Bari, 13-15 ottobre 2023

CONVEGNO ITALIANO SULLA DIDATTICA DELL'INFORMATICA

[ITADINFO]

Metodi e Pratiche Didattiche Innovative
Ricerca Scientifica
Esperienze
Laboratori Formativi

SPONSOR

CON IL PATROCINIO DI

Gentile E., Monga M. (a cura di).
[**ITADINFO**] *Convegno Italiano sulla Didattica dell'Informatica*. Bari, 13-15 ottobre 2023.

Prima edizione: ottobre 2023

Edito da: Università degli Studi di Bari Aldo Moro
www.uniba.it

ISBN: 978-88-6629-075-9

Gli articoli qui presentati sono stati tutti oggetto di valutazione interna.

COMITATO ORGANIZZATORE

Presidenza

Enrico Nardelli, Università di Roma "Tor Vergata"
Direttore del Laboratorio CINI

Veronica Rossano, Università di Bari

Programma Scientifico

Mattia Monga, Università di Milano

Sponsorizzazioni

Tullio Vardanega, Università di Padova

Relazioni Istituzionali

Enrico Nardelli, Università di Roma "Tor Vergata"

Iscrizioni

Francesco Lacchia, APS Programma il Futuro

Sito Web

Michael Lodi, Università di Bologna

Organizzazione Locale

Enrica Gentile, Università di Bari

Paola Plantamura, Università di Bari

Atti del convegno

Enrica Gentile, Università di Bari

Mattia Monga, Università di Milano

COMITATO SCIENTIFICO
E DEI REVISORI

PERCHÉ QUESTO CONVEGNO

La necessità di insegnare l'Informatica nella scuola è ormai un fatto riconosciuto in tutto il mondo. L'Informatica è essenziale per dotare i giovani delle competenze necessarie per partecipare attivamente alle nostre società tecnologiche e sempre più digitali, in modo responsabile e sicuro. Diversamente da altre discipline scientifiche di più antica tradizione, che hanno avuto decenni e secoli per creare e raffinare metodi e tecniche per l'insegnamento nella scuola dei loro concetti fondamentali, l'Informatica affronta questo problema relativamente da pochi anni. Ecco, quindi, che diventa ancora più importante la necessità della ricerca e della condivisione di risultati e buone pratiche.

Questo è l'obiettivo del primo convegno italiano dedicato a questo tema. Docenti di scuola e ricercatori universitari sono stati invitati a partecipare per discutere i vari approcci e scambiarsi opinioni, allo scopo di far progredire lo stato dell'arte relativo alla Didattica dell'Informatica nella scuola.

Il convegno ha previsto sessioni dedicate alla discussione di esperienze sul campo realizzate da insegnanti, alla presentazione di risultati di ricerca e alla formazione laboratoriale interattiva.

SESSIONE PLENARIA

INFANZIA

PRIMARIA

SECONDARIA DI PRIMO GRADO

SECONDARIA DI SECONDO GRADO

UNIVERSITÀ

ALTRI CONTRIBUTI

Di cosa parliamo quando parliamo di informatica

Violetta Lonati
Dipartimento di Informatica
Università degli Studi di Milano
lonati@di.unimi.it

Abstract

Il mondo della scuola si sta ormai convincendo che l'informatica debba avere un ruolo sempre più rilevante nella formazione di una cittadinanza consapevole e capace. Non è scontato però trovarsi d'accordo, negli ambiti scolastici e nei contesti dove si decidono le politiche scolastiche, su cosa sia questa "Informatica":

- nella scuola del primo ciclo l'informatica entra spesso in forma di "coding" o di "robotica educativa", attraverso ambienti di programmazione visuale che consentono di programmare il comportamento di personaggi su uno schermo o di piccoli automi nello spazio fisico;
- nella scuola secondaria di secondo grado, l'informatica è inclusa solo in alcuni percorsi formativi, quasi sempre come materia tecnica e professionalizzante;
- in certi documenti ministeriali recenti la parola "informatica" quasi non si trova più — avendo lasciato il posto all'espressione, forse più vaga, ma senz'altro più attraente, "pensiero computazionale" — oppure è utilizzata come sinonimo per intendere l'utilizzo delle tecnologie e dei dispositivi digitali (per cui sarebbe più appropriata l'espressione "competenze digitali").

In questo intervento proponiamo di tornare a pensare all'informatica come a quella "disciplina scientifica che studia i principi e i metodi per l'elaborazione automatica dell'informazione". Ripartiremo da alcune parole chiave (algoritmo, programma, dati) —e da attività didattiche, strumenti e materiali ad esse ispirati— per riflettere su alcune delle "idee fondanti" dell'informatica e sul ruolo che queste possano avere in ambito formativo. Siamo convinti infatti che la comprensione di queste idee informatiche sia molto più significativa della sola acquisizione di competenze operative o tecnologiche, costituendo invece il presupposto imprescindibile per formare cittadine e cittadini in grado di prendere decisioni informate su questioni personali e sociali connesse al digitale.

Ce lo chiede l'Europa!

Enrico Nardelli
Università di Roma "Tor Vergata"
Direttore Laboratorio Nazionale CINI "Informatica e Scuola"
Presidente di Informatics Europe
nardelli@mat.uniroma2.it

Abstract

In questo intervento presentiamo il Quadro di riferimento per l'informatica nella scuola elaborato dalla coalizione "Informatics for All".

Partendo dalla constatazione che alla base di ogni processo di trasformazione digitale vi deve essere la conoscenza della disciplina scientifica che lo rende possibile, la coalizione persegue l'obiettivo di far sì che l'informatica sia vista come disciplina essenziale nella formazione scolastica.

A questo scopo, la coalizione ha preparato un quadro di riferimento comune europeo per l'insegnamento dell'informatica nella scuola[1]. Esso offre una guida di alto livello, che chi definisce i curricoli scolastici dovrebbe tener presente per un corretto approccio all'insegnamento dell'informatica.

Il nucleo di questo quadro di riferimento è concepito come un insieme di aree tematiche fondamentali con le relative pratiche informatiche in cui ci si aspetta che tutti gli alunni siano competenti entro il termine dell'istruzione secondaria superiore.

La proposta è volutamente sintetica e breve, per fornire un insieme minimo di requisiti comuni di alto livello e lasciare spazio alle comunità nazionali per ricavare curricoli specifici e concreti. In ogni Paese essi dovranno essere definiti tenendo conto di tradizioni, lingua, cultura e, in particolare, la sinergia con lo sviluppo delle competenze digitali di base e l'uso dell'informatica in altre materie.

Tuttavia, fornire un comune riferimento condiviso in tutta Europa, è estremamente prezioso per aiutare l'accettazione dell'informatica come materia fondamentale per insegnamento nella scuola..

[1] Il quadro di riferimento è consultabile al link: https://www.informaticsforall.org/the-informatics-reference-framework-for-school-online-it/

 ISBN: 978-88-6629-075-9

INFANZIA

PRIMARIA

SECONDARIA DI PRIMO GRADO

Apprendere la programmazione in un micromondo musicale

Gabriele Pozzan, Costanza Padova, Chiara Montuori, Barbara Arfè, and Tullio Vardanega

Università degli studi di Padova
{gabriele.pozzan,costanza.padova,chiara.montuori}@phd.unipd.it
{barbara.arfe,tullio.vardanega}@unipd.it

Abstract

Questo lavoro dettaglia una esperienza di apprendimento di concetti base della programmazione condotta in due scuole medie della città di Padova durante l'anno scolastico 2022/2023. I partecipanti coinvolti avevano poca o nessuna familiarità con concetti di programmazione. Abbiamo basato la sperimentazione su un micromondo a tema musicale con il quale studentesse e studenti hanno potuto interagire tramite un linguaggio di programmazione a blocchi. Il micromondo si è rivelato adatto ad aiutare l'apprendimento del concetto di iterazione ma è stato meno immediato per costruire esercizi adatti all'apprendimento delle istruzioni condizionali.

1 Introduzione

Un tipico ambiente di apprendimento per i concetti fondamentali dell'Informatica propone attività di programmazione con *linguaggi a blocchi* in cosiddetti *micromondi*.

I linguaggi di programmazione a blocchi sono utili per programmatori principianti in quanto eliminano il rischio di errori sintattici e offrono un supporto visivo che può ridurre il carico cognitivo di chi apprende, ad esempio assegnando colori diversi a diverse funzionalità (per altri esempi cfr. [1]).

I micromondi sono rappresentazioni digitali di sistemi con cui è possibile interagire tramite programmazione. Un classico esempio di micromondo consiste di problemi di navigazione in uno spazio bidimensionale: gli utenti programmano i movimenti di un robot attraverso una mappa, con l'obiettivo di raggiungere una destinazione prefissata senza colpire eventuali ostacoli (ad esempio cfr. il corso *Express* di Code.org[1]). Le attività condotte in un micromondo possono essere più o meno strutturate. Le attività meno strutturate incoraggiano la creatività e l'esplorazione libera dei costrutti di programmazione (per esempio, gli ambienti di Scratch[2] e Alice[3]); le attività strutturate danno precise condizioni di partenza e obiettivi da raggiungere: qui ci concentriamo su questa seconda tipologia (per una rassegna approfondita di problemi ben strutturati cfr. [2]).

In questo lavoro discutiamo gli esiti di una esperienza didattica condotta in due scuole medie della città di Padova, durante l'anno scolastico 2022/2023. Per questa sperimentazione abbiamo creato un micromondo a tema musicale, programmabile con linguaggio a blocchi.

Il resto dell'articolo è strutturato in questo modo: la Sezione 2 descrive il linguaggio di programmazione e il micromondo, la Sezione 3 la sperimentazione e le relative attività, la Sezione 4 i risultati ottenuti; la Sezione 5 le conclusioni tratte dall'esperienza.

[1] https://studio.code.org/s/express-2023
[2] https://scratch.mit.edu/
[3] https://www.alice.org/

2 Il micromondo musicale

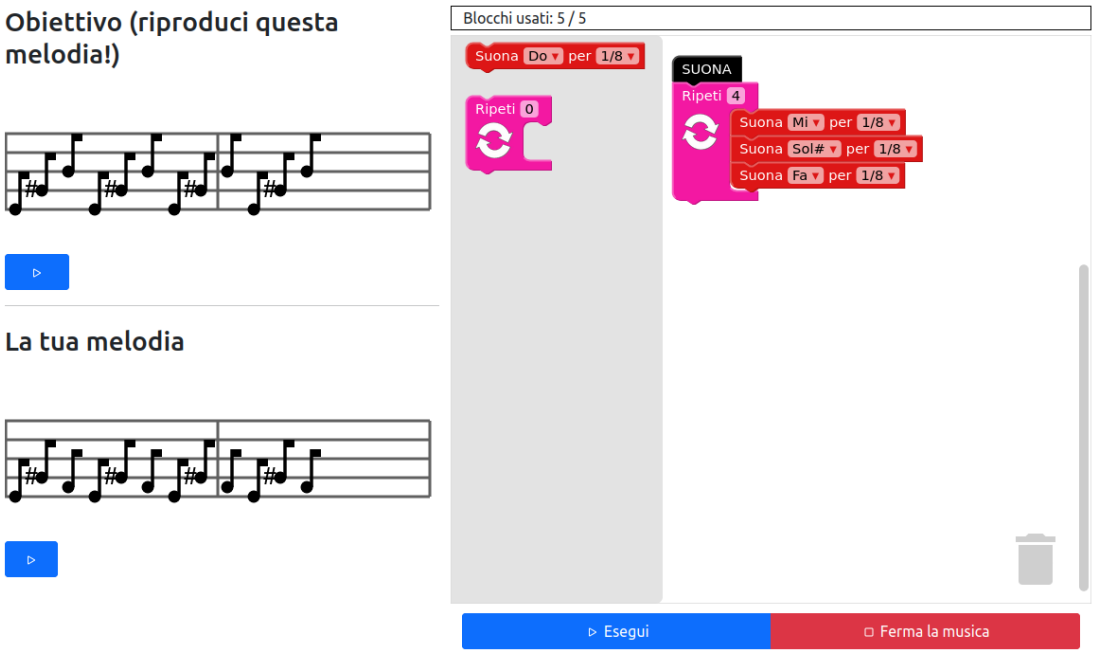


Figura 1: Esempio di esercizio del micromondo musicale

La Figura 1 mostra un esempio del micromondo musicale oggetto di questo studio[4]: ogni esercizio presenta una melodia *obiettivo* e fornisce un insieme di blocchi di codice da comporre con lo scopo di riprodurla.

Gli utenti possono ottenere informazioni su nome e durate delle note posizionando il cursore sulla loro rappresentazione nello spartito. Inoltre, possono ascoltare la melodia premendo il pulsante *Play* sottostante allo spartito.

Una volta composto un programma, gli utenti possono *eseguirlo*: questo fa sì che la melodia generata venga riprodotta in audio e trascritta sullo spartito nella sezione "La tua melodia".

Il micromondo considera una soluzione *corretta* se questa riproduce esattamente la melodia obiettivo: l'uso di specifici costrutti di programmazione (iterazione, condizionale) è incoraggiato ma non obbligatorio.

Ogni esercizio è basato su diversi costrutti di programmazione, a seconda del problema da risolvere. La Tabella 1 elenca tutti i costrutti disponibili complessivamente nel micromondo.

Per questa sperimentazione, abbiamo predisposto un totale di 46 esercizi che coprono tutti i costrutti elencati nella Tabella 1.

3 La sperimentazione

La sperimentazione ha coinvolto due classi di seconda media frequentanti due scuole della città di Padova, per un totale di 42 studentesse e studenti ($F = 15$ $M = 27$). Alcuni partecipanti

[4]Codice sorgente disponibile su GitHub https://github.com/cornacchia/blockly-music-microworld

Costrutto	Funzione
`Suona(n,d)`	Suona la nota `n` per la durata `d`.
`Pausa(d)`	Pausa di durata `d`.
`CambiaTono(t)`	Cambia tonalità delle note successive in `alta` o `bassa`.
`Ripeti(n){istruzioni}`	Ripeti `n` volte una sequenza di una o più `istruzioni`.
`SeIterazione(i){istruzioni}`	Esegue determinate `istruzioni` solo se l'iterazione corrente è la `prima`, l'`ultima`, `pari` o `dispari`.
`SeTono(t){istruzioni}`	Esegue determinate `istruzioni` solo se il tono corrente è `alto` o `basso`.
`DefinisciProcedura(nome){istruzioni}`	Definisce una procedura che esegua determinate `istruzioni`.
`ChiamaProcedura(nome)`	Esegue le `istruzioni` della procedura chiamata per `nome`.

