“Low floor high ceiling” Computer Science: Riflessioni su un curriculum per un primo corso d’informatica

Francesco Maiorana[1, 2, 3, 4]

[1]Univerità degli Studi di Catania – Dipartimento di Scienze della Formazione
[2]IISS G.B. Vaccarini, Via Orchidea, 9 Catania
[3] Scientix Ambassador
[4] CoderDojo Champion
fmaioran@gmail.com

Abstract. Guardando il panorama mondiale nel campo educativo è evidente una forte tensione: da un lato una spinta etica e democratica vorrebbe rendere l’educazione inclusiva ed accessibile a tutti lungo tutto l’arco della vita; dall’altro la rapida evoluzione tecnologica richiede ai futuri lavoratori conoscenze, abilità e competenze sempre più elevate. Questo lavoro vuole suscitare, a partire da una descrizione di contenuti d’informatica, di approcci pedagogici e di tecnologie didattiche, suscitare una riflessione tra gli educatori formali, informali e non formali, sull’insegnamento interdisciplinare e transdisciplinare di un primo corso d’informatica. Pur auspicando che l’insegnamento della disciplina cominci fin dalla scuola primaria, le riflessioni si adattano a chiunque, sia studenti sia docenti, si confronti con la disciplina in uno studio e ricerca lungo tutto l’arco della vita.

Keywords: Curriculum, Pedagogie, Tecnologie Informatiche, Pensiero Computazionale, Problem solving, Didattica dell’Informatica, Risorse didattiche.

1 Introduzione

Una riflessione internazionale ha portato in primo piano l’importanza di un’educazione di qualità per tutti [1] [2], inclusiva ed accessibile [3] [4]. Nel contempo la forte spinta tecnologica ha portato ad una revisione dei curricula in Inghilterra, [5], Australia [6], Nuova Zelanda [7], Computer Science Teachers Association (CSTA) [8] e alla nascita di movimenti internazionali quali Code.org [9], CorderDojo [10], Europe Code Week [11], Programma il futuro [12], CodeWeek.it [13] e reti di docenti come Scientix [14] che ha prodotto una forte spinta verso l’introduzione dello studio dell’informatica fin dalla scuola primaria attraverso l’educazione sia formale sia informale.

Il confronto internazionale partito dal 2006 [15] ha prodotto un confronto serrato durato più di un decennio [16] che ha messo in luce l’importanza del pensiero computazionale come “state of mind”, attitudine e strumento concettuale utile in tutte le discipline. Nel contesto di un corso introduttivo di programmazione la ricerca ha una storia lunga più di 50 anni [17] [18].

Il lavoro presenta un'ipotesi di curriculum che si basa su progressioni d'apprendimento con un facile punto d'ingresso, inclusivo e accessibile a tutti, ma che, nel contempo, offre svariate possibilità di apprendimento rendendo le risorse di apprendimento "low floor and high ceiling" [18] descrivendo i contenuti, gli approcci pedagogici e gli strumenti tecnologici usati con lo scopo di favorire un confronto e una discussione sulle tematiche d'interesse. Nella sezione 2 vengono descritti i contesti e gli scenari di applicazione del curriculum, nella sezione 3 sono presentati i contenuti, l'approccio pedagogico e le tecnologie, nella sezione 4 sono approfondite le conoscenze, le abilità e le competenze che il curriculum aspira a potenziare, nella sezione 5 sono brevemente presentate possibili traiettorie d'apprendimento, ed infine nella sezione 6 sono presentate le conclusioni e descritti possibili sviluppi futuri.

