante di questo esercizio non è l'uso delle Liste quanto la facilità con la quale si può confezionare l' istruzione SQL, nel nostro caso molto semplice, ed eseguirla. Altrettanto facile è riversare il risultato dell'SQL nella lista. Questa facilità ed immediatezza di uso fanno del Visual Basic 3.0 lo strumento più produttivo per applicazione che facciano da Front End verso banche dati da consultare frequentemente (attenzione! solo consultare non aggiornare).

Basta con la programmazione... per ora

Visual Basic è un linguaggio di programmazione, facile e divertente da usare, ma pur sempre un linguaggio. Ora vediamo qua e là, in altri prodotti, evoluti e per Windows, come siano affrontabili le liste.

Cominciamo con l'123 versione 5.0 per Windows che dispone di una serie

di istruzioni Macro che vanno inserite in una Macro e che producono controlli Windows. Nella figura 8, anche questa è un collage, ne vediamo due:

– la SCEGLI-UNO, che presenta una Box con una serie di pulsanti di opzione. La sintassi dell'istruzione prevede l'indicazione della zona con le opzioni tra le quali scegliere, composta da tre righe. La prima con le etichette delle opzioni, la seconda, con 0 e 1 (o ND) per indicare la scelta di default e le scelte disabilitate, la terza con le istruzioni da eseguire se viene scelta quella opzione. La sintassi prevede anche l'indicazione della cella in cui viene riversata la risposta, in pratica il numero progressivo della sede scelta. Prevede infine l'indicazione del testo e del titolo che appaiono sulla Box.

– la SCEGLI-ELEMENTO, che presenta una Box con una Lista scorrevole. La sintassi prevede l'indicazione dell'elenco da mostrare nella lista (può al solito essere digitato direttamente nell'istruzione, oppure può far riferimento ad una zona del foglio, indicata con le sue coordinate, o con il suo nome se è stata battezzata). Serve poi la zona per il risultato. Anche in questo caso si possono infine indicare testo e titolo che appaiono nella Box.

Le otto sottoliste, una per Sede, sono ottenute componendo le stringhe con Cognome e Nome, e ponendo in alto a sinistra il nome della Sede. Questo sistema permette, al solito, il facile battesimo della zona con la lista.

In pratica eseguendo la macro appare la Box nella quale scegliere la Sede e poi subito dopo la Box con l'elenco delle persone di quella sede.

Altrettanto interessante è l'esperimento condotto con MS Access 2.0. Abbiamo in pratica realizzato lo stesso esercizio di prima, fatto con il Visual Basic, questa volta, grazie alle sinergie tra Scheda e Query in Access 2.0, senza una riga di programma (fig. 9).

In Access la Lista o la Combo si disegnano sulla scheda. Poi ne vanno definite le caratteristiche, tra le quali la principale è quella che specifica come la Lista venga alimentata. Le possibilità sono tre. Alimentata direttamente a mano ed in questo caso si scrivono nella Box delle caratteristiche le varie stringhe corrispondenti alla varie opzioni. Alimentata da una Tabella o da una Query ed in tal caso va indicata la tabella o la query. L'ultimo caso, più particolare, è quello in cui viene mostrata una lista di campi. Ci interessa la seconda modalità.

La figura mostra «a galla» la scheda in cui ci sono i pulsanti per scegliere le iniziali, i pulsanti di opzione per scegliere la sede, tre caselle di testo di appoggio, e la lista, a due colonne, che viene alimentata da una Query.

