ATTI del Convegno Internazionale ANTEM, 30/31 Gennaio - 1 Febbraio 1985, a cura di Diòikema, CLUEB. Bologna (ltaJy)

STRUMENTI PER L'INTRODUZIONE DELL'INFORMATICA NELLA FORMAZIONE SCOLASTICA

D. Persico, L. Sarti, M. Tavella

C.R.N./Ist. Tecnologie Didattiche - Genova (Italia)

R. Degl'Innocenti

I.T.I.S. Gastaldi - Genova (Italia)

How to cite:
Persico, D., Sarti, L., Tavella, M., Degl'Innocenti, R. (1986). Strumenti per l'introduzione dell'informatica nella formazione scolastica. In: Dioikema (a cura di), *Atti Convegno Internazionale ANTEM*, Bologna, 30/1-1/2/1985 (pp. 177-185). Bologna (Italia): CLUEB.

1.

La seguente comunicazione verte sulla sperimentazione di un Corso di Introduzione all'Informatica nella formazione scolastica di base.

Il Corso è stato progettato e realizzato da due Istituti del C.N.R. di Genova, l'I.T.D. (Istituto per le Tecnologie Didattiche) e l'I.M.A. (Istituto per la Matematica Applicata).

Il Corso è già stato presentato in precedenti convegni, ma in questa sede verrà affrontato con nuovi elementi di conoscenza.

La novità consiste nei risultati che la prima fase di sperimentazione ha fornito, per la verifica delle ipotesi scientifiche e per la valutazione della qualità dei materiali didattici prodotti.

Il Corso è inserito nel progetto IRIS del CEDE di Frascati e fa parte delle unità di base rivolte al biennio della scuola media superiore.

Il Progetto IRIS prevede che la gestione del Corso sia affidata agli insegnanti delle classi in cui viene svolto, senza preclusioni disciplinari, favorendo anzi la partecipazione di più insegnanti e quindi l'incontro delle diverse materie con l'informatica ed il loro stesso incontro attraverso l'informatica.

Questo aspetto corrisponde ad una delle ipotesi principali e più qualificanti del progetto IRIS, quella secondo cui l'informatica dovrebbe diventare un nuovo elemento della formazione di base non già come disciplina a sè, bensì integrandosi nelle discipline ordinarie, offrendo loro degli strumenti per rinnovare il loro campo concettuale e metodologico, e ampliando i canali della didattica attraverso le opportunità offerte dal computer in tema di applicazione e di comunicazione.

In particolare, il valore formativo dell'informatica andrebbe ricercato *nei* contributi che essa sarebbe in grado di offrire, di determinati modelli di pensiero e linguaggi di espressione, assieme alle tecnologie per la loro realizzazione in processi automatici.

E' evidente che, se verificata, tale ipotesi svaluterebbe decisamente l'angusta concezione dell'informatica come mera innovazione tecnica, ancorchè raffinata e moderna, in un contesto che si dà, in maniera rassicurante, come acquisito.

D'altronde, il Corso ha come oggetto un tema eminentemente informatico quale è la Programmazione e non già un argomento disciplinare qualsivoglia entro cui passino temi informatici.

Questo da una parte è il frutto di una scelta che attribuisce ai principi ed ai metodi della Programmazione strutturata delle valenze formative in grado di travasarsi nella didattica della materia in cui vengono inseriti in forme che varieranno a seconda del contesto, con un ruolo che assomiglia a quello svolto in altri tempi e nell'ambito di obiettivi culturali diversi dall'insegnamento del latino.

Dall'altra parte c'è il fatto che se oggi si è in grado di avanzare almeno una proposta in questo senso sulla scorta dell'esperienza accumulata sul campo dell'insegnamento specialistico, non è così se si vuole già puntare a rendere

trasparente l'informatica nelle varie discipline, campo invece in cui le esperienze sono ancora troppo recenti e spesso parziali.

Probabilmente, la misura verrà raggiunta allorché si giungerà a coniugare sul terreno curriculare la tendenza ad accogliere il metodo generale della 'formalizzazione' che da alcuni anni si va manifestando nella scuola, anche in ambiti tradizionalmente estranei quali quelli umanistici, con la scienza informatica che tale metodologia generale incorpora nei suoi strumenti teorici ed operativi.

La sperimentazione sinora condotta non è giunta ad affrontare organicamente gli aspetti relativi all'ipotesi dell'IRIS quale l'abbiamo esposta precedentemente, anche se il Corso la condivide in linea di principio.