2 Il contesto

Il curriculum è da intendersi come ausilio didattico per un primo corso d'informatica per:

1) gli studenti del primo anno del primo biennio delle scuole superiori italiane, dove è prevista una disciplina inerente o le tecnologie informatiche o le scienze informatiche.
2) gli studenti della scuola superiore dove non è prevista una disciplina inerente o le tecnologie informatiche o le scienze informatiche come ausilio didattico interdisciplinare
3) gli studenti universitari in discipline umanistiche o, più in generale, in discipline dove le tecnologie e le scienze informatiche non rivestono il ruolo predominante
4) studenti delle scuole medie inferiori
5) in generale come ausilio didattico in un primo corso d'informatica

Con il primo termine, tecnologie informatiche, intendiamo tutto quello che ruota attorno agli aspetti ingegneristici ed applicativi legati alla progettazione e realizzazione di dispositivi sia software sia hardware.
Con il termine scienze informatiche intendiamo tutto quello che ruota attorno agli aspetti scientifici legati alla progettazione di modelli logico-matematico sia per il software sia per l'hardware.

3 Contenuti, approcci pedagogici e tecnologie

Per raggiungere l'ambizioso obiettivo si fa ricorso ad un arsenale si strumenti software ed hardware che privilegiano lo sviluppo di abilità e competenze nella risoluzione di problemi e nello sviluppo del pensiero computazionale [15] [16] inteso come un insieme di abilità cognitive da utilizzare in tutte le attività quotidiane, durante tutto l'arco della vita e in contesti formali, non formali ed informali.
In particolare la scelta di usare:

1) attività unplugged
2) puzzle

3) sfide algoritmiche e logico matematiche tratte dalle Olimpiadi di Problem Solving [20] e dalle Olimpiadi d'Informatica [21]
4) attività plugged che richiedono lo sviluppo di software usando linguaggi visuali a blocchi

vuole essere un tentativo di sintesi della tensione a livello internazionale verso i due approcci: unplugged e coding.
Il curriculum si avvale di un arsenale di linguaggi visuali a blocchi tra i quali segnaliamo:

1) Scratch
2) App Inventor per la programmazione di dispositivi mobili
3) Scribble: per il disegno di figure complesse.
4) Cellular: per simulazioni si sistemi
5) Snap! per la sua maggiore astrazione e per consentire di svolgere applicazioni in tutti i maggiori paradigmi di programmazione: imperativo, funzionale, logico e ad oggetti
6) Pixley: per l'elaborazione d'immagini
7) Tunely: per l'elaborazione di suoni
8) NetsBlox; pre la programmazione di rete e l'uso di primitive di comunicazione come i messaggi e le procedure remote
9) bloP: versione visuale a blocchi del linguaggio C
10) Edgy per la gestione di strutture dati come i grafi

Nonché di strumenti per la progettazione di diagrammi di flusso come flowgorithm che consentono la traduzione in svariati linguaggi di programmazione testuali. Altri linguaggi visuali a blocchi sono presentati nei materiali online e nei test online usando un approccio pedagogico costruttivista [22] che attraverso delle domande guida lo studente verso l'esplorazione dell'interfaccia, l'individuazione delle caratteristiche peculiari e la risoluzione di problemi. Tra le pedagogie che spingono verso questo approccio ricordiamo: Inquiry Based Learning (IBL) [23], Peer Instruction [24-25], Process Oriented Guided Inquiry Learning (POGIL) [26-27], Challenge based Instruction [28], Flipped classroom [29]. Tra questi linguaggi ricordiamo, giusto per fare qualche esempio:

1) Alice: per lo storytelling digitale in tre dimensioni
2) BeetleBlocks: per la stampa 3D
3) Snap4Arduino: per interfacciarsi con il microcontrollore Arduino
4) Enchanting: per programmare un robot
5) Blocky: linguaggio visuale a blocchi con codifica immediata in un linguaggio testuale come ad esempio Javascript e PHP
6) Bubble: linguaggio di programmazione visuale per creare applicazioni web

Nei materiali online e nei test interattivi saranno forniti degli esempi per uno o più di questi linguaggi. I test online privilegiano l'ottica formativa piuttosto che quella valutativa. Usando l'approccio pedagogico sopra indicato, approccio alla base dei recenti progetti di ricerca in ambito educativo in Europa [30] e nel mondo [24] [27]

gli studenti saranno guidati sia verso la soluzione di problemi e la realizzazione di software usando, come strumenti, diversi linguaggi di programmazione.

