Bibliografia ragionata su coding e pensiero computazionale nel contesto scolastico

Andreas R. Formiconi

Aprile 2017

v. 0.3

Lo scopo di questo testo è di commentare articoli scientifici apparsi sul tema del *coding* a scuola. L'idea generale è quella di costruire una bibliografia ragionata su un tema ampiamente dibattuto, forse a ragione, ma spesso in maniera troppo superficiale e partigiana. Le opinioni abbondano, i fatti scarseggiano. Il motivo specifico per cui mi sono messo a analizzare questi lavori è la superficialità con cui alcuni di questi lavori sono stati citati. L'analisi vuole quindi anche suggerire maggiore cautela e maggiore riflessione nell'impiego dei cosiddetti risultati scientifici.

Si tratta solo di una bozza iniziale, scritta in fretta per circostanze contingenti, sicuramente carente, afflitta da errori e refusi. Ci sono però motivi per condividerla anzitempo, anche solamente per invitare chiunque a segnalare articoli e contributi interessanti. In futuro introdurrò una classificazione mediante *tag*, per facilitare la ricerca e forse utilizzerò anche un altro formato. Per ora mi sono limitato a redigere la bozza in un documento ODT e a distribuire i lavori in ampie categorie.

In fondo si riportano due distinte liste di riferimenti bibliografici: la prima contiene le voci commentate nella presente bibliografia ragionata, nella seconda invece riporto altre voci che ho trovato opportuno citare ma che non ho ancora discusso.

0. Premessa intorno alla letteratura scientifica: delle critiche di Ioannidis

Questa sezione è a beneficio di chi non conosce bene la letteratura scientifica, in particolare di quella riferita ai domini delle scienze biologiche e sociali. Ne raccomando la lettura per mettersi in grado di valutare con maggior profitto i lavori commentati. In particolare le considerazioni svolte qui verranno riprese in dettaglio nella discussione dei lavori di Scaffidi e Chambers (2016) e Matias et al (2016), nelle sezioni 1.1 e 1.2 rispettivamente.

Nella letteratura scientifica domina il paradigma dell'esperimento scientifico, nel quale il ricercatore cerca di fissare tutte le condizioni in modo da interrogare solo alcune variabili al variare di altre. È il paradigma che ha sostenuto con grande successo il progresso delle scienze, in particolare di quelle fondamentali, quali la fisica e la chimica. La sua applicazione si fa tuttavia più problematica con la complessità dei fenomeni indagati, massimamente di quelli oggetto delle scienze biomediche e delle scienze sociali. In quest'ultime in particolare si assiste ad un paradosso: cercando di determinare precisamente parametri ben definiti si finisce con l'ottenere informazione quantitativa ma relativamente poco significativa ai fini della comprensione del fenomeno; allargando invece l'attenzione su aspetti più generali e significativi si perde in determinazione. L'applicazione del

primo paradigma conduce a ampie moli di informazioni che si risolvono in una somma di microconoscenze, non facilmente integrabili e riassumibili, e molto facilmente mistificabili se decontestualizzate e citate a sproposito. Il secondo paradigma si attaglia meglio alla descrizione dei fenomeni complessi ma offusca i crismi delle indagini scientifiche classiche, ispirati all'obiettività e alla riproducibilità.

Per capire bene il passaggio occorre approfondirlo meglio. Nelle scienze di base gli esperimenti servono a verificare determinati rapporti di causalità suggeriti dalla teoria. Grazie alla relativa semplicità dei fenomeni è possibile tentare di creare una situazione dove tutti i parametri in gioco sono in qualche maniera controllati in modo da poterne misurare altri per verificare le previsioni teoriche. Per "relativa semplicità" si intende il fatto di poter creare, in generale, condizioni sperimentali con maggior libertà; per esempio di poter ricorrrere al metodo *divide et impera*, suddividendo un problema in sottoproblemi più semplici, con l'obiettivo di mettere alla prova specifici rapporti di causalità. Esperimenti di questo tipo non sono scevri da errori: tutte le misure sperimentali sono affette da errori ma esistono metodi per valutare come questi si propagano nelle quantità derivate da quelle misurate.

La situazione nelle scienze biologiche o sociali è invece radicalmente diversa. Qui gli esperimenti si devono effettuare su sistemi di enorme complessità, che non possono certo essere scomposti e non possono nemmeno essere sradicati dal contesto. Tutto quello che si può fare è distinguere fra situazioni in cui i sistemi (animali, persone, popolazioni) sono sottoposti a nuovi fattori e situazioni in cui questi sono assenti. Per esempio si possono creare gruppi sottoposti a un trattamento e gruppi in cui il trattamento è assente, i cosiddetti gruppi di controllo; tutto ciò nella speranza che fra i due tipi di gruppi non vi siano altre differenze o che queste siano comunque trascurabili. In un simile contesto lo strumento di analisi principe è la statistica che fornisce oggi metodi molto sofisticati per l'analisi dei risultati degli esperimenti in una grande varietà di condizioni. Deve essere però chiaro un fatto fondamentale: nessun metodo statistico è in grado di dimostrare alcun rapporto di causalità ma può semplicemente verificare la probabilità di un risultato, con una valutazione della sua solidità statistica. Per chiarire: io posso scoprire una correlazione fra due fenomeni completamente diversi, per esempio fra la quantità di urine escrete da una popolazione in Italia e il flusso d'acqua delle cascate del Niagara ma niente mi autorizza a dichiarare che le urine prodotte da quella popolazione "dipendono" dal flusso delle cascate del Niagara. Il risultato può essere emerso per puro caso e non esiste alcun metodo certo per mettersi al riparo da simili cantonate.

Certo, esistono strategie per diminuire il rischio ma non è affatto banale farvi sistematicamente ricorso. E non è quello che succede nella realtà, purtroppo. *Why most published research findings are false* (Ioannidis, 2005) è il titolo di un popolarissimo articolo di John Ioannidis, epidemiologo greco alla Stanford University. Forse uno degli articoli più citati nella letteratura internazionale di questi anni, scaricato oltre un milione di volte, oggetto di attenzione anche da parte della *mainstream information*, soprattuto in campo economico. È semplice la ragione: per dare un'idea, nel 2010 solo negli Stati Uniti i finanziamenti alla ricerca nel settore biomedico sono stati dell'ordine di 240 miliardi di dollari, ebbene, è stato stimato che l'85% di questi investimenti sono andati sprecati a causa della scarsa rilevanza dei risultati – qualcosa come 200 miliardi di dollari. Il motivo risiede nel fatto che la grande maggioranza dei risultati pubblicati sembrano promettenti ma poi solo raramente si traducono in benefici concreti (Macleod et al, 2014). Questo non significa che la scienza non progredisca, è la letteratura ad essere bulimica e in larga parte inessenziale, e questo certo non giova al progresso stesso. Dello stesso male è afflitta la letteratura "quantitativa" nelle scienze sociali, anche se in questo caso i benefici concreti sono assai più difficili da rilevare.

Cerchiamo di capire in cosa consista la debolezza quantitativa delle ricerche in questi campi

approfondendo un poco i ragionamenti di Ioannidis. Prima però occorre dire due parole sulla natura degli errori statistici in questo genere di studi. Si distinguono due generi di errori[1]: gli errori di tipo I, quelli che si commettono quando un'ipotesi viene ritenuta vera mentre non lo è (falso positivo); gli errori di tipo II sono quelli che si commettono quando si ritiene che un'ipotesi sia falsa mentre invece è vera (falso negativo). Occorre accettare un compromesso fra i due errori perché non è possibile ottimizzare ambedue indipendentemente: riducendo l'occorrenza dei falsi positivi si aumenta quella dei falsi negativi e viceversa. Tuttavia la letteratura è fortemente sbilanciata sulla rilevazione dei risultati positivi, concentrandosi sulla riduzione dei falsi positivi. Di fatto nella quasi totalità degli articoli ci si limita a stabilire il "livello di significatività". Si dice che un risultato è "statisticamente significativo" se questo ha una probabilità inferiore a un certo valore limite di essere dovuto al caso, anziché alle cause indagate nello studio. I livelli di significatività dipendono dal campo di ricerca: nelle scienze biomediche si tende a richiedere l'1% mentre in quelle sociali ci si accontenta del 5%, in generale. È così che si assicura la significatività statistica nella quasi totalità della letteratura scinetifica.

Ma vediamo quali sono le obiezioni principali di Ioannidis. Sono tre: la potenza del test statistico; la scarsa verosimiglianza delle ipotesi testate; lo sbilanciamento nei confronti dei risultati positivi.

La potenza del test statistico consiste nella capacità di rivelare anche piccoli effetti nei dati. Il fattore principale è la dimensione delle indagini: la potenza migliora se gli studi sono ripetuti più volte o vengono effettuati su campioni molto estesi. La potenza si misura con un numero che va da 0 a 1. Per esempio un valore di 0.8, ritenuto genericamente accettabile, significa che testando dieci ipotesi vere due di queste sfuggiranno; un prezzo che viene genericamente ritenuto accettabile. Purtroppo molto spesso questo requisito non è garantito, per vari motivi fra i quali, non ultimi, la forte pressione alla pubblicazione e il controllo dei costi: per un ricercatore una mancata pubblicazione costa molto più della pubblicazione di un risultato sbagliato o ininfluente. Marjan Bakker et al (2012) hanno stimato che nel campo delle scienze psicologiche la potenza dei test statistici di fatto si aggira intorno a 0.35; in altri campi si può arrivare anche a valori intorno a 0.20.

Facciamo un esempio per capire bene come stanno le cose. Supponiamo di testare 1000 ipotesi delle quali solo 100 sono vere (quadrati verdi nella figura[2]). Con un test di potenza 0.80 ne troveremo 80 (80% delle 100 ipotesi vere) e le 20 rimanenti finiranno fra i falsi negativi (quadrati rossi). Delle 900 ipotesi false (quadrati gialli) il 5% verranno scambiate per vere, a causa dell'accettazione del 5% di falsi positivi, quindi 45 nel nostro esempio (quadrati neri). Come risultato in totale noi "vediamo" 80+45=125 (verdi+neri) risultati positivi, dei quali circa un terzo (nero) sbagliati, e ci siamo persi 20 (rosso) ipotesi positive, che sono così diventate falsi negativi.

