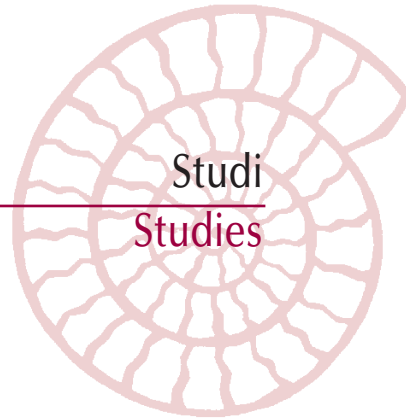

Studi
Studies

Proposte emergenti per lo studio della valutazione del pensiero computazionale

New framework for assessing the development of computational thinking

Michele Baldassarre
Università degli Studi di Bari "Aldo Moro" - michele.baldassarre@uniba.it
Immacolata Brunetti
Università degli Studi di Bari "Aldo Moro" - Immacolata.brunetti@uniba.it
Maria Brunetti
MIUR - maria.brunetti@istruzione.itt

ABSTRACT

Several study of educational research (Wing, 2006; Papert, 1975, 1981; Wilensky, 2001; Wilensky e Reisman,2006; Wilensky e Resnik, 1999; Kolodner et all, 2003; Puntambekar and Kolodner, 2005; Y. Kali and M.C. Linn, 2009; Linn, H.S. Lee, Tinker, Husic, and Chiu, 2006, Kafai, 2006; Kafai and Ching, 2001) have shown that there is little agreement about what computational thinking encompasses, and even less agreement about strategies for assessing the development of computational thinking in young people. We are interested in the ways that design-based learning activities – in particular, programming interactive media – support the development of computational thinking in young people. Our context is Scratch – a programming environment that enables young people to create their own interactive stories, games, and simulations, and then share those creations in an online community with other young programmers from around the world.

The first part of the paper describes the key dimensions of our computational thinking framework: computational concepts (the concepts designers engage with as they program, such as iteration, parallelism, etc.), computational practices (the practices designers develop as they engage with the concepts, such as debugging projects or remixing others' work), and computational perspectives (the perspectives designers form about the world around them and about themselves). The second part of the paper describes our evolving approach to assessing these dimensions, including artifact-based interviews, and design scenarios. We end with a set of suggestions for assessing the learning that takes place when young people engage in programming[1].

Diversi studi sulla ricerca educativa (Wing, 2006; Papert, 1975, 1981; Wilensky, 2001; Wilensky e Reisman,2006; Wilensky e Resnik, 1999; Kolodner et all, 2003; Puntambekar and Kolodner, 2005; Y. Kali and M.C. Linn, 2009; Linn, H.S. Lee, Tinker, Husic, and Chiu, 2006, Kafai, 2006; Kafai and Ching, 2001) dimostrano che gli studenti apprendono le strategie del pensiero computazionale mentre studiano e gli insegnanti riescono a modellare il pensiero verso queste strategie e adeguare misure di orientamento (K. Brennan, & M. Resnick, 2012). In molti casi un elemento chiave per l'insegnamento di queste strategie di pensiero è costituito dalla funzionalità offerta dagli ambienti di apprendimento idonei, particolarmente utili nel promuovere il pensiero computazionale (Wing, 2006); Affinché, però, ciò non risulti solamente una mera applicazione di giochi interattivi, il nostro studio verte principalmente sulla valutazione di quest'attività come prodotto e processo.

La ricerca proposta, è stata svolta in una classe prima di una scuola primaria nella provincia di Bari. Seguendo il programma del corso base di coding dal sito implementato dal Miur su "Programma il futuro", la docente ha sottoposto agli allievi un programma di coding attraverso giochi interattivi e collaborativi nel quale gli studenti si sono cimentati in un vero e proprio studio di programmazione informatica. Il nostro obiettivo principale verte sul modo in cui tali attività sostengono il pensiero computazionale e come valutarle. Abbiamo individuato delle dimensioni chiave nel processo di insegnamento-apprendimento di coding rinvenibili nei concetti, nelle pratiche e nelle prospettive computazionali che verranno analizzate insieme alle interviste degli studenti e al loro prodotto. Tali dati saranno successivamente triangolati con il pensiero dell'insegnante e i relativi indicatori sulla trasposizione didattica interna ed esterna.