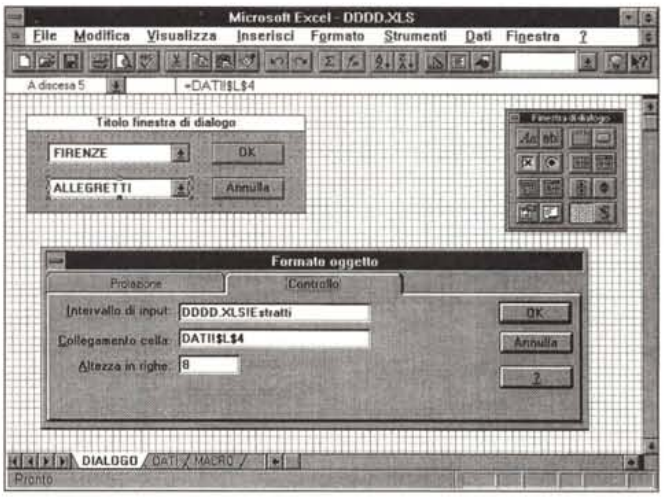


Figura 12 - Excel 5.0 - Alle prese con il Visual Basic for Application - Dialog Box. La finestra, con le due liste, viene disegnata nell'ambiente Editor. Della lista possono essere indicate alcune caratteristiche: da quale zona di celle del foglio sono alimentate, e in quale cella, una sola, riversare il dato scelto. Se la zona può variare occorre far riferimento ad un Nome di Zona e poi occorre via via aggiornare tale nome, quando il numero di dati cambia.

Figura 13 - Excel 5.0 - Alle prese con il Visual Basic for Application - Macro. Le macro sono due. La prima è quella che serve per attivare la finestra di dialogo ed è lanciata dal pulsante posto sul foglio Dati. La seconda è quella che viene eseguita in uscita dalla prima lista. Dapprima esegue l'estrazione, poi evidenzia i dati estratti e li rinomina in modo che l'aggancio con la lista continui a funzionare correttamente.

In secondo piano vediamo la struttura della Query in cui nella riga dei criteri sono stati piazzati dei riferimenti agli elementi della scheda.

In pratica il criterio sul cognome preleva i valori delle caselle di testo DEBI e DEBF della Scheda chiamata SCHEDA e il valore della casella di testo DEBC, che contiene la città di sede. Queste tre caselle sono alimentate al variare dei pulsanti con le opzioni.

Contrariamente a quanto annunciato occorre programmare (fig.10). Con delle Macro eseguiamo una serie di istruzioni ImpostaValore con le quali trasferiamo i valori scelti sui pulsanti nelle caselle di testo di destinazione. Altra procedura necessaria è quella che serve per scodificare i valori della Sede, che in questo caso corrisponde a valori numerici, in valori stringa (es. BARI corrisponde a 1, nell'archivio c'è 1, nella Form c'è BARI).

L'ultima procedura necessaria è quella che serve per rieseguire la Query al verificarsi dell'evento attivazione della Lista. In pratica, eseguite le scelte sui pulsanti, nel momento in cui si entra, ad esempio con il tasto TAB, nella lista, viene eseguita di nuovo la Query.

Excel 5.0 e Visual Basic for Application

Abbiamo visto un po' di programmazione in Visual Basic e qualche Lista costruita con altri sistemi. Ora vediamo di affrontare il nostro problema, lista alimentata sulla base di un criterio, con Excel 5.0.

In figura 11 vediamo in primo piano una piccola Box in cui si seleziona la Sede e la lista conseguente, in cui appaiono i cognomi delle persone di quella se-

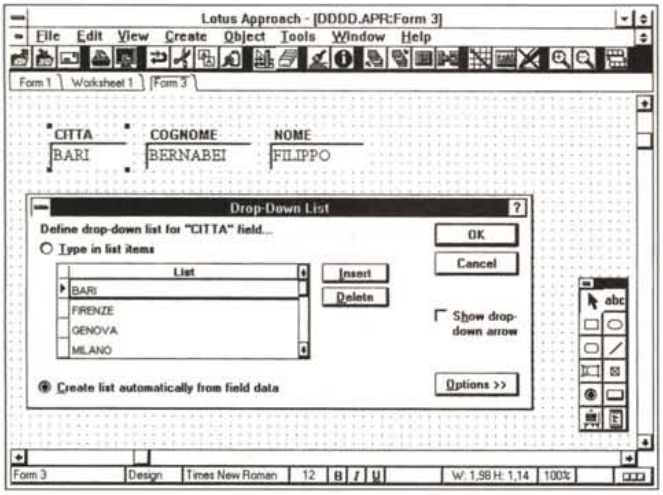


Figure 14,15 - Lotus Approach 3.0 - Due Liste in serie e ... gratis.
Una delle caratteristiche più interessanti di Approach 3.0, un DBMS per utenti finali, consiste nella possibilità di definire delle Liste in serie. Il contenuto della lista può essere letto direttamente dall'archivio, poi può dipendere da un filtro posto su un campo precedente. Insomma è possibile, direttamente in fase di definizione delle caratteristiche del campo, impostare un meccanismo per realizzare il quale con altri prodotti abbiamo dovuto programmare qualche cosa.

de. In secondo piano il foglio vero e proprio in cui vediamo la tabella con i dati, sulla sinistra, e messa in ordine alfabetico. Sulla destra vediamo degli elementi che servono per realizzare il sottoelenco. C'è l'elenco delle Sedi.