Le considerazioni che seguono sono tratte dalla valutazione dei risultati relativi agli obiettivi diretti del Corso e non agli effetti di ricaduta nel contesto curricolare, ma hanno il pregio di provenire da una particolare esperienza di sperimentazione del Corso, condotta nell'ambito della materia di Italiano.

Si tratta perciò di considerazioni che vanno ad investire comunque il problema dei rapporto tra informatica e formazione scolastica, segnatamente quella linguistica.

Prima di passare ad esaminare la sperimentazione, è necessario illustrare alcune caratteristiche del Corso, in particolare quelle relative al software su cui si fondano le prime 5 fasi delle 9 che compongono il Corso, le fasi prodotte dall' I. T. D.

Di questo software è già stato detto e scritto molto in ambito di ricerca. Le considerazioni svolte in questa sede mirano a presentarlo nella chiave di lettura che è stata data durante la sperimentazione a cui si fa riferimento in questa sede.

2.

Il corso si configura come 'un'unità' didattica, articolata in fasi, rivolta a studenti del biennio superiore, senza alcuna conoscenza prerequisita nel campo dell'informatica.

Ogni fase affronta ed esaurisce un'unità di contenuto, perseguendo tre ordini di obiettivi:

1) obiettivi relativi all'acquisizione di concetti e metodi generali della programmazione strutturata;
2) obiettivi relativi all'acquisizione di elementi di un linguaggio di programmazione, segnatamente del PASCAL;
3) obiettivi relativi al raggiungimento di abilità operative su un sistema di elaborazione di tipo 'personal computer'.

I tre ordini di obiettivi vengono perseguiti contestualmente, attraverso una logica didattica che può riassumersi nel seguente schema:

1) si parte da un problema;
   lo si analizza sulla scorta di metodi di analisi e di logiche di soluzione che si applicano nell'ambito della programmazione strutturata;
   si enuncia l'algoritmo di soluzione;
2) la soluzione viene codificata in un linguaggio di programmazione che ne interpreti la logica e ne permetta la comunicazione al computer;
3) il programma viene scritto al computer che viene comandato ad eseguirlo, cioè a realizzare la soluzione del problema;
4) si controlla l'esecuzione del programma;nel caso di errori, si ritorna al problema o al programma, a seconda della natura dell'errore, e quindi di nuovo alla esecuzione.

3.

Durante le prime fasi, il Corso si avvale di un particolare ed originale software (ispirato a 'Karel' di R.E. Pattis), che va sotto il nome di 'Martino ed il suo mondo'.

Martino costituisce un ambiente didattico, a carattere ludico, studiato per facilitare l'apprendimento dei primi elementi di informatica.

Martino è un robot, rappresentato sullo schermo del computer, dalla forma di un angolino, sufficiente a fornirne la posizione e la direzione verso cui è rivolto in ogni istante.

Martino vive in un mondo configurato come un reticolo di linee perpendicolari. Su queste linee, Martino si muove spostandosi, passo dopo passo, da un incrocio all'altro, nelle 4 direzioni cardinali verso cui può ruotare.

Gli unici elementi presenti del mondo di Martino sono:

1) *gli ostacoli,* che sono segmenti posti di traverso alle linee che uniscono i punti sulle coordinate; gli ostacoli condizionano negativamente i movimenti di Martino in quanto fanno fallire il programma nel caso in cui il robot li vada a scontrare;

2) *gli oggetti,* che sono quadratini posti agli incroci del reticolo, che invece condizionano positivamente *i* movimenti di Martino che può cercarli, prenderli, trasportarli ed infine posarli.

Sulla base di questo schema fondamentale, possono essere create innumerevoli storie in cui ambientare i programmi che 'guidano' Martino a muoversi nel suo mondo.

Gli scenari che fanno da sfondo alle storie possono essere creati attraverso un particolare software che permette di disporre sul reticolo gli ostacoli e gli oggetti e di collocare Martino nella posizione e nella direzione di partenza.

Tali scenari prendono il nome di 'mondi', e gli elementi a disposizione permettono di costruirne innumerevoli e particolarmente simili a mappe di labirinti, che come è noto, sono campi elettivi di applicazioni algoritmiche.

Per programmare Martino, invece, il software mette a disposizione un particolare insieme di istruzioni e di espressioni che formano il cosiddetto 'Linguaggio di Martino'.

Si tratta di parole italiane il cui Significato corrisponde alle azioni che Martino è in grado di fare e alle condizioni che è in grado di verificare, cioè al suo repertorio.

Tali parole vanno ad estendere il repertorio del linguaggio PASCAL, del quale del resto si insegna solo un sottoinsieme significativo.

