Pensiero Computazionale: dalle "scuole di samba della computazione" ai CoderDojo

Michael Lodi

Pensiero Computazionale: dalle "scuole di samba della computazione" ai CoderDojo

Michael Lodi[0000−0002−3330−3089]

Dipartimento di Informatica - Scienza e Ingegneria (DISI)
Alma Mater Studiorum - Università di Bologna & INRIA Focus
michael.lodi@unibo.it

Sommario In questo lavoro risaliamo alle origini dell'espressione "pensiero computazionale", calandola nel contesto originale (la teoria costruttivista di Seymour Papert) che ha portato a coniarla. Analizziamo poi le eredità di tale contesto (apprendimento creativo e informatica creativa), per trarne utili princìpi, validi ancora oggi. Presentiamo poi un esempio (i club di programmazione CoderDojo) in cui tali princìpi sono visibili. Concludiamo osservando che l'origine dell'espressione mette in luce aspetti culturali diversi (ma non necessariamente contrapposti) rispetto a quelli cui di solito si fa riferimento oggi parlando di *pensiero computazionale*.

Keywords: Pensiero Computazionale · Computational Thinking · Costruzionismo · Seymour Papert · Mindstorms · CoderDojo · Creative Learning · Scratch.

1 Pensiero computazionale

L'espressione "pensiero computazionale" (PC) è ormai diventata una "buzzword", tanto che anche il meno "tecnologico" tra gli insegnanti non può non essersene imbattuto almeno una volta.

La discussione su cosa sia il "modo di pensare degli informatici" e perché sia importante insegnarlo è vecchia quanto l'Informatica stessa [2, 4, 13].

Ma quando è stata utilizzata per la prima volta questa espressione e in che contesto? Quali princìpi sono validi e utili ancora oggi? In che contesti si trovano? Che differenze emergono rispetto all'utilizzo che si fa oggi dell'espressione?

2 Mindstorms

La prima apparizione accertata dell'espressione *computational thinking* (CT) risale al 1980, quando fu usata da Seymour Papert nel suo seminale libro Mindstorms [6, pag. 182]. Papert scrive:

> Non ho dubbi che nei prossimi anni vedremo la formazione di alcuni ambienti computazionali che dovrebbero essere chiamati "scuole di samba della computazione".
> Ci sono già stati i tentativi in questa direzione [...] ma hanno fallito [...] perché troppo primitivi. I loro computer semplicemente non avevano la potenza necessaria per le tipologie di attività più coinvolgenti e condivisibili. La loro visione su come integrare il **pensiero computazionale** nella vita di tutti i giorni non era sufficientemente sviluppata. Ma ci saranno altri tentativi, e altri ancora. E, infine, da qualche parte, tutti i pezzi verranno messi insieme e [tali scuole] prenderanno piede. Si può essere sicuri di ciò perché tali tentativi non saranno esperimenti isolati realizzati da ricercatori che potrebbero finire i fondi o semplicemente disilludersi e lasciar perdere. Saranno manifestazioni di un movimento sociale di persone interessate nella computazione personale, interessate nei loro bambini, e interessate nell'educazione. [6, pag. 182, traduzione mia, grassetto mio]

3 Costruttivismo e costruzionismo

Seymour Papert (1928-2016), matematico sudafricano, ha passato alcuni anni a studiare con l'eminente psicologo Jean Piaget.

Jean Piaget è il padre dell'epistemologia genetica: lo studio dell'origine della conoscenza. Secondo Piaget, il modo con cui la conoscenza viene acquisita determina quanto essa sia valida per noi. Egli incoraggia quindi *"l'uso di metodi attivi che diano ampio spazio alla ricerca spontanea del bambino o adolescente e richiedano che ogni nuova verità da apprendere debba essere riscoperta o almeno ricostruita dallo studente e non solo trasmessa a lui"* [8, pag. 15, traduzione mia]. Proprio per questo Piaget è considerato uno dei precursori del costruttivismo.