Poi c'è il sottoelenco dei nominativi di quella Sede, che viene ottenuto dinamicamente con delle istruzioni di Database sfruttando una zona di Criteri che serve per selezionare il sottoinsieme desiderato.

Sempre nel foglio con i Dati, gli altri due sono quello con la Box e quello con le procedure, troviamo il pulsante «Cerca» che manda in esecuzione la macro principale.

La figura seguente, la 12, mostra l'ambiente (si tratta di un particolare tipo di foglio) in cui si definisce la Box che chiede la Sede e mostra i Cognomi conseguenti. L'ambiente nel quale si costruisce la Box è caratterizzato dalla vistosa griglia e dal pannello che mostra gli oggetti inseribili nella Box. Abbiamo utilizzato solo le Combo.

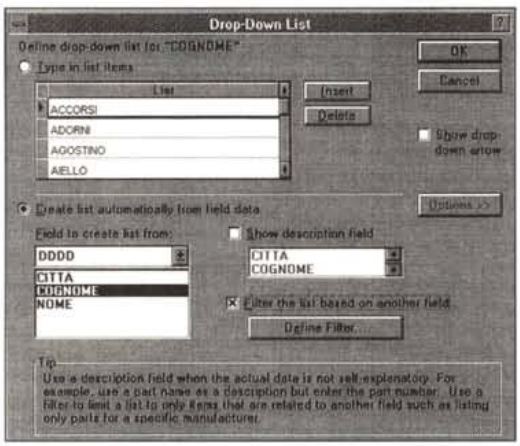
In basso vediamo il pannellino tramite il quale si definiscono i «rapporti» tra gli elementi della Box e il foglio con i dati. In particolare quale sia l'origine della Lista e dove va a finire il dato selezionato nella lista.

È chiaro che nel nostro caso l'indicazione dell'elenco delle Sedi è fissa. L'indicazione dell'elenco dei cognomi invece è variabile, in quanto l'elenco varia, in contenuto ed in lunghezza, a seconda della Sede scelta.

Per risolvere questo problema abbiamo legato l'elenco dei cognomi ad un nome di Zona e abbiamo costruito una procedura, richiamata quando si attiva la lista nella Box, che riesegue l'estrazione, sulla base del nuovo criterio, e che rinomina la zona in cui sono stati riversati i dati (il nome è Estratti).

Il problema più complicato è, al solito con Excel, un problema di riferimenti che non sono assoluti ma variabili. La zona da rinominare si estende dalla prima cella all'ultima cella estratta, raggiunta con il comando Fine in Basso. Il tutto è un po' troppo complesso in relazione alla semplicità del risultato da raggiungere.

In figura 13 vediamo il foglio con la relativa procedura.



Ulteriori varianti

Allora: le liste sono degli oggetti importantissimi nel rapporto tra utente ed applicazione. Ogni prodotto evoluto permette di crearne e di maneggiarne.

Ogni tanto purtroppo occorre fare un ulteriore sforzo per adattare la lista alle proprie necessità, ad esempio quando occorra fare i conti con una lista troppo lunga.

Nell'Approach 3.0, ultima versione dell'interessante DBMS della Lotus, è ovviamente possibile definire le Liste. Sono state introdotte una serie di novità, che intervengono al momento della definizione delle Liste, e che ne facilitano l'uso. La prima è l'alimentazione diretta della lista, nel caso si stia usando un file già pieno di dati. Nel nostro caso avremmo dovuto dire alla Box di Approach di caricarsi la lista leggendosi i dati delle Sedi nell'archivio con i dati (fig. 14).

La seconda, ben più interessante, è quella che consente di legare il contenuto di una lista ad una condizione verificatasi in un campo precedente. L'esempio è esattamente quello dell'elenco dei Cognomi relativo alle Sedi scelte nella Lista delle Sedi (fig. 15).