Obiettivo fondamentale delle prime fasi è acquisire i concetti relativi alle strutture di controllo (sequenza, IF-THEN-ELSE,REPEAT-UNTIL) e alle Procedure e imparare ad utilizzarle nella programmazione di Martino.

In queste fasi non sono considerati i Dati, che vengono invece introdotti nelle fasi successive, allorchè si abbandona l'ambiente di Martino per entrare nei mondo delle applicazioni reali.

Se le ragioni dell'esclusione *dei* Dati sono legate al particolare ambiente grafico di Martino, questo non significa che lo studente non abbia bisogno di comprendere con precisione quelli che sono i dati descrittivi del problema che gli è stato assegnato sulla base di un certo mondo, così come i dati descrittivi del 'mondo' che deve risultare alla fine della esecuzione del programma.

Esistono dunque i dati del problema, ma non c'è bisogno di considerarli come delle entità, tantomeno descriverli formalmente o compiere delle operazioni su di essi, in sede di programmazione.

Potendo così prescindere dai Dati, la programmazione avviene rivolgendo tutta l'attenzione alla logica di processo che deve guidare la soluzione del problema, focalizzandone gli strumenti chiave, cioè le strutture di controllo.

Oltre alle strutture di controllo, l'altro elemento fondamentale tratto dalla programmazione strutturata che viene introdotto è quello relativo alle Procedure.

Le Procedure vengono introdotte subito dopo la sequenza, allo scopo di favorire l'esercizio dell'applicazione del metodo dell'astrazione funzionale.

Le funzioni vengono astratte, nel senso che vengono estrapolate ed enunciate nella forma di sottoproblemi, che vengono affrontati e risolti separatamente.

Per realizzare questo procedimento, lo studente deve saper definire, nel senso autentico della parola, un sottoproblema, alla stregua della definizione degli elementi di un sistema, che esige l'enunciazione non tanto delle parti che lo costituiscono, quanto delle loro relazioni e delle loro funzioni relative rispetto agli obiettivi del sistema stesso.

Nel passaggio dalla sequenza alle successive istruzioni, lo studente viene ad acquisire un altro strumento fondamentale per trasformarne le capacità di analisi e di soluzione dei problemi.

Lo studente, attraverso l'apprendimento delle istruzioni condizionali prima, e di quelle iterative poi, è portato ad assumere una logica che gli permette di affrontare i problemi non come casi assoluti, bensì come particolari realizzazioni di casi generali.

In questo modo, allo studente viene richiesto di trovare la soluzione per un problema che si presenta come caso particolare di un problema generale, e quindi tale soluzione deve essere valida per tale generalità.

Nelle prime fasi, in cui si usano semplici sequenze, la esperienza di programmazione è ass1milabile ad un processo di comunicazione di tipo imperativo, equivalente ad una serie di comandi.

Nelle fasi successive, invece, per l'astrazione richiesta dal considerare contestualmente una pluralità di 'mondi' possibili, possibilità che si realizza con le istruzioni condizionali, la programmazione di Martino tende a diventare da attività di comunicazione con, ad attività di progettazione del robot a svolgere determinati compiti, più orientata cioè a creare modelli di soluzione piuttosto che a suscitare solo effetti della comunicazione.

L'attività di programmazione allontana così il suo accento dalla comunicazione con la macchina per acquistare invece una più decisa impronta quale mezzo di rappresentazione, di espressione e di documentazione di processi logici, rivolto a riflettere il pensiero del programmatore e a permetterne la socializzazione con coloro con i quali deve condividere, via via che il problema si fa più complesso, la ricerca di soluzione.

In questo senso, la scelta del PASCAL, in quanto interprete dei principi della programmazione strutturata, risulta essere una scelta felice, che emerge tra l'altro nella misura in cui durante il Corso si manifesta sotto una duplice 'natura':

1) la "natura" per cui è capace di evocare processi in una macchina;

2) la "natura" per cui estende le possibilità di interazione fra gli studenti.

4.

L'esperienza della programmazione pone lo studente di fronte ad un linguaggio che non è tanto importante imparare a leggere e capire, quanto perchè con esso si devono inventare e scrivere testi con una precisa funzione comunicativa, cioè programmi, con la possibilità altresì di verificarne immediatamente la correttezza formale e il valore comunicativo.

Le difficoltà che si sono riscontrate rispetto all'uso di metodologie di analisi e di sistematizzazione di un problema e alla capacità di costruire algoritmi sufficientemente generali vanno ascritte a nostro avviso in parte minore a insufficienze del corso, ed in parte maggiore all'incapacità degli studenti, di cui è responsabile soprattutto la nostra scuola, ad affrontare la conoscenza fornendosi delle capacità per estrinsecarla e organizzarla in forme testuali che ne permettono la realizzazione, cioè la possibilità di comunicazione.