I risultati attesi vogliono dimostrare che il pensiero computazionale ha natura interdisciplinare poiché sostiene il naturale processo di insegnamento e apprendimento di tutte le discipline ed è possibile valutarlo non come dato intuitivo ma come prodotto finale in relazione allo spazio, al tempo, alla lettura e alla scrittura.

KEYWORDS

Coding, Computational Thinking, Assessing, Knowledge Process, Tecnology.
Coding, Pensiero Computazionale, Valutazione, Processi Di Conoscenza, Tecnologie.

* **Pur condividendo gli autori la responsabilità complessiva dei contenuti del contributo, i parr. 1, 2, 2.1 sono da attribuirsi a Michele Baldassarre, Prof. Associato del Dipartimento di Scienze della Formazione, Psicologia, Comunicazione; i parr. 3, 4, 5, a Immacolata Brunetti dottore di ricerca del Dipartimento di Scienze della Formazione, Psicologia, Comunicazione e i parr. 6, 6.2, Conclusioni a Maria Brunetti, docente di scuola primaria.**

Formazione & Insegnamento XV – 2 – 2017
ISSN 1973-4778 print – 2279-7505 on line
doi: 107346/-fei-XV-02-17_19

1. Il pensiero computazionale

Computational thinking è il titolo di un'articolo pubblicato da Jeanette Wing (Wing, 2006) su Communication of ACM, nel quale si pone enfasi sul concetto di skills, indispensabili per tutti, non solo agli informatici. Questo modo di pensare viene proposto come quarta abilità di base, dopo la lettura, scrittura ed il calcolo da insegnare ai bambini. Già Saymond Papert(1996) introdusse la locuzione computational thinking dimostrando l'idoneità dell'insegnamento della matematica mediante la didattica della programmazione LOGO, proposta dal MIT, sulla base dei principi del costruzionismo. Dunque il computer risulterebbe essere un ottimo strumento per costruire gli artefatti cognitivi per mezzo del quale costruire rappresentazioni reali del mondo nel quale si interagisce. Successivamente viene declinato) come "is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent " (Cuny, Snyder, Wing, 2010).

In definitiva è un'attività mentale per la risoluzione di un problema che necessita di una soluzione computazionale ad opera sia di una macchina che dell'uomo oppure e sembrerebbe l'aspetto più importante dell'unione della macchina con l'uomo. La tecnologia deve sostenere la diffusione del pensiero computazionale per capire vari aspetti di un problema, valutare le diverse strategie computazionali applicando o adattando le diverse strategie in diversi usi. Gli studenti apprendono le strategie del pensiero computazionale mentre studiano e gli insegnanti riescono a modellare il pensiero verso queste strategie e adeguare misure di orientamento (K. Brennan, & M. Resnick, 2012). Questo è possibile perché il pensiero computazionale è astrazione, è un'abilità fondazionale, è un modo di pensare creativo, stimola il pensiero meta cognitivo, riguarda le idee, i concetti e gli approcci per risolvere un problema ma anche per gestire la propria vita. (Wilensky, 2001; Wilensky e Reisman,2006; Wilensky e Resnik, 1999; Kolodner et all, 2003; Puntambekar and Kolodner, 2005; Y. Kali and M.C. Linn, 2009). Il pensiero computazionale differisce dal pensiero matematico-critico perché pone nuove abilità nel risolvere un problema, usa tecniche per prove ed errori, iterazioni, il provare a caso ed infine è legato ad uno strumento (Barr e Conery, 2011).

L'idea della programmazione in se come linguaggio per imparare ad imparare non si è rivelata vincente; è, invece, vincente l'idea di legare gli oggetti e la programmazione come competenza contestualizzata generando nuovi ambienti che eliminano gli errori di sintassi e riducono la complessità del codice di programmazione (Linn, H.S. Lee, Tinker, Husic, and Chiu, 2006, Kafai, 2006; Kafai and Ching, 2001).

