EMMECI quadro
Scienza Educazione E Didattica

n. 60
marzo 2016

IL PENSIERO COMPUTAZIONALE E LA SCUOLA DELL'OBBLIGO

di Luca Botturi e Lucio Negrini***

Il recente Piano Nazionale Scuola Digitale (PNSD), introduce una nuova visione digitale della scuola italiana, che comprende l'introduzione al pensiero computazionale come componente importante dei percorsi didattici, in ogni ordine di scuola, ma a partire dalla scuola dell'obbligo.
L'autore chiarisce il significato del pensiero computazionale dal punto di vista del suo contenuto e del suo valore culturale.
Dopo un'analisi critica del suo possibile ruolo nei processi educativi e didattici, presenta alcune esperienze positive nei vari ordini scolastici.

** docente-ricercatore presso il Dipartimento formazione e apprendimento della Scuola universitaria professionale della Svizzera italiana con sede a Locarno .*

*** docente-ricercatore presso il Dipartimento formazione e apprendimento della Scuola universitaria della Svizzera Italiana*

Il *Piano Nazionale Scuola Digitale*, recentemente pubblicato dal MIUR (2015), traccia i contorni di una nuova visione della scuola digitale in Italia. Si tratta di una scuola basata su competenze, ricca di strumenti di co-creazione di contenuti, che richiede una parziale nuova definizione dei ruoli di chi insegna e di chi apprende.
Il documento propone di collocare in questo contesto «l'introduzione al pensiero logico e computazionale e la familiarizzazione con gli aspetti operativi delle tecnologie informatiche. In questo paradigma, gli studenti devono essere utenti consapevoli di ambienti e strumenti digitali, ma anche produttori, creatori, progettisti» (p. 29).
Si tratta di un'affermazione impegnativa, perché mette in gioco l'epistemologia stessa dei programmi scolastici, e assegna alla scuola il compito di introdurre un ulteriore tipo di pensiero.
A fianco del pensiero logico, che fa parte della fibra stessa della Matematica, della Filosofia e delle Scienze, ma che si trova in tutte le discipline (basti pensare al ruolo dei testi argomentativi), troviamo il pensiero computazionale, che non viene però definito esplicitamente. Ne viene identificata la finalità: la familiarizzazione con gli aspetti operativi delle tecnologie informatiche nella prospettiva multipla di utenti consapevoli, produttori, creatori e progettisti.
Il documento considera, peraltro in maniera piuttosto fumosa, anche i docenti, che «dalla loro parte e in particolare per quanto riguarda le competenze digitali, dovranno essere messi nelle giuste condizioni per agire come facilitatori di percorsi didattici innovativi basati su contenuti più familiari per i loro studenti»(p. 29).
L'intento di questo articolo è di fornire una pista di lavoro a docenti e dirigenti scolastici, cercando di dare corpo alla frase «pensiero computazionale».
Cercheremo dunque di formulare alcune ipotesi sui suoi contenuti culturali ed educativi, di distinguerlo dall'«informatica» come la troviamo oggi nella scuola (e come, in molti casi, ancora ce la immaginiamo), e di fornire esempi di possibili approcci didattici.

Contenuti, valore e valenza del pensiero computazionale

Contenuti

La locuzione *computational thinking* è stata usata per la prima volta da Seymour Papert (1928-…), autore del famoso libro *Mindstorms* (1980) e del linguaggio di programmazione LOGO. È però nel 2006 Jeannette Wing (1956-…), a definire e presentare il termine *computational thinking* come competenza per tutti, e non solo come una modalità di pensiero propria degli informatici.

Il *computational thinking* viene presentato come un modo di pensare che serve a «risolvere problemi, progettare sistemi, comprendere il comportamento umano basandosi sui concetti fondamentali dell'informatica» e mostra come alcuni concetti tipici dell'informatica siano applicabili anche a situazioni molto concrete: preparare la cartella per la scuola può essere interpretato come *pre-fetching*, cercare un gattino che si è perso come *back-tracking*, decidere se affittare o comprare un paio di sci è un *algoritmo*, eccetera.

Il suo argomento poi si estende: ciò che abbiamo imparato a proposito dei computer è diventato oggi un bagaglio culturale che può servire non solo tutti i giorni, ma che influenza anche altre discipline, scientifiche e non. L'analisi procede, e descrive i contenuti propri del pensiero computazionale: analizzare i problemi per capirne la complessità prima di iniziare a risolverli, scomporli in sotto-problemi, pensare ricorsivamente, identificare modalità di recupero in caso di fallimento, usare euristiche per identificare soluzioni, eccetera – in sostanza affrontare un problema «pensando come un informatico».