Ogni apprendimento, per essere reale e significativo, deve in qualche modo ristrutturarsi in forme testuali: la programmazione vincola lo studente a questa condizione.

Non solo, ma allo studente nel nostro caso si chiedeva anche l'applicazione al suo esercizio intellettuale e espressivo di una metodologia quale quella dell'astrazione funzionale e del top-down, che implica un rapporto con il problema che è estraneo alla nostra tradizione didattica.

Ancora una volta le difficoltà di apprendimento a scuola tendono a rivelare cause di ordine linguistico, piuttosto che legate ai particolari contenuti dell'una o dell'altra materia.

All'inizio di questa comunicazione, si è detto che il valore dell'informatica andrebbe ricercato, in via di prima ipotesi, nei modelli di pensiero e nelle forme di rappresentazione e di comunicazione di cui si avvale.

Ciò significa, in breve, parlare di logica e di linguaggio, *cioè* rivolgere l'attenzione alla sfera dei metodi cognitivi e dei mezzi di comunicazione che sono generalmente trascurati nella didattica, anche dell'italiano.

La sperimentazione del Corso, del software di Martino in particolare, ha fornito invece innumerevoli sollecitazioni in questo senso.

La prospettiva più interessante che il Corso ha aperto ha riguardato la

possibilità di procedere per analogia, confrontando il linguaggio di programmazione con il linguaggio. naturale.

In questo senso, è emersa per esempio la possibilità di mettere a confronto le proprietà di una lingua storico-naturale con quelle di un linguaggio artificiale.

Il linguaggio artificiale, in quanto linguaggio formale, e segnatamente il linguaggio di programmazione in quanto codificazione di un procedimento di pensiero, assume un obiettivo ruolo di strumento di analisi del linguaggio naturale.

Lo studente che deve sottoporre la sua lingua e l'ordine del suo discorso alle regole rigorose che il modello logico e la notazione formale impongono, è condotto a riconoscere le categorie cognitive e linguistiche con le quali dispone delle sue capacità di risolvere i problemi.

In realtà, infatti, data la natura profondamente linguistica dell'informatica ed il carattere scientifico di tale linguaggio, imparare a programmare equivale a impadronirsi di una teoria del linguaggio.

La scoperta dell'esistenza di una teoria del linguaggio applicata alla macchina porta con sè il bisogno spontaneo di verificare la corrispondenza di tale teoria alla lingua dell'uomo: di qui, le differenze e le similitudini, motivi di conoscenza e fonti di sviluppo delle capacità.

Sotto un altro aspetto, è emersa la possibilità di applicare il modello generale della comunicazione alla comunicazione uomo-computer.

La programmazione intesa come comunicazione uomo-macchina, porta in primo luogo a considerare la necessità del codice in quanto elemento che deve essere condiviso da emittente e ricevente.

In questo senso, tra l'altro, la storia dei computer può essere letta, ma meglio sarebbe dire che attende di essere scritta, come progressivo avvicinamento dei codici - quello dell'uomo con quello della macchina -, cercando di far si che né emittente né ricevente abbiano a rinunciare alla propria 'identità'.

Martino, nella sua qualità di robot assunto a ruolo di personaggio nelle storie che gli si possono creare attorno, avvicina sensibilmente l'uomo al computer.

Del resto, si programma Martino e non già il computer nella finzione del corso.

Per comunicare con Martino, similmente a quello che accade tra gli uomini, è necessario condividere un codice, cioè un sistema di segni - di significanti e di significati.

Il mondo dei significati rimanda al repertorio del robot, mentre il mondo dei significanti rimanda al linguaggio e alle sue regole, cioè alla sintassi.

La distinzione della dimensione sintattica da quella semantica è più facilmente riconoscibile in un computer che le distingue logicamente e temporalmente nell'analisi, contribuendo a fornire un modello, seppure semplice e per molti versi 'innaturale', dell'attività di comprensione del linguaggio.

5.

In conclusione, la sperimentazione ci sprona a proseguire sulla via tracciata dalle ipotesi che ci hanno guidato sin qui. Fra le tante ragioni che emergono da quanto esposto in precedenza, ce ne è una particolare per questo, e cioè l'avere verificato sul campo la natura intimamente logico-linguistica della materia informatica inserita in un processo didattico di tipo generale, cioè non mirato ad una competenza disciplinare in particolare.