2. Il progetto in una scuola primaria

La proposta parte dall'influenza di due lavori nel campo della ricerca informatica educativa. Il primo focalizzato su un costruttivismo radicale che sostiene l'importanza di tale pensiero non per l'apprendimento del linguaggio informatico, della sintassi o della semantica, ma per constatare la programmazione come un processo creativo ed intelligente di sintesi e problem solving per la risoluzione di problemi realistici in contesti autentici (Greening, 2000). Il lavoro di Ben-Ari (2001), invece, si focalizza su un costruttivismo più moderato e personale poiché da importanza alla conoscenza pregressa nell'apprendimento della programma-

zione. Lo studente che incontra un nuovo sistema, costruisce un nuovo modello mentale indipendentemente dal fatto che questo venga insegnato o no. Diversi studi mostrano quanto gli studenti (agenti computazionali) imparano le strategie del pensiero computazionale mentre studiano e i docenti adeguano misure di orientamento. Dunque il nostro primo obiettivo verte sul modo in cui tali attività promuovono il pensiero computazionale e come valutare tali attività. La domanda seguente è stata come valutare tali attività purché non risulti la sperimentazione solo una utilizzazione sterile e mera applicazione di giochi interattivi. Dunque il processo valutativo verte sul prodotto e sul processo.

La ricerca ha un impianto qualitativo; si è avvalsa di strumenti qualitativi tra i quali le video-osservazioni della lezione, le interviste semistrutturate e il prodotto progettuale con Scratch. Scratch è un linguaggio di programmazione che usa blocchetti colorati che rappresentano le istruzioni per raccontare una storia interattiva, giochi e animazioni; è anche una community per condividere le creazioni ed esplorare quelle degli altri membri. Scratch è stato creato dal MIT Media Lab per studiare e valutare il pensiero computazionale. In particolare attraverso questo linguaggio è possibile sviluppare il pensiero computazionale, nell'ambito del costruzionismo, mediante tre dimensioni chiave individuate nello stesso:

- Concetti computazionali: i concetti che gli sviluppatori utilizzano quando programmano (sequenze, eventi, parallelismo, condizionali, operatori, dati);
- Pratiche computazionali: pratiche che si apprendono programmando (essere incrementali e iterativi, testing e debugging, riuso e remixing, astrazione e modularizzazione);
- Prospettive computazionali: i modi di vedere il mondo e se stessi che i ragazzi sviluppano grazie alla programmazione (esprimersi, connettersi, farsi domande).

2.1. Fasi della ricerca.

Il percorso proposto al gruppo classe e' stato quello presentato nel progetto Programma il Futuro, Corso 1 rivolto ai bambini che non sanno leggere; questo per agevolare gli alunni e non sovraccaricarli di difficoltà in quanto l'abilità di lettura e comprensione era in via di acquisizione. Abbiamo seguito le diverse lezioni proposte in piattaforma che gradualmente hanno offerto agli alunni la possibilità di impadronirsi dei concetti necessari ad affrontare il compito finale del percorso: costruire una storia utilizzando dei comandi a blocchi per muovere e far parlare i personaggi utilizzando il programma scratch.

La ricerca si è articolata in tre step:

1. Fase 0: in questa fase ha avuto inizio l'esplorazione del coding a partire dal 7 Dicembre, Ora del codice, data stabilita dal Miur per iniziare il programma in tutta Italia. L'intero percorso si è articolato in ore di lavoro-gioco in laboratorio informatico, e ore di lezione tradizionale in palestra o in giardino. Gli alunni prima che al pc, hanno sperimentato fisicamente con esercizi di gioco l'attività del programmare movimenti e azioni come se loro stessi fossero il famoso cane o gatto del programma e un compagno di volta in volta diverso fosse il programmatore che scrive il codice. Allo step teorico/tradizionale svolto in palestra si è associato lo step pratico con l'ausilio del pc. Ciascun bambino ha realizzato la propria storia al PC; l'insegnante ha raccolto l'idea e le parole più coinvolgenti e chiamando l'autore di quelle particolari parole o

movimenti, ha chiesto di riprodurle su un altro PC. Questa operazione si è ripetuta fino a che abbiamo creato ciò che piaceva a tutti raccogliendo i suggerimenti di tutti. Alla fine è venuto fuori un programma di una decina di secondi dal tema: "Stare bene insieme anche nella differenza". Il lavoro si è svolto attraverso le seguenti caratterizzazioni:

- Decomposizione dei problemi in piccole parti facilmente risolvibili,
- Astrazione: semplificare e generalizzare i problemi,
- Negoziazione: lavorare in gruppi e poi lavorare per integrare le soluzioni in un tutto,
- Costruzione del consenso: lavorare per avere l'appoggio del gruppo attorno a un'idea.

