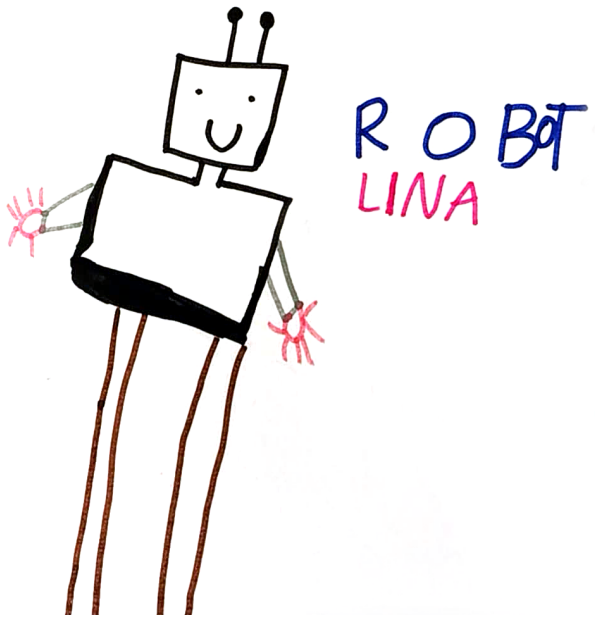
Il Pensiero Computazionale a Scuola

Pasquale Davide Rana
Università degli Studi di Bari, Università degli Studi di Udine
ranadavide@gmail.com



Abstract

Un articolo dal taglio teorico e sintetico che partendo dalla natura dell'Informatica e dalle origini del Pensiero Computazionale ne evidenzia il legame intrinseco. Dopo una breve esamina del loro spazio nel curriculum nazionale italiano, viene proposto un approccio integrato per favorire l'acquisizione delle abilità coinvolte nel processo di Pensiero Computazionale mediante l'utilizzo del Coding, della Robotica Educativa e delle Attività Unplugged evidenziandone la relazione con il Pensiero Creativo.

1 Informatica e Pensiero Computazionale

1.1 La natura dell'Informatica

Nel mondo della scuola l'informatica è inquadrata come *strumento* [2], come *uso di tecnologie* [16] e come *disciplina o scienza* [16]. L'oggetto di studio della scienza dell'informatica è l'informazione: l'elaborazione, la memorizzazione e la trasmissione dell'informazione in maniera automatica.

Il termine italiano reso, "informatica", deriva dal francese *informatique* ed è composto da *informat(ion)* 'informazione' e *(automat)ique* 'automatica'. Questa definizione non presuppone l'utilizzo di un elaboratore. Il termine inglese per indicare l'informatica è "*computer science*" che viene tradotto in italiano come scienza dei computer. Il termine inglese sottolinea lo studio del computer e dei sistemi computazionali nella definizione di informatica.

Nell'informatica intesa come scienza o disciplina convivono tre anime: *l'anima matematica*, *l'anima ingegneristica* e *l'anima scientifica*. L'anima matematica (paradigma razionalista), tipica dell'informatica teorica; l'anima ingegneristica (paradigma tecnologico), tipica dell'ambito dell'ingegneria del software; l'anima scientifica (paradigma scientifico), tipica dell'intelligenza artificiale e della modellazione e simulazione dei fenomeni.

1.2 Le origini del Pensiero Computazionale

Con il metodo pedagogico del costruzionismo di Papert, nascono il linguaggio della tartaruga ed il *Computational Thinking*. La locuzione "*Computational Thinking*" o *Pensiero Computazionale*, compare per la prima volta a pagina 182 del libro *"Mindstorms: children, computers, and powerful ideas"* [17], pubblicato da Seymour Papert nel 1980.

Nella sua declinazione più papertiana, il *Pensiero Computazionale* è uno *strumento del costruzionismo* che si avvale del computer e della tecnologia per costruire artefatti cognitivi e computazionali concreti a supporto dell'apprendimento.

L'espressione "*Pensiero Computazionale"* è divenuta popolare dieci anni dopo, nel 2006 [25], con un articolo pubblicato da Jeannette M. Wing (1956), direttrice del Data Science Institute e Prof.ssa di informatica alla Columbia University, in cui viene presentato, senza definirlo in maniera precisa, come la capacità di risolvere problemi, progettare sistemi e comprendere il comportamento umano, attingendo dai principi fondamentali dell'informatica. Tale definizione classica è stata concepita nel tentativo di diffondere la risoluzione dei problemi dall'informatica ad altre discipline e si compone di quattro pilastri: decomposizione, riconoscimento di pattern o modelli, astrazione e algoritmi. In estrema sintesi, secondo Wing, il Pensiero Computazionale equivale a "*pensare come un informatico*" [25].