Il comune (e un po' semplicistico) approccio all'epistemologia ritiene che esista un *mondo reale* indipendente dalla comprensione che noi abbiamo di esso, e che quindi la conoscenza consista nell'acquisizione di tale realtà esistente: il valore di verità di una proposizione dipende dal suo accordo con la realtà. Nel *costruttivismo*, invece, l'opzione ontologica è irrilevante: quel che conta è la costruzione attiva che il soggetto compie durante la sua interazione con il mondo e gli altri soggetti. La conoscenza non è copia fedele e oggettiva di una realtà indipendente, ma è una costruzione della mente dell'individuo, basata sul suo vissuto e sulla conoscenza che egli ha in sé già costruito [3].

Papert, dopo aver lavorato con Piaget, sviluppa una propria teoria dell'apprendimento: il **costruzionismo**. Esso condivide l'idea costruttivista della conoscenza *costruita tramite l'esperienza* attiva piuttosto che *trasmessa passivamente*, a cui aggiunge però che l'apprendimento è particolarmente efficace se lo studente è coinvolto nella costruzione attiva di prodotti che per lui sono significativi. Per compiere questa costruzione, egli ha bisogno di "materiali da costruzione". Papert, ad esempio, afferma di essersi "fissato" con gli ingranaggi quand'era bambino, e, in seguito, di aver usato i modelli mentali che si era for-

mato sul funzionamento di tali ingranaggi come strumento per comprendere il mondo e concetti complessi come le equazioni differenziali.

Secondo Papert, infatti, qualsiasi cosa può essere compresa facilmente se può essere assimilata alla collezione di modelli mentali già presenti in chi apprende. Si ha appunto bisogno dei cosiddetti "oggetti per pensare": una sua celebre frase recita "You can't think about thinking without thinking about thinking about something" ("Non puoi pensare al pensare senza pensare al pensare a qualcosa"). Papert sottolinea però l'importanza della componente affettiva: questi oggetti sono diversi da persona a persona (non ha senso quindi costruire, in stile Montessoriano, ingranaggi per tutti): ogni studente può essere "fissato" con cose molto diverse tra loro, che lo appassionano e gli piacciono. In questo, fortunatamente, entra in campo il computer: la sua universalità e capacità di "simulare" ed "eseguire" qualsiasi modello dà la possibilità di fornire ad ognuno i materiali da costruzione che meglio risuonano con ciò che egli ama di più.

Da quanto scritto è chiaro che il costruzionismo riserva una grande importanza al contesto in cui avviene l'apprendimento. Per fare un esempio, un bambino nato a Trento non ha problemi ad imparare spontaneamente a parlare l'Italiano, senza bisogno di un'educazione formale, almeno all'inizio. Avrà più difficoltà a imparare, ad esempio, il tedesco, studiandolo a scuola. Questo non perché, ovviamente, il tedesco sia intrinsecamente più difficile (se il bambino nascesse a Bolzano o in Austria, per esempio, imparerebbe il tedesco senza sforzi) ma perché non vive costantemente immerso in un ambiente germanofono.

Secondo Papert lo stesso vale per le altre discipline. In particolare egli fa l'esempio di "Matlandia": se i bambini potessero immergersi in un mondo in cui "si parla il linguaggio della matematica" potrebbero apprenderlo in maniera attiva e spontanea, così come fa un bambino con la sua lingua madre[1]. Ovviamente non si intende "imparare la terminologia tecnica della matematica" ma piuttosto comprenderne i principi e concetti fondanti, magari difficili da esprimere formalmente, ma assimilabili fin da bambini se si viene esposti ad essi e si ha la possibilità di farne un'esperienza pratica, attiva e "concreta".