Il pensiero computazionale «è un modo di pensare proprio degli umani, non dei computer», una modalità di conoscenza particolare e autonoma che procede per diversi livelli di astrazione e che combina il pensiero matematico, quello ingegneristico e quello scientifico.

Anche senza discutere le sfumature del testo originale, è centrale notare che il pensiero computazionale va ben al di là del saper usare le tecnologie in maniera critica e produttiva.

Il pensiero computazionale infatti è indipendente dalle tecnologie, anche se quest'ultime giocano un ruolo importante e vengono sfruttate intensivamente. Significa sviluppare un pensiero che affronti la realtà con strumenti formali, riportandola alle possibilità di calcolo per intervenire su di essa tecnologicamente. Non si tratta di imparare a usare degli strumenti professionali o personali, ma di acquisire la *forma mentis* che sta alla base del *pensiero tecnologico* (non solo digitale).

Se consideriamo valida la definizione della Wing, la domanda da farsi è se sia utile integrare nel percorso scolastico itinerari che promuovano lo sviluppo del pensiero computazionale. Possiamo riportare la questione a due interrogativi di carattere educativo.

Il pensiero computazionale presenta *un valore culturale* tale da renderlo elemento imprescindibile della crescita dei giovani, tanto quanto l'arte, la musica, o la letteratura?

Un contenuto curricolare ha valore culturale se permette di comprendere e giudicare meglio, in maniera più completa e adeguata, l'esperienza contemporanea di chi impara, e di conseguenza anche di agire in maniera più significativa. In questo senso, crediamo, ha importanza studiare i classici della letteratura o la storia della fisica, non certo per applicazioni dirette o «saper fare» [Calvino, 1991].

Il pensiero computazionale ha una sua *valenza formativa* originale?

Un contenuto curricolare ha una valenza formativa se il processo di apprendimento sviluppa delle competenze o degli atteggiamenti utili. È quanto comunemente si dice del Latino, che «apre la testa» o «insegna un metodo», o delle dimostrazioni dei teoremi (altrimenti semplicemente li prenderemmo come buoni, senza ripercorrerne le dimostrazioni).

Le scelte riguardo ai contenuti del *curriculum* sono giustificabili unicamente entro i termini di uno specifico contesto sociale e culturale, e non è possibile identificare risposte inconfutabili. Possiamo quindi tranquillamente affermare che l'assenza del pensiero computazionale dai programmi scolastici fino ad oggi non rappresenta un errore storico di cui pagheremo le conseguenze. Crediamo però che, proprio in forza dello specifico contesto attuale, la discussione attorno al pensiero computazionale meriti di essere aperta.

Valore culturale

Sicuramente la tecnologia in senso lato, e quella digitale applicata alla gestione delle informazioni in particolare, può essere considerata tra i tratti salienti della nostra esperienza: *internet*, nel bene e nel male, ha cambiato il mondo, e non potremmo immaginarci senza telefonia mobile, macchine a controllo numerico, banche dati, lettori di codici a barre, immagini digitali e applicazioni telematiche.

La penetrazione delle tecnologie digitali non concerne più solo il mondo del lavoro e degli adulti ma è sempre più presente anche nella vita dei giovani e dei bambini anche in tenera età. Non sorprende più vedere un bambino che usa il *tablet* per guardare un video su *YouTube*, che *chatta* con gli amici o registra la sua serie televisiva preferita.

In questo contesto l'alfabetizzazione tecnologica è innanzitutto un'esigenza imprescindibile per i cittadini di oggi e di domani: se cinquant'anni fa non potevamo immaginare partecipazione civica senza saper leggere e scrivere, oggi non possiamo immaginarla senza un uso base delle tecnologie digitali. Ma è anche un dovere culturale: il dominio tecnologico è un'immensa leva di potere.

I grandi temi di oggi – dalla finanza, alla famiglia, alla fine della vita, allo sviluppo economico e sociale – sono intrisi di tecnologia, come strumento e come mentalità, e non basta più «fermarsi fuori» per giudicarli senza capirne il funzionamento.

Prendiamo il caso della libertà di informazione. Potenzialmente, abbiamo oggi a disposizione più informazioni di quante ne possiamo consumare, e siamo anche liberi di pubblicare informazioni. Sappiamo bene però, che questa situazione non ha generato persone più informate, ma più «idioti digitali», ognuno chiuso nel suo microcosmo di informazioni selezionate in base ai propri interessi [Chamorro-Premuzic, 2014].