1 Coloro che hanno poca confidenza con la letteratura statistica, potrebbero confondersi leggendo le definizioni riportate nei testi base di statistica, dove gli errori sono riportati facendo riferimento all'accettazione o meno dell'ipotesi nulla, H_0, complementare all'ipotesi alternativa H_1: se è vera l'una è falsa l'altra e viceversa. Gli statistici sono usi riferire le argomentazioni all'ipotesi nulla H_0, che rappresenta per così dire il diniego del quesito sperimentale. Nel testo abbiamo descritto gli errori di I e II tipo riferendoci all'ipotesi H_1, seguendo Ioannidis (2005), perché ci sembra più intuitivo per chi non è abituato al gergo statistico.

2 Le figure sono state realizzate con la Tartaruga di Seymour Papert, ovvero con l'implementazione LibreLogo di Logo, che consente di inserire grafiche all'interno dei documenti in formato ODT in LibreOffice.

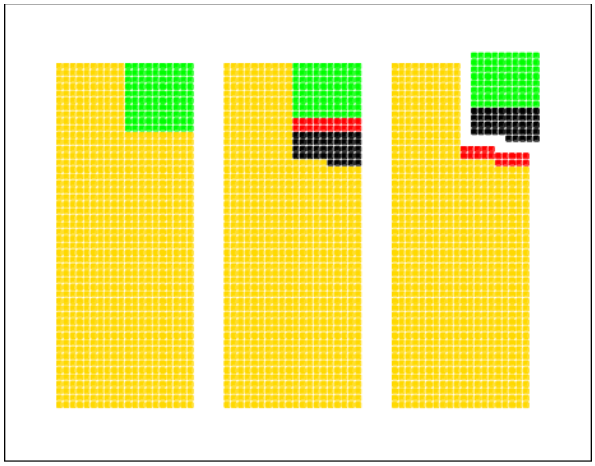

Illustration 1: Potenza del test pari a 0.80. Sinistra: proporzione reale di ipotesi vere (verde). Centro: in rosso il 20% delle ipotesi vere che sono così falsi negativi; in nero il 5% delle 900 ipotesi false che sono così falsi positivi. Destra: in evidenza i positivi che il ricercatore "vede".

Supponiamo ora di mantenere la significatività del 5% (accettazione di falsi positivi, neri) ma di lavorare con un test di potenza 0.2. In questo caso troviamo 20 (verde) ipotesi vere mentre 80 (nero) finiscono fra i falsi negativi. Questa volta "vediamo" 20+45=65 (verde+nero) risultati positivi, dei quali più di 2/3 (nero) sbagliati. Di concerto ci siamo persi 80 (rosso) ipotesi vere che sono invece diventate falsi negativi. Come dire che abbiamo fatto uno studio praticamente inutile!

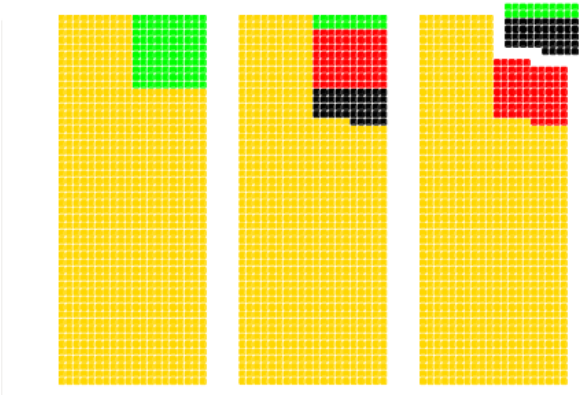

Illustration 2: Potenza del test pari a 0.20. Sinistra: proporzione reale di ipotesi vere (verde). Centro: in rosso l'80% delle ipotesi vere che sono così falsi negativi; in nero il 5% delle 900 ipotesi false che sono così falsi positivi. Destra: in evidenza i positivi che il ricercatore "vede".

Il problema di una grande quantità di studi è che ci si limita a stabilire la significatività senza curarsi

di valutare la potenza del test, talvolta per ignoranza, talvolta proditoriamente, perché migliorare la potenza “costa”, come vedremo successivamente.

La seconda obiezione di Ioannidis concerne la verisimiglianza delle ipotesi testate. È naturale che i ricercatori puntino preferibilmente a ottenere risultati sorprendenti ma questi comportano che la verisimiglianza delle ipotesi sia bassa. Il valore di una ipotesi vera su dieci, sul quale abbiamo costruito l'esempio precedente, è realistico in molti contesti ma vi sono campi in cui può essere molto più basso. Lo studio di ipotesi poco verosimili, ancorché attraenti, esaspera la criticità dei test.

Infine, la terza obiezione concerne ogni tipo di sbilanciamento, distorsione o pregiudizio (*bias*) delle indagini. Si intendono con questo i molteplici fattori (di progettazione, analisi e presentazione) che possono far comparire risultati laddove non ve ne sono o, all’inverso, possono nasconderli laddove invece esistono. Per esempio la manipolazione dei dati a posteriori, al fine di migliorare l’esito di un test, magari eliminando dati “troppo” strani (*outliers*), oppure la pubblicazione sistematica di risultati positivi e l’omissione di quelli negativi.

Una volta inquadrato il contesto, concentriamoci ancora un po’ sulla potenza, perché ci tornerà utile per svolgere alcune considerazioni nella bibliografia ragionata. La potenza di un testo è inluenzata (Chiandotto, 1978, p.102):

1. dal livello di significatività α prescelto;
2. dalla specificazione dell’ipotesi alternativa;
3. dalla numerosità del campione.

Al fine delle discussioni successive ci è utile soffermarci sulla numerosità del campione, anche perché è un fattore preminente nel determinare la qualità di un test statistico. La regola pratica è molto semplice: aumentando la dimensione del campione i test migliorano – la dimensione è un po’ il “carburante” dei test. La cura sembra quindi molto semplice ma, purtroppo, nella maggior parte dei casi è molto costosa, sia in termini di costi vivi che di tempo. Inoltre, non di rado, ci possono essere motivi obiettivi che rendono impossibile incrementare la dimensione del campione oltre certi limiti. Tuttavia, considerate le legittime e preoccupanti critiche alla Ioannidis, si va lentamente formando una consapevolezza crescente nei riguardi della problematica che ha dato luogo a articoli e strumenti software destinati alla determinazione della numerosità dei campioni un po’ in tutti i domini di ricerca pertinenti. Nel seguito (a proposito del lavoro di Kalelioğlu e Gülbahar, 2014) ci tornerà utile deteminare la potenza di un test bilaterale della media su uno stesso campione, senza e con “trattamento” (Chow et al, 2008, p. 51):

$$1-\beta=\Phi\left(z-z_{1-\alpha/2}\right)+\Phi\left(-z-z_{1-\alpha/2}\right) \quad , \quad z=\frac{\mu-\mu_0}{\sigma/\sqrt{n}} \qquad \text{(Eq. 1)}$$

dove:

n è la dimensione del campione

μ_0 è il valore medio di riferimento della variabile

μ è il valore medio della variabile

σ è la standard deviation

Φ è la distribuzione normale standardizzata

$z_{1-\alpha/2}$ è il valore della variabile standardizzata per cui la probabilità è inferiore o eguale a $1-\alpha/2$
α è l'errore di tipo I
β è l'errore di tipo II
$1-\beta$ è la potenza del test

Alternativamente, potremo determinare la dimensione del campionamento, data la potenza:

$$n = \left(\sigma \frac{z_{1-\alpha/2} + z_{1-\beta}}{\mu - \mu_0} \right)^2 \qquad \text{(Eq. 2)}$$

Per mezzo di queste e analoghe relazioni possiamo fare delle considerazioni interessanti sulla solidità statistica dei risultati riportati in alcuni lavori, ancorché questi riportino i dati necessari: descrizione accurata degli esperimenti, valori numerici misurati e relative standard deviation.

Chiudiamo questa premessa, per alcuni forse un po' pesante ma utile al fine di ricordare che, per quanto sia auspicabile fare riferimento allo stato dell'arte della ricerca, occorrano prudenza, accortezza e onestà intellettuale nella citazione dei risultati, ponendo grande attenzione ai contesti cui essi si riferiscono. A maggior ragione quando questi vengano utilizzati in arene, quali quelle dei *social network*, dove il dubbio e la ponderazione trovano scarsa cittadinanza.

1. Scraping di scratch.mit.edu

Si tratta di lavori nei quali si analizzano dati estratti dal database di progetti Scratch mediante appositi software, detti di *scraping*. Li ho inclusi perché alcuni di questi li ho visti citare a proposito del *coding* a scuola, o forse sarebbe meglio dire a sproposito, perché in realtà dicono poco o nulla su questo tema specifico ma piuttosto sull'efficacia di un ambiente di produzione e condivisione di software quale è Scratch. Alcuni di questi lavori sono metodologicamente buoni anche se i risultati che forniscono sono abbastanza prevedibili e in ogni caso pongono problemi che sono estranei al contesto della classe. Affrontano il contesto di un social network che è completamente differente: qui le persone sono abbandonate a se stesse, mentre nella classe sono assistite all'interno di una precisa dinamica didattica. Va anche detto che sono studi che si concentrano sulle "animazioni" quali prodotti elettivi delle attività in Scratch. Per ambedue questi motivi, si tratta quindi di studi di utilità limitata ai fini del coding quale attività scolastica.