La creazione di una comunità di pratica potrebbe arricchire notevolmente il testo attraverso la co-creazione sia di materiali online sia di test interattivi [31]. I contributi dei docenti che volessero partecipare renderanno possibile una collaborazione esplicita anche attraverso attività di ricerca educativa, come discusso nel capitolo 2.

La scelta di usare linguaggi visuali a blocchi è motivata dalla necessità dettata dal rapido progresso tecnologico e dall'enorme e in crescita esponenziale quantità di informazione disponibile, di potenziare le attività cognitive di ordine superiore ormai richieste costantemente dal mercato del lavoro. I linguaggi visuali a blocchi consentono di focalizzarsi sulla risoluzione dei problemi e sulle suddette attività cognitive di ordine superiore, relegando a contesti del tutto marginali attività di memorizzazione di dettagli poco significativi propri dei linguaggi testuali e delle sintassi spesso ostica e con notevoli differenze tra un linguaggio ed un altro.

L'arsenale di linguaggi usati è una dimostrazione lampante della tensione in questa direzione della comunità internazionale. La vastità degli strumenti presentati é resa possibile solo dal loro comune criterio di progettazione nato dalle teorie costruttiviste di Piaget che hanno trovato esemplificazione concreta ngli ambienti di programmazione quali Scratch, realizzato dal Massachusset Institute of Technology, e, successivamente, Build Your Own Block (BYOB) e Snap! entrambi realizzati da ricercatori dell' University of California, campus di Berkey). Tutti i linguaggi presentati rappresentano estensioni del linguaggio Scratch con il quale condividono la stessa interfaccia e lo stesso ambiente di programmazione consentendo agli studenti di lavorare sempre sostanzialmente con lo stesso ambiente di sviluppo. La maggiore astrazione e potenza espressiva dei vari linguaggi e la loro matrice comune consente agli studenti, pur mantenendo la familiarità dell'ambiente e dell'interfaccia, ma usando un approccio incrementale, di concentrarsi su attività cognitive a difficoltà crescente senza dover affrontare lo sforzo cognitivo di dover prima disimparare il vecchio linguaggio e poi imparare il nuovo.

La necessità di questo approccio pedagogico, come discusso nel secondo capitolo, trova riscontro in svariati framework educativi e curricula internazionali quali, ad esempio, (JRC, AICT, AP Computer Science Principles, CSTA, OCED, UNESCO) sforzo che ha portato ad una profonda revisione a livello mondiale dei curricula che rendono obbligatorio, anche attraverso atti legislativi, lo studio dell'informatica fin dalla scuola primaria, proseguendo per la scuola secondaria di primo e secondo grado e, in generale, lungo tutto l'arco della vita. Tra gli esempi in questo direzione citiamo i nuovi curricula della Gran Bretagna [5], dell'Australia [6] e della Nuova Zelanda [7].

Questo confronto con gli sforzi educativi in ambito mondiale è condiviso sia con gli studenti, attraverso i contributi delle attività Content Language Integrated Learning (CLIL), sia con i docenti, attraverso la pagine di questa guida.

Lo sforzo effettuato dovrebbe consentire agli studenti di sentirsi cittadini del mondo e di allargare gli orizzonti leggendo, attraverso i racconti dei diretti interessati, quando, che cosa e come gli altri studenti nel modo studiano l'Informatica. Ovviamente questa profonda rivoluzione a livello mondiale sia nei tempi, con un inizio

della formazione in informatica fin dalla scuola d'infanzia, sia nei modi, sia nei contenuti relativi ai curricula d'informatica necessita di una profonda revisione didattica, revisione per la quale è indispensabile il contributo indispensabile dei docenti che restano gli intermediari fondamentali nel processo di apprendimento degli studenti (Scientix). Questa revisione necessita inoltre, come discusso nel capitolo 2, di una ricerca educativa attiva (action research) attraverso una comunità di pratica di docenti che sperimenti e valuti l'efficacia di diverse traiettorie di apprendimento o approcci pedagogici basati su animazioni o visualizzazioni di algoritmi.