Tabella 1: Lista di costrutti di programmazione utilizzabili nel micromondo musicale

erano stati precedentemente esposti a concetti di programmazione in ambiente Scratch durante un laboratorio estivo facoltativo.

Nell'arco di circa un mese le due classi hanno separatamente partecipato a un totale di 6 incontri, di un'ora ciascuno, in cui hanno avuto la possibilità di interagire con esercizi del micromondo musicale. La Tabella 2 elenca le tematiche trattate durante ogni incontro.

Incontro	N. esercizi	Temi
1	6	Sequenze di note, cambi di tonalità
2	8	Iterazione
3	8	Iterazione e correzione
4	8	Istruzioni condizionali
5	8	Condizionali e procedure
6	8	Ripasso di tutti i concetti

Tabella 2: Tematiche dei sei incontri di preparazione

Durante ogni incontro di preparazione, gli studenti affrontavano autonomamente una sequenza di esercizi, con la possibilità di chiedere supporto agli istruttori (autori di questo lavoro e assistenti) che, per protocollo, non davano la soluzione ma cercavano di stimolare il ragionamento.

I partecipanti potevano tentare un esercizio tante volte quante necessarie per superarlo e, eventualmente, anche passare ad un esercizio successivo senza aver risolto quello corrente.

4 Risultati

La Tabella 3 mostra un aggregato di dati prestazionali, valori medi e deviazione standard (DS), per ogni incontro. La colonna *Risolto* mostra quanti studenti, in media, hanno superato un

esercizio dopo averlo tentato almeno una volta; la colonna *Numero tentativi* mostra i tentativi necessari, in media, per risolvere un esercizio. La Figura 2 mostra la distribuzione delle percentuali di superamento degli esercizi, evidenziando i valori medi con quadrati rossi.

Incontro	**Risolto**	**Numero tentativi**
1	90,56% (DS=4,49)	2,57 (DS=1,63)
2	80,51% (DS=10,93)	4,30 (DS=3,06)
3	77,12% (DS=14,66)	4,35 (DS=1,70)
4	60,58% (DS=19,67)	7,34 (DS=3,18)
5	74,42% (DS=21,76)	4,03 (DS=1,37)
6	84,65% (DS=11,74)	4,32 (DS=2,96)
Totale	77,43% (DS=17,19)	4,57 (DS=2,72)

Tabella 3: Percentuale di studenti che in media sono riusciti a superare gli esercizi e numero di tentativi

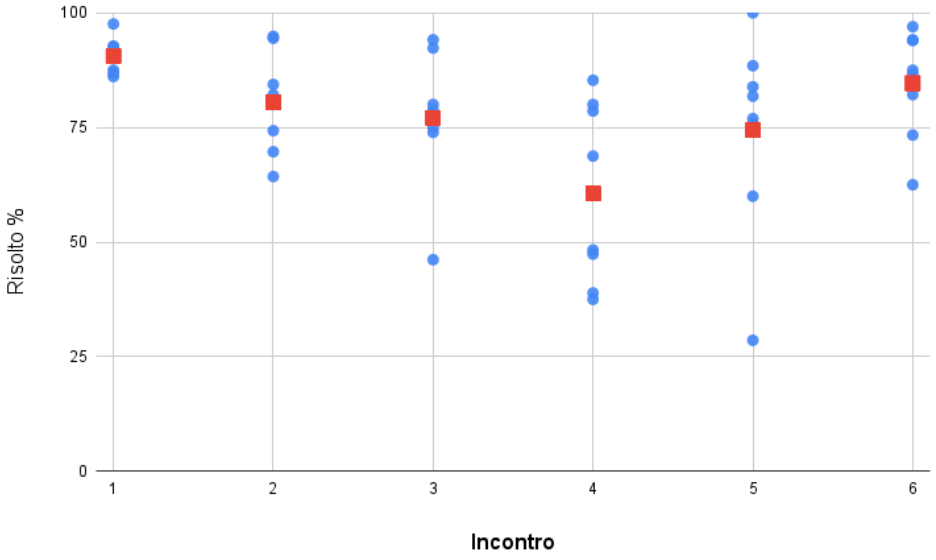


Figura 2: Percentuale di superamento degli esercizi (i quadrati rossi rappresentano i valori medi)

L'incontro 4, dedicato alle istruzioni condizionali, sembra essere stato il più difficoltoso. Questo risultato è rafforzato da alcune osservazioni qualitative che possiamo fare analizzando i programmi creati per risolvere gli esercizi di questo incontro: in generale, questi programmi, anche se corretti, *non contengono istruzioni condizionali*. La Figura 3 mostra un esempio: l'esercizio chiede di riprodurre una melodia che consiste della ripetizione di tre note, quattro volte di seguito, con la variante di alzare la tonalità all'ultima ripetizione. La soluzione più compatta, che usa il costrutto `SeRipetizione`, è stata fornita solo 4 volte, tutte le altre soluzioni corrette non usano costrutti condizionali.

Figura 3: Esempio di esercizio e di due soluzioni corrette: una compatta fornita solo 4 volte e una senza istruzioni condizionali, molto più comune

Un altro dato interessante relativo all'usabilità del micromondo è il numero di azioni degli studenti. Per *azioni* intendiamo tutte le operazioni coinvolte nella creazione di un programma: creazione e spostamento di blocchi, modifica di parametri, esecuzione del programma, ecc. Il numero di azioni medio per ogni esercizio è stato 45,63 (DS=18,76). Per confronto, le stesse studentesse e studenti hanno impiegato in media 19,16 azioni (DS=12,51) per risolvere esercizi di tipo navigazionale tratti da Code.org.

5 Conclusioni

Le difficoltà osservate negli gli esercizi dell'incontro 4 riflettono quelle che abbiamo avuto nell'immaginare esercizi di programmazione musicale che coinvolgessero istruzioni condizionali. In questa sperimentazione, abbiamo provato ad utilizzarle per esprimere in modo compatto melodie complesse in un modo simile alla comunicazione verbale: per esempio, la melodia di Figura 3 potrebbe essere raccontata come "ripetere per quattro volte Re, Fa# e La, però *l'ultima volta* alzare il tono". Questo è un approccio "dal basso verso l'alto" che dà massima importanza alla precisa comprensione del meccanismo di esecuzione sequenziale di singole istruzioni: la struttura dei blocchi messi a disposizione per risolvere un esercizio non ha relazione con la struttura del problema e quindi probabilmente non è di grande aiuto per principianti. Si può immaginare invece un approccio "dall'alto verso il basso" che offra istruzioni condizionali che permettano di creare soluzioni strutturalmente coerenti con il problema: per l'esempio di Figura 3 potremmo immaginare un blocco di iterazione `Ripeti` che includa un "ramo" acceduto tramite precondizione (*"se la ripetizione è l'ultima"*): la soluzione assumerebbe dunque una forma che riflette la struttura del problema, come mostrato in Figura 4.

Le prestazioni degli esercizi riguardanti invece l'*iterazione* (senza l'aggiunta di istruzioni condizionali o cambi di tono), sono risultate superiori alla media: ognuno degli 8 esercizi che trattano *solo* l'iterazione è stato superato in media dall'84,29% (DS=8,38) delle studentesse e studenti dopo 2,87 tentativi (DS=1,15). Inoltre, spesso abbiamo osservato un uso corretto dei cicli anche in esercizi dedicati ad altri concetti, come mostrato nella Figura 3. Queste migliori prestazioni legate all'uso dell'iterazione potrebbe esssere legate alla possibilità di *osservare* e *ascoltare* pattern melodici ripetuti nell'interfaccia del micromondo (cfr. Sezione 2).

Infine, il numero medio di azioni necessarie per risolvere un esercizio, alto se paragonato con un micromondo navigazionale, ci suggerisce che, mediamente, gli esercizi di questo micromondo

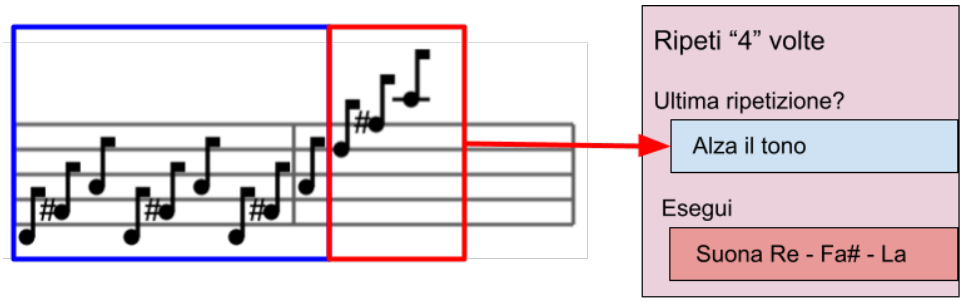


Figura 4: Esempio di blocco iterativo-condizionale che permette di esprimere una soluzione che riflette la struttura del problema

sono piuttosto laboriosi.

In conclusione, questo micromondo musicale ha rivelato un punto di forza interessante nella possibilità di stimolare un uso corretto del concetto di iterazione. Tuttavia servono ulteriori sperimentazioni relative alle istruzioni condizionali e, in generale, gli esercizi sono sembrati piuttosto complessi e laboriosi da affrontare. Questi risultati suggeriscono le potenzialità positive di un approccio misto che unisca esercizi di diversi micromondi in base ai concetti da apprendere.

Riferimenti bibliografici

[1] David Bau, Jeff Gray, Caitlin Kelleher, Josh Sheldon, and Franklyn Turbak. Learnable programming: blocks and beyond. *Communications of the ACM*, 60(6):72–80, 2017.

[2] Radek Pelánek and Tomáš Effenberger. The landscape of computational thinking problems for practice and assessment. *ACM Transactions on Computing Education*, 23(2):1–29, 2023.

I giochi Bebras dell'informatica nel curricolo della scuola del primo ciclo

Martina Palazzolo[1], Rita Signorini[1], Daniela Viganò[1]
[1] Istituto Comprensivo Ilaria Alpi di Milano
martina.palazzolo@icilariaalpi.edu.it
rita.signorini@icilariaalpi.edu.it
daniela.vigano@icilariaalpi.edu.it

Abstract

I giochi dell'informatica Bebras hanno l'obiettivo di divulgare, attraverso una gara non competitiva, i principi dell'informatica e il pensiero computazionale.

Il nostro Istituto Comprensivo partecipa alla gara Bebras nazionale dal 2015 e dal 2018 organizza un'attività basata sui quesiti Bebras chiamata *Allenatori Bebras* [1].

In questo articolo presentiamo le osservazioni fatte durante lo svolgimento dell'attività *Allenatori Bebras* in una seconda media. Durante l'attività i ragazzi utilizzano i quesiti Bebras per realizzare dei giochi cartacei. I manufatti servono quindi per allenare i compagni più giovani alla gara dell'anno successivo. L'obiettivo dell'attività *Allenatori Bebras* è quello di far analizzare ai ragazzi in modo approfondito i quesiti in modo che possano far proprie le abilità del pensiero computazionale richieste per la loro risoluzione. Analizziamo più manufatti realizzati a partire dallo stesso quesito *La tana di Mary* che quest'anno ha riscosso il maggior interesse. Le nostre osservazioni mostrano ancora una volta (vedi [1]) come l'abbinamento sinergico tra i quesiti Bebras e l'attività *Allenatori Bebras* sia proficua e permetta ai ragazzi di far propri i principi del pensiero computazionale. Inoltre, i giochi realizzati dai ragazzi evidenziano come gruppi diversi lavorino a livelli di competenza e abilità differenti traendone il massimo vantaggio per loro.

1 Introduzione

I giochi dell'informatica Bebras nascono nel 2004 in Lituania, con l'intento di divulgare i principi dell'informatica intesa come scienza. L'iniziativa è diventata presto internazionale e ad oggi partecipano alla comunità Bebras paesi di tutti i continenti [2] [3]. Delegato per l'Italia è il gruppo ALaDDIn (Laboratorio di Didattica e Divulgazione dell'Informatica) del Dipartimento di Informatica dell'Università Statale di Milano.

In Italia i giochi Bebras sono suddivisi in categorie per studenti dagli 8 ai 18 anni. La gara si svolge nel mese di novembre attraverso la piattaforma Bebras [4] che mette a disposizione anche le soluzioni e le spiegazioni dei giochi dopo la gara e per gli anni successivi.

Recentemente è stata proposta una classificazione dei giochi in base all'abilità del pensiero computazionale utilizzata per la loro risoluzione [5] e [6].

I quesiti, secondo tale classificazione, possono riguardare:

1 – organizzazione logica dei dati (uso di strutture per elaborare più facilmente i dati)
2 – analisi logica dei dati (ragionamento logico – deduttivo per trarre conclusioni)
3 – rappresentazione digitale delle informazioni (uso della rappresentazione simbolica di dati)
4 – pensiero algoritmico (uso di procedure passo – passo, metodi sistematici)
5 – identificazione di strategie risolutive (*problem solving* tramite strategie algoritmiche)
6 – analisi di soluzioni algoritmiche (confronto di strategie, problemi di ottimizzazione)
7 – implementazione di soluzioni algoritmiche (programmazione o *coding*)

Nel 2018 abbiamo organizzato, nel laboratorio scientifico curricolare, l'attività *Allenatori Bebras* con l'obiettivo di sfruttare le potenzialità didattiche offerte dai quesiti Bebras. Avevamo infatti notato che, terminata la gara, l'interesse dei ragazzi era principalmente legato alla conta dei quesiti corretti ed errati e meno a comprendere la soluzione e la spiegazione. Riscontrata l'efficacia dell'attività *Allenatori Bebras*, tale metodologia è stata adottata successivamente a M. Palazzolo [1] anche da altri insegnanti (R. Signorini e D. Viganò) assegnati ai laboratori scientifici. Il momento dell'allenamento agli alunni di quarta e quinta elementare, inoltre, viene utilizzato come attività di raccordo tra primaria e secondaria.