In figura 16 il recentissimo Paradox

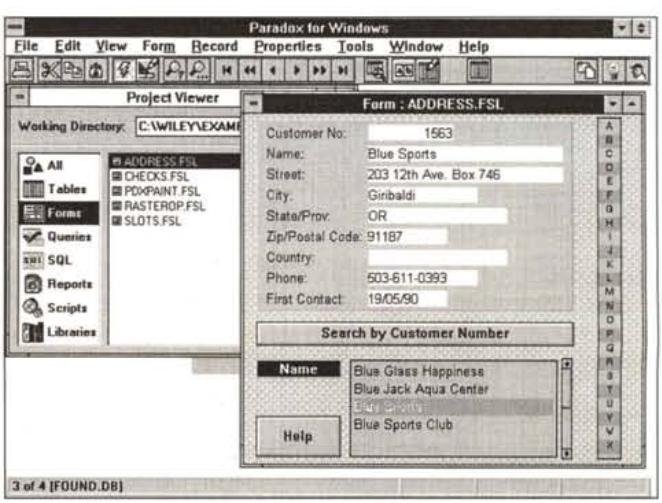


Figura 16 - Borland Paradox 5.0 per Windows - Ricerca «fine» del dato.
Che le liste siano uno strumento importantissimo nel disegno di qualsiasi interfaccia tra dati ed utente è confermato dal fatto che in tutti i nuovi prodotti, anche nel Paradox 5.0 per Windows, arrivatoci pochi minuti prima di «andare in macchina», ci sono tool per crearle facilmente. Guardate in questa Form come sia possibile dapprima scegliere una iniziale e poi, nella lista dei dati riferiti a quella iniziale, scegliere l'individuo voluto.

5.0, che abbiamo appena ricevuto in beta ed in inglese e che presentiamo in anteprima in altre pagine, che permette la costruzione di Form, contenenti liste condizionate.

Tornando al Visual Basic, per terminare, vediamo un altro Control, è uno di quelli aggiuntivi, utilissimo per definire una Lista «intelligente».

In figura 17 vediamo il risultato. La lista è una Struttura (il termine inglese è Outline) del tutto analoga a quella mostrata dal File Manager di Windows quando si vedono Directory e sottostanti File.

Noi abbiamo Sedi, primo livello, e Cognomi, secondo livello. Un doppio click sulla Sede (che mostra un segno più) e appaiono i Cognomi, un doppio click, ora che c'è un segno meno, e i Cognomi scompaiono.

Inizialmente esistono solo le Sedi, ognuna con un segnetto più. E poi via via si può esplodere la struttura per cercare il dettaglio desiderato.

Nella nostra applicazione supponiamo di avere la lista delle sedi e dei cognomi già ordinata e di leggerla da un file DBF.

È il programma di lettura del file che individua le Sede, gli attribuisce il primo livello (caratteristica Out.Indent=1) e individua i cognomi e gli attribuisce il secondo livello (fig.18).

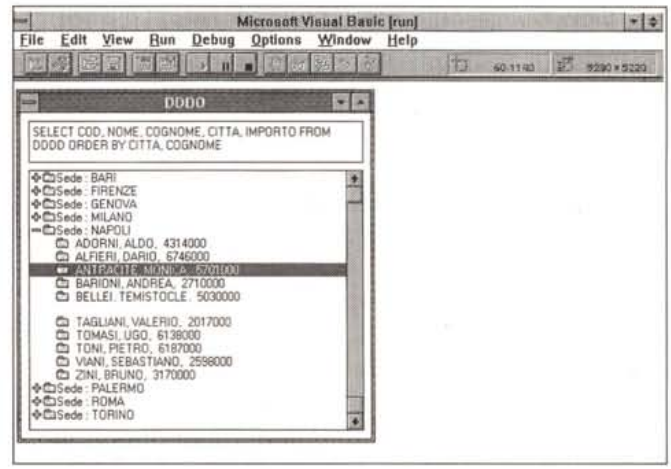


Figura 17 - Visual Basic 3.0 - Lista in modalità Outline.
In questo caso, con il tool aggiuntivo Outline, la lista viene risolta in maniera brillantissima. In pratica, direttamente in fase di lettura vengono create delle categorie (es. Sede: Bari) che costituiscono il primo livello della struttura. I dati normali invece costituiscono il secondo livello. I comandi per gestire la struttura, ben nota a chi usa il File Manager di Windows, sono tanti. Noi gestiamo solo l'evento Click, con il quale si provoca l'espansione o il collasso dell'Item puntato.