Tuttavia la successione dei due lavori seguenti è molto interessante perché illustra in modo esemplare la questione della numerosità del campione, che abbiamo discusso nella sezione precedente. In sostanza Scaffidi e Chambers (2016), lavorando su un sottoinsieme del database di progetti accumulati in Scratch, hanno rilevato che, sorprendentemente, alcune competenze di coding diminuiscono con il tempo passato a lavorare con Scratch, anche se debolmente. Nel lavoro successivo, Matias et al (2016) hanno ripetuto lo stesso studio ma utilizzando il database completo, 360 volte più numeroso, dimostrando come Scaffidi e Chambers avessero ottenuto quel risultato negativo per "sfortuna", a causa del campione troppo ridotto, quindi con una potenza del test insufficiente. Discuteremo la questione più in dettaglio commentando il lavoro di Matias et al (sezione 1.2).

1.1 Scaffidi e Chambers (2016) - Skill progression demonstrated by users in the Scratch animation environment

Scaffidi e Chambers indagano la progressione delle competenze degli utenti in Scratch.
Si tratta di un'analisi effettuata a posteriori sui materiali accumulati nel servizio Web di Scratch.
L'analisi è condotta su un campionamento casuale di 250 utenti, per un totale di 1791 animazioni.
Gli autori focalizzano l'attenzione sulla creazione di animazioni, non tanto come attività propedeutica alla programmazione professionale ma quale attività formativa per la vita e il lavoro.
Successivamente dichiarano che, in base ad altri studi citati e alla loro precedente esperienza, la creazione di animazioni in Scratch rivela il conseguimento di effettive competenze di programmazione. Quindi questi autori, dando per scontato l'utilità del *coding* e, in particolare dell'impiego di Scratch, si concentrano sullo sviluppo nel tempo delle competenze degli utenti della nota piattaforma di *coding*.

In sintesi i risultati sono questi:

1. tassi di abbandono molto elevati;
2. incremento con il tempo dei cosiddetti “social skills” fra coloro che insistono;
3. competenze stazionarie o decrescenti con il tempo determinate attraverso indicatori legati alle capacità di programmazione: in particolare “profondità” (numero di istruzioni usate nei progetti) e “ampiezza” (varietà delle tipologie di istruzioni). Ambedue gli indicatori diminuiscono leggermente con il tempo: la profondità in maniera non significativa al livello del 5%; l’ampiezza in modo significativo all’1%.

Gli autori, sulla base di tali risultanze, concludono domandosi se la semplice combinazione di un sistema per produrre animazioni con una comunità online, quale Scratch, sia sufficiente a coinvolgere le persone al fine di acquisire competenze di programmazione elementari.
Francamente, a una domanda del genere, posta in questi termini ci saremmo sentiti di rispondere abbastanza tranquillamente di no, senza aver bisogno di tutto quel lavoro. Lavoro peraltro caratterizzato da effetti piccoli e bassa significatività statistica – basta guardarli, i grafici che esprimono, ad esempio, la varietà di istruzioni usate o il loro numero in funzione del tempo di permanenza.

1.2 Matias et al (2016) - Skill Progression in Scratch Revisited

Matias et al riprendono il precedente lavoro di Scaffidi e Chambers eseguendo le stesse analisi ma su un database molto più esteso di 643246 progetti prodotti da 138321 utenti. Dei tre autori, J. Nathan Matias, Sayamindu Dasgupta e Benjamin Mako Hill, i primi due fanno parte del MIT Media Lab, dove è stato creato Scratch, e in effetti alla fine ringraziano per i feedback ricevuti anche Mitchel Resnick, leader del gruppo che ha sviluppato Scratch, quindi questo articolo rappresenta la risposta dello staff degli sviluppatori al risultato controverso di Scaffidi e Chambers. Avendo la possibilità di accedere pienamente a tutto il database di progetti gli autori non si sono fatti mancare nulla, ripetendo le analisi di Scaffidi e Chambers sull’intero insieme di dati disponibili, ovvero su 643246 progetti prodotti da 138321 utenti anziché su 1791 progetti di 250 utenti.

Il primo risultato è che le dipendenze della profondità e dell’ampiezza dal tempo sono risultate positive, contrariamente a quanto trovato da Scaffidi e Chambers, con un livello di significatività $p < 0.001$, quindi con una probabilità inferiore a $1/1000$ che il risultato sia dovuto al caso.
Evidentemente, considerata la numerosità del campione molto superiore, e quindi la maggiore significatività statistica, questo secondo risultato appare molto più credibile. Ma per documentare

meglio la questione, Matias e i colleghi hanno ripetuto le analisi, sezionando l'intero database in una quantità di sottocampioni e estraendoli dall'intero insieme esattamente come avevano fatto Scaffidi e Chambers, ottenendo 2000 campioni indipendenti di 1791 progetti ciascuno. Poi per ognuno di questi hanno ripetuto le stesse analisi, ottenendo i seguenti risultati: nel 70% dei campioni non è risultata alcuna dipendenza significativa dell'ampiezza dal tempo al livello di significatività dello 0.05, nel 13.5% dei casi è risultata una dipendenza positiva al livello dello 0.01, mentre nel 3.8% dei casi è emersa una dipendenza negativa, sempre al livello dello 0.01. La risposta è quindi chiara: Scaffidi e Chambers hanno avuto la sfortuna di imbattersi in uno di quei 3.8% di casi negativi. Un esempio vivido dei rischi denunciati da Ioannidis, istruttivo per comprendere come analisi statistiche di questo tipo vadano considerate con molta attenzione – una problematica calda specialmente negli ambiti delle scienze sociali e biomediche, dove una gran quantità di "conoscenza" viene acquisita sulla base di analisi statistiche.

1.3 Hermans e Aivaloglou (2016) - How Kids Code and How We Know: An Exploratory Study on the Scratch Repository

Hermans e Aivaloglou analizzano il codice prodotto dagli utenti di Scratch tratto da 250000 progetti scremati dal sito http://scratch.mit.edu (2% del totale di 14 milioni di progetti). Un buon lavoro dal punto di vista metodologico che tuttavia non studia gli effetti del *coding* nella scuola ma le caratteristiche del codice prodotto nel sito Scratch. In particolare si analizzano gli indizi di *smelling*, i cosiddetti *code smells*, ovvero gli indizi di pratiche di scarsa qualità, che in questo studio vengono ravvisati in script troppo lunghi, codice inutilizzato e codice duplicato. A favore della serietà degli autori depone il fatto che abbiano reso disponibile sia il codice del programma di *scraping* che l'intero *dataset* a disposizione in github: https://github.com/ScratchLover42/ICER-Data-Code. A favore della qualità del lavoro anche la dichiarazione esplicita e circostanziata dei limiti del lavoro, in particolare riguardo alla limitata rappresentatività del software utilizzato e del campione di soggetti coinvolto. I risultati dicono che 1) in grande maggioranza i programmi sono piccoli, 2) il livello di astrazione modesto e 3) la quantità di *code smells* è significativa. L'analisi è seria e accurata anche se sarebbe stato sorprendente trovare risultati diversi in una comunità online di questo tipo.

2. Studi controllati

Qui riprendiamo le considerazioni svolte intorno alla letteratura scientifica…

2.1 Hermans e Aivaloglou (2016) - Do code smells hamper novice programming – quanto vengono danneggiati i principianti lavorando su codice "sporco"

Hermans e Aivaloglou partono dalla constatazione che una sostanziale quantità di ricerche mostrano come i linguaggi a blocchi siano appropriati per l'apprendimento della programmazione, domandandosi se il fenomeno dei *code smells* affligga anche tali linguaggi e in che misura possa limitare l'apprendimento della programmazione. Fra le ricerche a supporto dell'utilità dei linguaggi a blocchi gli autori citano altri quattro lavori, compresi fra il 2003 e il 2015, uno dei quali (Price e Barnes, 2015) commentiamo successivamente, in quanto concerne un confronto diretto fra programmazione testuale e a blocchi, che ci interessa particolarmente. Hermans e Aivaloglou, dopo avere definito alcuni tipi di *code smells* in Scratch, si pongono tre obiettivi: 1) codici inutilmente lunghi o duplicati influenzano i tempi di comprensione dei problemi? 2) riducono inoltre la

correttezza delle soluzioni? 3) ci sono specifici quesiti di programmazione che risentono di tali *code smells*? Lo studio è stato condotto su 61 scolari di età compresa fra 12 e 14 anni, suddivisi in tre gruppi, “esposti” rispettivamente a codice pulito, codice afflitto da lunghezza eccessiva oppure da duplicazioni indebite, circa 20 studenti per gruppo quindi. Anche se la statistica è debole (20 persone per gruppo) gli autori riportano alcune correlazioni significative: gli studenti lavorano con maggiore difficoltà su programmi inutilmente lunghi o infarciti di sezioni di codice clonate e ridondanti; in particolare, i programmi lunghi risultano difficili da capire mentre quelli con duplicazioni risultano più difficili da modificare.
In sintesi un lavoro serio che indaga un aspetto molto specifico (le performance di persone che lavorano su codice di scarsa qualità) partendo dall'assunto della validità dei linguaggi a blocchi.

2.2 Kalelioğlu e Gülbahar (2014) - The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners’ Perspective

Kalelioğlu e Gülbahar si pongono due domande:

1. la programmazione in Scratch influenza le capacità di *problem solving*?
2. cosa pensano i bambini di V primaria sulla programmazione?

I risultati dichiarati sono:

1. l'analisi quantitativa non mostra differenze significative nelle capacità di problem solving in Scratch;
2. ai bambini è piaciuto e vogliono continuare.