Come docenti, sappiamo che, dato un qualsiasi tema di ricerca, la maggior parte degli allievi citerà le stesse fonti. Possiamo pensare di intervenire efficacemente in questo ambito senza conoscere come *Google* cerca, seleziona e ordina le informazioni? O senza comprendere il funzionamento di un'organizzazione digitale come *Wikipedia*? O senza capire quali sono i parametri economici che li fanno funzionare?

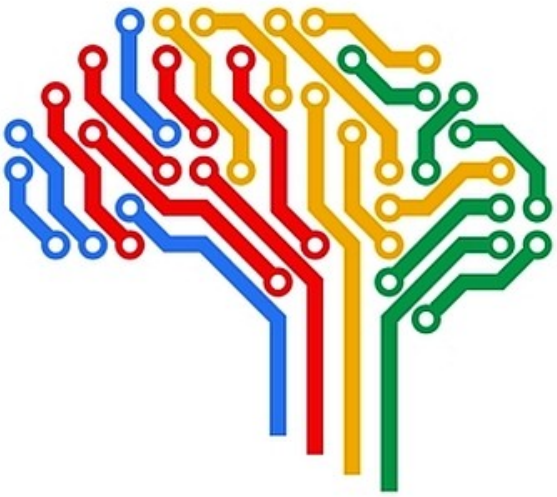
Potremmo farlo, ma in fondo resteremmo all'esterno del problema, e potremmo solo concludere la discussione sperando che «i tecnici» risolvano la questione.

Il pensiero computazionale ha dunque un suo valore culturale: capire le tecnologie permette di non esserne passivamente schiavi. Affermato icasticamente: o si impara a programmare o si verrà programmati [Rushkoff 2012].

Valenza formativa

Questo però non è sufficiente per giustificare l'inserimento del pensiero computazionale nei programmi scolastici. Ci siamo infatti anche chiesti se abbia una valenza formativa.

Chiunque abbia studiato o lavorato seriamente nel settore dell'informatica, sa che esso richiede un lavoro mentale particolare: quello accennato brevemente più in alto riportando le tesi della Wing. Analizzare i problemi, valutare le risorse, pianificare le soluzioni, presentare la soluzione intera prima di realizzarla, identificare eventuali ricorsività o snodi condizionali, eccetera.



Pensare computazionalmente significa assumere una *forma* particolare, numerica e algoritmica, per descrivere e risolvere i problemi.

È una forma diversa da quella delle lettere e delle arti, ma anche diversa dalla matematica e dalle scienze, prossima a quella delle ingegnerie. Riteniamo dunque che il pensiero computazionale abbia una sua valenza formativa originale, che lo rende interessante come ambito educativo e didattico nella scuola.

Vista la sua importanza, il pensiero computazionale viene anche indicato da alcuni autori come quarta abilità di base oltre a leggere, scrivere e calcolare [Lodi, 2014].

Alcuni studiosi ritengono inoltre che il pensiero computazionale sia l'abilità centrale di tutte le discipline STEM (*Sciences, Technology, Engineering, Mathematics*) e debba dunque venire portata nelle scuole [Henderson, Cortina, Hazzan, & Wing, 2007].