4 Le scuole di samba

Seymour Papert scrive profeticamente nel 1980 qualcosa che accade ancora oggi: i computer a scuola vengono utilizzati per somministrare esercizi, fornire feedback, dispensare informazioni agli studenti. "Il computer programma lo studente", per usare le parole di Papert. Ciò che egli propone è invece di far sì che sia lo studente a controllare il computer e non viceversa: lo studente programma il computer, e "insegnando" al computer come pensare, esplora la propria mente, riflette su di essa, regola il funzionamento del suo stesso pensiero.

[1] Magari fin dalla tenera età: studiosi come Daniela Lucangeli sostengono infatti che le dilaganti difficoltà in Matematica siano aggravate dal fatto che, a differenza dell'intelligenza linguistica, l'intelligenza numerica venga "allenata" formalmente solo all'inizio della scuola primaria, mentre sia quasi del tutto ignorata negli anni della prima infanzia.

La visione di Papert è molto radicale: egli ritiene che le scuole attuali (nel 1980, ma anche nel 2018) saranno inutili nel futuro, quando sarà possibile creare degli ambienti di apprendimento costruzionisti grazie alle tecnologie informatiche e agli strumenti di programmazione adatti.

Papert si chiede da cosa verranno sostituite le scuole tradizionali, e si dà una sorprendente risposta, ipotizzando un modello simile alle *scuole di samba* brasiliane. Tali scuole sono fondamentali per la preparazione del famoso carnevale di Rio. Non sono scuole nel senso tradizionale/occidentale del termine, ma piuttosto dei club che vanno da centinaia a migliaia di persone. L'età dei membri varia: si va dai bambini ai loro nonni, dai novizi ai professionisti. I membri di ciascun club si riuniscono ogni weekend per ballare e per incontrarsi con gli amici. Tutti ballano: i novizi imparano, i più grandi insegnano ma si esercitano anche per le mosse più difficili. Nelle scuole di samba c'è una grande coesione sociale, un grande senso di appartenenza, una forte idea di avere uno "scopo comune". Sebbene l'apprendimento sia spontaneo e naturale è anche deliberato: il risultato di un anno di lavoro sono infatti rappresentazioni spettacolari di livello professionale in cui ci sono richiami alla tradizione ma anche forti critiche politiche.

Possiamo ora facilmente comprendere la citazione iniziale: Papert si augura la nascita di ambienti, analoghi alle scuole di samba, in cui i ragazzi possano imparare i principi della computazione (e contemporaneamente usarli per imparare anche le altre discipline, riviste in un'ottica computazionale) in modi alternativi alla scuola, perché immersi in un ambiente in cui vengono svolte attività ricche di principi informatici e significative per la comunità che li frequenta.

Queste "profezie" rimangono solo l'utopia di un visionario matematico?

5 Il Lifelong Kindergarten e le 4P dell'apprendimento creativo

Mitch Resnick, ex studente di Papert e ora professore che occupa la cattedra che porta il suo nome, oltre che direttore del *Lifelong Kindergarten group* al MIT Media Lab, sostiene che le scuole dovrebbero assomigliare molto di più alle scuole dell'infanzia ("asili", "kindergarten").

Secondo Resnick [9, 11], invece di rendere gli asili più simili alle scuole, bisognerebbe rendere le scuole più simili agli asili, in cui avvengono eventi di apprendimento "spontaneo", "dal basso", sebbene con il supporto degli insegnanti. Resnick fa l'esempio di due bambini che giocano con dei blocchi, provano a costruire una torre, ma poi un altro bambino li fa cadere con una macchinina. Dopo avere discusso tra loro e fatto ipotesi sulla causa del crollo, vengono mostrate loro dall'insegnante foto di grattacieli ed essi notano che ha una base più ampia, e a questo punto ricominciano la loro costruzione...

Questo accade più facilmente tra i bimbi della scuola dell'infanzia, perché necessitano di semplici materiali come mattoncini o tempera; molto più complicato sarebbe per gli studenti più grandi, che devono imparare concetti via via più complessi e astratti.