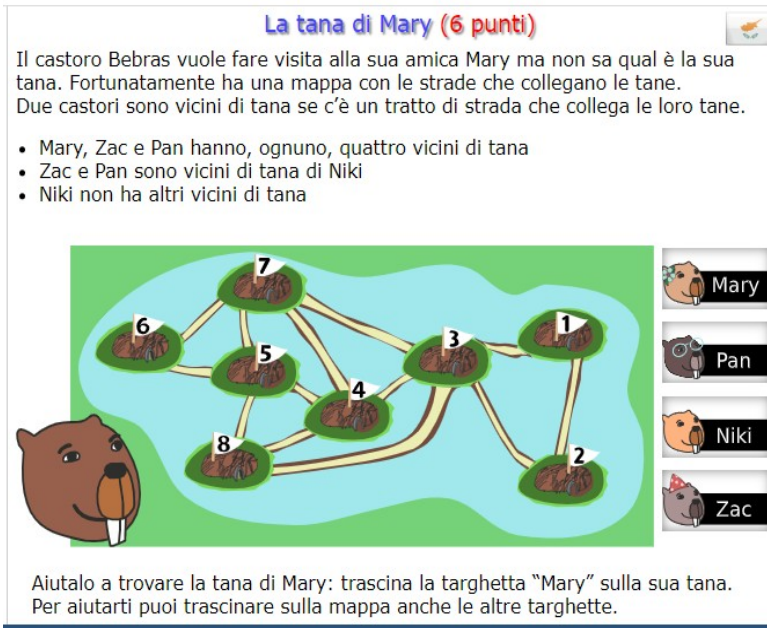
A testimonianza della ricaduta dell'attività sui ragazzi, presentiamo oggi le rielaborazioni del gioco *La tana di Mary* dell'edizione Mega 2022. Il gioco è stato scelto da 9 ragazzi su 24. I ragazzi hanno lavorato in gruppi di tre, costruendo quindi 3 giochi differenti a partire dallo stesso quesito. E' possibile osservare, come i ragazzi riescano a comprendere la logica del gioco e a rielaborarla facendo pratica con abilità del pensiero computazionale.

2 La Tana di Mary – Gioco dell'edizione mega 2022

2.1 – Il gioco originale

Nel quesito (figura 1) viene rappresentata la mappa delle tane dove abitano alcuni castori e delle strade che le collegano. I ragazzi devono individuare la tana di Mary osservando il disegno proposto e analizzando alcune informazioni sulle relazioni di vicinanza che intercorrono tra quattro castori: Mary, Pan, Niki e Zac. Si sa che Mary, Zac e Pan hanno ognuno quattro vicini di tana, Zac e Pan sono vicini di tana di Niky e che Niki non ha altri vicini di tana. La prima informazione dovrebbe portare i ragazzi a individuare quelle tane che hanno quattro collegamenti e che sono la tana 4, la 5 e la 7 . Una di queste sarà la tana di Mary. Se Niki ha solo Zac e Pan come vicini allora la sua tana dovrà avere solo due collegamenti. Tre tane hanno questa caratteristica: la tana 1, la 2 e la 6. Solo la 6 però è collegata alle possibili tane di Zac e Pan che potranno essere la 5 e la 7 indifferentemente. L'unica tana con quattro collegamenti rimasta, la 4, dovrà essere per forza la tana di Mary. Il quesito quindi richiede ai ragazzi di analizzare in modo logico i dati e ragionare per trovare la tana di Mary. Il gioco li fa lavorare su un tipo di rappresentazione (grafo) molto utilizzata per risolvere quei problemi la cui soluzione si trova analizzando le relazioni che intercorrono tra elementi. In questo caso la relazione di cui tener conto è "la vicinanza di tana".

Figura 1: Il gioco originale



2.2 - Rielaborazione 1 del gioco – Dai Castori alle scimmie

Il primo gruppo che abbiamo osservato ha scelto di cambiare l'ambientazione passando dalle tane dei castori agli alberi delle scimmie. Rifare il disegno (che non è stato banale) ha permesso loro di concentrarsi sui nodi e le relazioni tra di essi.

Figura 2: Foto del gioco LE SCIMMIE

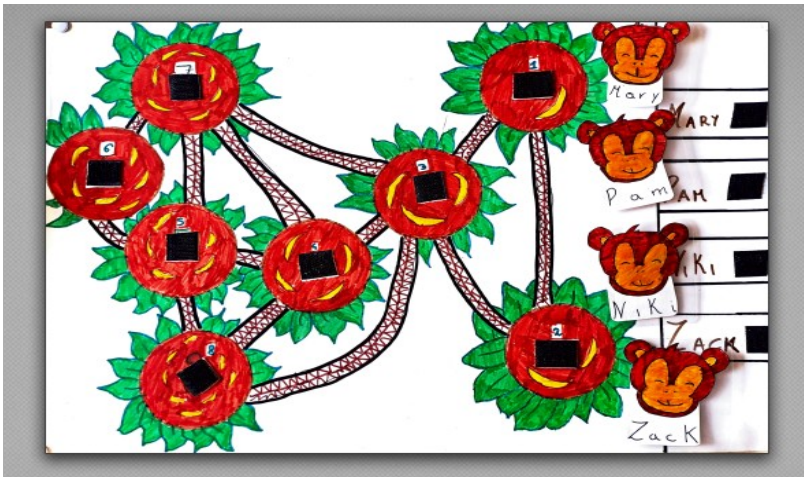
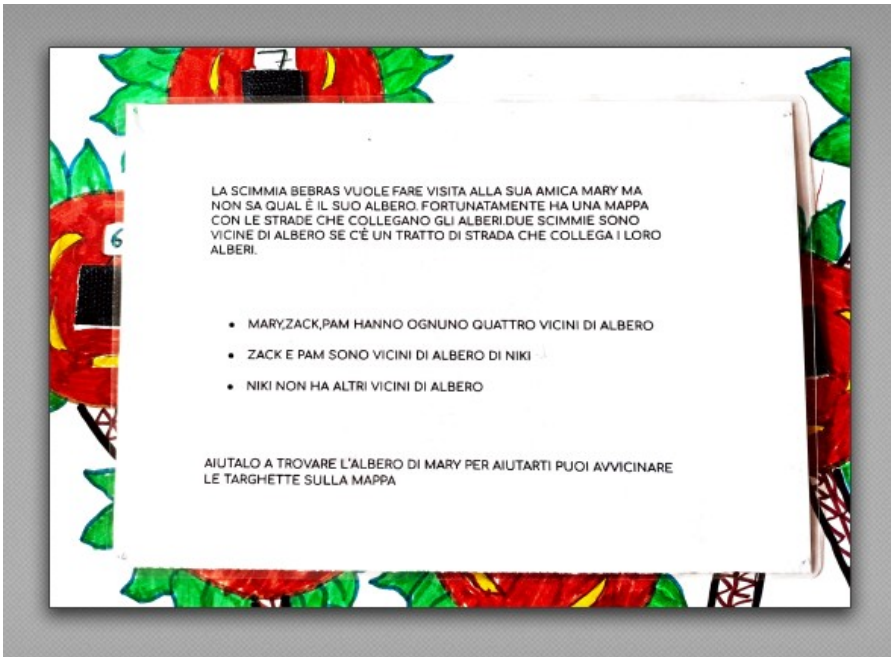


Figura 3: Il testo del gioco LE SCIMMIE



2.3 - Rielaborazione 2 del gioco – I CASTELLI

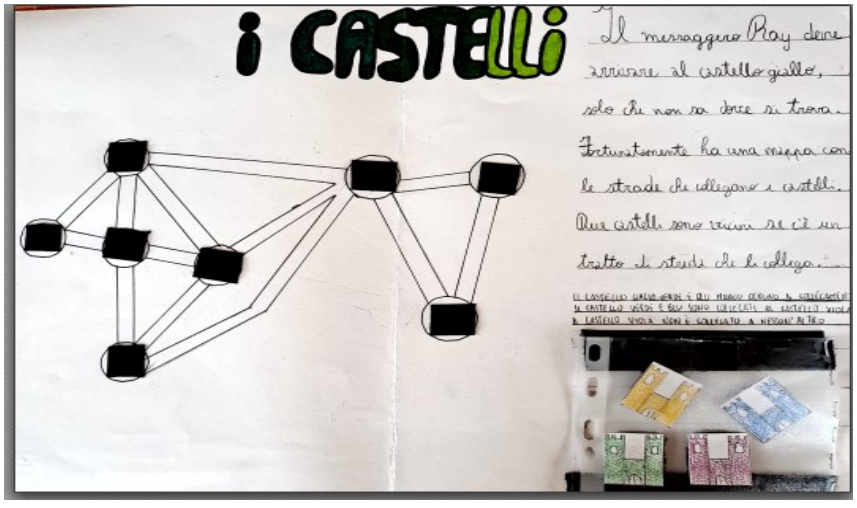


Figura 4: Foto del gioco I CASTELLI

In questa rielaborazione, i ragazzi inizialmente avevano pensato di variare la richiesta: posizionare correttamente tutti i castelli nella mappa. Dal confronto con l'insegnante è emerso

però che sono possibili più soluzioni e sono tornati sui loro passi. Hanno quindi proposto un messaggero alla ricerca di uno specifico castello. Riportiamo il testo del loro gioco poco leggibile nella figura:" Il messagero Ray deve arrivare al castello giallo, solo che non sa dove si trova. Fortunatamente ha una mappa con le strade che collegano i castelli. Due castelli sono vicini se c'è un tratto di strada che li collega.
I castelli giallo, verde e blu hanno ognuno 4 collegamenti.
I castelli verdi e blu sono collegati al castello viola
Il castello viola non è collegato a nessun altro".

2.4 - Rielaborazione 3 del gioco – IL DETECTIVE

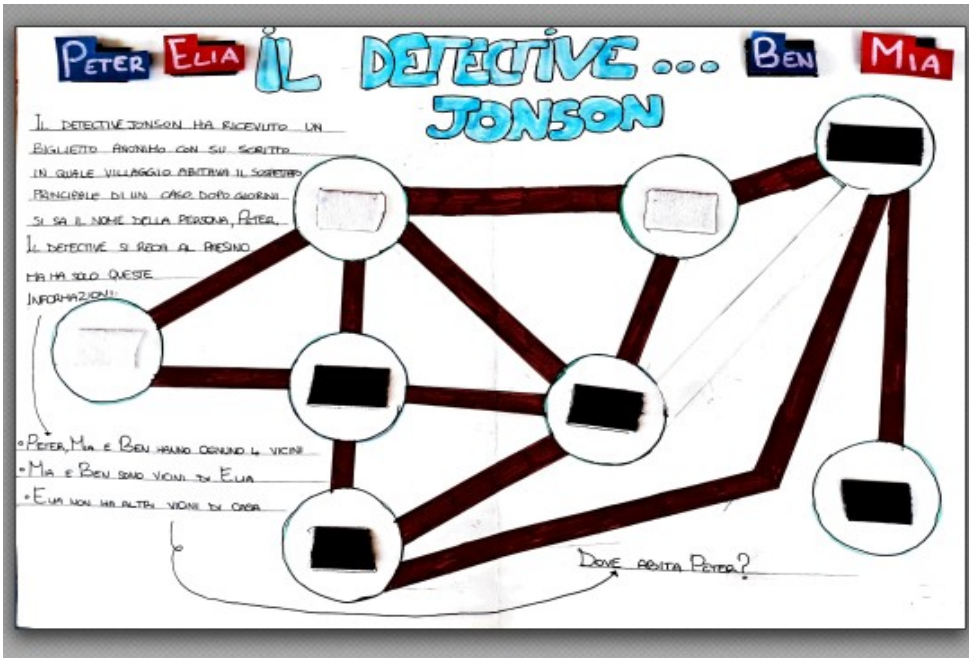


Figura 5: Foto del gioco IL DETECTIVE

In questo caso i ragazzi hanno proposto un detective alla ricerca di un fuggiasco. Il testo che scrivono è il seguente: "Il detective Jonson ha ricevuto un biglietto anonimo con su scritto in quale villaggio abita il principale sospettato di un caso. Dopo giorni si sa il nome della persona, Peter. Il detective si reca al paesino ma ha solo queste informazioni: Peter, Mia e Ben hanno solo quattro vicini; Mia e Ben sono vicini di Elia; Elia non ha altri vicini di casa. Dove abita Peter?".

Da un primo confronto con l'insegnante sono emerse alcune differenze tra la mappa originale e quella proposta: dei tre nodi disegnati in alto quello centrale (chiamiamolo nodo A) ha tre collegamenti invece di cinque del gioco originale; quello in alto a destra (nodo B) ne ha quattro (se ne vedono adesso chiaramente 3 ma uno è stato cancellato) invece di due; infine, il nodo in basso a destra (nodo C) ha un solo collegamento invece di due. La modifica nel nodo A non ha effetti sulla soluzione mentre il nodo B, avendo quattro collegamenti, ne diventa un ulteriore candidato. Anche il nodo C cambia il ragionamento da fare per la risoluzione del gioco in quanto si perde un possibile candidato per la casa di Elia, utilizzata per trovare la soluzione per esclusione. Dopo queste considerazioni i ragazzi sono tornati a ragionare per rendere coerente il loro prodotto al gioco originale. Invece di rifare tutta la mappa hanno trovato una soluzione efficace semplicemente cancellando un collegamento al nodo B. La mappa risulta differente

dall'originale, tuttavia il soddisfacimento dei vincoli permette di trovare in modo univoco la soluzione.