Wing, nelle sue pubblicazioni del 2010 [26] e del 2014 [27], definisce il *Pensiero Computazionale* come *l'insieme dei processi mentali usati per formulare i problemi e le loro soluzioni in modo tale che la descrizione delle soluzioni sia effettivamente eseguibile da un agente - uomo o macchina - che elabora informazioni*. (nota anche come la definizione di Cuny-Snyder-Wing).

La definizione di Wing è stata molto influente nella narrativa del Pensiero Computazionale. Wing si concentra sul *processo di risoluzione del problema*, ovvero sull'aspetto operativo di rendere i problemi calcolabili sfruttando i concetti fondamentali dell'informatica. In questo senso il *Pensiero Computazionale* può essere considerato come "*il nucleo scientifico dell'informatica*" [1]. Questo aspetto è essenziale ma non è l'obiettivo principale del *Pensiero Computazionale* di Papert.

Il Pensiero Computazionale di Papert, una sorta di pensiero procedurale, viene utilizzato per la costruzione di artefatti cognitivi concreti a supporto dei modelli mentali costruiti da chi apprende. Tale veduta, comprende implicitamente i concetti di Wing, ma non si limita ad essi. La sua veduta è più ampia. È uno strumento a supporto dell'apprendimento, che presuppone la conoscenza di alcuni concetti squisitamente appartenenti alla disciplina informatica per poterne usufruire a pieno. I

computer, considerati come estensori della mente e del pensiero, vengono usati per creare e scoprire. Attraverso essi, è possibile creare *micromondi* di matematica, di fisica e di altre scienze per facilitarne l'apprendimento. La geometria della tartaruga ne è un esempio.

Per una trattazione più completa sul pensiero computazionale come strumento pedagogico e come abilità trasversale indispensabile si rimanda alla lettura degli articoli di Rana, P.D. (2020) *Percorsi di Pensiero Computazionale nella scuola dell'Infanzia* [19] e (2021) *Quanti Computational Thinking? Il pensiero computazionale da Papert a Wing per l'insegnamento dell'informatica, della creatività e delle social skills nel biennio del liceo scientifico* [20] riportati in bibliografia.

1.3 Il Pensiero Computazionale nell'Istruzione

Selby e Woollard [24] hanno ricercato una definizione appropriata da utilizzare nell'istruzione scolastica per facilitare la progettazione dei curriculum di informatica e incoraggiare lo sviluppo del Pensiero Computazionale per tutti. Il loro lavoro di analisi della letteratura copre gli anni dal 2006 al 2013, e identifica termini, descrizioni e significati più ricorrenti, tenendo conto della motivazione dell'inserimento o dell'esclusione di un termine da parte di ogni singolo autore.

In *Computational Thinking - a guide for teachers* [9] Selby e Woollard definiscono il pensiero computazionale come un *processo cognitivo* o di pensiero che coinvolge il ragionamento logico, associato ma non limitato alla risoluzione dei problemi e riflette:

- la capacità di pensare algoritmicamente;
- la capacità di pensare in termini di scomposizione;
- la capacità di pensare per generalizzazioni, identificando e facendo uso di pattern;
- la capacità di pensare per astrazioni;
- la capacità di pensare in termini di valutazioni;

In altre parole, *il Pensiero Computazionale è un approccio focalizzato alla risoluzione dei problemi, che incorpora processi di pensiero che utilizzano astrazione, decomposizione, progettazione algoritmica, valutazione e generalizzazione* [24]. Tale definizione è stata successivamente adottata dal *Computing at School* (CAS) nel Regno Unito.

1.4 I Pilastri del Pensiero Computazionale

Sebbene ci siano punti di vista diversi su una definizione operativa sul processo di Pensiero Computazionale, c'è accordo sugli elementi fondamentali da cui è caratterizzato: pensiero algoritmico, pensiero logico, astrazione, generalizzazione e decomposizione. Talvolta viene considerata anche la valutazione come un elemento del Pensiero Computazionale. L'elenco degli elementi si basa sull'analisi di Selby e Woollard [24].

Il *pensiero algoritmico* è il processo di creazione di algoritmi, è la capacità di pensare in termini di sequenze e regole per risolvere problemi o capire situazioni. È la parte fondamentale del set di abilità del Pensiero Computazionale che lo rende diverso dalle capacità di pensiero di altre discipline come il pensiero scientifico, il pensiero matematico, il pensiero progettuale e così via, dove sorgono gli altri elementi costitutivi del Pensiero Computazionale.