Come già predetto da Papert, però, la disponibilità di questi materiali è ora data dagli ambienti virtuali che possono essere creati con i computer e dai linguaggi di programmazione adatti (come ad esempio Scratch[2]), che permettono di apprendere sperimentando con dei "blocchi" concettuali - che rappresentano le istruzioni - proprio come i bimbi dell'esempio precedente facevano con i blocchi "fisici" a loro disposizione.

Resnick sostiene inoltre che l'attuale distanza tra il modello di apprendimento degli asili e quello delle scuole sia dovuta anche al fatto che la società non riconosce ancora completamente l'importanza di insegnare agli studenti il *pensiero creativo*.

Un illustratore francese provò ad immaginare, nel 1900, come sarebbe stata la vita nel 2000. Tra le stampe, egli presentò una ipotetica classe in cui gli studenti vengono istruiti attraverso macchinari collegati al loro cervello all'interno del quale l'insegnante inserisce i libri (Fig. 1).

L'inquietante scenario rimanda da un lato all'"istruzione programmata", proposta negli anni Trenta dai comportamentisti quali Burrhus Skinner, e dall'altro ai computer usati come strumento alternativo alla lezione tradizionale, per dispensare comunque lezioni frontali e nozionistiche.

Ma l'immagine stimola anche un'altra riflessione: ipotizziamo che un insegnante del 1900 entri in una classe di oggi: probabilmente capirebbe benissimo le interazioni che avvengono e potrebbe persino tenere una lezione. Lo stesso non si può dire per altri mestieri: se, ad esempio, un chirurgo di 100 anni fa entrasse in una moderna sala operatoria non capirebbe probabilmente nulla di ciò che sta accadendo [1, 7].

Secondo molti osservatori la scuola di oggi è ancora modellata per una società che aveva grande bisogno di impiegati: lavoratori a cui è richiesto in particolare di essere abili e precisi nell'eseguire compiti spesso meccanici e ripetitivi impartiti loro dai capi. Ma questo modello è sempre più anacronistico per la società odierna: i compiti meccanici e ripetitivi sono e saranno sempre più automatizzati da robot. Di più: anche lavori che oggi richiedono comunque una componente "mentale" umana saranno rimpiazzati da intelligenze artificiali più accurate degli esseri umani stessi. Ciò che fa - e che farà sempre di più - la differenza non è più la conoscenza di una serie di nozioni o di tecniche, ma piuttosto la capacità di essere creativi, innovativi, la capacità di adattarsi a una società "liquida" e adattarsi a un mondo in costante cambiamento. Se un tempo il problema era l'accesso alle informazioni, ora viviamo il problema opposto: siamo sommersi da informazioni e sembra scarseggiare ciò che sarebbe invece determinante, e cioè il senso critico e la capacità di comprendere, elaborare (anche con l'ausilio delle tecnologie informatiche) e utilizzare tali informazioni per i propri scopi.

Per tutti questi motivi, Resnick propone un modello [10] per l'apprendimento creativo ("creative learning") anche tramite l'informatica creativa ("creative computing"). Tale modello basato è basato sulle cosiddette "4 P" (dalle iniziali inglesi dei quattro punti chiave che lo costituiscono).

[2] Ambiente di programmazione sviluppato dal gruppo di M. Resnick al MIT, ed erede del LOGO, linguaggio creato da Seymour Papert. `https://scratch.mit.edu/`