Accanto a questo gli strumenti di automazione d'ufficio per la comunicazione in forma scritta (videoscrittura), in forma verbale (presentazione) e in forma matematico-grafica (foglio di calcolo) sono presentati ed usati al fine di comunicare sia i processi sia i prodotti sviluppati durante il corso. In accordo con i framework internazionali l'uso di strumenti collaborativi sul cloud consente di potenziare le abilità collaborative. I linguaggi di programmazione e i fogli di calcolo sono usati estensivamente, come strumento per realizzare, dopo un'adeguata progettazione, sia algoritmi sia problemi a carattere interdisciplinare [32].

Scopo dello sforzo fatto è facilitare nei nativi digitali il passaggio dall'essere utilizzatore della tecnologia ad essere creatore di tecnologia usando la scienza.

In linea con quanto indicato da più parti a livello nazionale, dal Piano Nazionale Scuola Digitale, alle varie gare nazionali, il curriculum rappresenta una possibile risposta alla "chiamata per la costruzione di una visione di Educazione nell'Era Digitale, attraverso un processo che, per la scuola, sia correlato alle sfide che la società tutta affronta nell'interpretare e sostenere l'apprendimento lungo tutto l'arco della vita (life-long) e in tutti i contesti della vita, formali e non formali (life-wide)" (PNSD).

Un'attenzione, come discusso nel capitolo 3, è stata posta anche alle esigenze didattiche di studenti speciali. Anche in questo caso, confidando nel supporto di una comunità di pratica, si potrebbero realizzare diverse versione del curriculum, come ad esempio, un curriculum parlato (riferimento) allo scopo di facilitare l'apprendimento negli studenti ipovedenti o dislessici. Si auspica il coinvolgimento delle varie associazioni operanti nel settore.

L'approccio didattico, seguito nella progettazione del curriculum, prevede un'introduzione incrementale delle problematiche introducendo dettagli tecnico-scientifici quando sono necessari. Ogni capitolo vede l'intervallarsi di una contestualizzazione del problema e di attività pratiche in laboratorio. Utilizzando la parte introduttiva in modalità flipped (flipped classroom) e le attività pratiche laboratoriali, consentiranno agli studenti di avere sempre un ruolo attivo (constructivism) e al docente di concentrarsi, in qualità di allenatore, sul suo compito fondamentale: progettare esperienze di apprendimento [33]. Questo può essere fatto, ad esempio, selezionando le attività tra:

- ✓ Le attività sviluppate nel testo,
- ✓ Le verifiche sui contenuti del testo,
- ✓ Le verifiche con l'aiuto della rete,
- ✓ Gli esercizi e problemi a difficoltà incrementale,
- ✓ Le risorse didattiche nei materiali online organizzati online,

- ✓ I test online proposti attraverso una piattaforma online o attraverso test interattivi (runestone interactive)
- ✓ Le risorse educative (giochi educativi, libri, video, animazioni, articoli scientifici divulgativi, riviste per studenti anche in lingua inglese, etc…) citate nella ricca bibliografia e sitografia disponibile alla fine di ogni capitolo.

La scelta di una presentazione incrementale dei contenuti sia per approfondimenti sia per dettagli successivi lungo tutta l'opera è dettata oltre che dalla necessità di evitare un sovraccarico cognitivo dovuto a un superamento della capacità stimata della memoria di lavoro (citare work memory art tracing) anche, e soprattutto, dal voler favorire e spingere i giovani lettori a riprendere il contesto di apprendimento ogni qual volta si presenti un nuovo concetto, un approfondimento o un dettaglio dello stesso concetto. Studi dimostrano che quest'attività di recall favorisce l'apprendimento e la scalata di piramidi cognitive che portano gli studenti da una mera conoscenza all'acquisizione di abilità e competenze.

Il materiale contenuto nel curriculum offre parecchi spunti e suggerimenti sia per attività di recupero sia per attività di approfondimento confidando nella sapiente direzione dei docenti che sceglieranno le attività più idonee e confacenti gli interessi sia di gruppi sia dei singoli studenti.