2. Fase 1: individuazione e descrizione dei concetti, delle pratiche e delle prospettive computazionali individuate dal MIT Media Lab mediante l'utilizzo di Scratch.
3. Fase 2: valutazione di queste dimensioni attraverso la loro analisi; somministrazione delle interviste semistrutturate (Bichi, 2002) tese ad individuare le reazioni comportamentali degli studenti, le pratiche che si apprendono programmando, ovvero l'essere incrementali, iterativi, testing o debugging, riuso e remixing, astrazione e modularizzazione e le prospettive computazionali che ognuno di noi ha per organizzare la visione di se stessi e del mondo sviluppate grazie alla programmazione, come ad esempio l'esprimersi, il connettersi ed il farsi domande. Infine dati raccolti sulla riflessione dell'azione da parte del docente stesso e integrazione di queste con gli indicatori della trasposizione didattica (logica valoriale, didattica e dell'apprendimento)(Damiano, 2007; Altet, 2003; Develay,1995; Festermacher, 1986; Shulman, 1986).

3. Analisi dei dati: i concetti del pensiero computazionale

Quando si progetta un media interattivo con Scratch, entrano in gioco una serie di concetti computazionali, comuni a molti linguaggi e identificati come mappatura di blocchi di programmazione: sequenze, loop, parallelismo, eventi, condizionali, operatori e dati. Per ogni concetto si cercherà di spiegare i passi effettuati.

Concetto: sequenza
Un concetto fondamentale nella programmazione è che una particolare attività o compito viene espresso in una serie di singole fasi o istruzioni eseguite dal computer, come ad esempio una ricetta o una serie di istruzioni di programmazione che specificano un comportamento o un'azione. Ad esempio, un oggetto può essere programmato per muoversi a breve distanza attraverso una sequenza di istruzioni.

Concetto: loop
Nell'esempio precedente l'oggetto viene programmato per spostarsi e attendere alcuni secondi. Nel concetto di loop invece l'oggetto deve spostarsi e attendere molte più volte. Un loop è un meccanismo di istruzione di esecuzione della stessa sequenza più volte, ovvero una sequenza di istruzioni.

Concetto: eventi
Un evento, inteso come qualcosa che causa un'altra cosa, è molto importante nei media interattivi. Un esempio può essere il tasto di avvio di un video musicale oppure la collisione di due oggetti aumentano il punteggio di un gioco o ancora premendo il tasto spazio l'oggetto si muove nello spazio in su o in basso oppure cliccando il mouse verrà visualizzato il fumetto del personaggio per alcuni secondi.

Concetto: parallelismo
La maggior parte dei linguaggi di programmazione moderni supportano il parallelismo, ovvero sequenze di istruzioni che accadono allo stesso tempo. Scratch supporta il parallelismo tra gli oggetti ad esempio, una scena di festa da ballo potrebbe coinvolgere diversi personaggi che ballano contemporaneamente, ognuno con una sequenza unica di istruzioni di danza. Scratch supporta anche il parallelismo all'interno di un singolo oggetto. Nel nostro caso sperimentale mentre il cane si avvicina, il gatto dice ciao.

Concetto: condizionale
Un altro concetto chiave nei media interattivi è il condizionale, ovvero la capacità di prendere decisioni in base a determinate condizioni come espressione di molteplici risultati. Un esempio è stato utilizzare il blocco if per determinare la visibilità di un oggetto.

Concetto: operatore
Gli operatori permettono al programma di eseguire manipolazioni numeriche tra cui l'addizione, la sottrazione, la divisione e le operazioni di stringa come la concatenazione e la lunghezza delle stringhe.

Concetto: dati
Il dato comporta la memorizzazione, il recupero e l'aggiornamento dei valori. Scratch offre attuamente due contenitori per i dati: la variabile che può mantenere un singolo numero o una stringa e le liste che possono mantenere un insieme di numeri o stringhe. Calcolare il punteggio in un gioco è una motivazione frequente per i giovani designer per esplorare le variabili.