Il pensiero computazionale e l'ora di informatica

La breve storia dell'informatica come disciplina scolastica non ha generato un proprio canone di contenuti, ma ha visto succedersi almeno quattro diverse interpretazioni. È allora utile ripercorrerla per mettere a fuoco la novità che l'idea di pensiero computazionale introduce.
Negli anni Ottanta e inizio anni Novanta, prima della diffusione delle interfacce grafiche e di *internet*, l'informatica era legata al pensiero formale, e gli allievi apprendevano i primi rudimenti della programmazione lavorando con *Lotus* 123 o *Basic*. Era l'epoca di DOS e dei *floppy disk*, e i computer erano oggetti professionali delicati e specializzati in particolare per l'ambito scientifico.
Dopo la diffusione dei personal computer e in particolare di *Windows*, l'attenzione si è spostata sulla produttività personale.
L'uso esperto degli strumenti è diventato un obiettivo: scrivere una lettera, impostare un foglio di calcolo, creare una presentazione. In questa fase l'informatica è stata promossa come «quarto pilastro» della scuola, insieme a Italiano, Matematica e all'altro «nuovo arrivato» in questa veste, l'Inglese.
L'informatica però veniva considerata come l'apprendimento dell'uso del computer e dei vari *software* e non come una scienza che studia la modalità di raccolta, di trattamento e di trasmissione di informazioni. L'uso di *internet* e in particolare dei motori di ricerca si è poi inserito in questo contesto, in cui la prospettiva dell'utente avanzato, ma non del tecnologo, era prevalente.
In questo periodo si è anche diffuso l'uso dei media digitali (audio, video, fotografia) come modalità espressiva e di documentazione di esperienze didattiche e creative.
La terza fase è la più recente, e ci accompagna dalla metà degli anni Duemila, e potremmo definirla una fase di incertezza disciplinare.
Da un lato è chiaro che non ha più senso intendere l'informatica come introduzione all'uso di strumenti professionali, perché le tecnologie digitali sono diventate oggetto quotidiano nel privato e spesso gli allievi sono più esperti dei docenti; inoltre il mercato di sta popolando di piccoli programmi monofunzionali – le *app* – il cui tempo di apprendimento è minimo, a scapito di programmi ampi e multifunzionali come il pacchetto office [Beattie, 2015].
Dall'altro lato, non appare evidente quale sia il senso del rendere le tecnologie oggetto di studio, anche se si sente l'esigenza di educare a un uso sicuro, legale, critico e responsabile dei media digitali, come prevenzione dalle dipendenze e dai rischi della rete.
Questa nuova evoluzione della disciplina è nata insieme alla diffusione del *Web 2.0*, che poi è il *web* come lo conosciamo oggi: un luogo in cui tutti sono potenzialmente consumatori e produttori di informazioni e in cui i diversi media si intrecciano senza soluzione di continuità.
A questo si è poi aggiunto il *Web 3.0*, che è la risposta tecnologica alla presa di consapevolezza che la quantità di informazioni disponibili richiede che sia il sistema stesso, e non noi, a creare un'organizzazione sempre mutevole e aggiornata, in base alla sua rappresentazione delle nostre esigenze individuali.
In questo scenario arriva la proposta del pensiero computazionale, che in qualche modo ci riporta all'essenziale: pensare la tecnologia come parte del nostro mondo, e comprendere come ci rappresenti, al di là dell'uso di strumenti e di pratiche d'uso sicure.
Anche per chi si occupa di insegnare a usare strumenti specifici nel settore professionale o di educazione a un uso critico dei media, poter contare su uno «zoccolo duro» di reale comprensione delle macchine digitali significherebbe poter costruire su fondamenta più solide.

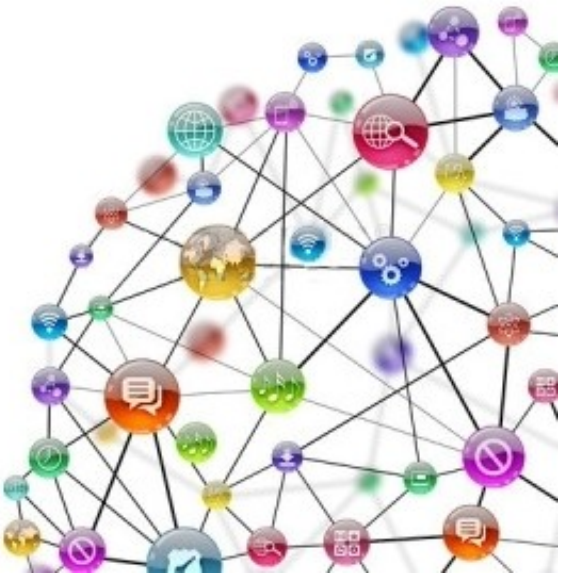
Il pensiero computazionale nella scuola primaria
È bene notare inoltre che l'informatica è sempre stata pensata per gli anni alti della scuola dell'obbligo (scuola media) o nella scuola secondaria superiore.
Tra le azioni previste nel già citato documento del MIUR, invece, la numero 17 prevede di «portare il pensiero logico-computazionale a tutta la scuola primaria», perché è «fondamentale partire dai giovanissimi, per almeno due ragioni: primo, anticipare la comprensione della logica della Rete e delle tecnologie, proprio perché l'avvicinamento alle

tecnologie stesse avviene prima, a partire dal contesto familiare; secondo, preparare da subito i nostri studenti allo sviluppo delle competenze che sono al centro del nostro tempo, e saranno al centro delle loro vite e carriere» (p. 81).
L'idea di base è che, trattandosi di una forma di pensiero, sia importante introdurla il prima possibile, quando non bisogna contrastare abiti mentali già strutturati (forse si potrebbe fare lo stesso ragionamento per la Filosofia).
Ci piacerebbe poter modificare lievemente le finalità: è importante anticipare la comprensione delle tecnologie, perché fanno parte di ogni attività che svolgiamo, e non per mettere una pezza alla lacunosa educazione familiare; inoltre perché gli allievi sviluppino le competenze necessarie per far stare le tecnologie al posto giusto nella loro vita, lasciando al centro a ciò che è bene che ci stia, sia nella vita sia nella carriera.