Il *pensiero logico* è la capacità di sviluppare e verificare ipotesi. Pensare in modo logico è necessario per risolvere problemi, per sviluppare algoritmi, implementarli come programmi e verificare se funzionano correttamente o meno, in modo informale o formale.

L'astrazione è il processo di semplificazione e di occultamento dei dettagli. L'essere umano astrae continuamente, senza di essa non sarebbe possibile vivere in un mondo oggi più complesso che mai. L'astrazione permette di governare questa complessità nel mondo come nella programmazione. Essere in grado di pensare a più livelli di astrazione e spostarsi tra i livelli è un'abilità chiave. Ciò è necessario quando si sviluppano soluzioni, spostandosi avanti e indietro, ad esempio, tra il livello del problema, i livelli di progettazione e i livelli di programmazione. Come parte del Pensiero

Computazionale, fornisce un modo per gestire la complessità al fine di facilitare la risoluzione dei problemi e consentire la progettazione di sistemi computazionali complessi.

La *decomposizione* o scomposizione, consiste nel suddividere un problema o un compito complesso in parti o sotto-problemi di complessità inferiore che possono essere risolti separatamente e in maniera indipendente gli uni dagli altri. È possibile scomporre un problema complesso fino a quando le parti più piccole sono così semplici da diventare facili da risolvere. Ciascuna sotto-soluzione, concorrerà alla soluzione del problema iniziale.

La *generalizzazione*, strettamente correlata con l'astrazione, implica il trovare la soluzione di un problema e la creazione di una versione più generale applicabile a un insieme più ampio di problemi. Tale concetto è applicabile agli algoritmi, ai problemi e alle soluzioni.

L'essere umano valuta il suo operato quotidianamente. La valutazione di quanto realizzato è una parte importante anche di qualsiasi approccio alla risoluzione dei problemi. La *valutazione* consiste nell'individuare le possibili soluzioni a un problema e giudicare quale sia la migliore da utilizzare, se funzioneranno in alcune situazioni ma non in altre, e come possono essere migliorate.

1.5 Un legame intrinseco

Il Pensiero Computazionale è contenuto nell'informatica intesa come disciplina scientifica e si differenzia dall'informatica intesa come professione. Il Pensiero Computazionale è trasversale alle altre discipline e racchiude concetti sia per leggere la realtà che per intervenire su di essa. Il Pensiero Computazionale non è prerogativa solo degli informatici, è insegnare a ragionare, a pensare come un informatico, a vedere il mondo con occhi computazionali e a trovare soluzioni creative. Il Pensiero Computazionale è al centro dell'informatica intesa come disciplina. Di conseguenza un contesto efficace per lo sviluppo del Pensiero Computazionale è l'apprendimento dell'informatica. Il Pensiero Computazionale e l'informatica sono intrinsecamente connessi.