Capito il meccanismo di assegnazione dei livello è facilissimo creare delle Strutture, anche a più livelli. Ad esempio Sedi, Iniziali e Cognomi, se l'elenco fosse molto lungo.

L'unico elenco gestito è il doppio click che verifica lo stato della riga. Se esplosa viene implosa e viceversa.

Note:

– se analizzate il programma di lettura potete notare il doppio ciclo, quello esterno su tutti i record del file, e quello interno sulla sede rispetto alla quale il file esterno è ordinato;

– il funzionamento di un elenco strutturato è assolutamente intuitivo. Forse è il più semplice sistema di manipolazione di un elenco finora inventato;

– il Control Outline del Visual Basic 3.0 dispone di numerose caratteristiche e metodi. È possibile, nel caso di struttura a più livelli, espandere tutto, collassare tutto. Ovviamente nel caso si vogliano sfruttare tutte queste possibilità, perché i livelli sono tanti e perché l'elenco è lungo, conviene costruire una pulsantiera di supporto.

Altra variante, più esteriore, consiste nel poter usare più tipi di simboli, proprio come quelli a cui siamo più abituati.

```
Dim Db As Database, SN As SNAPSHOT

Sub Form_Load ()
  Set Db = OpenDatabase("d:\DDDD", False, False, "dBASE III;")
  V$ = Chr(34): w$ = ", "
  s$ = "SELECT COD, NOME, COGNOME, CITTA, IMPORTO FROM DDDD"
  s$ = s$ + " ORDER BY CITTA, COGNOME"
  text1.Text = s$
  Set SN = Db.CreateSnapshot(s$)
  Do
  CT$ = SN("CITTA")
  Out1.AddItem "Sede : " + CT$
  Out1.Indent(Out1.ListCount - 1) = 1
  Do While CT$ = SN("CITTA")
    d$ = "  " + SN("COGNOME") + w$ + SN("NOME") + w$ + Str(SN("IMPORTO"))
    Out1.AddItem d$
    Out1.Indent(Out1.ListCount - 1) = 2
    SN.MoveNext
    If SN.EOF Then Exit Do
  Loop
  Loop While Not SN.EOF
End Sub

Sub Out1_DblClick ()
  If Out1.Expand(Out1.ListIndex) Then
    Out1.Expand(Out1.ListIndex) = False
  Else
    Out1.Expand(Out1.ListIndex) = True
  End If
End Sub
```

Figura 18 - Visual Basic 3.0 - Lista in modalità Outline.
La lista presentata nella box Outline viene costruita automaticamente in fase di lettura del file esterno in formato DBF. La lettura avviene attraverso una Query SQL che provvede anche a mettere nell'ordine voluto i dati. Da apprezzare la sinteticità delle istruzioni, sia di quelle di lettura del file con i dati, sia di quelle per il caricamento e la gestione della Lista Outline.

Conclusioni

Le conclusioni che possiamo trarre è che le liste sono, giustamente, molto diffuse, facili e comode da usare. Spesso costituiscono il modo più intuitivo ed immediato per vedere e per scegliere certi tipi di dati.

Purtroppo pur essendo sempre uguali dal punto di vista di chi le usa, ogni prodotto che permetta di personalizzarle, lo fa in un modo molto «personale». Addirittura alcuni tipi di lista, ad esempio quella che abbiamo chiamato «lista condizionata», in alcuni prodotti possono essere realizzati solo ricorrendo alla programmazione, in altri, i più recenti, possono essere costruite ben più semplicemente definendone, tra le caratteristiche, la condizione. È probabile che la diffusione, in atto, dei linguaggi Visuali, in cui si disegna l'oggetto e se ne impostano le varie caratteristiche, provocherà un allineamento sia in termini di possibilità che in termini di modalità operative per definirle. MC