Strumenti ed esempi

Uno dei primi a sostenere la necessità di promuovere il pensiero computazionale e di insegnare la programmazione fu Alan Perlis (1922-1990), che nel 1961 propose di creare un corso di programmazione in ogni università.
Scopo del suo corso non era «insegnare alle persone come programmare un computer o insegnare un linguaggio specifico», ma di «insegnare alle persone come costruire e analizzare processi» [Greenberger, 1964].
Tra queste prime iniziative e oggi sono stati sviluppati diversi *curricula*, attività, *framework*, strumenti tecnici hardware e software per insegnare il pensiero computazionale [Grover & Pea, 2013]. Il panorama rimane però ancora molto frammentato. Che cosa significa in pratica proporre attività che promuovano lo sviluppo del pensiero computazionale?
Le possibilità non mancano: sono molte le organizzazioni, in particolare private e *non profit*, che negli anni hanno sviluppato metodologie e strumenti per lavorare sulle tecnologie e con le tecnologie in maniera divertente e creativa.
Il punto centrale di lavoro è sempre quello di scoprire la logica delle macchine «dal di dentro», e di imparare a dare forma a un pensiero che le comprenda in forma non magica ma, appunto, logica.

Attività senza computer
Nella scuola elementare sono possibili diverse attività che promuovono il pensiero computazionale senza l'uso diretto di tecnologie.
Per esempio, il semplice esercizio di scrivere ricette o altre sequenze di azioni che devono poi essere riprodotte da altri può essere una prima introduzione all'idea di algoritmo. Si tratta di un genere testuale diverso dal componimento scritto classico: in questo caso vengono premiate precisione, assenza di ambiguità, ordine e completezza dell'informazione.
Altre attività, che appassionano grandemente i bambini, portano a scoprire come funzionano i sistemi informatici tramite simulazioni. Con poco sforzo è possibile organizzare gli allievi in modo che ognuno rappresenti una componente di un calcolatore (*hard-disk*, RAM, *mouse*, schermo, processore, eccetera) o di rete (*server*, *router*, terminale, servizio specifico, eccetera) e simulare le operazioni comuni che facciamo tutti i giorni.
Anche qui l'obiettivo non è tecnico (imparare i nomi delle componenti dei sistemi o il loro funzionamento), ma logico: i computer e le reti hanno comportamenti di un certo tipo perché sono progettate così, e realizzano la logica che noi gli abbiamo dato.
A un livello tecnico un po' più sofisticato, belle attività possono essere anche svolte lavorando alla costruzione di semplici circuiti per esempio, costruendo da zero un «sapientino» con 4 o 5 domande e risposte; o una mini-bambolina che si illumina se le schiacci il naso.
Si lavora con tavolette di legno, nastri metallici, lampadine LED e batterie, come per esempio in alcune attività di alfabetizzazione tecnologica che abbiamo visto proposte in Belgio da Maria Grazia Ciocci. Si tratta in questi casi di prendere familiarità con il fatto che comportamenti che ci paiono intelligenti (come segnalare la risposta giusta per il «sapientino»), sono in realtà frutto di programmazione (in questo caso, hardware).
Per esempio, gli allievi si dispongono sui nodi di una rete disegnata sul pavimento con dello *scotch* di carta. Ognuno di loro riceve poi un ruolo