L'approccio sarebbe interessante ma l'articolo è opaco. Le prove sono deboli, soprattutto quelle “quantitative”. È uno studio su 49 bambini in tre classi di V di una scuola privata che hanno fatto un'ora di computer a settimana. Il “trattamento”, durato 5 settimane - quindi 5 ore in totale – è consistito nell'esposizione a Scratch (variabile indipendente). La variabile dipendente dell'esperimento è invece costituita dall'esito del *Problem Solving Inventory* (PSI). Il PSI è un metodo concepito da P. Paul Heppner e Chris H. Petersen (1982) per valutare le capacità di *problem solving* in gruppi di studenti. È costituito da un insieme di affermazioni di tipo *likert* – affermazioni sia positive che negative intorno a fatti specifici. I risultati raccolti su un insieme di persone vengono poi elaboratori mediante l'analisi fattoriale. Quest'ultima è un'analisi statistica che consente di discernere le componenti principali fra le varie che possono concorrere alla risposta globale. Sono metodi che risolvono il problema mediante una trasformazione dello spazio a n dimensioni dove n sono i fattori presi in considerazione. Dal punto di vista matematico, si procede all'individuazione degli autovettori della trasformazione per determinare quali siano le componenti fondamentali dei dati, e in base all'ordine decrescente dei corrispondenti autovalori si può determinare se vi sia un piccolo sottoinsieme di tali componenti che preponderano su tutti gli altri. Analisi di questo tipo hanno successo quando il numero delle componenti principali risulta molto basso perché con queste si può spiegare l'essenza del fenomeno, consentendo di sbarazzarsi di tutte le altre componenti che contribuiscono in maniera inessenziale. Riducendo così drasticamente la dimensionalità del problema diventa molto più facile effettuare controlli di tipo quantitativo.
È necessario tuttavia attribuire il giusto significato a questo tipo di risultati. Si tratta infatti di analisi suggestive ma che non vanno assolutamente prese per dei “rivelatori magici di verità”. Non sta scritto da nessuna parte che la descrizione ottenuta attraverso i fattori principali risultanti rappresenti la realtà. Tutto quello che possiamo fare è attribuirle un valore meramente indicativo. Un singolo risultato del genere non rappresenta un avanzamento concreto nella conoscenza del fenomeno in questione ma una interessante indicazione per indagini future. In questo senso l'attributo “quantitativo” è assai opinabile e manifesta la tendenza invalsa di forzare le indagini nel campo delle scienze sociali verso il “quantitativo”, ma spesso tante indagini dichiarate

“quantitative” non lo sono affatto – numerico non è necessariamente quantitativo! - o comunque hanno una valenza quantitativa ben diversa da quella che un fisico o un ingegnere possono immaginare, e i relativi risultati vanno usati con grande prudenza.

I fattori individuati dagli autori sono (1) confidenza nella propria capacità di risolvere il problema, (2) autocontrollo e (3) tendenza ad evitare il problema. Gli autori confrontano gli esiti del PSI pre- e post-test. È questo il tipo di analisi quantitativa che consente loro di dire che le capacità di *problem solving* di quei 49 bambini sono rimaste le stesse dopo quel tipo di trattamento. Le altre rilevazioni fatte dagli autori sono di tipo qualitativo. Ed è questo risultato quantitativo quello che citano nell'abstract. Tuttavia nella discussione, dopo avere stabilito questo risultato, gli autori si affrettano a dichiarare che esso può essere ritenuto valido solo nell'ambito (scope) delimitato da questa ricerca e che con altri disegni sperimentali e in altri contesti i risultati potrebbero essere diversi. Poi dicono che, guardando i risultati in dettaglio, malgrado la durata assai breve dell’esperimento, il fattore 1 risulta essere migliorato, anche se non significativamente dal punto di vista statistico. Questo piccolo miglioramento, dicono, potrebbe avere un certo valore, supportando l'ipotesi che l'attività di programmazione possa invece influenzare le capacità di problem solving! Non il massimo della chiarezza per un paper scientifico.

Qui possiamo applicare le relazioni per la determinazione della potenza del test e della dimensione del campionamento (Eq. 1 e 2) ai dati che gli autori riportano fra i risultati, dove specificano di avere osservato un incremento del fattore “*self-confidence in their problem solving ability*”, anche se non significativo al livello del 5%. Facendo riferimento ai dati riportati in tabella 3 a pag. 43 per tale fattore, che estraiamo qui di seguito:

Fattore	Numero soggetti	Media	Std. Dev.	t-student	p
***Pre**-test self-confidence in their problem solving ability*	49	2.10	0.64	-0.67	0.506
Post-*test self-confidence in their problem solving ability*		2.17	0.69		

Inserendo tali dati in equazione 1 si ottiene una potenza pari 0.10[3]. Abbiamo quindi una situazione ancora più grave di quella illustrata in fig. 2, poiché troviamo il 90% delle ipotesi vere che sono finite fra i falsi negativi. Possiamo avere percezione della debolezza dell’esperimento anche calcolando la dimensione del campione necessaria per ottenere una potenza accettabile di 0.80 mediante l’equazione 2. Si ottiene così una dimensione del campione pari a 763, ben maggiore dei 49 soggetti dichiarati dagli autori. Cosa vuol dire esattamente questo? Significa che per avere speranza di di appurare se il valore 2.17 sia da ritenere frutto del “trattamento”, solo con una probabilità inferiore al 5% che sia invece dovuto al caso (falso postivo), e avendo tollerato una probabilità inferiore al 20% di incorrere in un falso negativo, ebbene per soddisfare tali condizioni occorre che nello studio vengano inclusi 763 soggetti.

3 I calcoli sono stati eseguiti mediante il seguente codice R, opportunamente arrangiato nei due casi:

```
mu=2.17
mu0=2.10
sd=0.69
alpha=0.05
beta=0.20
(n=(sd*(qnorm(1-alpha/2)+qnorm(1-beta))/(mu-mu0))^2)
ceiling(n)# 32
z=(mu-mu0)/sd*sqrt(n)
(Power=pnorm(z-qnorm(1-alpha/2))+pnorm(-z-qnorm(1-alpha/2)))
```

L'impressione è che nella discussione gli autori si siano arrampicati sugli specchi per farsi accettare il lavoro dai reviewers, i quali potrebbero avere eccepito l'entità veramente modesta del "trattamento", la numerosità del campione molto limitata per uno studio di questo genere e la conseguente scarsa attendibilità delle analisi statistiche. Una certa disinvoltura emerge anche dal fatto che per i dettagli relativi all'applicazione del PSI nel contesto specifico, gli autori citano un articolo in turco (Serin et al, 2010). Non il massimo dell'apertura per un articolo scientifico.

In conclusione, ammesso e non concesso che la valutazione proposta si possa considerare quantitativa, ci saremmo stupiti nel constatare un significativo miglioramento di qualsivoglia tipo di capacità di *problem solving* mediante un "trattamento" di cinque incontri doposcuola di un'ora ciascuno distanziati di una settimana, valutato per di più su di un campione così limitato.

2.3 Confronto tra linguaggi testuali e visuali (a blocchi)

2.3.1 Weintrop e Wilensky (2015) - Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs

Weintrop e Wilensky (2015, 2015 A) hanno scritto una coppia di articoli, sulla base della stessa sperimentazione e dello stesso *dataset*, uno orientato alla valutazione oggettiva delle differenze di comprensione del codice fra modalità a blocchi e testuale, l'altro alla percezione degli studenti delle due diverse modalità.
Il primo è un bel lavoro caratterizzato da 1) una robusta base di riferimenti teorici e non solo prevalentemente sperimentali – non usuale per questo genere di studi – con 76 voci di letteratura, una notevole consistenza statistica per uno studio controllato (90 studenti di scuola secondaria superiore in un corso di 10 settimane), una metodologia nuova e articolata. Il background teorico si articola intorno alla questione definita con "rappresentazioni e apprendimento". In un *excursus* sostenuto da una ventina di riferimenti, che prende le mosse dal concetto di mediazione strumentale e simbolica delle attività mentali di Vygotsky passando poi da Papert e diSessa, gli autori mettono in evidenza il consenso ormai raggiunto su come le rappresentazioni simboliche influenzino in maniera profonda le modalità del pensiero. Di particolare interesse il riferimento allo studio approfondito di B.L. Sherin (2000) sul confronto fra *algebra-physics* e *programming-physics* - che commentiamo a parte successivamente – perché, pur sulla base di aspetti molto specifici relativi all'insegnamento della fisica, propone una visione concreta e profonda, utile a circostanziare meglio l'abusata e nebulosa nozione di *computational thinking*. Tale *background* teorico serve agli autori per porsi la domanda se i nuovi sistemi grafici a blocchi di rappresentazione dei linguaggi comporti qualche significativa differenza rispetto a quelli testuali. Gli autori si pongono due domande:

1. Come fare per confrontare la comprensione degli studenti nelle due diverse modalità, a blocchi e testuale?
2. La modalità influenza significativamente la comprensione dei concetti basilari della programmazione? E se sì, come cambiano le cose con il tipo di concetti?

Alla prima domanda rispondono descrivendo e utilizzando un metodo di indagine specifico, denominato *commutative assessment*, con il quale agli studenti vengono proposti dei quesiti su algoritmi e costrutti nelle due modalità, in fasi alternate, quesiti che sono formulati in modo da enucleare i concetti fondamentali, in maniera quanto più indipendente possibile da aspetti legati alle specificità dei linguaggi e della loro rappresentazione. I linguaggi messi a confronto sono Snap!, Javascript e Java. I risultati che ottengono depongono a favore di un significativo vantaggio della modalità a blocchi per quanto concerne la comprensione di costrutti software specifici.

Curiosamente invece non sono emerse differenze per quanto riguarda la comprensione generale di cosa faccia un algoritmo. Gli autori riconoscono la necessità di progettare indagini specifiche per comprendere questo tipo di risultato.

2.3.2 Weintrop e Wilensky (2015 A) - To block or not to block, that is the question: students' perceptions of blocks-based programming

Weintrop e Wilensky in questo secondo articolo, utilizzando il medesimo *dataset*, cercando di mettere in luce il punto di vista degli studenti e ponendosi le seguenti domande:

1. Gli studenti pensano che la programmazione a blocchi sia più facile di quella testuale e, in caso affermativo, perché?
2. Quali sono le maggiori differenze messe in luce dagli studenti?
3. Quali sono invece gli svantaggi percepiti nella programmazione a blocchi?