Solo il docente potrà scegliere il percorso e le attività più adatte per i suoi studenti; attraverso il contatto quotidiano con gli studenti gli insegnanti potranno suggerire il percorso di apprendimento più adatto alle zone di "proximal development" (citare articolo) anche dei singoli, spingendoli, come suggerito anche da tutte le competizioni olimpiche (OPS) (OI), a dare il meglio di sé.

Ingredienti fondamentali di questo processo sono l'individuazione e la valorizzazione dei talenti di ogni studente (UNESCO quality education for all) (European schoolnet quality education for all is better). Solo la sapiente direzione dei docenti accompagnata dall'impegno e dallo studio motivato, serio, costante e resiliente di ogni studente determinerà il successo di questo lavoro.

Certo che quanto fatto rappresenta solo un punto di partenza per una comunità di apprendimento auguro a tutti una piacevole lettura.

4 Conoscenze, abilità e competenze

All'interno del curriculum possiamo individuare varie macro-aree: Pensiero computazionale; Pensiero critico; Programmazione e coding; Comunicazione e collaborazione; Ricerca, selezione, analisi, sintesi e visualizzazione di informazioni e dati anche in forma open; Società, senso civico, impegno sociale e cittadinanza digitale.

4.1 Lo sviluppo di concetti legati al pensiero computazionale

Tra i concetti legati allo sviluppo del pensiero computazionale possiamo ricordare:

1) Formulare problemi per una soluzione computazionale (CSTA, 2011) [34], in tutto il curriculum

2) ragionamento logico (CAS, 2015) [35]: in tutto il curriculum
3) Organizzare logicamente e analizzare dati (CSTA, 2011), in tutto il curriculum e, in modo particolare nel capitolo 12
4) Astrazione, inclusi i modelli e le simulazioni (CAS, 2015), (CSTA, 2011), (ISTE, 2016) [36], in tutto il curriculum. In particolare, per l'astrazione, il capitolo 10 sulle variabili, per i modelli e le simulazione attraverso il linguaggio Cellular e le sue applicazioni, e gli altri linguaggi di programmazione presentati nei materiali on-line.
5) Decomposizione (CAS, 2015), (ISTE, 2016) a partire dall'introduzione delle procedure e funzioni (ca. 8) nella sezione dei Puzzle, ad esempio con la tecnica del Divide and Conquer e le sue varianti, applicate a Puzzle e giochi matematici.
6) Generalizzazione (CAS, 2015): in varie sezioni del curriculum, come ad esempio le procedure e le funzioni e le tecniche algoritmiche di trasforma e conquista e di cambiamento di dominio
7) Pensiero Algoritmico (CAS, 2015) (CSTA, 2011), (ISTE, 2016): in tutto il curriculum. Nella sezione di puzzle dove sono presentate varie tecniche algoritmiche quali: divide et impera e le sue varianti (decrease and conquer, trasform and conquer), algoritmi avidi, miglioramenti iterativi, programmazione dinamica, backtracking, invarianti, e trasformazione in un altro dominio. Attraverso le attività e i problemi delle OPS e delle OI.
8) Valutazione delle efficienze e della correttezza (CSTA, 2011): nei capitoli 3, 4, 5, 6, 7, 8, 9 con particolare riferimento:
    a. all'efficienza degli algoritmi in termini di tempo, spazio e difficoltà di codifica
    b. all'efficienza e all'efficacia delle interfacce grafiche (materiali on-line)
    c. alla correttezza degli algoritmi attraverso le tecniche di debug, il concetto d'invariante e le tabelle di traccia
9) Valutazione (ISTE, 2016): attraverso la progettazione, lo sviluppo ed il confronto di più soluzioni e la scelta di quella migliore in funzione del campo di applicazione.
10) Generalizzazione e trasferimento in altri domini (CSTA, 2011): nella sezione dei Puzzle attraverso le tecniche algoritmiche di trasforma e conquista e trasformazione del problema in un altro dominino (come quello matematico), OI, OPS
11) Individuazione di ripetizioni (CAS, 2015) attraverso il disegno di figure e il concetto d'invariante
12) Analizzare dati (ISTE, 2016): uso delle liste, manipolazione di immagini in Pixley, manipolazioni di suoni e, in generale, nel trattamento di dati multimediali; attraverso l'uso dei fogli di calcolo