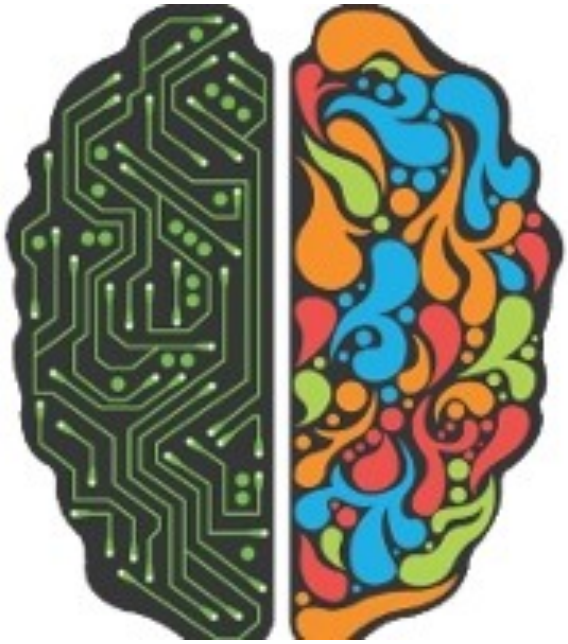
particolare: alcuni sono *server* di rete, altri *router,* altri terminali collegati via cavo e alcuni rappresentano *smartphones* o altri dispositivi *wireless*. Alcuni impersonano i *server* principali di Google, Facebook, YouTube e Snapchat.
Sul pavimento, quando possibile, si rappresenta anche l'oceano atlantico e le rispettive dorsali transoceaniche. Già solo prendere questo assetto significa riflettere sul fatto che *internet* non è uno «spirito buono» che aleggia nell'aria, ma un'opera di ingegneria umana, con contorni ben definiti.
È poi possibile «giocare» diverse simulazioni: la semplice visita a un sito *web*, una ricerca su Google (e come Google indicizza i siti), la concatenazione post→notifica su Facebook, eccetera. Nell'azione gli allievi si trovano a ragionare come computer: come trovo la strada giusta nella rete? Come traduco in un comando semplice l'interazione dell'utente? Ecc.
Una volta capito che si tratta dell'invio di un messaggio attraverso i nodi di una rete, gli allievi stessi traggono molte delle conclusioni che ci appaiono come difficili obiettivi comportamentali per un uso consapevole di *internet.*
Per esempio, risulta chiaro che, quando Google copia una nostra foto sui suoi *server*, quella foto esce dal nostro controllo; o che un contenuto pubblicato su Facebook viene duplicato N volte nella rete, diventando virtualmente non eliminabile. Conoscere davvero gli strumenti che usiamo vale molto di più che dare regole d'uso.

I robot in classe
Un filone interessante e altamente motivante è quello offerto dalla robotica educativa, cioè dell'uso di piccoli o grandi robot pensati appositamente per attività di apprendimento. I robot sono oggetti affascinanti, percepiti come «magici», e che hanno il grande pregio di ancorare gli aspetti logici e di programmazione alla concretezza dell'azione fisica e della meccanica.
I robot possono essere introdotti già a partire dalla scuola dell'infanzia, come dimostra per esempio il progetto di ricerca-azione PReSO. Scopo del progetto, che include una quindicina di docenti di scuola dell'infanzia e scuola primaria del Canton Ticino, è di favorire il pensiero computazionale introducendo i robot in classe.
Per i più piccoli sono interessanti i *BlueBots*, piccoli robottini che possono essere programmati per spostarsi su un piano (un tavolo o il pavimento). Usandoli si rende operativa l'idea di algoritmo e si sperimenta la modellazione, perché lo spazio piano in cui si muovono viene virtualmente suddiviso in una griglia.
Cubetto è un prodotto recente, sviluppato proprio per introdurre i piccoli alla programmazione: i bambini devono programmare il movimento di un'automobilina-cubetto, disponendo oggetti fisici in sequenza su una tavola. Tra gli oggetti che rappresentano il vocabolario di programmazione, oltre che «avanti», «indietro» e «gira», troviamo anche alcuni elementi di controllo di flusso come i cicli. Come per i *BlueBots*, *Cubetto* può essere integrato in diverse sfide da risolvere.
Thymio è un robot sviluppato presso il Politecnico di Losanna: ha due ruote, un altoparlante, pulsanti, LED colorati e diversi sensori. Ha due modalità di operazione che lo rendono interessante per il secondo ciclo della scuola elementare o per i primi anni di scuola media: comportamenti pre-programmati e programmazione.
Nella prima modalità, può assumere diversi comportamenti (per esempio: seguire un oggetto; evitare il contatto; eccetera), ed è un bell'esercizio cercare di decodificarli in maniera formale: quale azione viene eseguita come risposta a quale stimolo raccolto dall'ambiente? Con quali sensori? Il processo è di tipo scientifico (formulazione di ipotesi, creazione di mini-esperimenti, eccetera), ma applicato ad un oggetto artificiale e direttamente riproducibile.
Nella seconda modalità è possibile programmare *Thymio* con un'interfaccia (almeno inizialmente) grafica e quindi sia replicare i comportamenti base, sia programmare nuovi comportamenti, come per esempio rendere il robot un piccolo strumento musicale, o farlo «ballare» al ritmo del battito delle mani.
Uno strumento flessibile e con grandi potenzialità è il kit EV3 della Lego. Si tratta di un set di Lego tecnico che include una centralina digitale collegabile a sensori e motori e programmabile con un computer. È possibile creare robot autonomi di tutti i tipi, tramite un'interfaccia di programmazione a blocchi che può raggiungere anche un'elevata complessità.