3 Conclusioni

Le osservazioni dei lavori finiti e del processo che ha portato alla loro realizzazione ha confermato quanto pubblicato in [1]: i quesiti Bebras e l'attività *Allenatori Bebras*, in modo sinergico, permettono ai ragazzi di ragionare sfruttando le strategie e le abilità del pensiero computazionale. Il confronto dei manufatti realizzati a partire dallo stesso gioco amplia ulteriormente il quadro: i ragazzi usano le abilità richieste per la risoluzione del gioco mettendo in campo le proprie risorse personali e partendo dal proprio livello di competenza. Traggono quindi il massimo beneficio comprendendo a fondo il quesito con riflessioni e livelli di approfondimento differenti. Il primo gruppo, per esempio, ha tratto vantaggio rivisitando semplicemente l'ambientazione e ha avuto la necessità di lavorare ridisegnando correttamente lo stesso grafo e gli stessi vincoli. Il secondo gruppo invece voleva estendere il quesito ad una richiesta più ampia: invece di localizzare solo uno dei castelli, chiedeva di posizionare tutti i castelli in modo da soddisfare i vincoli posti. Purtroppo, dati i vincoli utilizzati, la modifica ammetteva più soluzioni corrette e è dovuto tornare sui propri passi. Il terzo gruppo, infine, ha tratto vantaggio dalla riflessione partita da una modifica fatta rispetto alla mappa originale che determinava una soluzione non univoca. Ha quindi ragionato e introdotto una correzione che ha creato una nuova configurazione della mappa, questa volta coerente con la soluzione proposta dal quesito originale.

Bibliografia

[1] C. Bellettini, V. Lonati, M. Monga, A. Morpurgo, M. Palazzolo. Situated Learning with Bebras Tasklets. *ISSEP 2019* pagg. 225 – 239 LNCS 11913, Springer 2019
[2] https://www.bebras.org/countries.html
[3] V. Dagiené and G. Stupuriene. Informatics education based on solving attractive tasks through a contest. *IFIP-KEYCIT* 2014, p. 51-62, 2014
[4] bebras.it
[5] A. Calcagni, V. Lonati, D. Malchiodi, M. Monga, A. Morpurgo. Promoting Computational Thinking Skills: Would You Use this Bebras Task? *ISSEP 2017* pagg. 102 – 113 LNCS 10696, Springer 2019
[6] V. Lonati, D. Malchiodi, M. Monga, A. Morpurgo. Bebras as a Teaching Resource: Classifying the Tasks Corpus Using Computational Thinking Skills. *ItiCSE '17*, july 3 – 5, 2017, Bologna, Italy

Crittografia e steganografia alla scuola primaria: un'attività di coding ricca di potenzialità.

Maria Cristina Carrisi
Dipartimento di Matematica e Informatica
Università degli studi di Cagliari
mariacri.carrisi@unica.it

Abstract

In questo lavoro si descriverà un'attività di coding legata alla crittografia e alla steganografia, proposta a studenti della terza classe di una scuola primaria. L'attenzione sarà rivolta sulle competenze che l'attività può sviluppare e/o potenziare e sui legami con le diverse discipline in un'ottica di reale integrazione del coding nel curricolo scolastico, come previsto dalla normativa vigente.

1 Introduzione

Nel 2019, con la mozione n. 1-00117 del 12 marzo 2019 [1], viene proposto per la prima volta in Italia l'inserimento del coding nel curricolo della scuola primaria come attività obbligatoria, integrandolo con le altre attività previste dalle Indicazioni Nazionali [2]. Tale mozione è stata recepita il 21 dicembre 2021, quando la Camera dei deputati ha approvato il *Decreto Recovery* (DL 152) [3], successivamente trasformato in legge [4] il 29 dicembre dello stesso anno.

Nel decreto e successiva legge, lo sviluppo delle competenze digitali è esplicitamente indicato come una priorità anche ai fini del raggiungimento e mantenimento degli obbiettivi del Piano Nazionale di Ripresa e Resilienza, cui il *Decreto Recovery* si riferisce. Si legge, infatti:

"Al fine di consentire l'attuazione […] del Piano nazionale di ripresa e resilienza, per favorire e migliorare l'apprendimento e le competenze digitali, a decorrere dall'anno scolastico 2022/2023 e per un triennio, il Piano nazionale di formazione dei docenti delle scuole di ogni ordine e grado, […] senza nuovi o maggiori oneri per la finanza pubblica, individua, tra le priorità nazionali, l'approccio agli apprendimenti della programmazione informatica (coding) e della didattica digitale.

Entro il termine dell'anno scolastico 2024/2025, con decreto del Ministro dell'istruzione sono integrati, ove non già previsti, gli obiettivi specifici di apprendimento e i traguardi di competenza delle Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo di istruzione e delle Indicazioni nazionali e delle Linee guida vigenti per le istituzioni scolastiche del secondo ciclo di istruzione.

A decorrere dall'anno scolastico 2025/2026, nelle scuole di ogni ordine e grado si persegue lo sviluppo delle competenze digitali, anche favorendo gli apprendimenti della programmazione informatica (coding), nell'ambito degli insegnamenti esistenti, con le risorse umane, strumentali e finanziarie disponibili a legislazione vigente e, comunque, senza nuovi o maggiori oneri per la finanza pubblica".

Nel decreto è previsto un adeguamento del piano di formazione dei docenti, ma non il reclutamento di nuovo personale specificatamente formato, come invece avviene per l'insegnamento della lingua inglese e, dall'anno scolastico 2022-2023, per l'insegnamento

dell'educazione motoria (introdotta con legge n. 234/2021 [5] nelle classi quarte e quinte della scuola primaria). È infatti ribadita più volte la necessità di avvalersi delle *"risorse umane, strumentali e finanziarie disponibili a legislazione vigente"*.

Questo significa che l'insegnamento del coding non viene considerato una disciplina a sé stante, con un quadro orario ben delineato e obbiettivi di apprendimento differenziati per livello scolastico (classi), ma gli insegnanti delle diverse discipline avranno l'obbligo di inserire attività di coding all'interno della loro programmazione annuale. Questo rischia di essere un punto di debolezza, sia per quanto attiene la motivazione dei docenti ma anche perché il coding potrebbe essere percepito come una tematica avulsa dai nuclei centrali della disciplina, tradizionalmente considerati forieri di maggiori strumenti nel bagaglio culturale degli studenti e quindi marginalizzato.
Tuttavia, proprio l'obbligatoria interazione con le varie discipline consente di sfruttare le potenzialità del coding nelle sue diverse sfaccettature e applicazioni, dal problem-solving allo story-telling, dalla pixel art alla coordinazione motoria, con i vantaggi ben delineati in letteratura.

Sebbene l'obbligo di erogazione agli studenti decorra dall'anno scolastico 2025-2026, è già possibile trovare proposte di attività di coding nei libri di testo della scuola primaria. Da un'analisi di alcuni testi (vedi per esempio [6] e [7]), si rileva che le proposte di attività

- si trovano nei volumi di Matematica e Tecnologie, focalizzandosi quindi sui legami con le competenze di ambito logico-matematico;
- sono quasi esclusivamente legate all'individuazione di percorsi all'interno di griglie predefinite, oppure all'analisi del testo di un problema, successiva individuazione delle diverse tipologie di dati (mancati, inutili ecc.) e schematizzazione della procedura risolutiva;
- spesso non si differenziano nei diversi livelli scolastici. Per esempio, nei testi [6] volume 4 e [7] volume 2 è presente la medesima tipologia di attività sui dati del problema.

Emerge quindi la necessità di proporre ulteriori e nuove attività, che possano essere utilizzate in svariati contesti disciplinari con lo scopo di potenziare o rafforzare diversi tipi di competenze e non solo quelle maggiormente legate all'ambito logico-matematico. Nella prossima sezione verrà descritta e analizzata un'attività proposta nell'A.S. 2022-2023 in una scuola primaria della provincia di Cagliari.

2 Descrizione ad analisi dell'attività

L'attività, nata dal desiderio delle docenti di far partecipare gli allievi all'ora del codice [8], è stata proposta il 15 febbraio 2023 a due classi terze della scuola primaria dell'istituto comprensivo Selargius 1 di Selargius (Ca), per un totale di 29 alunni, di cui 5 con disabilità (DVA), 5 con Disturbi Specifici dell'Apprendimento (DSA) e 2 con Bisogni educativi Speciali (BES). Per consentire la partecipazione attiva di tutti gli studenti si è lavorato in un'ottica di Gamification e gli studenti sono stati suddivisi in piccoli gruppi di massimo 5 componenti per favorire l'attivazione di processi di peer tutoring.

L'attività dal titolo "Giochiamo a fare le spie" ha avuto come oggetto la presentazione ed il successivo uso da parte degli allievi di semplici tecniche di crittografia e steganografia ed è stata progettata per essere svolta in due ore, come da richiesta delle insegnanti.

La proposta si è articolata in una prima, breve, fase introduttiva e successivi 3 tasks. Nella prima parte si è discusso con gli studenti del processo di comunicazione e della necessità di proteggere i dati. Si sono utilizzati alcuni esempi storici particolari o divertenti, e semplici esempi di vita quotidiana alla portata di studenti di 8/9 anni. Successivamente si è attivato un brain storming guidato al fine di validare, mostrare i punti di debolezza e le incoerenze delle tesi o degli ulteriori esempi proposti dagli studenti. Infine, sono stati proposti agli studenti i tre compiti:

1. Uso dell'inchiostro simpatico. Con un cotton fioc (o un dito) intinto nel succo di limone, gli studenti hanno scritto le loro iniziali in un foglio. Dopo che il foglio si è asciugato le lettere non sono più facilmente individuabili e riconoscibili ma avvicinando il foglio ad

una fonte di calore (si è utilizzato un lumino di cera, tipo tea light) è possibile rivelare la scritta.

2. Uso delle griglie di Cardano. Agli studenti sono stati consegnati due fogli: uno contenente un testo (vedi Figura 1) e l'altro la griglia (vedi Figura 2). Gli allievi avevano il compito di ritagliare i quadratini blu della griglia per ricavare delle finestrelle e poi sovrapporre la griglia al testo in modo da rivelare il messaggio nascosto (vedi Figura 3).

UNA VOLPE AFFAMATA GIUNSE IN UN VIGNETO. "UVA? CON LA FAME CHE HO, MEGLIO CHE NIENTE..." SI DISSE LA VOLPE. COSÌ SALTÒ CON AGILITÀ MA NON RIUSCÌ A RAGGIUNGERE L'UVA. RIPROVÒ PIÙ E PIÙ VOLTE, SENZA ALCUN SUCCESSO. QUEST'UVA È TROPPO ACERBA! POCO IMPORTA SE NON RIESCO AD AFFERRARLA... " CONCLUSE AD ALTA VOCE LA VOLPE.

Figura 1: Testo contenente il messaggio nascosto

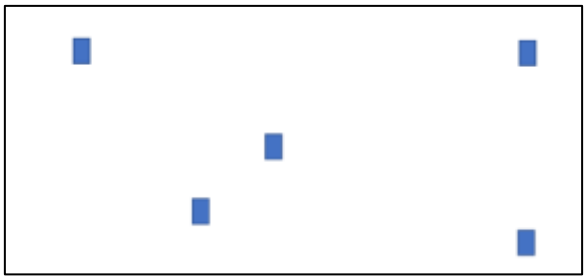


Figura 2: Schema di griglia. I quadratini segnano le fessure che dovranno essere ricavate per rivelare il messaggio nascosto

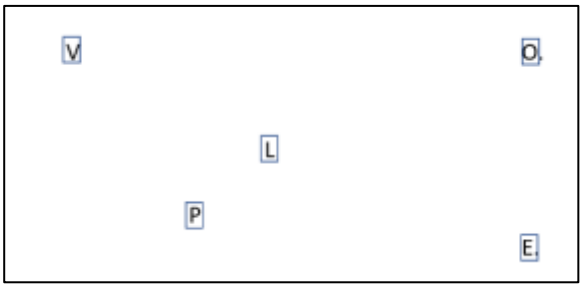


Figura 3: Sovrapponendo la griglia al testo e leggendo le lettere evidenziate in ordine lessicografico è possibile scoprire il messaggio nascosto

3. Uso del disco di Alberti. Agli studenti è stato consegnato un foglio contenente lo schema del disco di Alberti (vedi Figura 4). Dopo aver ritagliato i dischi e averne forato il centro con una matita, i due dischi sono stati assemblati e fissati con un ferma campioni in modo che potessero ruotare, al fine di ottenere un determinato abbinamento delle lettere. Stabilito l'abbinamento di partenza gli allievi hanno effettuato la codifica di una parola (scritta alla lavagna ed uguale per tutti) e successiva decodifica del crittogramma precedentemente ottenuto.

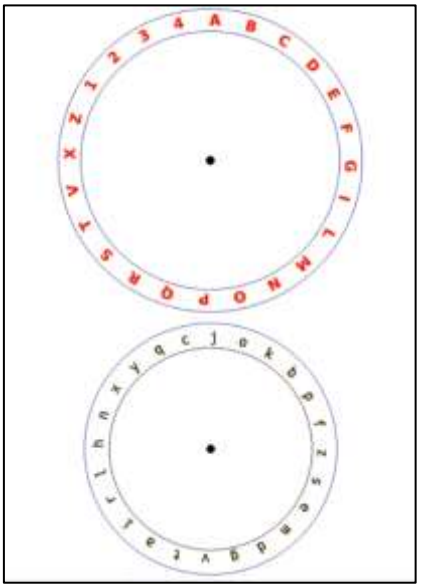


Figura 4: Componenti del disco di Alberti.

Tutti i tasks sono risultati molto attrattivi. Gli studenti hanno partecipato con interesse, risolvendo tutti gli enigmi e completando l'attività, sebbene alcuni abbiano dovuto affrontarli più volte prima di portarli a termine in modo corretto e/o completo.

L'attività non ha messo in gioco solo la capacità di eseguire un dato algoritmo seguendo in modo rigoroso i passi indicati ma molte altre competenze, come vedremo nei paragrafi seguenti.