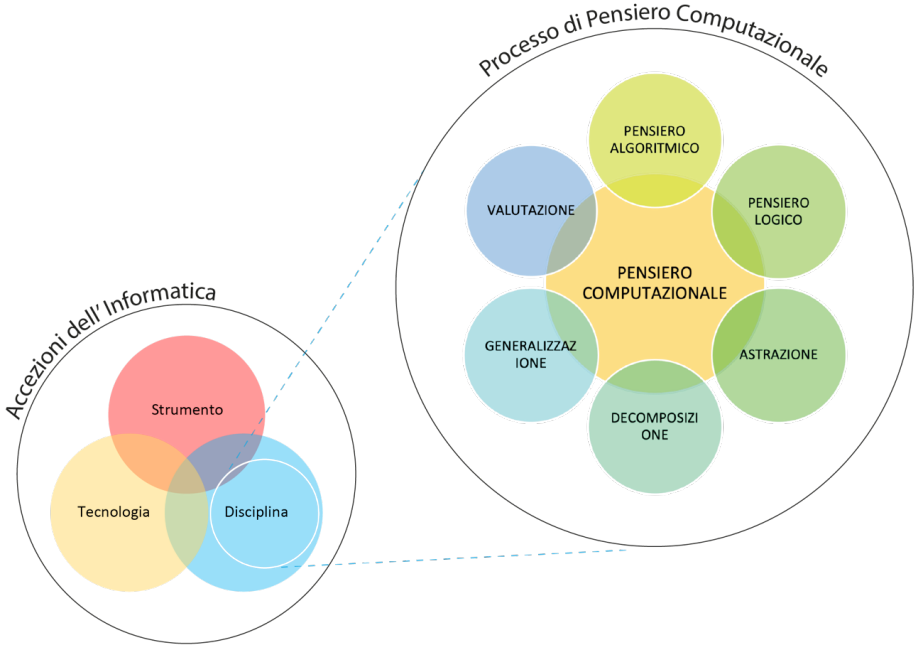


Figura 1. Relazione fra Informatica e Pensiero Computazionale

1.6 La trasversalità del Pensiero Computazionale

Il *Pensiero Computazionale* è uno strumento mentale di utilità generale che rimane valido per tutta la vita, come avviene per il linguaggio naturale e la matematica [10]; l'apprendimento di tali concetti investe tutti i livelli di istruzione e tutti i campi di applicazione. Da questo si deduce il ruolo cruciale rivestito dal *Pensiero Computazionale* e quindi dall'informatica, come disciplina trasversale in grado di fornire un valido contributo per una migliore comprensione di altre discipline, come già avviene per l'italiano e per la matematica.

Il Pensiero Computazionale viene considerato da Wing [25] come la "quarta abilità di base" oltre a saper leggere, scrivere e calcolare. Ed è per questo che dovrebbe essere insegnato sin dalla tenera età. Questo non significa che i bambini e più in generale l'essere umano debbano imparare a pensare come il computer: il Pensiero Computazionale è il modo in cui gli esseri umani insegnano al computer a risolvere problemi e non viceversa.

1.7 Il curriculum nazionale italiano

Il sistema educativo italiano comprende il sistema integrato zero-sei anni, il primo ciclo di istruzione, il secondo ciclo di istruzione e l'istruzione superiore [12].

Il *Reviewing Computational Thinking in Compulsory Education* [11] (2022) descrive gli sviluppi significativi riguardanti l'integrazione delle competenze del Pensiero Computazionale nella scuola dell'obbligo in Europa tra il 2016 e il 2021.

In Italia, le competenze del Pensiero Computazionale non fanno parte delle linee guida ufficiali del curriculum nazionale per l'istruzione obbligatoria come invece avviene in altri stati europei. Tuttavia, il Ministero dell'Istruzione ha pubblicato due importanti documenti sull'educazione digitale che affrontano il tema del Pensiero Computazionale: la *Strategia Nazionale per la Scuola Digitale* [13] pubblicata nel 2015 e le *Indicazioni Nazionali e Nuovi Scenari* [14], per le scuole dell'infanzia e primari, pubblicata nel 2018. In questi documenti, il Pensiero Computazionale è trattato come un tema chiave da promuovere nelle scuole. Inoltre, il Ministero ha sostenuto iniziative sia a livello nazionale, Code Week e code.org, che a livello locale. Diverse scuole primarie stanno includendo le abilità del Pensiero Computazionale, la programmazione e le attività di robotica nei curricula concettuali-pedagogici che esse stesse hanno definito. Nella scuola secondaria di secondo grado, l'informatica è una materia curriculare nelle scuole che offrono specifici indirizzi di studio in tal senso.

2 Il Pensiero Creativo

Resnick, sul solco del costruzionismo di Papert, propone la sua visione di apprendimento rappresentata dalla "*spirale dell'apprendimento creativo*". Il motore del pensiero creativo. Il contenuto del paragrafo è basato sul saggio di Resnick: *Come i bambini. Immagina, crea, gioca e condividi*. *Coltivare la creatività con il Lifelong Kindergarten del MIT* [22].

La "*spirale dell'apprendimento creativo*", è un processo che incoraggia i bambini a *immaginare* cosa vogliono fare, a *creare* progetti *giocando* con strumenti e materiali, a *condividere* idee e creazioni con gli altri, a *riflettere* sulle loro esperienze con l'aiuto dell'insegnante. Il processo è iterativo, a seguito della riflessione e delle esperienze compiute, i bambini immaginano nuove idee e nuovi percorsi.

Figura 2. La "spirale dell'apprendimento creativo" è il motore del pensiero creativo [22].

Nei giardini di infanzia la spirale dell'apprendimento creativo si ripete continuamente: i materiali cambiano (blocchi di legno, pastelli, cartoncini, colori), e cambiano le cose create (disegni, castelli, storie, canzoni), ma l'essenza del processo rimane immutata.