4. Le pratiche del pensiero computazionale

Per avere un quadro chiaro ed esaustivo del pensiero computazionale è necessario ottenere altri dati che corrispondono alle pratiche progettuali, ovvero al processo del pensare e dell'imparare, dunque all'andare oltre ciò che si sta imparando. Le dimensioni chiave delle pratiche sono state raccolte con l'aiuto delle interviste tese ad identificare appunto i processi di costruzione. Ci sono 4 gruppi principali di pratiche:

1. Incrementali e iterativi;
2. Testing e debugging;
3. Riutilizzo e remix
4. Astrazione e modularizzazione.

Pratica: Essere incrementale e iterativo
Progettare un media interattivo non è un processo privo di errori e sequen-

ziale poiché è necessario prima di tutto identificare un concetto per delineare il progetto, poi sviluppare un piano per la progettazione, e poi attuare il progetto in codice. È un processo adattivo perché la progettazione iniziale potrebbe cambiare in risposta all'avvicinarsi della soluzione che si svolge a piccoli passi. Nelle interviste si nota fortemente l'essere iterativi attraverso il desiderio di immaginazione e costruzione. Quindi giocare e creare, provare, ed infine sviluppare ulteriormente, in base alle proprie esperienze e nuove idee.

Pratica: test e debug
Le cose raramente (se non mai) funzionano esattamente come le immaginiamo; è fondamentale per i progettisti sviluppare strategie per affrontare se non anticipare i problemi. Nelle video-osservazioni, gli alunni descrivono le loro diverse pratiche provando e riprovando attraverso tentativi ed errori o il sostegno del docente.

Pratica: riutilizzo e remix
Questa pratica è stata possibile poiché i bambini hanno adattato alla loro storia e soprattutto alle loro idee, i personaggi presi dalle linee guida che il Miur ha identificato in questo programma (Code.org) su ciò che la comunità di Scratch condivide.

Pratica: astrazione e modularizzazione
L'astrazione e la modularizzazione sono pratiche importanti per la progettazione di un problem solving. In Scratch, gli alunni dovevano progettare una situazione problematica o storia, mettendo insieme piccole parti aventi ciascuno un obiettivo da raggiungere. I progettisti impiegano l'astrazione e la modularizzazione a più livelli ovvero dal lavoro iniziale di concettualizzazione del problema al passo successivo di traduzione del concetto in pile di codici che definiscono comportamenti o azioni. Ci può essere la pila di codice che controlla il movimento dell'oggetto sullo schermo; la seconda pila di codice controlla l'aspetto visivo dell'oggetto; la terza pila di codice controlla i vari eventi associati con gli ostacoli, come ripristinare il livello se l'oggetto collide con un pericolo. In questa fase gli alunni si pongono dal punto di vista di chi legge.

5. Le prospettive del pensiero computazionale

Nelle interviste abbiamo voluto rilevare l'evoluzione nella comprensione di se stessi, nelle relazione con gli altri e con il mondo tecnologico che li circonda, ovvero i cambiamenti di prospettiva.

- La prima prospettiva la definisco di espressione perché gli alunni hanno visto questa strategia di pensiero come un mezzo per creare giocando: "posso giocare e creare"; "scrivere la storia" ed infine "mi piaceva quando programmavo di eseguire".
- La seconda prospettiva la definisco di collaborazione poiché la totalità dei bambini ha risposto che è piaciuto molto collaborare e ricevere aiuto da altri compagni.

Questo metodo ha avuto esiti favorevoli perché ha rafforzato l'impegno di ciascuno a superare le difficoltà che di volta in volta rendevano l'esercizio "difficile" tanto quanto lo sono i giochi ai quali i nostri bambini sono abituati, questo

per fare in modo che l'attenzione e la concentrazione non calassero; inoltre ha alimentato la capacità collaborativa del gruppo classe in una situazione di gioco altamente accattivante.

6. La valutazione del pensiero computazionale

Dopo aver analizzato il pensiero computazionale mediante i concetti, le pratiche e le prospettive, si è cercato di effettuare un'analisi valutativa sul processo di insegnamento della pratica computazionale attraverso alcuni indicatori della trasposizione didattica. Riporto di seguito due episodi della lezione tenuta dal docente in aula tecnologica.