Analogamente a quanto accade nell'insegnamento linguistico, in cui la lingua è allo stesso tempo oggetto e mezzo di apprendimento, l'informatica si presenta come oggetto ma fornisce altresì gli strumenti per apprendere e sono strumenti il cui valore non resta confinato rispetto all'oggetto stesso.

Questa ragione ci sprona, perchè dimostra che l'informatica può concorrere ad offrire all'insegnamento metodi che favoriscano l'unità delle conoscenze, una necessità sempre più sentita nella scuola e che sinora è stata erroneamente affrontata con il metodo dell'interdisciplinarietà, cioè ricercando aree di intersezione dei vari contenuti disciplinari.

Si tratta invece di rovesciare la prospettiva, mettendo in comune linguaggi e strumenti, per la ricerca, l'analisi, la rappresentazione e la soluzione dei problemi.

In questo senso, a nostro avviso, l'introduzione della informatica può fornire i suoi migliori strumenti alla formazione scolastica.

BIBLIOGRAFIA

Andrews, G.R., Schneider, F.B. (1983). Concept and notation for concurrent programming. *Computing Surveys*, 15, 3-43.

Ferraris, M., Midoro, V., Olimpo, G., Persico, D., Sarti, L., Tavella, M., Bottino, R.M., Forcheri, P., & Molfino, M.T. (1985). *Corso di Introduzione all'informatica*. Torino (Italy): SEI.

Bayman, P., Mayer, R.E. (1983). A Diagnosis of Beginning Programmers' Misconceptions of BASIC Programming Statements, *Communications of ACM,* 26, 677-679.

Chen, P.P.(1976). The Entity - Relationship Model: Toward a Unified View of Data. *ACM Transactions on data-base systems*, 1.

Clocksin, W.F., Mellish, C.S.(1981). *Programming in PROLOG.* New York: Springer Verlag.

Dijkstra, E.W.(1982). How do we tell truths that might hurt?, *SIGPLAN Notices*, 17, 13-15.

Ferraris, M., Midoro, V., Olimpo, G. (1984). Petri net as a modelling tool in the development of CAL courseware, *Computo Educ*, 8, 41-49.

Fierli, M., (1982). Aspetti formati vi dell'informatica di base. *Atti della giornata AICA su Contenuti formativi dell'informatica di base*, Padova.

Gilbert, L.A. (1982). Microelectronics in education: two types of innovation, two strategies, *Int. J. Man-Machine Studies*, 17, 3-14.

Jantzen, M.(1980). Structured Representation of Knowledqe by Petri nets as an aid for teaching and research. In *Net Theory and Application*, Berlin: Springer Verlag.

Killam, R., et al(1981). Computer Assisted Instruction in music. Pipeline, 6, 3-4.

Kowalski, R.A.(1982). Logic as a computer language for children, *Proc. European conference on Artificial Intelligence*. Orsay, France

LiSkov, B.H., Zilles, S.(1974). Programming with abstract data types. *SIGPLAN Notices*, 9, 50-59.

Moshell, M.(1982). Computer Power: a first course in using the computer. N.Y.: Mc. Graw Hill.

Olimpo, G. (1982). Aspetti formativi del metodo informatico. Atti della giornata AICA su Contenuti formativi dell'Informatica di base, Padova.

Olimpo, G.(1984). Informatica nella formazione di base: obiettivi, contenuti e metodi. In *Proc. Simposio sobre Informatica y Educacion*, Tucuman.

Olimpo, G., Persico, D., Sarti, L., & Tavella, M. (1985). An experiment in introducing the basic concepts of informatics. In *Proceedings WCCE'85*. North-Holland Amsterdam.

Papert, S.(1980). *Mindstorms: children, computer and powerful ideas*. N.Y.:Basic Books.

Pattis, R.E.(1981). Karel the Robot. New York: John Wiley & Sons.

Peterson, L.(1981). *Petri net theory and the modelling of systems.* Englewood Cliffs: Prentice Hall.

Scotti, F. (1982). The conceptual schema as didactic tool. *Atti di ACM - SIGSE Technical Symposium on Computer Science Education*, Indianapolis.

Slidel, R.J. Anderson, R.E., Hunter, B. (1982). *Computer Literacy*. New York: Academic Press.

The Open University (1982), *Micros in school: an awareness pack far teachers, Case studies*. The Open University Press.

Wexelblat, R.L.(1981). The consequences of one's first programming language. *Software-Practice and Experience*, 11, 733-740.

Woodhouse, D.(1983). Introductory courses in computing: aims and languages. *Comput. Educ*, 7, 79-90.