La spirale dell'apprendimento creativo è il motore del pensiero creativo. Muovendosi lungo di essa, i bambini sviluppano e perfezionano le loro capacità di pensare creativamente. Imparano a sviluppare le proprie idee, a metterle alla prova, a sperimentare alternative, ad accogliere gli spunti degli altri e a generare nuove idee sulla base delle proprie esperienze. Purtroppo, dopo la scuola dell'infanzia, la maggior parte delle scuole si allontana dalla spirale dell'apprendimento creativo concentrandosi sul fornire insegnamento e informazioni.

Come è possibile incoraggiare e sostenere le esperienze di apprendimento creativo? Resnick ed il suo gruppo di ricerca del MIT hanno definito quattro principi guida per aiutare i bambini ed i ragazzi a sviluppare il pensiero creativo: *Project*, *Passion*, *Peers* e *Play*, conosciuti anche come le "quattro P". Il modo migliore per coltivare la creatività è sostenere le persone nel lavorare su progetti basati sulle loro passioni, in collaborazione con i pari e in uno spirito giocoso.

3 Come favorire il *processo di Pensiero Computazionale*?

Come si conducono i bambini ed i ragazzi nel viaggio verso l'acquisizione delle abilità del *processo di Pensiero Computazionale*? È vantaggioso adottare nell'insegnamento del Pensiero Computazionale un approccio costruttivista e costruzionista. È conveniente mostrare aspetti del Pensiero Computazionale adatti e adattati all'età dei discenti.

A prescindere da quale sia l'approccio adottato, l'attività di programmazione è un passaggio chiave nell'insegnamento del Pensiero Computazionale partendo da contesti familiari e stimolanti, come creare e animare una storia, guidare un robot, programmare un gioco o trovare la strada per uscire da un labirinto. Per fare questo è possibile usare un computer, un tablet, un robot programmabile, costruzioni, attività di drammatizzazione, giochi cinestetici e percorsi creativi. È da tenere presente che senza una esperienza di programmazione alcuni concetti rimangono sfuggevoli.

La strategia proposta per favorire il processo di Pensiero Computazionale è un approccio integrato che combina esperienze indipendenti o intersecate di Coding, Robotica Educativa e Attività Unplugged per la creazione di artefatti cognitivi e computazionali, inserite all'interno della cornice pedagogica del gioco.

Seguendo la spirale dell'apprendimento creativo di Resnick nella realizzazione di un artefatto cognitivo e computazionale è favorito anche il Pensiero Creativo. È questo il punto di incontro fra il Pensiero Computazionale di Papert e il Pensiero Creativo di Resnick. Con i bambini questo si verifica in maniera spontanea, mentre con il crescere è necessario coinvolgere i ragazzi in un progetto per loro appassionante, promuovendo la collaborazione tra pari e un contesto di gioco.

Di seguito vengono analizzati il Coding, la Robotica Educativa e le Attività Unplugged dal punto di vista teorico. Per una loro applicazione pratica in contesti scolastici si rimanda alla lettura degli articoli riportati in bibliografia di Rana, P.D. (2020) [19] per la scuola dell'infanzia, (2021) [20] e (2022) [21] per la scuola secondaria di secondo grado.

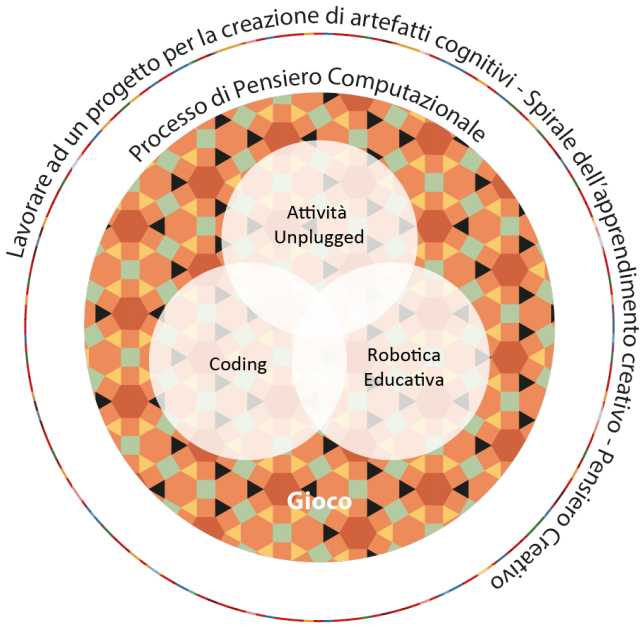


Figura 3. Metodologia integrata per favorire il processo di Pensiero Computazionale e il Pensiero Creativo

3.1 Coding

Il *coding*, come anche la *robotica educativa*, educano ad un utilizzo consapevole della tecnologia, permettono di sperimentare in prima persona, sostituiscono *l'imparare per usare* con *l'usare per imparare,* permettono il controllo dell'errore e l'apprendere da esso, di lavorare in autonomia senza l'aiuto dell'adulto, di sviluppare e potenziare la creatività, i processi logici, la concentrazione, l'attenzione, la precisione e l'esercizio del Pensiero Computazionale.