13) Rappresentare dati (ISTE, 2016): nel capitolo 12
14) Testing (ISTE, 2016): nei capitoli con attività legate al coding – 3, 4, 5, 6, 7, 8, 9.
15) Parallelizzazione (ISTE, 2016): nei materiali online presentando costrutti di calcolo parallelo di un'estensione di Snap! e semplici esempi del loro uso.
16) Raccogliere dati: nei materiali online attraverso l'uso dei Form di Google
17) Automazione (ISTE, 2016): nei materiali online con applicazione di robotica, e Internet of Things

4.2 Pensiero critico

Il pensiero critico viene sviluppato in tutto il curriculum anche attraverso la ricerca, selezione, analisi e sintesi di informazioni disponibili in rete; negli esercizi con l'aiuto della rete, disponibili in tutto il capitolo; nei materiali online con materiali quali "Impariamo ad Imparare.

4.3 Programmazione e coding

La programmazione e il coding sono trattati nei capitoli 3, 4, 5, 6, 7, 8 e 9 attraverso la codifica, usando linguaggi visuali a blocchi, di tutti i problemi presentati; nei materiali online attraverso la soluzione di problemi proposti nelle sezioni di verifica presenti nel testo, attraverso le attività CLIL.

4.4 Comunicazione e collaborazione

La comunicazione e la collaborazione vengono facilitate in tutto il curriculum dove, ad esempio, si è preferito usare il linguaggio naturale per progettare la soluzione dei problemi con l'obiettivo di migliorare le capacità di descrizione e comunicazione degli algoritmi. Questo è reso possibile anche dalla scelta dei linguaggi visuali a blocchi dove le indicazioni testuali ricalcano sostanzialmente il linguaggio naturale. Altre tecniche di progettazione grafica, come i diagrammi a blocchi, sono presentati al fine di un confronto e di un ampliamento degli strumenti a disposizione. I capitoli sulla comunicazione in forma scritta (cap. 5 sulla videoscrittura) e sulla comunicazione orale (cap. 11 strumenti di presentazione) evidenziano gli aspetti collaborativi e l'uso di strumenti di collaborazione on-line.

4.5 Ricerca, selezione, analisi, sintesi e visualizzazione di informazioni

La ricerca, selezione, analisi, sintesi e visualizzazione di informazione e dati anche in forma open é sviluppata in tutto il curriculum attraverso l'approccio di ricerca ed esplorativo sollecitato.

4.6 Società, senso civico, impegno sociale e cittadinanza digitale

Varie tematiche a sfondo civico e sociale sono presentate nel testo, nelle attività CLIL e nei materiali online quali, ad esempio: Adozioni e affidamento; Humanitarian FOSS Project (FOSS); ICT e Turismo sostenibile; ICT, salute e benessere; Etica professionale; Sviluppo sostenibile; Riciclo e inquinamento; Parità di genere; Digital divide; Importanza dello sport.

Il materiale a disposizione dovrebbe essere sufficiente a coprire un ampio spettro d'interessi dei giovani lettori e tale da offrire ai docenti un'adeguata varietà di risorse tra le quali selezionare, anche variando di anno in anno, di classe in classe e da studente a studente, i materiali più adatti alla situazione in esame. L'autore conta, sia in

prima persona, sia con l'aiuto della comunità di pratica dei docenti [37] [38] che vorranno usufruire di quest'opera, di ampliare la sezione dei materiali online anche in un'ottica interdisciplinare.
La sezione delle verifiche comprende inoltre problemi più complessi anche sotto forma di possibili progetti da svolgere sia in gruppo sia singolarmente, sparsi tra i vari capitoli del curriculum. Questo necessita negli studenti di "soft skills" quali, ad esempio:

1) Persistenza nel confrontarsi con problemi complessi (CSTA, 2011)
1) Interesse verso problemi e progetti sviluppabili in un arco temporale più lungo
2) Tecniche per progettare, codificare, analizzare, riflettere ed applicare (CAS, 2015)
3) Tecniche per la creazione di conoscenze

grazie anche ad un responsabilizzazione ed incoraggiamento dei docenti [39].