I risultati mostrano che in generale viene preferita la programmazione grafica a quella testuale: per l'abitudine già acquisita alla manipolazione grafica degli oggetti e per la facilità con cui si possono cercare i comandi fra i vari blocchi disponibili. Tuttavia alcuni studenti mettono in evidenza anche alcuni svantaggi molto precisi:

- La sensazione che la programmazione a blocchi sia meno potente – *With text based programming you can do a lot more – Blocks are limiting, like you can't do everything you can with Java, I guess*.
- La maggiore lentezza e la maggiore complicazione nel comporre il codice con i blocchi quando gli obiettivi si fanno più ambiziosi.
- La sensazione che la programmazione a blocchi sia meno “vera”. Spesso ai giovani piace fare le cose “per davvero”.

Gli autori enfatizzano l'importanza di questo dato e auspicano che se ne tenga conto sia per quanto riguarda la progettazione di ambienti didattici per la programmazione che per le pratiche di insegnamento in classe.

2.3.3 Lewis (2010) - How programming environment shapes perception, learning and goals: Logo vs. Scratch

L'autrice, Colleen Lewis, per inciso anche docente in un noto MOOC su Scratch – Programming in Scratch – nella piattaforma edX (https://www.edx.org/course/programming-scratch-harveymuddx-cs002x-1), pone direttamente a confronto Scratch con il suo illustre predecessore, Logo. In questo studio controllato vengono confrontati due gruppi di studenti di 10-12 anni che partecipano ad un corso estivo di programmazione di 36 ore in 12 giorni. Un gruppo viene introdotto alla programmazione con Logo e l'altro con Scratch (successivamente alla raccolta dei dati i ruoli vengono poi invertiti). L'autrice mette a prova le seguenti ipotesi sui vantaggi di Scratch:

1. Maggiore facilità nella programmazione.
2. Maggiore confidenza nelle competenze acquisite e maggiore propensione a continuare lo studio della programmazione.
3. Maggiore capacità di comprensione di costrutti specifici quali i cicli e le istruzioni condizionali.

Il risultato imprevisto, per il quale questo lavoro è ampiamente citato, consiste nel fatto che il livello

di confidenza nelle proprie capacità di programmazione è risultato significativamente superiore negli studenti che avevano seguito il percorso con Logo.

2.3.4 Price e Barnes (2015) - Comparing textual and block interfaces in a novice programming environment

Price e Barnes hanno confrontato esplicitamente due ambienti di programmazione, uno visuale e uno testuale in due classi di 6° e 7° grado (11-13 anni), rispettivamente di 17 e 14 alunni. Alla prima classe è stato affidato l'ambiente visuale, alla seconda quello testuale. Hanno utilizzato l'ambiente "Tiled Grace" che offre sia un'interfaccia a blocchi che una testuale, consentendo di passare instantaneamente da una all'altra. In questo modo gli autori hanno inteso ridurre al minimo le differenze di ambiente di lavoro al di fuori della contrapposizione blocchi-testo. I quesiti dello studio sono:

1. L'interfaccia a blocchi influisce sull'attitudine degli studenti verso le attività computazionali?
2. Influisce sulla difficoltà percepita nella programmazione?
3. E influisce sulle capacità di programmazione?

Gli autori concludono che sì, la programmazione a blocchi non sembra influenzare attidudine e difficoltà percepite ma ha influito positivamente sulla quantità di problemi risolti e il tempo impiegato a risolverli.

2.3.5 Homer e Noble (2014) - Combining tiled and textual views of code

Quello di Homer e Noble non è uno studio controllato ma lo poniamo qui perché descrive lo strumento utilizzato nel precedente articolo di Price e Barnes. Gli autori propongono un ambiente di programmazione dotato di duplice interfaccia, testuale e visuale. Può essere utilizzato direttamente su http://homepages.ecs.vuw.ac.nz/~mwh/minigrace/tiled, oppure scaricato a partire dalla home di uno degli autori (http://homepages.ecs.vuw.ac.nz/~mwh/) grazie alla licenza GPL v3.0 con cui viene distribuito. È interessante ma ancora più interessanti sono le motivazioni degli autori, che giustificano la citazione in questa rassegna:

1. Succede che i giovani percepiscano i sistemi a blocchi come "non veri" (vedi anche Weintrop e Wilensky, 2015, ma gli autori citano anche Lewis (2010) e Lewis et al (2014), mentre notoriamente preferiscono agire nella "realtà" piuttosto che in contesti che percepiscono "da bambini".
2. Gli autori citano evidenze secondo le quali la transizione precoce da un linguaggio ad un altro può disorientare e rallentare la progressione dell'apprendimento. Si sostiene che questo sia un problema frequente perché spesso, quando i codici si fanno più complessi, gli insegnanti tendono a spostarsi sui linguaggi usuali quali, Python, Java, C++. Gli autori sostengono che con il loro sistema la transizione sia facilitata consentendo di saltare fra le due versioni di un codice avanti e indietro con immediatezza.
3. Infine, Homer e Noble sembrano aderire più strettamente al principio papertiano del low floor and high ceiling: linguaggi come Scratch possono risultare frustranti quando le ambizioni si fanno più sostanziose mentre la soluzione proposta con Tiled Grace è pensata per consentire l'accesso a funzionalità più complesse.

2.3.6 Lewis et al (2014) - Children's perceptions of what counts as a programming language

Questo di Lewis et al rappresenta il desiderio di comprendere la percezione dei giovani (12-13 anni) del fatto che un ambiente visuale sia una forma di vera programmazione o no. L'indagine prevede prove che inducono gli studenti a estrapolare delle semplici istruzioni nei linguaggi C++, Java e Python sulla base di ciò che hanno imparato in una settimana di esposizione a Scratch.
I risultati ottenuti dagli autori sono tuttavia ambigui, con una lieve propensione a ritenere che no, Scratch e simili non vengano tanto visti come una forma di programmazione del computer bensì come un particolare tipo di gioco.

3. Studi empirici

3.1 Giannakos e Jaccheri (2013) - What Motivates Children to Become Creators of Digital Enriched Artifacts?

L'indagine di Giannakos e Jaccheri concerne una sperimentazione dove 66 bambini di 12 anni di varie scuole hanno partecipato a dei laboratori presso il centro ReMida di Reggio Emilia, attrezzato con vari strumenti e ausili didattici. Nei laboratori sono stati impiegati Scratch, Scratch for Arduino (S4A), Arduino e materiali riciclati. Si tratta di uno studio empirico sul ruolo della creatività nella formazione. Le motivazioni teoriche prendono le mosse dalla visione di Papert in materia. Lo strumento di indagine consiste nell'elaborazione statistica degli esiti di interviste. L'articolo non è molto nitido sulla descrizione delle analisi, o comunque forse va studiato con ulteriore attenzione. Gli autori concludono che le attività laboratoriali basate su software e hardware proposti hannno un impatto positivo riguardo alla concezione della tecnologia, stimolano il desiderio di esplorazione e migliorano la collaborazione. Si riporta inoltre che la facilità delle attività e la loro utilità percepita migliora il desiderio di partecipazione. Gli autori ammettono che lo studio statistico possa essere indebolito dalla limitatezza del campione.

4. Pensiero computazionale

4.1 Sherin (2001) - A comparison of programming languages and algebraic notation as expressive languages for physics

Questo non si presenta esplicitamente come un lavoro sul pensiero computazionale ma in realtà dice molto a riguardo. Un articolo di grande interesse, sia sperimentale che teorico. Sessanta pagine, 57 voci bibliografiche, spessore tecnico e pedagogico. Lo studio è indirizzato all'insegnamento della fisica. In tale contesto è naturale occuparsi di un tema del genere. Da un mezzo secolo a questa parte settori sempre più ampi della scienza hanno visto la luce grazie a nuove tecniche computazionali: “Applied computer science is now playing the role which mathematics did from the seventeenth through the twentieth centuries: providing an orderly, formal framework and exploratory apparatus for other sciences” (Djorgovski, 2005). A partire dal Seicento fino alla prima metà del Novecento l’analisi matematica ha fornito il sistema di rappresentazione fondamentale di tutta la scienza, sviluppandosi di pari passo e in modo dialettico con le varie e rapidamente crescenti esigenze delle scienze classiche, fisica e chimica in primis. Poi, già nel primo Novecento, la scienza si è dovuta occupare di sistemi sempre più complessi, difficilmente descrivibili con il linguaggio dell’analisi matematica. All’inizio i sistemi complessi e i fenomeni che presentavano caratteristiche intrattabili, se non caotiche, venivano considerati “eccezioni intrattabili”. Tuttavia presto si è dovuto riconoscere che invece tali sistemi rappresentavano la norma mentre erano proprio i casi che potevano essere affrontati con il linguaggio matematico

classico ad essere le eccezioni. È in questo stato di cose che sono apparsi i computer digitali, che con un processo dirompente, dall'essere immaginati da menti visionarie (Turing, von Neumann) negli anni 40-50 sono diventati i protagonisti del mondo scientifico e commerciale già a partire dagli anni 60. Da lì il ruolo del digital computing è letteralmente esploso, generando campi di ricerca del tutto nuovi in tutte le scienze di base. Ma il ruolo del "pensiero computazionale" che si è sviluppato intorno al digital computing non deve essere visto in concorrenza alla matematica classica, bensì deve essere visto come un ulteriore e possente arricchimento dell'armamentario matematico a disposizione dei ricercatori. Amplissime branche della scienza odierna si sostengono oggi esclusivamente sul digital computing. Esemplari sono tante storie della mia generazione. Io durante i quattro anni di studio della fisica, fra il 1974 e il 1978, non avevo mai sentito parlare di niente che avesse a che vedere con il mondo digitale. Se fosse stato per i 18 insegnamenti universitari non avrei avuto idea di cosa ci fosse dentro a un computer e non avrei avuto idea di cosa fosse un bit. Ma già lavorando alla tesi entrai in contatto con queste nuove, e assai costose a quei tempi, macchine digitali. Non solo, per arrivare in fondo al progetto di tesi, mi ritrovai a studiare e applicare un metodo dei calcolo di parametri fisici che faceva ricorso alla possibilità di simulare fenomeni casuali con i computer digitali, il cosiddetto metodo Monte Carlo.