Il primo task è quello che ha fatto emergere maggiori problemi esecutivi. In particolare, le difficoltà riscontrate degli allievi hanno riguardato la capacità di scrivere imbevendo adeguatamente il foglio, la capacità di scrivere con punte così grosse pur mantenendo la leggibilità, la capacità di comprendere quanto prossimo al lumino dovesse essere il foglio per consentirne il riscaldamento ma evitare che bruciasse. Questo ha reso possibile aprire una discussione con la classe, in cui analizzare le cause delle problematiche riscontrate ed elaborare strategie risolutive alternative, svolgendo quelle attività di analisi della soluzione, back-tracking e rielaborazione di nuova strategia che sono tipiche del coding ma spesso sottovalutate o rimandate ai livelli scolastici successivi. Si è potuto lavorare sull'analisi delle differenze con quanto fatto dai compagni, sul perché una strategia o uno strumento hanno funzionato meglio, sui nessi di causa ed effetto, tutte competenze sviluppabili col coding ed in generale riferite al metodo scientifico.

Tutti i tasks consentono di lavorare su competenze di area linguistica, prevedendo il riconoscimento di lettere in situazioni non standard (lettere poco riconoscibili a causa della scrittura col cotton fioc, in parte nascoste dalla griglia, variamente orientate nel disco di Alberti) e non tutti gli allievi sono stati immediatamente in grado di individuare in modo corretto le lettere o di attivare strategie per compensare il problema di riconoscimento (per esempio, ruotare il disco in modo da visualizzare le lettere in posizione standard).

Ricordiamo che il riconoscimento delle lettere fa parte delle abilità alfabetiche emergenti, il cui possesso si valuta nell'ultimo anno della scuola dell'infanzia e la cui carenza è un forte predittore di rischio per DSA [9]. Non stupisce, quindi, che in un gruppo classe come quello in cui si è operato si sia verificata questa circostanza ma è importante che gli allievi, seppur con tentativi ed errori siano riusciti a portare a termine l'attività.

Tutti i tasks consentono, inoltre, di lavorare su competenze di area visuo-spaziale, motoria e prassica. L'avvicinamento del foglio al lumino ma anche l'uso delle forbici ha consentito di lavorare su motricità fine, coordinazione dei piccoli movimenti della mano, oculo-manuale e bilaterale (nell'uso delle forbici, infatti, le mani svolgono azioni diverse: una tiene il foglio e lo muove, l'altra taglia). Oltre a quanto già descritto per il primo compito, si sono rilevate difficoltà nel ritagliare le finestrelle della griglia o seguire il bordo curvilineo del disco. Alcuni allievi hanno

attivato strategie compensative che hanno permesso, pur eseguendo tagli non necessari, di preservare comunque l'integrità della griglia, come evidenziato dai riquadri arancioni in Figura 5.

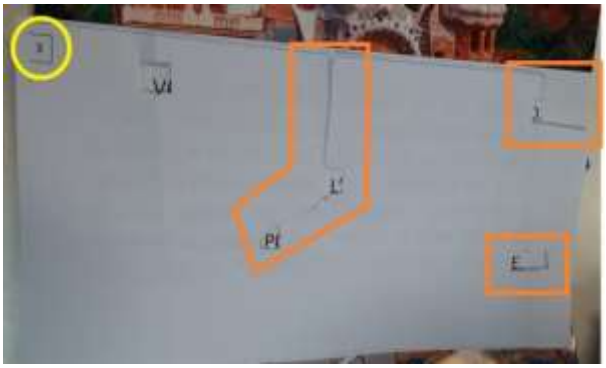


Figura 5: Griglia realizzata da uno studente

Nonostante il tagliare sia un'attività che sviluppa la concentrazione, soprattutto in casi come questi in cui la precisione è fondamentale per ottenere il risultato, essa è anche fortemente rilassante e distensiva e quindi molto utile in bambini che, per diversi motivi, possono mostrare nervosismo, agitazione o rabbia.
Sempre restando nell'area delle competenze visuo-spaziali, è utile sottolineare che il compito 2 consente anche di lavorare sul rafforzamento dei principali concetti topologici. Infatti, è possibile osservare in Figura 5 come il cerchio giallo evidenzi un simbolo che consente il corretto posizionamento della griglia rispetto al testo. Una collocazione errata non consente di visualizzare il messaggio.

È bene sottolineare l'importanza di effettuare entrambe le fasi di codifica e decodifica, soprattutto nel task 2 dove potrebbe apparire non necessario, in quanto è indispensabile che gli studenti abbiano contezza della reversibilità (invertibilità) del processo per poterlo considerare un efficace metodo di protezione delle informazioni che ne consenta un successivo recupero. Questo tipo di riflessione, insieme a quanto già esposto relativamente all'analisi critica delle procedure messe in atto nei diversi compiti, consente di sviluppare una corretta cultura informatica, permettendo ai discenti di comprendere il fondamento scientifico della disciplina che non consiste in un mero uso di strumenti digitali.

3 Conclusioni

È stata proposta una attività di coding che, per le sue caratteristiche, permette di sviluppare o rafforzare competenze nelle diverse aree motorio-prassica, linguistica, logico-matematica e quindi di essere proposta dai docenti in vari contesti.

Attività analoghe, con questi o altri tipi di codifica, possono essere proposte in tutte le classi, raggiungendo l'obbiettivo, previsto dalla legge, dell'integrazione del coding con i nuclei tematici delle diverse discipline. A titolo di esempio menzioniamo il fatto che

- la realizzazione di manufatti rientra appieno nelle competenze previste nella materia Tecnologia
- utilizzando il codice di Cesare o la scitala Lacedemonica si sarebbe potuto creare un interessante intermezzo nelle lezioni di storia su Sparta o sulla Repubblica romana in una classe quinta
- le griglie possono essere usate come strumento di verifica della risposta alle tipiche domande di analisi del testo che prevedono di individuare, tra le altre cose, protagonista, antagonista, oggetto del desiderio.

La scelta dei diversi compiti da proporre alla classe è stata vincolata dal tempo a disposizione (si ricorda che l'attività doveva svilupparsi e concludersi in due ore), dalle conoscenze, abilità e competenze in possesso di studenti della terza classe di una scuola primaria ma soprattutto dalla volontà di consentire la partecipazione di tutti gli studenti in una classe così numerosa e con una presenza così significativa di alunni con difficoltà. Per questo motivo si è scelto di focalizzare l'attenzione verso quella parte del coding che è così tipica e legata al metodo scientifico ma anche la più difficile da ritrovare tra le attività classicamente proposte e cioè il testing dell'algoritmo, l'analisi delle criticità rilevate e la ricerca di soluzioni alternative e/o più generali. Inoltre, prevedendo difficoltà nel tenere a lungo l'attenzione, si è preferito far svolgere attività con una forte componente manuale e costituita da compiti diversi con un totale cambiamento di scenario nel passaggio dall'uno all'altro, piuttosto che attività che fossero tra loro legate, onde evitare che problemi nello svolgimento di uno dei compiti potessero limitare o impedire lo svolgimento degli altri.

Ci si propone per il futuro, di sviluppare l'attività costruendo un breve percorso per studenti di scuola primaria che abbia l'obbiettivo di mostrare l'evoluzione delle tecniche di protezione di dati mediante un processo che, partendo dall'analisi delle criticità di un metodo consenta di introdurre nuove tecniche che superano quelle criticità avvalendosi di metodologie e strumenti presenti in letteratura (vedi per esempio [10]) che possono aiutare a veicolare tali concetti anche a studenti che non hanno ancora competenze solide in ambito informatico.

Bibliografia

[1]https://documenti.camera.it/leg18/resoconti/assemblea/html/sed0139/leg.18.sed0139.allegato_a.pdf

[2] https://www.miur.gov.it/documents/20182/51310/DM+254_2012.pdf

[3] https://www.gazzettaufficiale.it/eli/id/2021/11/06/21G00166/sg

[4] legge 29 dicembre 2021, n. 233, https://www.gazzettaufficiale.it/atto/serie_generale/caricaDettaglioAtto/originario?atto.dataPubblicazioneGazzetta=2021-12-31&atto.codiceRedazionale=21G00257&elenco30giorni=true

[5] https://www.gazzettaufficiale.it/eli/id/2021/12/31/21G00256/sg

[6] Nuovi traguardi, E. Costa, L. Doniselli, A. Taino, Gruppo editoriale La Spiga

[7] Acchiappa Storie, S. Bordiglioni, E. Rizzo Licori, M. Tognana, Mondadori Education

[8] https://programmailfuturo.it/come/ora-del-codice

[9] B. Fioravanti, E. Savelli, S. Franceschi, La conoscenza delle lettere nell'ultimo anno della scuola dell'infanzia come indice predittivo dell'apprendimento della letto-scrittura, Dislessia, vol. 9 n. 2, Maggio 2012 p.79-101

[10] M. Lodi, M. Sbaraglia, S. Martini, Cryptography in Grade 10: Core Ideas with Snap! and Unplugged, ITiCSE '22: Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education, Vol. 1, July 2022, pp. 456-462, doi.org/10.1145/3502718.3524767

Coding Mind

Martina Ruscica
Istituto Comprensivo Statale "Martiri della Libertà" Sesto San Giovanni (MI)
martirusci126@gmail.com

Abstract

Il progetto si è posto come finalità lo sviluppo del pensiero computazionale attraverso la programmazione (*coding*) all'interno di una *situazione a-didattica*, intesa come *gioco* e con la realizzazione di diversi *artefatti culturali materiali*.

In una classe seconda di scuola primaria, le giovani *menti codificanti* hanno appreso le basi della programmazione informatica attraverso diverse attività, di coding unplugged, di coding e di tinkering.

1 Introduzione

Nelle vesti di insegnante di sostegno in anno di prova, durante l'A.S. 2022/2023, ho condotto una progettazione, in cui si è fatto uso di diversi strumenti digitali (computer, LIM) e piattaforme online (Programma il Futuro, Code.org, Scratch) per consentire agli alunni l'acquisizione di specifiche conoscenze e abilità utili ad imparare le basi della programmazione informatica e a saperle padroneggiare.

Il progetto è stato attuato nella classe in cui presto servizio, una seconda di scuola primaria. I bambini possedevano dei prerequisiti su alcuni concetti di informatica, acquisiti l'anno precedente: *pixel art,* percorsi con il Bee-bot, ecc. Il progetto, quindi, ha fatto leva sulle loro preconoscenze, interessi e conoscenze personali, arricchendole di maggiore *significato*, incrementando la loro curiosità, autostima, motivazione e autodeterminazione.

2 Quadro teorico di riferimento

Guy Brousseau nell'ambito della "Teoria delle situazioni didattiche" sostenne che la *situazione a-didattica* sembrava essere la più consona e più proficua in aula, poiché grazie ad essa avviene il processo di *devoluzione*, cioè l'allievo diventa attivo, interessato, responsabile e cosciente del proprio apprendimento. Risulta, dunque, vincente ciò che si riesce a mettere sotto forma di situazione a-didattica, alla quale Brousseau attribuisce il sinonimo di *gioco*, poiché l'intenzionalità educativa non è esplicitata e soprattutto l'azione di ogni singolo alunno non è condizionata dalle attese del docente, realizzandosi così la rottura del *contratto didattico*; producendo un tipo di apprendimento più consapevole e profondo: una vera e propria conoscenza, capace anche di *transfer cognitivi* (Brousseau, 1986).

L'ingegneria didattica del progetto, si fonda sul *learning by doing*, letteralmente "apprendere facendo". L'esperienza rappresenta il *fattore abilitante dell'apprendimento*, poiché lo rende davvero significativo e orientato al cambiamento della persona (Dewey, 2014). L'esperienza, inoltre è valida e fertile nella misura in cui è accompagnata dalla consapevolezza e dalla riflessione sulle connessioni (*il significato*) tra l'attività che il soggetto compie e le conseguenze che ne derivano. Al fare, dunque, va sempre accompagnato il pensare, perciò non solo "learning by doing" ma anche "thinking by doing"; in quanto l'esperienza insieme alla capacità di *prendere decisioni* sono gli elementi che determinano il 90% dell'apprendimento (Dale, 1946).

Nell'ultima fase del progetto è stato creato un ambiente di apprendimento all'interno del quale si è instaurato un particolare tipo di relazione tra l'insegnante e l'allievo, seguendo una prospettiva costruttivista, poiché l'alunno ha costruito non solo metaforicamente, ma concretamente, con le proprie mani, oggetti che hanno sollecitato la sua conoscenza (D'Amore & Marazzani, 2011). In laboratorio, quindi, gli allievi hanno indossato le vesti dei veri e propri protagonisti dell'atto didattico, costruendo diversi *artefatti culturali*, utili mediatori cognitivi per la comprensione di specifici concetti (Bartolini Bussi & Mariotti, 2009). In queste attività di *tinkering*, gli alunni hanno "armeggiato", manipolato e giocato con diversi strumenti, usando materiale di riciclo, materiali conduttori e componenti elettrici; sviluppando competenze importanti, come il saper analizzare, mantenere la concentrazione, il lavorare in maniera autonoma, il riconoscere i propri limiti e quelli delle situazioni con cui ci si confronta (Fondazione Mondo Digitale, 2019).

3 Il progetto

Il progetto ha avuto una durata di circa tre mesi. I destinatari sono stati 21 alunni, di cui uno con disabilità, sei di origine straniera e due con altri Bisogni Educativi Speciali.

Il progetto è stato pensato, declinato e attuato proponendo delle attività che fossero adatte a tutti, nel rispetto delle proprie capacità e propensioni, con l'intento di mettere a frutto e sviluppare i talenti di ogni alunno. Nelle diverse attività sono stati strutturati dei ruoli, come il *programmatore* e il *robot*; il *tutor* e il *tutee*; il *progettista* e il *costruttore*, che a turno, a seconda del tipo di richiesta, tutti gli alunni hanno ricoperto, con le adeguate personalizzazioni.

Le diverse attività si sono susseguite seguendo un ordine rispetto al tipo di contenuto e concetto proposto, dal più semplice al più complesso.

3.1 Io, Robot

In questa prima fase, svolta inizialmente in aula, si è partiti con la presentazione delle carte CodyRoby e della loro modalità di utilizzo.

Le prime lezioni si sono svolte senza l'ausilio di strumenti tecnologici con un'attività di coding unplugged, volta a far sperimentare agli alunni i due diversi ruoli: quello di *programmatore* (Roby) e quello di *robot* (Cody), in un gioco di movimento sulla scacchiera. L'alunno-Roby è stato lasciato libero di realizzare sulla scacchiera il percorso desiderato e ha utilizzato le carte CodyRoby per programmare il percorso del compagno-Cody, il quale ha dovuto eseguirlo leggendo il blocco di programmazione costruito con le *carte-codice* (Figura 1).