Il coding non è il Pensiero Computazionale. Il coding è un mezzo attraverso il quale è possibile esercitare e sviluppare il Pensiero Computazionale. Fare coding vuol dire scrivere codice, scrivere un programma. La traduzione in codice è solo una parte del processo di risoluzione del problema e del Pensiero Computazionale.

Il coding si differisce dalla programmazione in senso stretto. Quest'ultima è una skill tecnica tipica degli informatici ed è utilizzata nel mondo del lavoro. Il coding, la programmazione e l'informatica in generale hanno un elevato valore meta-cognitivo poiché permettono di riflettere sul proprio processo di pensiero e di risoluzione dei problemi.

3.2 Robotica Educativa

La *robotica educativa* permette di "parlare" con un *artefatto cognitivo concreto*, ad esempio un robot, portando il *coding* nel mondo del reale.

Nel libro "*Mindstorms: Children, Computers, and Powerful Ideas"* [17], Papert evidenzia come l'utilizzo della robotica, non trasmettendo in maniera diretta competenze informatiche, genera curiosità, stimola la creatività e la motivazione all'apprendimento, permettendo di costruire ed entrare in contatto con "*idee potenti*". Tali idee potenti si riferiscono a nuovi modi di pensare, nuovi modi di utilizzare la conoscenza e nuovi modi di stabilire connessioni personali ed epistemologiche con altri domini della conoscenza [18].

Possiamo ritrovare tracce dei dispositivi robotici all'avanguardia oggi, come i robot Bee-Bot o Cubetto, all'interno degli studi di Piaget, di Papert e del lavoro di ricerca di Radia Perlman [23].

Il robot Cubetto, prodotto nel 2016 dalla Primo Toys, utilizza un *linguaggio di programmazione tangibile*, ed è considerato sia come l'evoluzione del sistema *Tortis*, ideato nel 1974 da Perlman, e quindi come un artefatto cognitivo del costruzionismo, sia come un materiale di costruzione montessoriano poiché *auto-educativo* e *auto-correttivo.* Nel sistema *Tortis,* la "*button box"* e la "*la slot machine*", consentivano ai bambini di controllare un robot chiamato *Turtle*. In *Cubetto,* la scheda, che ospita le istruzioni tangibili, svolge la stessa funzione.

Il concetto di idee potenti è stato ripreso come elemento chiave nel lavoro di *Marina Umaschi Bers*, una pioniera nel campo di studi sulla tecnologia per la prima infanzia. Bers ha conseguito il dottorato di ricerca nel 2001 presso il MIT Media Lab sotto la supervisione di Papert. Bers afferma che le idee potenti "offrono nuovi modi di pensare" in relazione sia al contenuto che al processo, ovvero sia a ciò che i bambini imparano sia a come lo imparano [7].

Bers [6] descrive sette "idee potenti" collegate al Pensiero Computazionale, che sono adatte allo sviluppo dei bambini nella scuola dell'infanzia (pre-K) e che non sono legate a un particolare ambiente di programmazione del computer, ma piuttosto alla disciplina dell'informatica e alle sue abitudini mentali associate. Queste idee sono: algoritmi, modularità, strutture di controllo, rappresentazione, hardware/software, processo di progettazione e debug.

Sia Papert che Bers, utilizzano il termine "idee potenti" per descrivere il grande impatto delle esperienze di informatica su altri tipi di apprendimento. Il percorso Piaget - Papert - Bers è un collegamento diretto dallo sviluppo originale della teoria costruttivista alle attività di robotica educativa [11] e ai programmi per l'insegnamento dell'informatica e del Pensiero Computazionale.