Sherin nel suo articolo affianca – si badi bene, non contrappone – la *programming-physics* all'*algebra-physics*. Nelle note precedenti ho parlato di analisi matematica perché questo è lo strumento più potente e generale; Sherin, occupandosi di fisica a livello di scuola secondaria superiore, può fare riferimento più semplicemente alla rappresentazione algebrica della fisica. L'impianto teorico del suo lavoro si
fonda sul ruolo giocato dalle rappresentazioni strumentali e simboliche che supportano la conoscenza nella formazione stessa di tale conoscenza – concetti a cui abbiamo accennato commentando il lavoro di Weintrop e Wilensky (2015). L'autore documenta i propri argomenti con alcuni esercizi di fisica presentati sia nella forma algebrica convenzionale che nella forma computazionale. Per quest'ultima riferisce di esperienze svolte con l'ambiente Boxer, sviluppato da Sherin stesso e di Sessa, sulla base di Logo. Con questo lavoro l'autore sostiene la tesi che sistemi di rappresentazione diversi influiscono in modo differente sui meccanismi del pensiero e possono indurre un diverso tipo di comprensione dei medesimi fenomeni. In capo a un'analisi minuziosa di esperimenti didattici, condotti su gruppi di studenti sia mediante l'algebra-physics che la programming-physics, giunge alla conclusione per cui con la conoscenza algebrica si tende a enfatizzare gli equilibri mentre con quella computazionale si è portati a comprendere meglio gli aspetti dinamici. È estremamente interessante la prospettiva nella quale Sherin pone questa conclusione. Non si tratta, dice, di giudicare l'effetto di un metodo o dell'altro secondo una singola metrica e di confrontarli sulla base di tale metrica – ovvero non si tratta di stabilire quale sia "meglio" - bensì di accettare, comprendere e utilizzare proficuamente il fatto che il nuovo paradigma offra una mutata visione della conoscenza degli stessi fenomeni e di come, in ultima analisi, la cosa più sensata da fare sia quella di affiancare questa nuova forma di conoscenza a quelle preesistenti. E non si può evitare di osservare che la nuova prospettiva computazionale – qui nel senso della programming-physics di Sherin, possa essere di grande giovamento per la comprensione dei fenomeni fisici. Infatti lo strumento matematico costituisce indubbiamente il fondamento imprescindibile delle scienze di base – il linguaggio che consente di porre domande alla natura, per dirla con Galileo - ma il processo con il quale un giovane giunge a creare senso compiuto a partire da un linguaggio formale è molto complesso e faticoso. Pochi studenti arrivano ad apprezzare il formalismo matematico come uno strumento utile per comprendere e esprimere pensieri sul mondo fisico o altro. Per la grande maggioranza i formalismi matematici rappresentano al più una quantità di regole da applicare a memoria negli specifici contesti creati dalla scuola: qual era la formula da usare qui...? A questo proposito è interessante ricordare un noto articolo scritto da Enrico Persico (Persico, 1956), maestro di Enrico Fermi, dove ci si domandava cosa non andasse

con quella studentessa che procedeva come una locomotiva quando sciorinava le equazioni di Maxwell alla lavagna ma che non sapeva dire perché, con quel certo valore di corrente, una lampadina si sarebbe fulminata – non a caso Sherin rileva esattamente lo stesso problema a pag. 43 del suo lavoro, e proprio a proposito delle equazioni di Maxwell. La questione della comprensione dei fenomeni attraverso il linguaggio matematico non è, e non da ora, semplice. È esattamente qui che il "nuovo" approccio computazionale, nel quale peraltro vengono declinati settori sempre più ampi della fisica e delle altre scienze, può venire in aiuto. Infatti, l'approccio computazionale induce ad analizzare e scomporre i fenomeni fisici nella dimensione temporale, enfatizzandone così la natura dinamica, spesso più accessibile all'intuizione. Non solo, l'analisi computazionale costringe ad utilizzare precisi valori numerici da assegnare ai parametri fisici coinvolti e questa è una pratica che induce più facilmente gli studenti a ricavare un senso da ciò che studiano.

4.2 Weintrop et al (2016) – Defining Computational Thinking for Mathematics and Science Classrooms

Il lavoro di Weintrop et al generalizza quello di Sherin. La formulazione del pensiero computazionale – perché di questo si trattava, in sostanza – di Sherin è estremamente interessante perché mostra come sia scaturito spontaneamente nel corso dell'evoluzione della disciplina: oggi la fisica è pensata e creata sia mediante gli strumenti matematici tradizionali che con quelli computazionali. Il fenomeno concerne anche la matematica, dove nel corso del '900 sono nate intere nuove branche che trovano la ragione della propria esistenza solamente per il fatto che si sono resi disponibili computer, reti di collegamento e sistemi di memorizzazione sempre più pervasivi e potenti. Ad esempio, a fianco dell'analisi matematica si è evoluta molto l'analisi numerica e la maggior parte degli sviluppi di questa sono funzionali alla risoluzione di problemi caratterizzati da grandissime quantità di dati numerici, che sono trattabili solo attraverso macchine di calcolo. È in queste discipline che il pensiero computazionale si è formato, già a partire dalla metà del secolo scorso. Ma quasi immediatamente tutti gli altri campi scientifici hanno visto la proliferazione di metodi numerici. E, successivamente, non solo scientifici, perché oggi l'impatto degli approcci computazionali investe settori come quello della linguistica e delle scienze sociali in generale, specialmente, in questo caso, attraverso il trattamento dei *big data*. In questa luce, è difficile pensare che la scuola, nel suo insieme, possa rimanere indifferente a tutto questo.
Weintrop et al affrontano il tema del pensiero computazionale nell'ambito delle discipline scientifiche alla larga – le cosiddette discipline STEM: Science, Technology, Engineering, Mathemathics. Gli autori, sulla base di una ricca messe di riferimenti bibliografici, documentano come le tecniche computazionali abbiano avuto da mezzo secolo a questa parte un ruolo crescente in tutti i campi della scienza. Successivamente descrivono un lavoro sistematico di ampio respiro con il quale hanno cercato di definire una tassonomia di riferimento del pensiero computazionale. Concludono con alcuni esempi di attività in classe esemplificative dell'applicazione di alcune delle categorie definite in tale tassonomia.
Quindi quello di Weintrop et al rappresenta uno di quei lavori con i quali si cerca, ancora oggi, di definire l'identità del pensiero computazionale, quando lo si voglia estendere al di là di quei contesti scientifici ben circoscritti nei quali è invece naturalmente e perfettamente definito. Un lavoro esteso: oltre 20 pagine sostenute da 147 voci bibliografiche con l'obiettivo di produrre una definizione del pensiero computazionale per la matematica e le scienze nella forma di una tassonomia composta da quattro tipi di pratiche: manipolazione di dati, realizzazione di modelli e simulazioni, soluzione di problemi computazionali, pensiero sistemico. Il lavoro si inquadra nella tendenza generale tesa a introdurre il pensiero computazionale nell'insegnamento di tutte le discipline scientifiche nella scuola secondaria.

Il processo con cui gli autori hanno costruito la loro tassonomia è articolato in quattro fasi. Nella

prima fase gli autori hanno condotto un'ampia ricognizione della letteratura in materia di pensiero computazionale in ambito scientifico, giungendo a definire un primo insieme di dieci abilità fondamentali di pensiero computazionale. Successivamente, da questo materiale hanno estratto una varietà di pratiche didattiche destinate all'introduzione del pensiero computazionale nell'insegnamento della matematica e delle scienze; in questa fase, due revisori indipendenti, hanno analizzato 208 aspetti diversi desunti da 34 diverse attività, selezionandone 45. La tassonomia così ottenuta è stata proposta a 16 insegnanti di matematica e scienze nell'ambito di un workshop estivo per progettare nuove attività nelle proprie classi. Sulla base dei feedback ricevuti da questi insegnanti, oltre quelli ottenuti da altri esperti di curricula nelle discipline STEM, la tassonomia è stata ulteriormente sintetizzata in 22 pratiche, suddivise nelle quattro summenzionate categorie. Nel corso dell'intero processo la messe di feedback sulla tassonomia che andava formandosi è stata ulteriormente arricchita mediante una serie di interviste a ricercatori dell'accademia, dell'industria e a studenti nell'ambito STEM.

Gli autori passano quindi a discutere in dettaglio le caratteristiche delle pratiche emerse dal punto di vista dei vari portatori di interesse – studenti, insegnanti, progettisti di curricula, amministratori – chiudendo con l'esposizione di tre di tali pratiche situate in contesti reali. Nel corso di tale sezione finale emergono alcuni concetti interessanti. La pratica di lavorare con modelli computazionali ha un valore didattico rilevante perché, in ultima analisi, la scienza non spiega nulla e a fatica interpreta i fenomeni, più che altro formula modelli (von Neumann, 1955, p. 628). Oggi la computer science applicata gioca lo stesso ruolo che era della matematica fra il VII e XX secolo, fornendo il contesto formale ordinato e lo strumentario per l'esplorazione necessari alle altre scienze (Djorgovski, 2005). La ricerca ha ormai appurato che mediante pratiche di problem solving computazionale, sviluppo di algoritmi e astrazioni computazionali gli studenti possono sviluppare una conoscenza profonda dei fenomeni e dei fatti matematici. Pur non dovendosi ovviamente aspettare che tutti gli studenti diventino esperti di programmazione, una preparazione base di programmazione è una componente importante nell'indagine scientifica del XXI secolo. La capacità di pensare in modo sistemico è un atteggiamento mentale importante non solo per coloro che si dedicheranno a professioni tecnico-scientifiche ma anche per formare la cultura scientifica (ancora largamente assente) di qualsiasi cittadino. I concetti caratterizzanti il pensiero sistemico, come retroazione, emergenza, stock e flussi, sono trasversali e li possiamo ritrovare in campi assai diversi come la fisica, l'economia e la storia.