- Lavorare per progetti (*Projects*): è importante far sì che gli studenti lavorino attivamente su progetti, più che su esercizi. Un esercizio infatti è spesso artificiale e scollegato dalla realtà, mentre un progetto è di solito legato in modo sensato a qualcosa di reale e concreto. Si può apprendere testando nuove idee, realizzandone prototipi, migliorandoli, per poi arrivare a un prodotto finito (che può comunque sempre essere migliorato). Lavorare per progetti permette inoltre di imparare *mentre* si usa uno strumento, anche nuovo, piuttosto che seguire la tradizionale ma artificiosa direzione del "prima imparo, poi faccio/uso".
- Collaborare tra pari (*Peers*): è importante far sì che gli studenti collaborino, si scambino idee, costruiscano cose nuove basandosi sul lavoro dei compagni. In tutti gli ambiti, inoltre, si studiano i lavori dei "grandi" di tale settore: il codice sorgente - o i blocchi visuali - di programmi realizzati da altri possono rappresentare la "letteratura" dell'Informatica e studiarli può, di nuovo, favore un approccio basato sulla pratica e sulla sperimentazione.
- Seguire i propri interessi e le proprie passioni (*Passion*): se gli studenti sono impegnati in attività legate a ciò che li appassiona, che ritengono utile o importante, si concentreranno di più e per un tempo maggiore, non si scoraggeranno se incontreranno delle difficoltà, impariamo in modo più autentico e profondo. Per citare ancora Papert, "l'educazione ha poco a che fare con la 'spiegazione' e molto di più con il 'coinvolgimento'" (*explanation* contro *engagement*, in Inglese). Se affrontiamo un compito sfidante, ma alla nostra portata, magari che ci viene presentato *prima* che ci venga spiegata una tecnica per risolverlo, saremo molto più motivati a imparare la teoria e le tecniche necessarie, perché ne comprendiamo l'utilità in quel momento [12].
- Mettersi in gioco (*Play*): i bambini e i ragazzi non giocano soltanto per divertirsi, ma sopratutto per apprendere: è importante stimolarli affinché sperimentino in modo giocoso nuove idee, prendano rischi, procedano per prove ed errori, esplorino i propri limiti e le proprie potenzialità. Resnick sostiene che invece di parlare di "edutainment" (*education+entertainment*, cioè educazione+intrattenimento), che sono due concetti passivi, è meglio parlare di "learn and play" (imparare e giocare), che sono due concetti attivi.

Durante la Scratch Conference 2017, Resnick ha proposto una "quinta P": *Purpose* (che potremmo tradurre come un "obiettivo alto", "scopo", "fine", "missione"). Resnick infatti sottolinea come insegnare a programmare non sia solo finalizzato a far sì che gli studenti sviluppino skill tecniche, ma soprattutto a dare a tutti la possibilità di esprimersi attraverso la tecnologia. Per fare un paragone, lo scopo delle scuole di samba non è quello di insegnare a ballare *come fine*, ma quello di esprimere le proprie idee e la propria arte durante il Carnevale. Non a caso lo stesso Papert riconosce nello "scopo collettivo" uno degli elementi caratteristici e distintivi di tali scuole.

Secondo le teorie sviluppate al MIT [9,11] la creazione di un qualsiasi artefatto (esempio un programma per computer) segue la "spirale dell'apprendimento creativo" (Fig. 2): chi vuole creare qualcosa *immagina quello che vuole fare, crea un progetto basato sulla sua idea, gioca con la sua creazione, condivide la sua*

idea e la sua creazione con altri, riflette sull'esperienza e sul feedback ricevuto da altri, e tutto questo lo porta a immaginare nuove idee, nuove funzionalità, nuovi miglioramenti per il proprio progetto, o nuovi progetti [5].

Gli informatici non tarderanno a riconoscere nella spirale anche una metafora dei metodi di sviluppo software iterativi e incrementali (cosiddetti "agili") e, dunque, è essa stessa una pratica di PC.