Questi kit sono usati come strumento didattico di riferimento in diversi contesti, tra cui alcuni percorsi opzionali nella scuola media in Ticino, e anche nel torneo internazionale di robotica *First Lego League*.
Oltre ai robot fisici esistono anche varie applicazioni e software dove i robot vengono simulati. Possiamo per esempio citare *LightBot* e *Robot School*, due versioni di robot che vivono sullo schermo di un computer; sicuramente più economici, ma meno concreti.
Esistono poi vari prodotti di robotica per la didattica nella scuola superiore, specificatamente pensati per sviluppare competenze di programmazione, come per esempio i *Nao* – ma qui ci spostiamo già verso un altro settore.
Da diversi anni uno degli autori di questo articolo è *coach* di una squadra di ragazzi tra i dieci e i tredici anni che partecipa ai tornei della *First Lego League* usando il kit Lego EV3.
Il lavoro con il robot su una sfida reale (la competizione) innesca una motivazione altissima: da un lato si costruisce e si programma un oggetto tecnologico di alto livello, dall'altro ci si confronterà con le soluzioni di altri ragazzi direttamente sul campo di gioco. Il tempo di apprendimento della programmazione è piuttosto lungo, perché il sistema permette controlli di flusso, iterazione, l'uso di variabili, eccetera.
Siamo convinti però che il guadagno principale di queste attività consista proprio nello sviluppo di un approccio tecnologico rigoroso: lo stupore di fronte alla tecnologia viene spogliato di ogni aura magica, e subito si capisce che per far fare qualcosa al «nostro robot» bisognerà esprimerla in un linguaggio formale, a lui comprensibile, formato di poche parole univoche e con una grammatica rigida.
Piuttosto rapidamente si impara anche a valutare la complessità di realizzazione dei programmi, per esempio quando si trova una buona soluzione ma si capisce che sarebbe difficile da programmare, o quando si riscrive un programma in una forma più elegante.

Coding

Un filone recente, ma molto promettente e che ha acquisito rapidamente vasta popolarità, è quello del *coding*. Si tratta di attività che introducono alla programmazione come competenza specifica, e nel farlo creano le basi di una solida alfabetizzazione tecnologica.
Uno dei primi passi in questa direzione è stato intrapreso dal gruppo *Lifelong Kindergarten* del MIT Media Lab, che ha sviluppato *Scratch*, un ambiente di programmazione grafico pensato appositamente per i ragazzi.
Su questo ambiente di programmazione sono stati sviluppati numerosi programmi didattici, per esempio dalla rete americana *Computer Clubhouse Network* che hanno portato poi a eventi regionali, nazionali e internazionali come la *International Code Week*. Sull'onda di questo successo lo stesso MIT ha rilasciato più recentemente *AppInventor*, un servizio online che permette di programmare piccole *app Android* che sfruttano i servizi disponibili sui dispositivi (microfono, sintetizzatore vocale, GPS, eccetera) ed effettivamente usabili.
Una declinazione di questi strumenti in itinerari didattici è quella disponibile su Code.org, ripreso in lingua italiana dall'iniziativa *Programma Il Futuro*.
Uno strumento simile a *Scratch*, ma legato al mondo dei videogiochi, è *GameMaker*, una piattaforma che permette di sviluppare semplici videogiochi con un'interfaccia grafica di programmazione.
Sempre nell'ambito della creazione di videogiochi è nato il concetto dello *Scalable Game Design* per portare la progettazione di videogiochi nella scuola primaria lavorando con software come *Agentsheets* e *Agentcubes* [Repenning, Webb & Ioannidou, 2010].
Così facendo si impara a programmare in modo ludico e avendo uno scopo ben definito.
Ancora una volta, il tema di questi strumenti non è tanto l'apprendimento di competenze tecnologiche, quanto l'offrire una possibilità accessibile di considerare gli oggetti elettronici quotidiani non solo come utenti, ma come progettisti, e quindi di incuriosire a scoprirne la logica, di ragionare per realizzare ciò che si è pensato.
Recentemente chi scrive ha assistito a un workshop con *AppInventor* per le scuole medie superiori, e ha seguito un progetto di sviluppo di una *app* con questo strumento da parte di un gruppo di ragazzi di scuola media.
Anche in questo caso, le competenze tecniche richiedono tempo per



essere sviluppate, ma il guadagno immediato è la scoperta che le *app* che usiamo sono prodotti, sono pensate come strumenti software che si appoggiano sulle opportunità offerte dai dispositivi hardware, che dialogano con un sistema operativo (in questo caso, *Android*): in poco tempo viene rimossa la patina di opacità che rende gli *smartphone* oggetti incomprensibili, e si inizia a ragionare di valori che passano da una variabile all'altra, di formati, di interfacce e di livelli di architettura software.