Figura 1: Cody e Roby in *azione*

In seguito, allo stesso tipo di attività, è stata aggiunta la richiesta per l'alunno-Roby di riproporre sulla scacchiera i percorsi del Labirinto Classico de l'Ora del Codice, presente all'interno della piattaforma Programma il Futuro, visionabili sulla LIM (Figura 2).

Figura 2: Labirinto Classico sulla scacchiera

L'alunno-Roby dopo aver programmato il compagno-Cody è stato invitato a programmare anche il Labirinto sulla LIM (Figura 3).

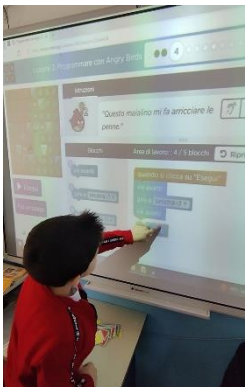


Figura 3: Realizzazione di un blocco di programmazione con i diversi codici a disposizione

Accogliendo una proposta della classe, sono state costruite le carte *#Raccogli* (Figura 4) per assortire il mazzo di carte CodyRoby e per poter riproporre sulla scacchiera il percorso del Corso Rapido: lezione 5 "Programmare con la collezionista" di Code.org, proiettato sulla LIM.

Figura 4: Carte *#Raccogli*

Attraverso una conversazione guidata è stata recuperata ed arricchita di significato la terminologia, fino ad ora incontrata, riferita all'attività di programmazione: *linguaggio* e *blocchi di programmazione; pensiero computazionale; codice binario; bug e debugging*.

Nel laboratorio di informatica, si è poi passati ad un'attività di coding "puro". Attraverso l'uso del computer, gli alunni divisi in coppie hanno completato i diversi percorsi, programmando online Il Labirinto Classico e il Corso Rapido: lezione 2 "Programmare con Angry Birds" e lezione 5 "Programmare con la collezionista" (Figura 5).

Le coppie sono state formate affidando, a turno ad ogni alunno, a seconda del tipo di percorso, il ruolo di *tutor* e di *tutee*, in modo da permettere a tutti di sperimentare e ricavare dei vantaggi da entrambe le funzioni di questo approccio cooperativo dell'apprendimento.

Figura 5: Programmando in coppia i diversi percorsi

3.2 Pronti partenza…sfida!

Nella seconda fase, le attività di coding sono state sviluppate attraverso il conseguimento di *cinque "sfide"* proposte su Scratch, utili all'esplorazione di questa piattaforma di programmazione.

Le attività si sono svolte nel laboratorio di informatica e gli alunni, anche in questo caso, sono stati divisi in coppie, sempre col ruolo interscambiabile di *tutor* o di *tutee* (Figura 6).

Figura 6: *Sfide* su Scratch

Le sfide sono state ideate e strutturate seguendo un ordine di complessità, dalla consegna più semplice a quella più articolata (programmazione del movimento, dell'aspetto, del movimento associato all'aspetto, dello sfondo, del suono).

Gli alunni accedendo a Scratch con le credenziali associate ad ogni coppia, hanno inserito i loro *progetti*-sfida all'interno delle *gallerie*-sfida, presenti nella loro classe virtuale (Figura 7).

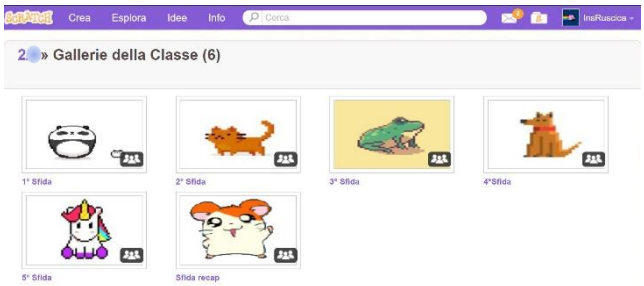


Figura 7: Le cinque *gallerie*-sfida + *galleria*-sfida recap (verifica)

3.3 Coding tangibile

In questa ultima fase, è stata presentata la scheda Makey Makey e le sue modalità di utilizzo più creative, sperimentando il suo funzionamento attraverso diversi materiali conduttori, come frutta, verdura, acqua, piante, corpo umano, grafite (matita), carta alluminio, ecc. (Figura 8).

Figura 8: Sperimentazione del funzionamento della scheda Makey Makey

Successivamente, in assetto laboratoriale, le potenzialità della piattaforma Scratch sono state sviluppate attraverso l'utilizzo della scheda Makey Makey.

Gli alunni, divisi in dieci gruppi, hanno realizzato diversi artefatti materiali, ricoprendo il ruolo di *progettista* o di *costruttore.* Tali artefatti autocostruiti sono stati associati, attraverso la scheda Makey Makey, a diversi programmi realizzati su Scratch, inserendo in alcuni casi la loro voce, in altri le note musicali o dei suoni, in altre ancora le parti della frase o della pianta, ecc. (Figura 9).

Figura 9: I nove programmi realizzati su Scratch

Di seguito i diversi prodotti realizzati con materiali di riciclo e materiali conduttori come graffette, carta alluminio, fili di stagno, fermacampioni, plastilina, grafite, ecc. (Figura 10; Figura 11; Figura 12; Figura 13; Figura 14; Figura 15; Figura 16; Figura 17; Figura 18).

Figura 10: Gruppi 1 e 2 - Il sistema solare

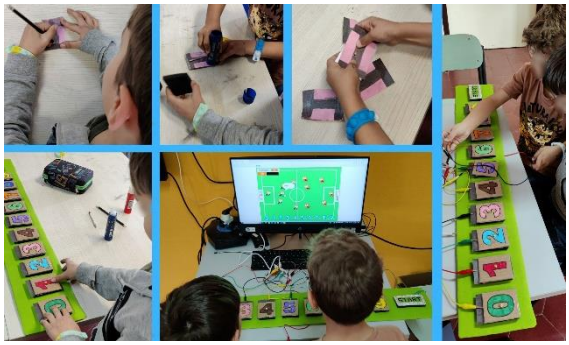


Figura 11: Gruppo 3 - Calcio con le tabelline

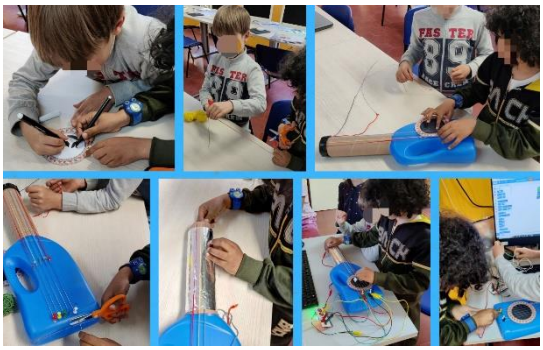


Figura 12: Gruppo 4 - Chitarra

Figura 13: Gruppo 5 - Joystick

Figura 14: Gruppo 6 - Dance Revolution

Figura 15: Gruppo 7 - Batteria

Figura 16: Gruppo 8 - Pianoforte

Figura 17: Gruppo 9 - Analisi grammaticale Montessori

Figura 18: Gruppo 10 - Le parti della pianta Montessori

A conclusione del percorso, gli alunni si sono impegnati in un progetto di classe:

"Far suonare le scale".

Dopo essersi divisi in piccoli gruppi, ognuno di loro ha realizzato un tassello del progetto: prendere le misure delle scale e dei cartoni; colorare i cartoni che avrebbero rappresentato i tasti del pianoforte; ricoprire i tasti e il corrimano con carta alluminio per garantire il funzionamento del circuito; assemblare il prodotto finale associandolo ad un programma realizzato su Scratch (Figura 19; Figura 20).

Figura 19: *Tasselli* del progetto di classe

Figura 20: *"Far suonare le scale"*

4 Conclusioni

Le tre fasi del progetto sono state valutate attraverso tre verifiche di tipo orale e pratico, in cui gli alunni hanno dovuto dimostrare di *saper generalizzare* e utilizzare le conoscenze e abilità acquisite per: programmare il Bee-Bot usando le carte CodyRoby (Figura 21); realizzare un progetto finale su Scratch mettendo insieme più codici (movimento, aspetto, suono, ecc.) (Figura 22); illustrare e programmare in un piccolo Expo i materiali autocostruiti anche da altri gruppi (Figura 23), utilizzando la *raccolta delle programmazioni* (Figura 24).

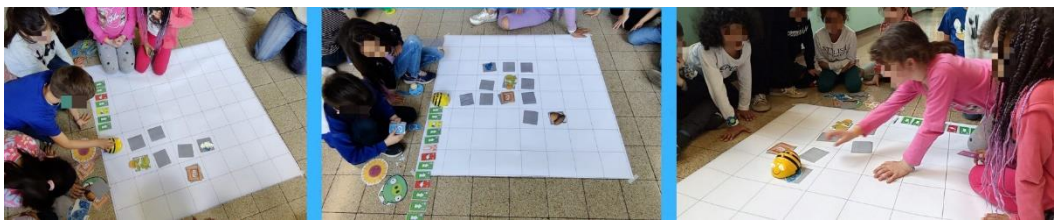


Figura 21: Verifica fase *Io, Robot*

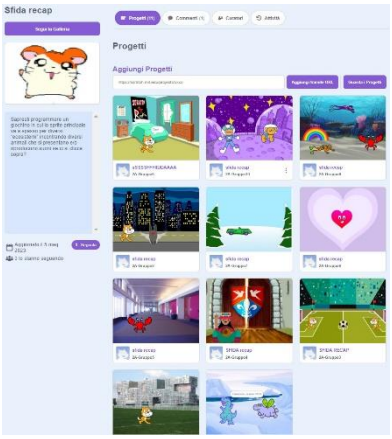


Figura 22: Verifica fase *Pronti partenza…sfida!*

Figura 23: Verifica fase *Coding tangibile*

Figura 24: Raccolta delle programmazioni

Bibliografia

Bartolini Bussi, M. G., & Mariotti, M. A. (2009). Mediazione semiotica nella didattica della matematica: artefatti e segni nella tradizione di Vygotskij. *L'insegnamento della matematica e delle scienze integrate*, 32, 269-294.

Brousseau, G. (1986). Fondements et méthodes de la didactique des mathématiques. *Recherches en Didactique des Mathématiques, 7*(2), 33-115.

Dale, E. (1946). *Audio-visual methods in teaching.* New York: The Dryden Press.

D'Amore, B., & Marazzani, I. (2011). *Problemi e laboratori. Metodologie per l'apprendimento della matematica.* Bologna: Pitagora.

Dewey, J. (2014). *Esperienza ed Educazione.* Milano: Raffaello Cortina.

Fondazione Mondo Digitale. (2019). *Tinkering, coding, making per bambini da 4 a 6 anni.* Trento: Edizioni Erickson.

Una strategia didattica per favorire l'apprendimento di concetti di Robotica Educativa in microlingua L2 e L3 in un contesto plurilingue

Gennaro Iaccarino[1], Sara Tosi[1], Daniel Gallo[2], and Ilenia Fronza[3]

[1] I.I.S.S. "G. Galilei", Bolzano, Italia
{gennaro.iaccarino, sara.tosi}@scuola.alto-adige.it
[2] Liceo delle Scienze Umane e Artistico "G. Pascoli", Bolzano, Italia
daniel.gallo@scuola.alto-adige.it
[3] Libera Università di Bolzano, Italia
ilenia.fronza@unibz.it

Abstract

L'attività didattica basata su *learning by teaching* favorisce l'apprendimento, promuove la rielaborazione delle informazioni e stimola la curiosità rispetto alle nozioni apprese. In questa *esperienza sul campo* abbiamo utilizzato questa strategia per creare un legame di competenze tra alunni del primo e del secondo ciclo d'istruzione, proponendo a studenti del secondo anno di Liceo Scientifico opzione Scienze Applicate di insegnare a dei bambini di 6-10 anni i principi base della programmazione applicati alla Robotica Educativa. Il progetto è stato diviso in tre fasi ed è stato arricchito dall'acquisizione di competenze linguistiche di microlingua che, in un territorio plurilingue come l'Alto Adige, rappresenta un valore aggiunto. Come risultato, abbiamo osservato l'efficacia della strategia adottata nel promuovere l'apprendimento dei principi di programmazione sia da parte dei bambini che dei liceali, stimolando ad approfondire e rielaborare quanto appreso per riproporlo in maniera chiara e semplificata.

1 Introduzione

La *didattica dell'Informatica* nel percorso scolastico italiano è un tema aperto già da qualche anno, che il Ministro dell'Istruzione e del Merito affronta guardando al panorama internazionale. Negli Stati Uniti è attiva dal 2015 l'iniziativa *Computer Science for All*, che inserisce l'informatica nell'istruzione scolastica alla pari delle altre discipline scientifiche e tecnologiche. Nel Regno Unito già dal 2014-15 la disciplina *Computing* è obbligatoria in tutte le scuole, a partire dalle elementari. Iniziative simili sono in corso in molti altri paesi [4]. In Italia, il Consorzio Interuniversitario Nazionale per l'Informatica (CINI) dal 2017 propone degli obiettivi che, fin dal primo ciclo d'istruzione, favoriscono la partecipazione a pieno titolo alla società digitale e lo sviluppo della creatività, dello spirito critico, e dell'autonomia intellettuale attraverso l'informatica [4].

Questo lavoro descrive un'esperienza sul campo basato su *learning by teaching* [7, 10] che, partendo dal principio "teaching others is a true enabler to learn" [12], mira all'acquisizione di competenze trasversali (digitali, linguistiche comunicative) e disciplinari (programmazione block based), puntando sulla naturale rielaborazioni delle informazioni e la necessaria semplificazione richieste in fase di teaching [16]. Il progetto inoltre propone la realizzazione di un *curricolo verticale delle competenze*, incluse quelle digitali e linguistiche, che coinvolge allievi del primo e secondo ciclo d'istruzione. Abbiamo osservato l'efficacia di *learning by teaching* chiedendo ad un gruppo di studenti del secondo anno di un Liceo Scientifico opzione Scienze Applicate di

insegnare i principi base della programmazione applicati alla Robotica Educativa a dei bambini di 6-10 anni.