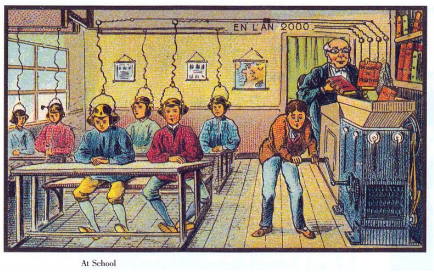

Figura 1: "A scuola": il 2000 immaginato nel 1900.
By Jean Marc Cote (if 1901) or Villemard (if 1910)
Public domain, via Wikimedia Commons
http://publicdomainreview.org/2012/06/30/france-in-the-year-2000-1899-1910/

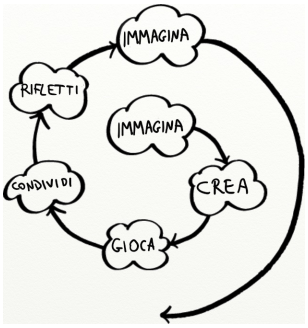

Figura 2: Spirale dell'apprendimento creativo
Traduzione di Michael Lodi, dall'originale di M. Resnick [CC BY-SA], pubblicata in [5]

6 CoderDojo[3]

Dal 1980 sono passati 37 anni, e le scuole (sebbene con molte spinte innovative) mantengono ancora la loro struttura tradizionale. Le tecnologie e le applicazioni informatiche hanno permeato ormai tutti gli aspetti della vita umana. Le persone però sembrano utilizzarle in modo passivo - in alcuni casi subendole, spesso senza conoscere per nulla "cosa ci sta dietro".

Nessuna "scuola di samba della computazione" in cui si sviluppa il pensiero computazionale ha davvero preso piede.

Ma nel mondo sono nati moltissimi club che hanno alcune caratteristiche descritte da Papert: luoghi in cui genitori e appassionati sono interessati all'educazione dei propri figli, e si ritrovano per far sì che i bambini possano "allenarsi" ad essere attivi e creativi con l'Informatica. In Italia, in particolare, dal 2013 hanno avuto grande diffusione i CoderDojo.

Coder significa programmatore e *Dojo* è il luogo in cui si praticano le arti marziali. Dunque un CoderDojo è una "palestra" per programmatori. Una "palestra" nel senso orientale del termine: in cui i più esperti insegnano ai novizi, ma in cui tutti vanno per allenarsi e imparare, proprio come nelle scuole di samba.

[3] Alcune parti di questa sezione sono riadattate e aggiornate a partire dal testo di cui sono co-autore: [5, cap. III, par 3.]

Si tratta di un movimento (internazionale, volontario, no-profit) di club gratuiti di programmazione per bambini e ragazzi (di solito 7-17, ma esistono eventi per bambini anche più piccoli). CoderDojo nasce nel 2011 in Irlanda dal diciottenne James Whelton che, trovatosi per caso a insegnare a programmare ad alcuni compagni di scuola e vedendo il suo "modello a club" crescere vertiginosamente nel numero di partecipanti, riceve aiuto per strutturare il progetto dal benefattore Bill Liao.

I club si sono diffusi rapidamente per l'Irlanda e poi nel mondo: a fine 2017 si contavano quasi 1500 Dojo in più di 60 paesi nel mondo. Il movimento è supportato dalla CoderDojo Foundation, ogni Dojo è però autonomo.

Chiunque può creare un proprio club, a patto di rispettare alcuni principi: gratuità dei laboratori, tutela dei ragazzi, condivisione delle risorse, collaborazione e senso civico, così via. La prima regola, però (ed anche il motto del movimento) è "*Above All, Be Cool: bullying, lying, wasting people's time and so on is uncool*" ("Prima di tutto, sii in gamba. Fare il bullo, mentire, far perdere tempo agli altri e così via non è da persone in gamba").

Aprire un CoderDojo non richiede molte risorse, basta trovare uno spazio fisico che ospiti gli eventi (biblioteche, FabLab/MakerSpace, scuole, aziende e così via). La risorsa più importante è costituita però dai mentor volontari: appassionati che donano il loro tempo per supportare i bambini.

Di solito i mentor sono sviluppatori o comunque esperti di tecnologia, ma anche insegnanti e genitori. Non è fondamentale avere competenze avanzate di programmazione, è invece importante la capacità di relazionarsi con i giovani "ninja", cioè i bambini che partecipano. Il mentor infatti non è un insegnante "tradizionale", ma un facilitatore (o un "allenatore"), che sta accanto ai ragazzi affinché siano loro stessi a imparare tramite la scoperta, la sperimentazione e la collaborazione con gli altri.