1. I: Me ciao gatto spiegalo
2. A: Io come faccio a scrivere?
3. I: Nicolo hai risolto?…….Ciao gatto……Martina lì scrivi e lì lo porti
4. A: Maestra io ho capito come bisogna fare la storia……..bisogna fare che se vuoi prendere un altro personaggio devi…fare di nuovo.
5. I: Spiega spiega
6. A: O mostragli fare oppure puoi scrivere nascondi
7. I: Ok…se lo vuoi fare sparire ovviamente
8. A: Andrea era troppo facile.
9. A2: dove devo metterlo?
10. A1: devi metterlo la. È facilissimo.
11. I: Pierpaolo che storia stai facendo?
12. A: sto ancora
13. I: quanti ne stai mettendo? È una partita di pallone?
14. A: no
15. I: hai messo lo sfondo con un campo di calcio?
16. A: no. Ah! devo mettere uno sfondo allora!
17. I: ma chiara glielo devi anche spiegare altrimenti lui non capisce e non lo saprà fare da solo. Spiegagli come si fa.
18. A: devi mettere……
19. I: dove? Ma perché devi cliccare Gaetano! Clicca dentro…prima avrai cliccato senza accorgertene….dai Gabri scervellati!!
20. A: il gatto arancione…..dai Marti devi aggiungere una cosa.
21. I: ricomincia perché probabilmente Chiara non hai eseguito bene la consegna. Hai letto bene la consegna? Cosa dice?
22. A: dice…fai arrivare il cane fino al gatto, fai dire ciao al gatto quando il cane arriva la
23. I: quindi esegui, prova a fare esegui, Clicca su esegui, ricomincia…il cane non è arrivato al gatto ma tu hai spostato il gatto, poi fai dire ciao al gatto quando il cane arriva……stai attenta. Tu hai fatto prima quando arriva il cane, invece prima devi fare arrivare il cane. Dopo.. ..Il blocco prima dice quando arriva il cane e il cane, vedi. C'è qualcosa che non va. Allora……elimina quei blocchi e ricomincia. Leggi bene la successione delle azioni.
24. A: fai arrivare il cane fino al gatto e fai dire ciao al gatto quando il cane arriva là.
25. I: e Vai!
26. A: ma questo lo devo mettere in mezzo maestra?
27. I: e penso proprio di si. Riprova c'è qualcosa che non va. A te come va Gaetano? Gabri ci stai riuscendo?
28. A:….
29. I: bravo!

In queste battute l'insegnante apre e chiude l'episodio. L'obiettivo del docente è la creazione della storia mediante la programmazione. Nelle battute dal numero 11 al numero 14 si evidenzia l'invito del docente alla verbalizzazione della storia incoraggiando l'alunno al ragionamento. Si evince la logica valoriale basata sulla ragione. Dalle prime battute si evidenzia l'impostazione della lezione basata sul porsi le domande. Alla battuta 5 e alla battuta 17 , c'è una chiara evidenza dell'invito della docente alla collaborazione e alla partecipazione di tutti gli allievi ma anche ad un implicito riferimento alla promozione della zona di sviluppo prossimale. Ciò dimostra una tendenza verso valori democratici e partecipativi. Dalla battuta 19 alla battuta 21 l'insegnante funge da guida, finché alla battuta 23 esplicita le direzioni da seguire mettendo in evidenza la domanda della consegna. In questo episodio si evidenzia molto chiaramente la logica didattica dell'insegnante che guida seppur in maniera molto marginale verso la risoluzione del problema ponendo domande indirizzanti verso il suo obiettivo. La logica dell'apprendimento si basa sulla costruzione della conoscenza ed anche sullo sviluppo del pensiero logico e computazionale.

6.1. Prospettive valutative

La piattaforma dava l'opportunità di valutare il prodotto finale attraverso l'attribuzione di punteggi per avere una chiara visione d'insieme, ma non ha offerto la possibilità di valutare la correttezza dei nodi concettuali interessati alla comprensione della lettura degli stessi. È opportuno, infatti, esprimere una valutazione delle competenze quando lo studente dimostra il possesso di un sapere che è utilizzato in contesti reali e complessi in modo preciso e pertinente. In tale processo tutte le risorse del soggetto vengono orchestrate e mobilitate per produrre soluzioni efficaci. Quindi la valutazione deve fare riferimento alle risorse possedute, alle strutture di interpretazione (modelli con cui il soggetto interpreta determinate situazioni) e alle strutture di autoregolazione (modi con cui riflette sulla proprie interpretazioni) (Trinchero, 2012). Resnick e Brennan nella progettazione di Scratch, hanno proposto diversi approcci per valutare il prodotto finale in relazione alle competenze (Brennan K., &Resnich, 2012). Seguendo, invece, la procedura del progetto "Programma il Futuro" , non è possibile trasferire l'acquisizione di tali attività in altri contesti, poiché se da un lato è possibile applicare facilmente gli esercizi, dall'altro si nega quella flessibilità richiesta dall'azione educativa.

Così abbiamo pensato di relazionare gli obiettivi di apprendimento all'acquisizione delle competenze di base legate al pensiero computazionale (tab 1).