Nei paragrafi successivi analizzeremo le motivazioni che hanno spinto la realizzazione del progetto, la sua implementazione, e le prospettive future.

2 Motivazioni

2.1 Insegnamento della Robotica Educativa

L'obiettivo primario di questo progetto è l'introduzione di una strategia didattica per l'insegnamento (nella scuola secondaria superiore) efficace e duraturo dei principi di programmazione applicati alla Robotica Educativa: comandi, sequenze di comandi, input/output, eventi (e funzionamento dei sensori), condizioni, cicli, e funzioni. L'ambizioso obiettivo di questa strategia didattica è quello di favorire l'apprendimento, promuovere la rielaborazione delle informazioni, e stimolare la curiosità rispetto alle nozioni apprese. A questo scopo, la colonna portante della strategia adottata è il learning by teaching [7, 10]: oltre ad acquisire i costrutti di base della programmazione, agli studenti viene richiesto di preparare il materiale didattico per prepararsi a spiegare i concetti appressi a bambini di 6-10 anni, sfida che richiede una profonda comprensione dei concetti di Robotica Educativa e l'abilità di rielaborare e semplificare la spiegazione in modo maggiore (e con grande responsabilità) rispetto alle regolari "interrogazioni" in classe.

Inoltre, motori principali del nostro progetto sono l'esigenza di sviluppare un *curricolo verticale* delle competenze e l'importanza del potenziamento linguistico, in un contesto plurilingue come quello dell'Alto Adige.

2.2 Curricolo verticale

Le indicazioni nazionali specificano che *"l'itinerario scolastico, pur abbracciando tipologie di scuola caratterizzate ciascuna da una specifica identità educativa e professionale, è progressivo e continuo"* [18], implicando una doverosa continuità di intenti tra primo e secondo ciclo d'istruzione. La progettazione verticale è infatti già presente negli istituti comprensivi, nei quali le linee guida prevedono la costruzione di un unico curricolo verticale, mentre è ancora poco sviluppato un curricolo verticale unico di raccordo tra il primo e il secondo ciclo. La presenza di un curricolo verticale tra i due cicli riveste tuttavia un ruolo chiave nello sviluppo delle competenze. Esso, infatti, permette di creare un quadro organico mettendo in evidenza gli elementi di continuità tra i saperi e promuovendo le competenze nel tempo [20]. Il curricolo verticale è infatti strutturato per singola competenza, che viene sviluppata dagli alunni e dalle alunne attraverso la progettazione didattica sul lungo periodo [20]; ciò significa che esso segue lo sviluppo verticale della competenza nell'alunno (secondo i livelli del QNQ [9]) ed è basato sul saper fare, punto cardine del nostro progetto.

L'attenzione alla progressività e la continuità dello sviluppo di competenze è un obiettivo strategico per l'Unione Europea e all'interno delle Raccomandazione del Consiglio dell'Unione Europea del 22 maggio 2018. Viene infatti affidato agli Stati Membri il compito di *"sostenere e rafforzare lo sviluppo delle competenze chiave per tutti, a partire dalla giovane età e durante tutto l'arco della vita, nel quadro delle strategie nazionali di apprendimento permanente"* [8], individuando otto competenze chiave, due delle quali sono obiettivi del nostro progetto:

1. *aumentare* il livello delle competenze linguistiche e fornire sostegno nell'apprendimento di lingue diverse che siano utili nella vita lavorativa e personale;

2. *innalzare e migliorare* il livello delle competenze digitali in tutte le fasi dell'istruzione e della formazione per tutti i segmenti della popolazione.

Uno sviluppo verticale delle competenze richiede l'utilizzo di metodologie di apprendimento innovative e non convenzionali, basate sulla ricerca-azione in compiti autentici, ovvero attraverso *"prove che mirano a richiamare contesti di realtà, diretti o simulati, nei quali utilizzare il proprio sapere per affrontare i problemi posti. Agendo così sul superamento del modello trasmissivo di insegnamento che rende difficile la personalizzazione dell'azione educativa ed alla base di molti fenomeni di abbandono e dispersione"* [1, 2].

2.3 Plurilinguismo

Il Consiglio d'Europa definisce il *plurilinguismo* come capacità dei parlanti di usare più di una lingua, considerando dunque le lingue dal punto di vista di coloro che le parlano e le apprendono; il termine *multilinguismo*, invece, rimanda alla presenza di più lingue in un'area geografica, indipendentemente da coloro che le parlano [3]. Questa differenza chiarisce l'importanza di apprendere le lingue parlate in un territorio plurilingue (come l'Alto Adige, area della nostra esperienza) in cui la seconda lingua L2 (italiano o tedesco a seconda del gruppo linguistico di appartenenza) e l'inglese L3 sono uno strumento fondamentale di comunicazione e un mezzo indispensabile per il motore economico. Per questo motivo l'insegnamento della seconda lingua e della lingua straniera, in Alto Adige, parte già dai primi anni del primo ciclo d'istruzione e continua fino al diploma. Per facilitare la comunicazione dei sempre più diffusi team di sviluppo software globali, è inoltre importante che le future generazioni di sviluppatori acquisiscano competenze di microlingua e sappiano utilizzare gli strumenti di comunicazione remota del lavorare agile [13].

Il Comitato dei Ministri del Consiglio d'Europa raccomanda *"l'uso del QCER come strumento per un'educazione plurilingue coerente, trasparente ed efficace che consente di promuovere la cittadinanza democratica, la coesione sociale e il dialogo interculturale"* [5]. Ed è proprio in questa direzione che è stato deciso di creare uno strumento di politica linguistica a livello europeo per mettere il discente al centro del processo di apprendimento e promuovere il plurilinguismo. Alla base del QCER c'è la domanda riguardante gli strumenti linguistici necessari per un determinato atto comunicativo e la ricerca di elementi per l'individuazione delle competenze da raggiungere in una lingua straniera. Per questo, infatti, al centro della didattica moderna delle lingue (L2 e L3) c'è l'individuo, che deve interagire in/con un dato contesto. Per compiere azioni comunicative, l'individuo necessita di determinate competenze, metodologie, e conoscenze linguistiche e non-linguistiche adeguate. Pertanto il lavoro in classe è basato su compiti (task) che hanno una valenza comunicativa e mirano al *lifelong learning* e alla competenza plurilingue. Lo stesso documento del Comitato dei Ministri del Consiglio d'Europa [5], che è stato integrato dal Companion Volume with New Descriptors [6], dedica una parte alla mediazione linguistica e alla competenza linguistico-comunicativa pragmatica, secondo l'approccio orientato al *fare*, seguendo il paradigma dei *learning by doing*.

Oltre alla competenza plurilingue vi è l'intento di un'educazione pluriculturale che passi attraverso l'interazione quotidiana tra individui di diverse culture e attraverso le (micro)lingue. Questo tipo di didattica è raccomandata come strumento per un'educazione plurilingue [5], per acquisire sin da bambini competenze di *microlingua*, ossia la terminologia specifica dei settori disciplinari e professionali [11]. Seguendo queste indicazioni, nel nostro progetto, il processo di apprendimento passa dall'acquisizione delle competenze nella microlingua in L1 (madrelingua), per poi essere integrate con la microlingua in L2/L3 attraverso la mediazione linguistica e la *"plain language"* [5, 15].

La mediazione linguistica si discosta dal concetto di traduzione letterale e verte sulla comunicazione/trasmissione di concetti acquisiti in (micro)lingua, nel nostro caso tra un discente esperto (in L2) e un discente/apprendente (in L1). Oltre alla comunicazione in microlingua, basata su connotati culturali, che si discosta sia dalla lingua quotidiana che dalla lingua di formazione, risulta importante la semplificazione linguistica di concetti acquisiti dal discente esperto. Questo doppio passaggio rappresenta anche verifica di comprensione e assimilazione dei contenuti. Per favorire questi processi di avvicinamento tra lingue e culture è necessario acquisire dunque competenze linguistiche e di microlingua sin da bambini, lì dove per microlingua si intende la terminologia specifica nei diversi settori disciplinari e professionali [11].

3 Esperienza sul campo

La Figura 1 mostra la struttura dell'attività didattica che è stata svolta con un gruppo di 12 studenti di seconda Liceo Scientifico Scienze Applicate (percorso quadriennale). La fase di *training* (circa 12 ore), è dedicata all'acquisizione dei costrutti di base della programmazione e dell'utilizzo del *racconto* come strumento didattico. In questa fase interviene l'insegnamento della microlingua nella propria madrelingua (L1) per acquisire la terminologia specifica e consolidare i concetti teorici. La microlingua in L2 e L3 non entra ancora in gioco, vista la complessità e la specificità dei contenuti disciplinari.

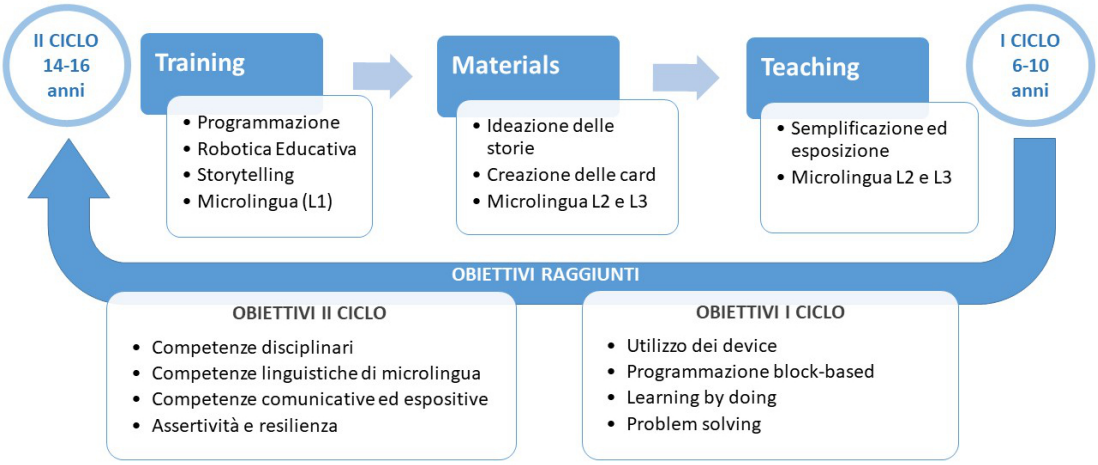


Figure 1: Schema dell'attività didattica. Nella parte superiore, le tre fasi del progetto e i loro obiettivi operativi. Nella parte inferiore, gli obiettivi disciplinari e trasversali raggiunti dagli allievi del primo e del secondo ciclo d'istruzione, gli uni grazie agli altri.

Durante l'esperienza sul campo, gli studenti hanno imparato a costruire e a programmare robot educativi LEGO-EV3, hanno appreso le fasi del racconto, ed hanno acquisito il linguaggio specifico (microlingua). Per ogni kit di sviluppo LEGO gli studenti avevano a disposizione motori e sensori per il movimento e l'input, ed un ambiente di sviluppo visuale. Gli studenti hanno lavorato in coppie ed hanno risolto problemi di difficoltà crescente che richiedevano di estendere le implementazioni precedenti. Dal semplice movimento temporizzato dei motori alle decisioni basate su stimoli esterni (sensori), dal movimento per raccogliere oggetti alla registrazione e riproduzione di suoni. Al termine del percorso tutti i partecipanti erano in

grado di costruire e programmare il proprio robot e risolvere problemi di semplice e media difficoltà (seguilinea, raccolta di oggetti, movimenti articolati, riproduzione di suoni).

Nella seconda fase (circa 8 ore), gli studenti realizzano i materiali didattici per il momento di teaching, ossia una serie di *card* in cui i bambini devono costruire e programmare i robot per completare dieci missioni in una storia fantastica. Viene introdotta la microlingua in L2 e L3 attraverso la mediazione linguistica nel lavoro di gruppo, con il supporto del docente di lingue.

La terza fase (circa 8 ore) è dedicata alla didattica. Nell'esperienza svolta, durante un festival delle scienze locale, gruppi di 2-3 studenti hanno insegnato ai bambini presenti (6-10 anni) utilizzando il materiale didattico prodotto. Essendo i bambini di madrelingua italiana e tedesca (e molti turisti stranieri), gli studenti dovevano riuscire a comunicare in maniera efficace nelle tre lingue. Anche in questa fase la mediazione linguistica e il *plain language* [15] sono state la modalità perseguite per una comunicazione efficace.

L'attività è stata portata a termine in maniera soddisfacente, tutti i partecipanti hanno dimostrato di aver acquisito le conoscenze e le competenze richieste dal progetto e tutti i bambini hanno imparato le parti fondamentali della programmazione, imparando a riprodurre autonomamente azioni di movimento e semplici operazioni decisionali con i propri robot.

È stato inoltre interessante osservare la variazione di registro nel passaggio da una lingua all'altra e il tentativo di semplificazione che continuamente si rendeva necessario, a conferma della concreta comprensione degli argomenti disciplinari proposti. La gratificazione degli studenti di liceo era visibile continuamente sui loro volti e il feedback di fine attività è stato molto positivo.

4 Conclusioni e lavori futuri

L'esperienza presentata in questo lavoro prevede una fase di training in cui i partecipanti acquisiscono le conoscenze e le competenze minime necessarie alla programmazione applicata alla Robotica Educativa. Il training avviene però nel contesto di una strategia didattica orientata alla creazione di un curricolo verticale, che stimoli ad approfondire i concetti, rielaborandoli in maniera chiara e semplificata, con la sfida aggiuntiva della microlingua in L2 e L3.

I risultati sono promettenti e suggeriscono di continuare in questa direzione, considerando anche che il *learning by teaching* sviluppa numerose competenze trasversali, favorendo la creazione di un clima meno giudicante (dove esporsi liberamente all'errore) e la crescita emozionale dell'alunno e la sua intelligenza emotiva [17]. Inoltre, poiché la didattica laboratoriale ed interdisciplinare supporta il contrasto alla dispersione implicita e favorisce l'inclusione [14], l'approccio descritto in questo lavoro dovrebbe essere applicato nei casi di studenti con bisogni educativi speciali o a rischio dispersione.