3.3 Attività Unplugged

Il progetto *Computer Science Unplugged* (CS Unplugged), nato inizialmente nell'Università di Canterbury come un programma di sensibilizzazione per coinvolgere gli studenti delle scuole primarie all'informatica, è adesso una raccolta di materiale didattico gratuito ampiamente utilizzato per coinvolgere una varietà di pubblico con i concetti dell'informatica attraverso giochi e puzzle coinvolgenti che utilizzano carte, spago, pastelli e tanto altro senza dover imparare a programmare o persino utilizzare un dispositivo digitale e per aiutarli a capire che l'informatica non è solo programmazione [4]. Il progetto ha avuto origine inizialmente con una raccolta di attività condivise online dai primi anni '90 e culminate nel 1999 con il libro online gratuito "*Computer Science Unplugged: Off-line activities and games for all ages"* [5] redatto da *Tim Bell*, Mike Fellows e Ian Witten e destinato agli accademici coinvolti nella divulgazione della disciplina. Le attività CS Unplugged forniscono un'impalcatura per un approccio costruttivista all'introduzione di argomenti di informatica, senza la necessità di imparare prima la programmazione.

Le attività di CS Unplugged si basano su una serie di principi [8] fra cui: l'informatica senza computer, il learning by doing, l'utilizzo del gioco e della sfida, l'importanza della resilienza, l'uso dello storytelling, un approccio costruttivista, attività altamente cinestetiche e spiegazioni brevi e semplici. Linee guida specifiche su come progettare tali attività sono approfondite in "*A CS unplugged design pattern"* [15].

Ci sono molte attività "*unplugged*" che non sono necessariamente basate su csunplugged.org, ma che condividono gli stessi elementi: i computer non sono richiesti anche se tutti i concetti provengano dall'informatica, gli studenti sono impegnati in attività cinestetiche e qualsiasi attrezzatura necessaria è disponibile a basso costo.

Sebbene l'approccio unplugged si discosti nettamente dal costruzionismo di Papert che conferisce alla programmazione un ruolo guida come meta-strumento per la costruzione della conoscenza, si possono evidenziare alcune relazioni. Le attività unplugged sono concrete piuttosto che formali e mirano a insegnare ai bambini idee di informatica complesse, idee che di solito vengono rimandate fino a quando non diventano pensatori adulti/formali/astratti; i concetti di informatica non sono semplificati, ma resi accessibili con esperienze pratiche [3]; nelle attività unplugged i bambini usano i loro corpi o la manipolazione fisica di oggetti per eseguirle.

Allo stesso modo, Papert mirava a insegnare idee matematiche profonde molto prima che i bambini avessero la competenza di astrazione per afferrarle formalmente: "La mia congettura è che molto di ciò che ora vediamo come troppo 'formale' o 'troppo matematico' sarà appreso facilmente quando i bambini cresceranno in un mondo ricco di computer in un futuro molto prossimo" [46 pag.7]. Inoltre, ha progettato LOGO in modo che fosse "corpo sintonico": i bambini potevano usare il proprio corpo per impersonare la Tartaruga disegnando sullo schermo: "lavorare con la Tartaruga mobilita l'esperienza e il piacere del bambino nel movimento. Si basa sulla conoscenza consolidata del bambino della 'geometria corporea' come punto di partenza per lo sviluppo dei ponti nella geometria formale" [46 pag.58] .

Il Progetto CS Unplugged è nato con lo scopo di aiutare i bambini a capire cosa fa un informatico. Abbinare le attività CS Unplugged con le idee di Pensiero Computazionale è utile per mostrare come entrambi servono a uno scopo simile. In effetti, l'articolo di Wing del 2010 cita specificamente CS Unplugged sotto il titolo "Il Pensiero Computazionale nell'istruzione" [26].

4 Conclusioni

Il Coding, la Robotica Educativa e le Attività Unplugged giocano un ruolo fondamentale nel coltivare le abilità coinvolte nel processo di Pensiero Computazionale, poiché oltre a favorire un apprendimento attivo mediante esperienze pratiche che stimolano la logica, la creatività e la

risoluzione di problemi, permettono di creare artefatti cognitivi e computazionali concreti a supporto dell'apprendimento. In questo modo, il Pensiero Computazionale ed il Pensiero Creativo, diventano una chiave essenziale per preparare le future generazioni ad affrontare creativamente le sfide di un mondo in continua evoluzione.