4.3 Kalelioğlu et al (2016) - A Framework for Computational Thinking Based on a Systematic Research Review

Anche questo articolo è espressione di quel pensiero computazionale che insegue la propria identità, ma in maniera meno mirata rispetto al lavoro di Weintrop et al, estendendo l'indagine al di là delle discipline STEM.
Si tratta di una review della letteratura sul tema del pensiero computazionale aggiornata al 2015. I tre ricercatori autori dell'articolo, F. Kalelioğlu, Y. Gülbahar e V. Kukul, si sono suddivisi il compito di analizzare qualitativamente i 125 lavori, scremati in base alla pertinenza al tema del pensiero computazionale a partire da un totale di 274 reperiti nelle librerie digitali Ebscohost, ScienceDirect, Web of Science, Springer, IEEE Digital Library e ACM Digital Library. La maggior parte di tali contributi sono dedicati a attività che promuovano il pensiero computazionale nei curricula, il contesto prevalente è quello della scuola primaria e secondaria. Game-based learning e costruttivismo sono i riferimenti teorici principali.
L'obiettivo degli autori è quello di definire un "framework" di riferimento che faciliti l'innesto di pratiche relative al pensiero computazionale nei curricula. Qui di seguito riporto il framework risultante.

Identificazione del problema	Raccolta, rappresentazione e analisi dei dati	Generazione, selezione e pianificazione delle soluzioni	Implementazione delle soluzioni	Verifica delle delle soluzioni e aggiornamenti successivi
Astrazione	Raccolta	Ragionamento matematico	Automazione	Testing
Decomposizione	Analisi	Costruzione di algoritmi e procedure	Simulazioni e creazione di modelli	Debugging
	Rilevazione di schemi e modelli	Parallelizzazione		Generalizzazione
	Concettualizzazione			
	Rappresentazione			

Confrontando questo framework con la tassonomia di Weintrop et al, seppur riferita alle sole discipline STEM, si vede come vi sia una certa convergenza ma vi siano anche delle differenze, a testimonianza della fluidità del concetto. Kalelioğlu et al rilevano come la letteratura sull'argomento non sia ancora matura, anche per la necessità di estendere il concetto di pensiero computazionale ad altri campi, al di là delle discipline STEM. È una letteratura che risale a non più di dieci anni fa, ammontando, dicono gli autori, a circa 500 contributi – un valore molto piccolo rispetto a settori più consolidati. Di conseguenza manca sia di supporti teorici adeguati che di sostanziali riscontri sperimentali.

Sia il lavoro di Kalelioğlu che quello di Weintrop aiutano a chiarire molti aspetti del pensiero computazionale ma non consentono di delimitarne esattamente il dominio nell'ambito dell'istruzione, questione che rimane ancora sostanzialmente aperta.

4.4 Bocconi et al (2016) - Developing computational thinking in compulsory education

Quello di Bocconi et al non è un articolo scientifico ma un documento programmatico della Commissione Europea dedicato allo sviluppo del pensiero computazionale nella formazione obbligatoria. Lo includo perché l'ho visto citare in vari gruppi di discussione. È un buon documento per avere un'idea dello stato dell'arte, che gli autori hanno scritto a partire dallo studio delle azioni in corso presso i ministeri dell'istruzione di tutto il mondo, della letteratura scientifica e da una serie di interviste a esperti del settore.
Leggendo la descrizione degli autori del concetto di pensiero computazionale si percepisce come oggi il mondo corra ad una velocità eccessiva rispetto alla nostra capacità di metabolizzare il nuovo che incalza. Il caso del pensiero computazionale è emblematico. Mentre da un lato i Ministeri dell'Istruzione di tutti i paesi, e la medesima Unione Europea, sentono la cogenza della questione sino al punto di intraprendere azioni concrete, dall'altro, ancora oggi, non esiste un consenso su cosa si debba intendere di fatto per pensiero computazionale. Per la loro analisi gli autori fanno riferimento a cinque review di autori di riferimento: Barr e Stephenson (2011), Lee et al (2011), Grover e Pea (2013), Selby e Woollard (2013), Angeli et al (2016). Sulla base di queste e delle ultime proposte di Jeannette Wing, colei che per prima ha introdotto, nel 2001, questo termine, propongono la seguente definizione: il pensiero computazionale descrive i processi del pensiero connessi con la formulazione di un problema in maniera tale da poter individuare una soluzione

computazionale che richieda astrazione, pensiero algoritmico, automazione, decomposizione, debugging e generalizzazione. Questi ultimi concetti sono gli aspetti che secondo la maggior parte degli studiosi caratterizzano l'idea di pensiero computazionale. Anche qui vediamo una discreta ma non perfetta sovrapposizione con le definizioni proposte nei due articoli precedenti.

In particolare, intersecando i cinque articoli citati da Bocconi et al vediamo, che tutti includono il concetto di astrazione e 4 su 5 includono pensiero algoritmico e decomposizione. Fra i vari temi dibattuti emergono: relazione con l'alfabetizzazione digitale, ruolo del coding, modalità di inclusione del pensiero computazionale nei curricula, pratiche di insegnamento, valutazione degli studenti, preparazione degli insegnanti e inclusione di formazione non formale.

Gli autori rilevano come nella letteratura l'alfabetizzazione digitale venga prevalentemente discussa in chiave critica e come l'interesse verso il pensiero computazionale derivi proprio dall'insufficienza di un insegnamento che si è troppo concentrato sull'impiego superficiale di strumenti specifici. Con l'introduzione del pensiero computazionale si intende condurre i giovani verso la conoscenza della scienza e delle idee che sottendono le nuove tecnologie.

Per quanto riguarda il coding, si rivela come questo venga spesso erroneamente identificato con il pensiero computazionale. Invece quest'ultimo è un concetto molto più generale. Con il coding si intende l'attività di comporre una serie di istruzioni in un linguaggio di programmazione per eseguire determinate operazioni mediante un computer, localmente o attraverso un apposito servizio Web. Il coding è ovviamente un'attività necessaria per creare un software ma non è sufficiente. La creazione di un software comporta attività di analisi, progettazione e infine implementazione. Quest'ultima comprende effettivamente il coding ma anche il testing e il debugging. In realtà per molti autori il pensiero computazionale è un concetto più generale, che si estende al di là del dominio specifico della computer science. Ma anche questa visione allargata è ancora lungi dall'essere condivisa. Su questo tema cito ancora il lavoro di Sherin (2001) e rimando alle mie precedenti considerazioni, in particolare all'opportunità di non confrontare diversi paradigmi sulla base di una qualche metrica unidimensionale. Si tratta di porre la questione in una prospettiva inclusiva e non di ricercare una dicotomia.

Riguardo all'inclusione nei curricula scolastici si distinguono due orizzonti. Uno più ampio che prevede di attivare nuove modalità di pensiero negli scolari, sviluppare la capacità di esprimersi con una varietà di media, risolvere problemi in contesti reali. L'altro invece è più orientato a concepire il pensiero computazionale come un mezzo per favorire la crescita economica e potenziare l'occupazione nel settore dell'Information and Computing Technology. L'impegno da parte delle istituzioni è sostenuto nella maggior parte dei Paesi. I Paesi che hanno intrapreso o stanno intraprendendo azioni concrete per lo sviluppo del pensiero logico in questo contesto sono Austria, Svizzera, Cecoslovacchia, Danimarca, Finlandia, Francia, Gran Bretagna, Ungheria, Italia, Lituania, Polonia, Portogallo, Turchia. Sette di questi enfatizzano lo sviluppo del coding: Finlandia, Francia, Lituania, polonia, Svizzera e Turchia. L'attenzione dei Ministeri per l'Istruzione nei vari Paesi si focalizza, in generale, sulla promozione dei seguenti aspetti:

1) capacità di pensiero logico
2) capacità di problem solving
3) percorsi verso la computer science
4) coding
5) occupazione nel settore dell'Information and Computing Technology.

Per quanto concerne gli aspetti pedagogici, gli autori evidenziano i metodi tipo "computer science

unplugged" (attività di natura computazionale effettuate senza computer o simili), le simulazioni computerizzate di fenomeni fisici (complesse per le conoscenze matematiche avanzate che quasi sempre richiedono), modelli computerizzati (affini alle simulazioni ma molto più semplici, e quindi fattibili - vedi Weintrop et al, 2016, per questioni inerenti a simulazioni e modelli) e le questioni di inclusività, ivi comprese le questioni di genere - vedi le svariate iniziative e i movimenti per favorire il coinvolgimento femminile nel campo della computer science.

Se il concetto di pensiero computazionale non è ancora universalmente definito altrettanto si può dire per la questione della valutazione. Fra le pratiche citate troviamo: portfolio degli studenti, discussione di eventuali artefatti prodotti dagli studenti - grafici, multimediali, software - e prove su scenari dati - ad esempio progetti software da studiare, descrivere, estendere, correggere e remixare. A questo proposito, potremmo citare l'esperienza del nostro Laboratorio di Tecnologie Didattiche presso il CdS in Scienze della Formazione Primaria. Qui la valutazione si basa su un diario delle attività effettuate durante l'insegnamento, la verifica di almeno un codice prodotto autonomamente con LibreLogo (il logo inserito da ciascun studente nel diario), una discussione di tali artefatti all'esame orale, con eventuale prova di coding al computer.