Il fenomeno è arrivato anche in Italia, e ha preso piede: nel 2016 si contavano più di 150 CoderDojo. Ogni Dojo è indipendente, ma in Italia (anche grazie all'influenza, tra gli altri, di Carmelo Presicce[4] e del Dojo di Bologna) di solito gli eventi sono strutturati in 3 ore: un breve tutorial iniziale a tema (es. Natale, vacanze, ecc.), per coloro che partecipano la prima volta o per chi è in cerca di idee nuove; una pausa merenda, per staccare e giocare con gli altri partecipanti; infine la maggior parte del tempo lasciata alla sperimentazione libera, in cui i bambini possano realizzare il proprio gioco, storia, animazione, collaborando e ricevendo se necessario il supporto dei mentor.

All'evento, i partecipanti devono portare: un computer portatile, un genitore (che resta per tutto il tempo, sia per questioni di responsabilità sia perché questo permette di farlo avvicinare ai mondi spesso sconosciuti e "spaventosi" in cui i suoi figli si muovono con dimestichezza), e una merenda.

Per iniziare con la programmazione, lo strumento più utilizzato in Italia è Scratch, senza però tralasciare i più piccoli, coinvolti con strumenti quali ScratchJR o Lightbot, o con attività *unplugged*. Non mancano attività per i ragazzi più grandi, a cui si insegna la costruzione di pagine web (HTML e Java-

[4] Champion del CoderDojo Bologna e ora studente di Mitch Resnick al MIT.

script), o videogiochi con Python, o attività di *tinkering* e *making* con Arduino, RaspberryPi, Micro:bit, e così via.

Gli eventi si pongono in un contesto di apprendimento non formale, dunque distante dal modello di scuola tradizionale e strettamente affini alla didattica laboratoriale, per lo sviluppo di competenze autentiche. Appare chiara l'ispirazione data dalle "4 P" e dal modello dell'apprendimento creativo.

Proprio perché si tratta di un modello di educazione non formale, la sua trasposizione nelle scuole non è immediata: i *ninja* sono "auto-motivati", spesso figli di genitori sensibili al tema, non ci sono valutazioni, obiettivi di apprendimento formali da raggiungere, e così via. La vera sfida è proprio quella di cogliere gli aspetti positivi e peculiari di questi modelli per adattarli al contesto scolastico, senza svuotarli di senso e, soprattutto, di efficacia. È importante comunque sottolineare che la diffusione dei Dojo sta "contaminando le scuole": in Italia si organizzano i cosiddetti TeacherDojo, cioè Dojo in cui al posto dei ragazzi ci sono gli insegnanti che vogliono formarsi e portare il modello nelle loro classi; molti mentor dei Dojo sono loro stessi insegnanti; spesso i membri del Dojo vengono chiamati nelle scuole per tenere laboratori o corsi di formazione, e così via.

Poiché si tratta di un movimento gestito da volontari nel loro tempo libero, difficilmente essi potranno raggiungere tutte le scuole. L'invito per gli insegnanti è allora quello di partecipare al Dojo più vicino: per osservare, per diventare mentor, senza paura di mettersi in gioco e di imparare dai ninja stessi, per poi portare nelle loro classi tanto le competenze informatiche quanto le metodologie innovative che hanno sperimentato in prima persona durante i CoderDojo.

Non è detto che il modello dei club CoderDojo scali mantenendo la sua natura. Sebbene la Fondazione voglia preservare i valori originali, per offrire l'opportunità di partecipare a più ragazzi e per mantenere la qualità dei laboratori, a livello internazionale si sta cercando di dare più struttura ai club e si stanno facendo accordi con grandi multinazionali tecnologiche.

Ciò che non si può negare è però il fatto che tali club stanno gettando i semi nella società perché si creino quelle "computer cultures" sognate da Seymour Papert.