5 Traiettorie di apprendimento

Le varie attività possono essere sviluppate seguendo varie "traiettorie di apprendimento", alcune delle quali sono mostrate nella figura 1.

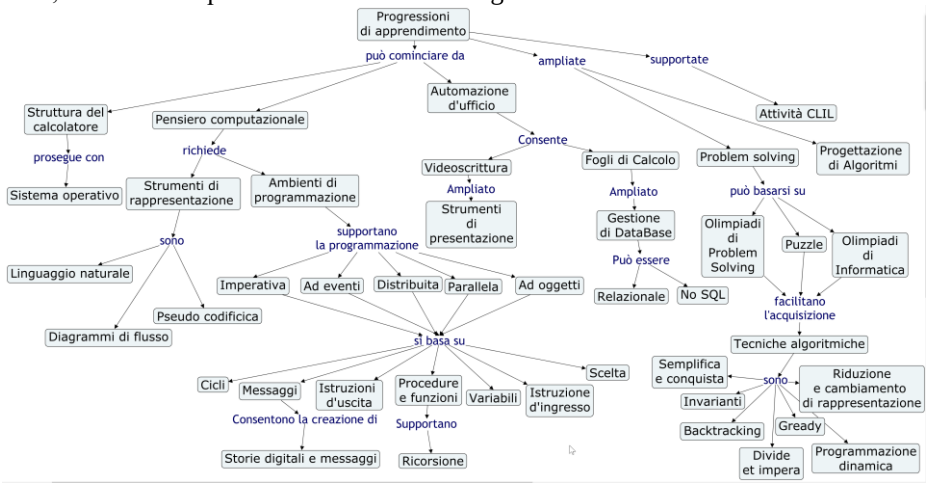

Figura 1. Esempi di traiettorie di apprendimento

6 Conclusioni e sviluppi futuri

In questo lavoro sono stati descritti i contenuti, gli approcci pedagogici e le tecnologie per un curriculum adatto per un primo corso d'informatica offrendo spunti di riflessione e confronto per la comunità italiana di educatori. L'autore, grazie anche alle riflessioni che scaturiranno da questo lavoro e da un confronto internazionale [40] aspira ad ampliare il lavoro anche con contenuti on-line, video, animazioni, test inte-

rattivi [41] e materiali di supporto grazie anche al contributo degli educatori che volessero contribuire alla sviluppo e alla creazione di risorse.

References

1. UNESCO. Director-General, 2009-2017 (Bokova, I. G) Unesco moving forward the 2030 Agenda for Sustainable Development
2. Owens, T. L. (2017). Higher education in the sustainable development goals framework. *European Journal of Education*, *52*(4), 414-420.
3. Burgstahler, S. E., & Cory, R. C. (Eds.). (2010). *Universal design in higher education: From principles to practice*. Harvard Education Press.
4. Burgstahler, S. (2013). Universal design in higher education: Promising practices. *Seattle: U of Washington*.
5. National curriculum in England: computing programmes of study https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study
6. Australian curriculum https://www.australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies/pdf-documents/
7. New Zealand Technology curriculum http://nzcurriculum.tki.org.nz/The-New-Zealand-Curriculum/Technology
8. CSTA K-12 Computer Science Standards, Revised 2017 https://www.csteachers.org/page/standards
9. Code.org https://code.org/
10. Coder Dojo https://coderdojo.com/
11. Europe code week https://codeweek.eu/
12. Programma il futuro https://programmailfuturo.it/
13. Codeweek it http://codeweek.it/
14. Scientix http://www.scientix.eu
15. Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.
16. Denning, P. J. (2017). Remaining trouble spots with computational thinking. Communications of the ACM, 60(6), 33-39.
17. Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., ... & Szabo, C. (2018, July). Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (pp. 55-106). ACM.
18. Becker, B. A., & Quille, K. (2019, February). 50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 338-344). ACM.
19. Harvey, B., & Mönig, J. (2010). Bringing “no ceiling” to scratch: Can one language serve kids and computer scientists. *Proc. Constructionism*, 1-10.
20. Olimpiadi di problem solving https://www.olimpiadiproblemsolving.it
21. Olimpiadi italiane d’informatica https://www.olimpiadi-informatica.it
22. Guzdial, M. Constructivism vs. Constructivism vs. Constructionism https://computinged.wordpress.com/2018/03/19/constructivism-vs-constructivism-vs-constructionism/
23. E Hazelkorn, R Charly. (2015) Science education for responsible citizenship. Report to the European Commission of the Expert Group on Science Education.