Obiettivi di apprendimento	Processi	Abilità di base con il pc
Leggere e ricavare informazioni da istruzioni	Procedure ed algoritmi. Collezione ed analisi dei dati. (Dati)	Decodificare istruzioni e creare istruzioni con linguaggio naturale e/o con linguaggio visuale a blocchi
Impiegare alcune regole del disegno tecnico per rappresentare semplici oggetti. Riduzione della complessità per fare emergere solo alcuni aspetti e tralasciandone altri.	Rappresentazione di dati; Riconoscimento di modelli e astrazione. (astrazione)	Rappresentare le soluzioni in forma sequenziale che sia effettivamente eseguibile e riproducibile.
Rappresentare i dati dell'osservazione attraverso tabelle, mappe, diagrammi, disegni, testi.	Scomposizione di un problema	Rappresentare anche attraverso l'uso di strumenti tecnologici dopo aver riconosciuto modelli, compiuto processi di astrazione, mappato un fenomeno complesso scomponendolo
Prevedere le conseguenze di decisioni o comportamenti personali. Individuare un metodo che raggiunga un risultato	Calcolabilità e complessità	Uso di diagrammi di flusso per rappresentare sequenze di azioni e conseguenze; trasformare decisioni o comportamenti in storie: storytelling e coding
Immaginare possibili miglioramenti. Modellare un processo ed eseguire esperimenti per individuare problemi/errori e correggerli.	Procedure e algoritmi (simulazione test e debugging)	Analisi metacognitiva di errori con la programmazione visuale a blocchi
Cambiare punto di vista: mettersi dalla parte del lettore	Scomposizione di un problema nei suoi elementi di astrazione e modularizzazione: divisione del problema in piccole parti.	Analisi metacognitiva del problema con la programmazione visuale a blocchi.

Tab.1. Obiettivi di apprendimento e pratiche del pensiero computazionale.

Conclusioni

Pochi hanno mostrato interesse incostante per l'alternanza di momenti "facili " ed altri " difficili". La maggior parte ha mostrato interesse, curiosità, e partecipazione ed ha contribuito alla risoluzione di alcuni quesiti di tipo tecnico da sciogliere. In generale ciascuno in opera davanti al proprio PC è riuscito ad acquisire i primi strumenti di pensiero computazionale per programmare una piccola storia che non solo doveva rispondere ai criteri della consegna ossia far muovere il cane e il gatto per far dire qualcosa, ma doveva dare un senso compiuto e piacevole alla storia stessa utilizzando i pochi elementi a disposizione in piattaforma; in particolare sfondo e suono. Gli alunni hanno compreso che solo se il codice(cioè la successione di comandi) è breve, semplice, e chiara l'azione sarà immediata.

Alla classe è piaciuto molto il risultato finale soprattutto per la sorpresa di aver costruito personalmente un piccolo programma. Gli inconvenienti sono stati tanti soprattutto quello legato alla sottomissione del prodotto al concorso.

In ogni caso la sperimentazione ha sortito una ricaduta positiva lasciando negli alunni la curiosità di continuare il percorso il prossimo anno scolastico.

Didatticamente parlando si ritiene che si possano fare diverse osservazioni al riguardo:

Il pensiero computazionale è un processo mentale per la risoluzione di problemi costituito dalla combinazione di metodi caratteristici e strumenti intellettuali, entrambi di valore generale. Questa generalità è il motivo principale del perché insegnare coding (Wing, 2006, pp.33-35).

I metodi caratteristici includono:

- Analizzare e organizzare i dati del problema in base a criteri logici;
- Rappresentare i dati del problema tramite opportune astrazioni;
- Formulare il problema in un formato che ci permette di usare un "sistema di calcolo" (nel senso più ampio del termine, ovvero una macchina, un essere umano, o una rete di umani e macchine) per risolverlo;
- Automatizzare la risoluzione del problema definendo una soluzione algoritmica, consistente in una sequenza accuratamente descritta di passi, ognuno dei quali appartenente ad un catalogo ben definito di operazioni di base;
- Identificare, analizzare, implementare e verificare le possibili soluzioni con un'efficace ed efficiente combinazione di passi e risorse (avendo come obiettivo la ricerca della soluzione migliore secondo tali criteri);

Generalizzare il processo di risoluzione del problema per poterlo trasferire ad un ampio spettro di altri problemi.