References

1. Armoni, M. (2015), *Computer Science, Computational Thinking, Programming, Coding: The Anomalies of Transitivity in K-12 Computer Science Education*, ACM Inroads, 7(4), 24-27.
2. Baudé J. (2007), *Le développement de l'informatique et des TIC*, Association EPI
3. Bell, T., & Lodi, M. (2019). *Constructing computational thinking without using computers*. Constructivist foundations, 14(3), 342-351. https://constructivist.info/14/3/342
4. Bell, T., & Vahrenhold, J. (2018*). CS unplugged—how is it used, and does it work?*. In Adventures between lower bounds and higher altitudes (pp. 497-521). Springer, Cham.
5. Bell, T., Witten, I.H., Fellows, M. (Original Book) (1999). *Computer Science Unplugged: Off-Line Activities and Games for All Ages* - https://classic.csunplugged.org/
6. Bers, M. (2020), *Coding as a Playground. Programming and Computational Thinking in the Early Childhood Classroom*, Routledge, https://doi.org/10.4324/9781003022602
7. Bers, M. U. (2008). *Blocks to Robots: Learning with Technology in the Early Childhood Classroom*. New York: Teachers College Press.
8. CS Unplugged Principles - https://www.csunplugged.org/en/principles/
9. Csizmadia, A.P., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C.C., & Woollard, J. (2015). *Computational thinking - a guide for teachers*.
10. Forsythe, G.E. (1968), *What To Do Till The Computer Scientist Comes*, The American Mathematical Monthly, 75(5), 454-462.
11. Gadzikowski A. (2018), *Robotics for Young Children: STEM Activities and Simple Coding*, Redleaf Press.
12. Ministero dell'Istruzione, dell'Università e della Ricerca, Il Sistema educativo di istruzione e formazione in Italia. https://www.miur.gov.it/sistema-educativo-di-istruzione-e-formazione, ultima consultazione in data 02/2025.
13. Ministero dell'Istruzione, dell'Università e della Ricerca. (2015). *Piano Nazionale Scuola Digitale*.. http://www.istruzione.it/scuola_digitale/allegati/Materiali/pnsd-layout-30.10-WEB.pdf
14. Ministero dell'Istruzione, dell'Università e della Ricerca. (2018) *Indicazioni nazionali e nuovi scenari*. https://www.miur.gov.it/documents/20182/0/Indicazioni+nazionali+e+nuovi+scenari/
15. Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T., Kuno, Y.: *A CS unplugged design pattern*. In: Fitzgerald, S., Guzdial, M., Lewandowski, G., Wolf- man, S.A. (eds.) Proceedings of the 40th SIGCSE Technical Symposium on Com- puter Science Education, SIGCSE 2009, Chattanooga, TN, USA, pp. 231–235. ACM (2009)
16. Paoletti F. (1993), *Épistémologie et technologie de l'informatique*, Le bulletin de l'EPI
17. Papert S. (1980), *Mindstorms: children, computers, and powerful ideas*, Basic Books, Inc. New York, tr.it., *Mindstorm. Bambini, computers e creatività*, Emme Edizioni, Milano 1984.
18. Papert, S. (2000), "*What's the big idea? Toward a pedagogy of idea power*", IBM Systems Journal, Vol. 39/3.4, pp. 720-729, https://doi.org/10.1147/sj.393.0720
19. Rana, P.D. (2020), *Percorsi di Pensiero Computazionale nella scuola dell'Infanzia*, Atti Convegno Nazionale DIDAMATiCA 2020 "Smarter School for Smart Cities", ISBN: 978-8-89-809161-4.
20. Rana, P.D. (2021), *Quanti Computational Thinking? Il pensiero computazionale da Papert a Wing per l'insegnamento dell'informatica, della creatività e delle social skills nel biennio del liceo scientifico*, Atti Convegno Nazionale DIDAMATiCA 2021 "Artificial Intelligence for Education", ISBN 978-88-98091-62-1
21. Rana, P.D. (2022), *Un artefatto cognitivo per la costruzione delle social skills Il pensiero computazionale ed il pensiero creativo nel biennio del liceo scientifico*, Atti Convegno Nazionale DIDAMATiCA 2022 "La

trasformazione digitale nella Scuola, negli ITS, nell'Università e nella formazione professionale", ISBN 978-88-98091-63-8

22. Resnick M. (2018), *Come i bambini. Immagina, crea, gioca e condividi*. *Coltivare la creatività con il Lifelong Kindergarten del MIT*, Erickson, Trento.
23. Perlman, R. (1976), *Using computer technology to provide a creative learning environment for preschool children*. https://dspace.mit.edu/handle/1721.1/5784
24. Selby, C., & Woollard, J. (2013). *Computational thinking: The developing definition*.
25. Wing, J.M. (2006), *Computational Thinking*, Communication of the ACM, vol. 49, no. 3, March 2006, pp. 33-35.
26. Wing, J.M. (2010), Research Notebook: *Computational thinking - what and why?*, The Link Magazine, 20-23.
27. Wing, J.M. (2014), *Computational thinking benefits society*. 40th Anniversary Blog of Social Issues in Computing, *2014*.