7 Conclusioni

Abbiamo visto come l'espressione *pensiero computazionale* sia nata da una specifica visione di Seymour Papert. Sebbene il suo testo del 1980 risulti profetico soprattutto negli aspetti negativi (es. l'uso dei computer come strumenti calati in un contesto di didattica tradizionale), in esso viene prospettata la possibilità che gli strumenti informatici diventino quei materiali da costruzione per apprendere in modo pratico e costruttivo concetti complessi, ad ogni livello scolastico e per ogni disciplina.

Oggi sosteniamo che l'Informatica debba diventare una disciplina scolastica autonoma, così come lo sono le altre scienze, fondamentale per comprendere il mondo di oggi [4]. Ma questa ricostruzione storica ci fa capire che il *pensiero computazionale* ha anche un alto valore trasversale (così come, ad esempio, la

lettura, la scrittura e il far di conto): usare principi, strumenti e metodi informatici per creare, simulare ed eseguire mondi e astrazioni altrimenti impossibili da "concretizzare", per imparare "immergendosi" in tali mondi, e dunque facendone un'esperienza pratica, autentica, creativa.

In questo momento di grande attenzione per il tema, sarà importante far vivere e sostenere entrambe le visioni, facendole coesistere senza ridurre l'una all'altra, per mantenere sia la componente scientifica dell'informatica, che porta i suoi linguaggi e i suoi metodi caratteristici come saperi fondamentali, sia la profonda trasversalità e innovazione didattica, sociale e culturale che la diffusione dell'informatica può favorire.

Ringraziamenti

Grazie Simone Martini, che supporta la mia ricerca; a Carmelo Presicce, per avermi coinvolto in CoderDojo; grazie ad entrambi per i suggerimenti su questo articolo. Grazie a Renzo Davoli, per le sue visioni su scuola e informatica, che hanno ispirato alcune mie riflessioni.

Riferimenti bibliografici

1. Capponi, M.: Un giocattolo per la mente. L'≪informazione cognitiva≫ di Seymour Papert. Morlacchi Editore (2009)
2. Lodi, M.: Imparare il pensiero computazionale, imparare a programmare. Mondo Digitale **13**(51), 822–833 (2014)
3. Lodi, M.: Imparare il pensiero computazionale, imparare a programmare. Master's thesis, Università di Bologna, Corso di Studio in Informatica (2014)
4. Lodi, M., Martini, S., Nardelli, E.: Abbiamo davvero bisogno del pensiero computazionale? . Mondo Digitale (72), 1–15 (Nov 2017)
5. Marchignoli, R., Lodi, M.: EAS e pensiero computazionale. ELS LA SCUOLA (2016)
6. Papert, S.: Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Inc., New York, NY, USA (1980)
7. Papert, S.: The children's machine: Rethinking school in the age of the computer. ERIC (1993)
8. Piaget, J.: To Understand is to Invent: The Future of Education. Penguin Books (1973)
9. Resnick, M.: All I Really Need to Know (About Creative Thinking) I Learned (by Studying How Children Learn) in Kindergarten. In: Proc. of the 6th ACM SIGCHI Conf. on Creativity and Cognition. pp. 1–6. ACM, New York, NY, USA (2007)
10. Resnick, M.: Give p's a chance: Projects, peers, passion, play. In: Proceedings of Constructionism and Creativity Conference, Vienna, Austria (2014)
11. Resnick, M.: Lifelong Kindergarten: Cultivating Creativity Through Projects, Passion, Peers, and Play. MIT Press (2017)
12. Schwartz, D.L., Bransford, J.D.: A time for telling. Cognition and Instruction **16**(4), 475–5223 (1998)
13. Tedre, M., Denning, P.J.: The long quest for computational thinking. In: Proc. of the 16th Koli Calling Int. Conf. on CER. pp. 120–129. Koli Calling '16, ACM, New York, NY, USA (2016)