Gli autori rilevano inoltre come probabilmente il problema maggiore in questa fase storica sia costituito dalla formazione degli insegnanti, che nella grande maggioranza dei casi non ha mai ricevuto una formazione intorno a questi temi nel corso dei propri studi. In tutti i Paesi si stanno cercando soluzioni ma le difficoltà sono rilevanti. Non è semplice formare masse così ampie di persone e in così breve tempo, senza nemmeno avere del tutto chiari i confini delle nozioni e delle competenze occorrenti e le modalità di intervento. Da alcune esperienze in corso pare che gli approcci più efficaci siano quelli dove gli eventi didattici vis-à-vis sono accompagnati e sostenuti da una comunità di discussione online.

Infine, in questo contesto, non si può non menzionare la disponibilità delle risorse di studio disponibili in Internet, di tipo informale. Ad esempio usare Scratch implica già l'uscita dalla classe, per certi versi, perché si tratta di un ambiente con una duplice valenza, tecnica e social. Ma sono veramente molte e diversificate le risorse disponibili in rete, quali code.org o csunplugged.org, giusto per fare due esempi fra i più noti. Da notare che csunplugged.org non è una novità in quanto risale agli anni '90.

In conclusione, secondo gli autori il mondo della formazione si trova in una fase di rapidi e inevitabili cambiamenti. Numerose iniziative intorno ai temi del pensiero computazionale stanno emergendo in tutto il mondo, sia istituzionali che spontanee. In tale contesto pare scontato che il pensiero computazionale rientri a pieno titolo nei curricula dell'insegnamento formale. Si ritiene di fondamentale importanza che vengano presi provvedimenti adeguati per l'aggiornamento professionale degli insegnanti intorno a quella che chiamano *Computational Thinking Pedagog*y.

4.5 S. G. Djorgovski (2005) - Virtual Astronomy, Information Technology, and the New Scientific Methodology

Includo questo lavoro in risposta, anche qui, a tante discussioni dove vedo che si continua a ridurre il mondo in categorie disgiunte: l’informatica, la fisica, la psicologia e via dicendo fino alla tanto ubiquitaria quanto trita divisione fra umanisti e scientifici. Questo è un articolo di astrofisica ma spiega bene cosa stia succedendo alla scienza, e di conseguenza al pensiero umano. Tutte le scienze, astronomia inclusa, si trovano immerse nell’era dell’abbondanza delle informazioni. Il volume e la complessità dei dati stanno crescendo in maniera esponenziale; le pratiche scientifiche verranno trasformate ma si pongono anche nuove sfide tecnologiche, comuni a tutti gli ambiti. Il concetto di

Osservatorio Virtuale discusso da Djorgovski rappresenta la risposta della comunità degli astronomi a queste sfide, con la quale da un lato si sfruttano i progressi della *information technology*, dall'altro si fornisce un laboratorio importante per lo sviluppo dell'*information technology* e per la *computer science* applicata. I metodi e le tecniche utilizzati per organizzare le mostruose quantità di dati che oggi vengono raccolti quotidianamente in tutti i campi scientifici (scienze sociali incluse) in repositori condivisi e accessibili via web, i sistemi per la scoperta di schemi, regolarità, informazioni utili (*data mining*) stanno dando vita a un nuovo campo composito al quale devono necessariamente concorrere gli specialisti delle varie discipline – qui gli astrofisici, là i chimici, i biologi, i sociologi etc. – con quelli della computer science, della information technology, con gli statistici, i matematici. Ma non si tratta di cooperazioni tradizionali, per così dire, dove ognuno arriva con la sua scatola degli attrezzi, non basta questo. Sta emergendo una potente sinergia fra le nuove scienze, tutte ormai largamente computazionali, e la nuova computer science, sempre più indirizzata dalle sfide poste nelle diverse arene scientifiche. Una sinergia che genera culture ibride e nuovi modi di pensare, perché questo è ciò che è richiesto oggi per affrontare qualsiasi problema scientifico, ma anche organizzativo, economico, sociologico. Tutte le categorizzazioni sono nefaste nel contesto attuale. Non si tratta di insegnare prima la matematica e poi il *coding*. Messa in questi termini la questione è del tutto sterile. Si tratta di insegnare la matematica includendovi i nuovi paradigmi computazionali, perché la matematica di oggi è fatta anche di questo, è fatta anche fondamentalmente di questo. E così per tante altre discipline, anche insospettate.

Le voci discusse nella presente bibliografia ragionata

Non tutti i file relativi agli articoli citati sono disponibili al pubblico. Chi desiderasse accedere ad uno di questi articoli me li può richiedere in via privata: arf@unifi.it.

S. Bocconi, A. Chioccariello, G. Dettori, A. Ferrari, K. Engelhardt (2016), *Developing computational thinking in compulsory education* – ED. *P. Kampylis e Y. Punie*
Science for Policy report by the Joint Research Centre (JRC), the European Commission's science and knowledge service.

S. G. Djorgovski (2005), *Virtual Astronomy, Information Technology, and the New Scientific Methodology, Computer Architecture for Machine Perception*, In: Di Gesu V, Tegolo D (eds) IEEE Proc. of CAMP05: Computer Architectures for Machine Perception, p. 125-132.

M.N. Giannakos e L. Jaccheri (2013), *What Motivates Children to Become Creators of Digital Enriched Artifacts?*
Proceeding C&C '13 Proceedings of the 9th ACM Conference on Creativity & Cognition Pages 104-113.

F. Hermans, E. Aivaloglou (2016), *How Kids Code and How We Know: An Exploratory Study on the Scratch Repository*
Proceeding ICER '16 Proceedings of the 2016 ACM Conference on International Computing Education Research Pages 53-61
(Il frontespizio del PDF è errato: si riferisce ad un altro articolo degli stessi autori)

F. Hermans e E. Aivaloglou (2016), *Do code smells hamper novice programming*
IEEE 24th International Conference on Program Comprehension (ICPC)

M. Homer e J. Noble (2014), *Combining tiled and textual views of code*
Proceeding VISSOFT '14 Proceedings of the 2014 Second IEEE Working Conference on Software Visualization, 1-10.

J.P. Ioannidis (2005), *Why most published research findings are false*
PLoS Med. 2: 696-701.

F. Kalelioğlu, Y. Gülbahar (2014), *The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective*
Informatics in Education, 13: 33-50.

F. Kalelioğlu, Y. Gülbahar, V. Kukul (2016), *A Framework for Computational Thinking Based on a Systematic Research Review*
Baltic J. Modern Computing, Vol. 4, No. 3, 583-596.

C.M. Lewis (2010), *How programming environment shapes perception, learning and goals: Logo vs. Scratch*
Proceeding SIGCSE '10 Proceedings of the 41st ACM technical symposium on Computer science education, Pages 346-350.

C. Lewis, S. Esper, V. Bhattacharyya, N. Fa-Kaji, N. Dominguez, and

A. Schlesinger (2014), *Children's perceptions of what counts as a programming language*
J. Comput. Sci. Coll., 29(4): 123-133.

J.N. Matias, S. Dasgupta, B.M. Hill (2016), *Skill Progression in Scratch Revisited*
Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, 1486-1490.

T.W. Price, T, Barnes (2015), *Comparing textual and block interfaces in a novice programming environment*
Proceeding ICER '15 Proceedings of the eleventh annual International Conference on International Computing Education Research , 91-99.

C. Scaffidi, C. Chambers (2016), *Skill progression demonstrated by users in the Scratch animation environment*
Proceedings of the Conference on Computer Human Interaction (CHI), 1-39.

B.L. Sherin (2001), *A comparison of programming languages and algebraic notation as expressive languages for physics*
Int. Journal of Computers for Mathematical Learning, 6: 1-61.

D. Weintrop, U. Wilensky (2015), *Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs*
Proceeding ICER '15 Proceedings of the eleventh annual International Conference on International Computing Education Research, 101-110.

D. Weintrop, U. Wilensky (2015 A), *To block or not to block, that is the question: students' perceptions of blocks-based programming*
Proceeding IDC '15 Proceedings of the 14th International Conference on Interaction Design and Children, 199-208.

D. Weintrop, E. Beheshti, M. Horn, K. Orton, K. Jona, L. Trouille, U. Wilensky (2016) *Defining Computational Thinking for Mathematics and Science Classrooms*, J Sci Educ Technol, 25:127–147

Riferimenti bibliografici ancora non discussi o comunque utili

Non tutti i file relativi agli articoli citati sono disponibili al pubblico. Chi desiderasse accedere ad uno di questi articoli me li può richiedere in via privata: arf@unifi.it.

C. Angeli, J. Voogt, A. Fluck, M. Webb, M. Cox, J. Malyn-Smith, & J. Zagami. (2016). A K-6 Computational Thinking Curriculum Framework- Implications for Teacher Knowledge. Educational Technology & Society, 19(3), 47–57.

M. Bakker, A. van Dijk, J.M. Wicherts (2012), *The rules of the game called psychological science*, Perspectives on Psychological Science 7: 543-554.

V. Barr, & C. Stephenson (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? ACM Inroads, 2(1), 48–54.

Chiandotto B. (1978), *Elementi di inferneza statistica*, in “Biometria e metodi” a cura di F. Salvi e B. Chiandotto, Piccin.

Chow S., Shao J., Wang H. (2008). *Sample Size Calculations in Clinical Research*. 2nd Ed. Chapman & Hall/CRC Biostatistics Series.

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

P. Heppner e C.H. Petersen (1982). *The development and implications of a personale problem-solving inventory*, Journal of Counseling Psychology, 29: 66-85, 1982

I. Lee, F. Martin, J. Denner, B. Coulter, W. Allan, J. Erickson, L. Werner. (2011). Computational thinking for youth in practice. ACM Inroads, 2(1), 32–37.

M.R. Macleod, I. Roberts, U. Dirnagl, I. Chalmers, et al. (2014) Biomedical research: increasing value, reducing waste. Lancet 383: 101–104.

C. C. Selby, & J. Woollard (2013). Computational Thinking: The Developing Definition. University of Southampton (E-prints).