Conclusioni

I riferimenti presentati in quest'ultima sezione dimostrano che le opportunità di lavorare sul pensiero computazionale in tutti gli ordini scolastici non mancano: sono molte le università, le organizzazioni *non profit* e anche le imprese che hanno messo a tema le tecnologie e l'educazione tecnologica con un non comune spirito di apertura e di condivisione.
La questione fondamentale, tuttavia, viene prima, e riguarda l'opportunità di inserire lo sviluppo del pensiero computazionale tra gli obiettivi della scuola dell'obbligo.
In questo articolo abbiamo cercato di dimostrare come sia importante innanzitutto definirlo chiaramente, e non appiattirne la definizione sulle competenze funzionali d'uso delle tecnologie, né sulle regole per un uso sicuro, legale, critico e responsabile dei media.
Il dibattito sui contenuti dei programmi scolastici è sempre un salutare segno di vita culturale; per quanto riguarda il pensiero computazionale siamo convinti che, in termini di valore culturale e di valenza formativa, ci siano sicuramente argomenti rilevanti per aprire la discussione e cercare nuove vie di sperimentazione.

Luca Botturi e Lucio Negrini***

*(*docente-ricercatore presso il Dipartimento formazione e apprendimento della Scuola universitaria professionale della Svizzera italiana con sede a Locarno dove coordina le attività legate all'ambito Tecnologie e media ed è responsabile del Servizio Risorse didattiche, eventi e comunicazione.*

***docente-ricercatore presso il Dipartimento formazione e apprendimento della Scuola universitaria della Svizzera Italiana).*

Indicazioni bibliografiche

Beattie, A. (2015). *How Apps Are Changing The World*. In *Investopedia*, 18 maggio 2015.

Calvino, I. (1991). *Perché leggere i classici*. Milano: Mondadori.

Chamorro-Premuzic, T. (2014). *How the web distorts reality and impairs our judgement skills*. In *The Guardian*, 13 maggio 2014.

Greenberger, M. (1964). *Computers and the World of the Future*. Cambridge: MIT Press.

Grover, S., & Pea, R. (2013). *Computational Thinking in K-12: A Review of the State of the Field*. In *Educational Researcher*, *42*(1), 38–43.

Henderson, P. B., Cortina, T. J., Hazzan, O., and Wing, J. M. (2007). *Computational thinking*. In *Proceedings of the 38th ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '07)*, 195–196. New York, NY: ACM Press.

Lodi, M. (2014). *Imparare il pensiero computazionale, imparare a programmare*. In *Atti del convegno Didamatica 2014*, 822-833.

MIUR (2015). *Piano Nazionale Scuola Digitale*. Roma: MIUR.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.

Repenning, A., Webb, D., & Ioannidou, A. (2010). *Scalable game design and the development of a checklist for getting computational thinking into public schools*. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*, 265–269. New York, NY: ACM Press.

Rushkoff, D. (2012). *Programma o sarai programmato.* Milano: Postmedia.

Wing, J. M. (2006). *Computational Thinking*. *Communications of the ACM, 49*(3), 33-35.

Indicazioni sitografiche

AppInventor	*http://appinventor.mit.edu/explore/*
BlueBot	*www.bee-bot.us/bluebot.html*
Code.org	*www.code.org*
Computer Clubhouse Network	*www.computerclubhouse.org/*
Cubetto	*www.primotoys.com/*
First Lego League	*www.firstlegoleague.org/*
Game Maker	*www.yoyogames.com/gamemaker*
International Code week	*https://csedweek.org/*
Lego EV3	*www.lego.com/it-it/mindstorms*
LightBot	*https://lightbot.com/*
Nao	*www.aldebaran.com/en/cool-robots/nao*
PReSO	*http://cerdd.ch/preso*
Programma Il Futuro	*http://programmailfuturo.it*
Robot School	*www.robotschoolapp.com/*
Scalable Game Design	*http://sgd.cs.colorado.edu/wiki/Scalable_Game_Design_wiki*
Scratch	*https://scratch.mit.edu/*
Thymio	*www.thymio.org/*

EMMECI quadro
Scienza Educazione E Didattica

DIRETTORE RESPONSABILE Mario Gargantini
www.emmeciquadro.eu

Aut. Trib. Milano n. 226 del 24.3.2000
ISSN 2240-0389
Emmeciquadro (online)