Questi metodi sono importanti per tutti, non solo perché sono direttamente applicati nei calcolatori (computer), nelle reti di comunicazione, nei sistemi e nelle applicazioni software ma perché sono strumenti concettuali per affrontare molti tipi di problemi in diverse discipline.

Gli strumenti intellettuali includono:

- Confidenza nel trattare la complessità
- Ostinazione nel lavorare con problemi "difficili";
- Tolleranza all'ambiguità (da riconciliare con il necessario rigore che assicuri la correttezza della soluzione);
- Abilità nel trattare con problemi definiti in modo incompleto;
- Abilità nel trattare con aspetti sia umani che tecnologici, in quanto la dimensione umana (definizione dei requisiti, interfacce utente, formazione, ...) È essenziale per il successo di qualunque sistema informatico;

Capacità di comunicare e lavorare con gli altri per il raggiungimento di una meta comune o di una soluzione condivisa.

Riferimenti bibliografici

Altet, M. (2003), *La ricerca sulle pratiche di insegnamento in Francia*. Brescia: La Scuola.

Barr, D., Harrison, J. and Conery, L. (2011). Computational thinking: A digital age skill for everyone. Learning &amp. *Leading with Technology* 38(6), 20–52.

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20(1), 45–73.

Bichi, R. (2002). *L'intervista biografica. Una proposta metodologica*. Milano: Vita e Pensiero.

Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. *Paper presented at annual American Educational Research Association meeting*. Vancouver, BC: Canada.

Cuny, J., Snyder, L., and Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. *Unpublished manuscript in progress, referenced*. In http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf.

Damiano, E., (2007). *L'insegnante etico. Saggio sull'insegnamento come professione morale*. Assisi: Cittadella.

Develay, M. (1995). *Savoir scolaires et didactique des disciplines*. Paris: ESF.

Fenstermacher, G. (1986). Philosophy of research on teaching. Tree aspect. In Wittrock M. C. (ed). *Handbook of research on teaching*, 3. New York: Macmillan.

Greening, T. (2000). Emerging Constructivist Forces in Computer Science Education: Shaping a New Future? *Computer science education in the 21st century*, 47–80. Berlin: Springer.
Kafai, Y. B. (2006). Playing and Making Games for Learning: Instructionist and Constructionist Perspectives for Game Studies. *Games and Culture,* 1(1), 36-40.
Kali, Y. and Linn, M. C. (2009). Designing Effective Visualizations for Elementary School Science. *Elementary School Journal* 109(5), 181-198.
Kolodner, J. L. et al. (2003). Problem-Based Learning Meets Case-Based Reasoning in the Middle-School Science Classroom: Putting Learning by Design into Practice. *Journal of the Learning Sciences,* 12(4), 495-548.
Linn, M. C., Lee, H. S., Tinker, R., Husic, F. and J. L. Chiu (2006). Teaching and Assessing Knowledge Integration in Science. *Science* 313, 1049-1050.
Papert, S. (1975). Teaching Children Thinking. *Journal of Structural Language*, 4, 219-29.
Papert, S. (1981). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.
Puntambekar, S. and Kolodner, J. L. (2005). Toward Implementing Distributed Scaffolding: Helping Students Learn Science by Design. *Journal of Research in Science Teaching,* 42(2), 185-217.
Kafai, Y. B. and Ching C. C. (2001). Affordances of Collaborative Software Design Planning for Elementary Students' Science Talk. *Journal of the Learning Science,* 10(3), 323-363.
Shulman, S. L. (1986). Those who understand: Knowledge Growth in teaching. *Educational Researcher*, 15(1).
Trinchero, R. (2012). *Costruire, valutare, certificare competenze. Proposte di attività per la scuola*. Milano: Franco Angeli.
Wilensky, U. and Resnick, M. (1999). Thinking in Levels: A Dynamic Systems Approach to Making Sense of the World. *Journal of Science Education and Technology,* 8(1), 3-19.
Wilensky, U. (2001). Modeling Nature's Emergent Patterns with NetLogo. *Proceedings of the Eurologo 2001 Conference*. Linz, Austria.
Wilensky, U. and Reisman, K. (2006). Thinking Like a Wolf, a Sheep, or a Firefly: Learning Biology Through Constructing and Testing Computational Theories—an Embodied Modeling Approach. *Cognition and Instruction*, 24(2), 171-209;
Wing, J. M. (2006). Computational Thinking. *Communications of the ACM,* 49(3), 33-35.