24. Leo Porter, Dennis Bouvier, Quintin Cutts, Scott Grissom, Cynthia Lee, Robert McCartney, Daniel Zingaro, and Beth Simon. 2016. A multi-institutional study of peer instruction in introductory computing. ACM Inroads 7, 2 (May 2016), 76-81. DOI: https://doi.org/10.1145/2938142
25. Peer Instruction for Computer Science https://www.peerinstruction4cs.org/
26. Education ambivalence. *Nature* **2010,** 465, (7298), 525-526.
27. Computer Science POGIL http://cspogil.org/Home
28. Johnson, Laurence F.; Smith, Rachel S.; Smythe, J. Troy; Varon, Rachel K. (2009). Challenge-Based Learning: An Approach for Our Time. Austin, Texas: The New Media Consortium
29. JL Bishop, MA Verleger . (2013). The flipped classroom: A survey of the research. ASEE National Conference Proceedings, Atlanta.
30. Licht, A.H, Tasiopoulou, E., Wastiau, P. (2017). Open Book of Educational Innovation. European Schoolnet, Brussels.
31. Giordano, D., Maiorana, F., Csizmadia, A. P., Marsden, S., Riedesel, C., Mishra, S., & Vinikienė, L. (2015, July). New horizons in the assessment of computer science at school and beyond: Leveraging on the viva platform. In Proceedings of the 2015 ITiCSE on Working Group Reports (pp. 117-147). ACM.
32. Scarabattolo, N. (2010). ECDL for problem solving. Un passo Avanti verso la scuola digitale. Mondo Digitale, 33 (1), 27-46
33. Brooks, F. P. (2012). The teacher's Job is to Design Learning Experiences; Not Primarily to Impart Information. *SIGCSE.* Raleigh, North Carolina, USA: ACM.
34. Computational Thinking. 2011; http://www.csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf
35. Computing at School, a subdivision of the British Computer Society (BCS). 2015. Computational Thinking: A Guide for Teachers; http://community. computingatschool.org.uk/files/6695/original.pdf 4. CSTA. Operational Definition of
36. International Society for Technology in Education. ISTE Standards for Students, 2016; http://www.iste. org/standards/standards/for-students-2016
37. Taylor, C., Spacco, J., Bunde, D. P., Butler, Z., Bort, H., Hovey, C. L., ... & Zeume, T. (2018, July). Propagating the adoption of CS educational innovations. In Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (pp. 217-235). ACM.
38. Maiorana, F., Richards, G., Lucarelli, C., Miles, B., Ericson. B. (2019). Interdisciplinary Computer Science Pre-service Teacher Preparation. In Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (In Press). ACM.
39. Hug, S., Guenther, R., & Wenk, M. (2013). Cultivating a K12 computer Science Community: a case study. *44th ACM technical symposium on Computer Science Education* (p. 275-280). Denver, Colorado: ACM.
40. Falkner, K., Sentence, S., Vivian, R. (2019). An International Benchmark Study of K-12 Computer Science Education in Schools. In Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (In Press). ACM.
41. Maiorana, F., Richards, G., Lucarelli, C., Miles, B., Ericson. B. (2019). Interdisciplinary Computer Science Pre-service Teacher Preparation. In Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (In Press). ACM.