Come risultato rilevante di questo lavoro, la strutturazione dell'attività didattica (Figura 1) permette la riproducibilità e il raggiungimento degli obiettivi anche partendo da altri contenuti disciplinari. Ad esempio, si può immaginare la realizzazione di materiale didattico plurilingue per discipline come biologia e fisica, ma anche per quelle più professionalizzanti (come meccanica, elettronica, e chimica), coinvolgendo i docenti di L1 e di lingue straniere.

Infine, è possibile utilizzare questo approccio per progettare attività orientative, ormai diventate parte della proposta formativa della scuola italiana: gli istituti del secondo ciclo potrebbero proporre attività per gli alunni del primo ciclo durante le giornate di orientamento, porte aperte, festival delle scienze o simili. Attività esperienziali come quella proposta sono riconosciute anche all'interno delle recenti linee guide per l'orientamento pubblicate dal Ministero dell'Istruzione e del Merito [19], nel quale tra le modalità in cui è possibile svolgere le 30 ore di orientamento vengono individuati *"tutti quei laboratori che nascono dall'incontro tra studenti di un ciclo in-*

feriore e superiore per esperienze di peer tutoring, tra docenti del ciclo superiore e studenti del ciclo inferiore, per sperimentare attività di vario tipo, riconducibili alla didattica orientativa e laboratoriale" [19]. Le attività di orientamento e riorientamento devono infatti essere progressive durante i cicli di istruzione sviluppandosi ad esempio, all'interno di un curricolo verticale, quale strumento utile per individuazione dei talenti e delle attitudini degli alunni.

References

[1] Autorità Garante per l'Infanzia e l'Adolescenza. La dispersione scolastica in italia: un'analisi multifattoriale - documento di studio e di proposta, 2022. https://www.garanteinfanzia.org/sites/default/files/2022-06/dispersione-scolastica-2022.pdf.

[2] Mario Castodi. *Valutare e certificare le competenze*. Carocci ed., 2016.

[3] Sabrina Colombo, Maria Stopfner, and Flavia De Camillis. Plurilinguismo (dossier). Eurac researc [online], 2020. https://www.eurac.edu/it/dossiers/dossier-plurilinguismo-alto-adige#id-25302273.

[4] Consorzio Interuniversitario Nazionale per l'Informatica. Proposta di indicazioni nazionali per l'insegnamento dell'informatica nella scuola, 2017. https://www.consorzio-cini.it/index.php/it/component/attachments/download/745.

[5] Council of Europe. Common european framework of reference for languages (cefr) and the promotion of plurilingualism, 2008. https://search.coe.int/cm/Pages/result_details.aspx?ObjectId=09000016805d2fb1.

[6] Council of Europe. Common european framework of reference for languages (cefr): Learning, teaching, assessment, 2020. https://rm.coe.int/common-european-framework-of-reference-for-languages-learning-teaching/16809ea0d4.

[7] Amy Debbané, Ken Jen Lee, Jarvis Tse, and Edith Law. Learning by teaching: Key challenges and design implications. *Proceedings of the ACM on Human-Computer Interaction*, 7(CSCW1):1–34, 2023.

[8] European Education Area. Raccomandazione del consiglio relativa alle competenze chiave per l'apprendimento permanente, 2018. https://education.ec.europa.eu/focus-topics/improving-quality/key-competences.

[9] European Education Area. Quadro di riferimento nazionale delle qualifiche, 2023. https://eurydice.eacea.ec.europa.eu/it/national-education-systems/italy/quadro-di-riferimento-nazionale-delle-qualifiche.

[10] Logan Fiorella and Richard E. Mayer. *Learning by Teaching*, page 151–166. Cambridge University Press, 2015.

[11] Giovanni Freddi. *Psicolinguistica, sociolinguistica, glottodidattica. La formazione di base dell'insegnante di lingue e di lettere*. UTET Università, 1999.

[12] Ilenia Fronza, Luis Corral, Gennaro Iaccarino, and Claus Pahl. Enabling peer-led coding camps by creating a seed effect in young students. In *Proceedings of the 22nd Annual Conference on Information Technology Education*, SIGITE '21, page 117–122, New York, NY, USA, 2021. Association for Computing Machinery.

[13] Gennaro Iaccarino, Lucia Bartoli, Ilenia Fronza, and Luis Corral. PCTO per l'acquisizione di competenze di smart working. In *Proceedings of the Annual AICA Conference "DIDAMATICA 2021: Artificial Intelligence for Education". October 7-9, 2021, Palermo (Italy)*.

[14] Gennaro Iaccarino, Sara Tosi, and Ilenia Fronza. Didattica non convenzionale: un approccio inclusivo per la lotta alla dispersione implicita. In *Didattica e Inclusione Scolastica. Bressanone-Brixen (Italy)*., 2023. http://www.iisgalilei.eu/dis23/paper.pdf.

[15] ISO 24495-1:2023. Plain language — part 1: Governing principles and guidelines, 2023. https://www.iso.org/standard/78907.html.

[16] Aloysius Wei Lun Koh, Sze Chi Lee, and Stephen Wee Hun Lim. The learning benefits of teaching: A retrieval practice hypothesis. *Applied Cognitive Psychology*, 32(3):401–410, 2018.

[17] John D. Mayer and Peter Salovey. *Emotional Development and Emotional Intelligence: Implications for Educators (pp. 3–34). Basic Books.*, chapter What is emotional intelligence? 1997.

[18] Ministero dell'Istruzione e del Merito. Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione, 2012. https://www.miur.gov.it/documents/20182/51310/DM+254_2012.pdf.

[19] Ministero dell'Istruzione e del Merito. Linee guida per l'orientamento, 2023. https://www.miur.gov.it/documents/20182/6735034/linee+guida+orientamento-signed.pdf/d02014c6-4b76-7a11-9dbf-1dc9b495de38?version=1.0&t=1672213371208.

[20] Milena Ronzoni. Progettare il curricolo verticale per competenze e per assi culturali. Ministero dell'Istruzione e del Merito, 2020. https://www.istruzioneer.gov.it/wp-content/uploads/2020/02/Progettare-il-curricolo-verticale-per-competenze-e-per-assi-culturali.pdf.

Introduzione al *Coding Unplugged* per la Fascia 3–7 Anni con il The Coding Box Laptop

Lorena Cosatto[1], Eric Medvet[2,1]

[1] The Coding Box, Trieste, Italia
[2] Dipartimento di Ingegneria e Architettura, Università degli Studi di Trieste, Trieste, Italia

Sommario

Per una società con solide competenze digitali, è opportuno che l'informatica sia insegnata fin dalla scuola dell'infanzia. D'altra parte, è sconsigliabile che i bambini utilizzino troppo i dispositivi elettronici. In questo contributo, presentiamo un'attività giocosa ed educativa che permette di insegnare i concetti fondamentali del coding in maniera *unplugged*, cioè senza ricorrere a veri computer. L'attività si basa infatti su un finto laptop di legno che permette ai bambini di immedesimarsi nel ruolo di programmatore ed esecutore, in una sorta di gioco di ruolo.

1 Insegnare l'informatica: presto e *unplugged*

Il recente rapporto della rete Eurydice [7] evidenzia come in tutti i Paesi europei sia in corso un profondo processo di riforma dei curricoli scolastici per prevedere lo studio dell'informatica come materia a sé stante. L'Italia, con l'art. 24-bis Decreto Legge n. 152/2021, ha introdotto l'insegnamento del coding e della didattica digitale fin dalla scuola primaria. Nel Digital Education Action Plan 2021–27[1] si evidenzia, tuttavia, che per ottenere un ecosistema altamente efficiente di istruzione digitale e migliorare le competenze e le abilità per la trasformazione digitale è fondamentale introdurre l'insegnamento dell'informatica fin dalla scuola dell'infanzia. D'altra parte, illustri pedagogisti e centri di ricerca per l'educazione innovativa raccomandano di limitare l'esposizione ai dispositivi elettronici ai bambini dai 3 ai 6 anni [5, 9–11].

Per soddisfare entrambi i requisiti, cioè (a) insegnare l'informatica fin dalla scuola dell'infanzia e (b) non imporre l'uso di dispositivi elettronici, una possibilità è quella di ricorrere ad attività formative *unplugged* erogate secondo la metodologia dell'*edutainment* (educare giocando) [1]. In questo contributo, presentiamo un'attività didattica basata su uno strumento concreto, che ricorda un calcolatore portatile, ma non è un dispositivo elettronico.

L'attività qui presentata ha l'obiettivo di introdurre alcuni concetti basilari dell'informatica, in particolare quello della separazione dei ruoli fra chi programma e chi esegue, concetto chiave del *coding*: le due parti si accordano, in una sorta di gioco di ruolo [2, 3], per utilizzare un linguaggio non ambiguo in modo che le istruzioni date portino al risultato atteso. Proprio perché è unplugged, l'attività proposta esercita primariamente il pensiero computazionale [12] inteso come gli aspetti culturali e scientifici dell'informatica, a prescindere da quelli strumentali o tecnologici [6, 8].

2 Coding con il The Coding Box Laptop

L'attività che proponiamo è ideata per bambini dai 3 ai 7 anni. I partecipanti operano a coppie e in autonomia, ma supervisionati da un adulto animatore che fornisce loro una preparazione

[1] https://education.ec.europa.eu/focus-topics/digital-education/action-plan

 ISBN: 978-88-6629-075-9

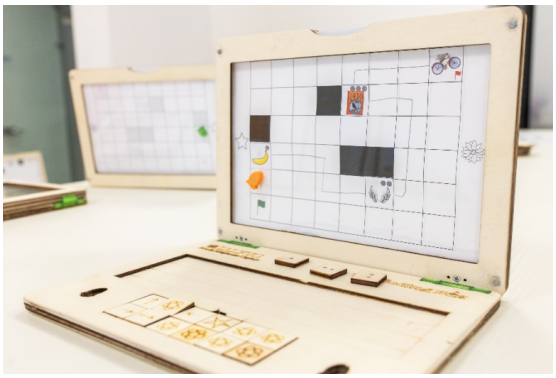


Figura 1: Il TCB-Laptop visto frontalmente con un foglio griglia già in posizione. Si distingue, in arancione, la pedina magnetica di Heki e, riposte "nel touchpad", le tessere istruzione. Sullo sfondo si intravede un altro TCB-Laptop visto da dietro, con la griglia non riempita, come la vede l'esecutore.

iniziale. Le coppie interagiscono utilizzando un finto computer portatile di legno, che chiamiamo The Coding Box Laptop (TCB-Laptop)[2], dal nome dell'associazione[3]. Per i bambini i giochi di imitazione sono un modo per indagare e capire il mondo che li circonda, in particolare quello degli adulti: utilizzando il TCB-Laptop essi "fanno finta di" essere dei veri programmatori.

Il TCB-Laptop è un oggetto delle dimensioni di un laptop da 15 pollici, con la base costituita da tre fogli di compensato incollati (Figura 1). Il primo foglio, più esterno, è intero; il secondo, centrale, presenta un foro in corrispondenza e con le dimensioni del touchpad, e due fori ai lati corti dell'ideale touchpad per ospitare due pedine magnetiche. Il terzo strato ricalca il secondo, ma presenta anche un foro rettangolare in corrispondenza dell'ideale tastiera. Una volta assemblato, si ottengono degli spazi contenitivi in cui posizionare delle tessere monostrato di compensato, quadrate, sulle quali abbiamo inciso al laser dei simboli corrispondenti alle istruzioni (freccia su, freccia giù, stella, fiore, alcuni disegni rappresentanti delle attività). Il monitor è costituito da una cornice in compensato, doppio foglio incollato, che sostiene due fogli in plastica semirigida sottile e trasparente, fra i quali è possibile inserire un foglio di carta sul quale è stampata la griglia di gioco. Abbiamo scelto di realizzare il nostro TCB-Laptop in compensato perché è un materiale di facile reperibilità, economico, leggero e resistente, adatto ad essere trasportato agilmente e manipolato in sicurezza anche da bambini piccoli. Abbiamo realizzato i primi TCB-Laptop, partendo da un prototipo in cartone, presso lo SciFabLab[4] dell'ICTP di Trieste, e i successivi in collaborazione con la Scuola di Artigiano Digitale dello IALFVG[5] di Pordenone.

L'attività consiste in due fasi. Nella prima, la coppia di bambini prepara assieme la griglia in modo che rappresenti uno scenario con alcuni elementi da raggiungere, secondo una storia suggerita dall'animatore. Nella seconda fase, uno dei due bambini assume il ruolo del programmatore e l'altro quello dell'esecutore: il primo, seduto di fronte al TCB-Laptop, dovrà comporre la sequenza di istruzioni che permette ad una pedina mobile di compiere la storia raccontata in precedenza; l'esecutore eseguirà materialmente le istruzioni muovendo la pedina.

[2] TCB-Laptop è stato ideato da Silvia Faion e Barbara Razzini con licenza CC BY-NC-ND 4.0.
[3] https://thecodingbox.org/
[4] http://scifablab.ictp.it/
[5] https://www.ialweb.it/

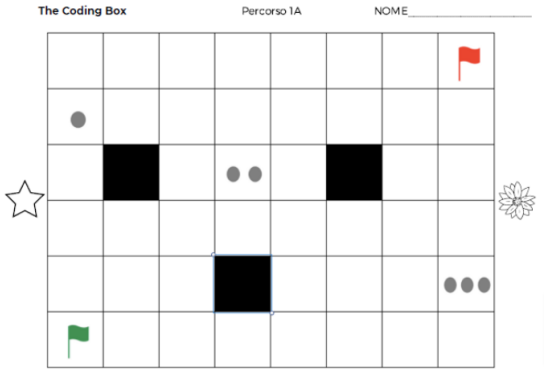


Figura 2: Un esempio di foglio griglia parzialmente compilato. Le caselle nere rappresentano degli ostacoli; le caselle con dei punti grigi degli elementi di interesse; le bandiere il punto di partenza e di arrivo.

Nel dettaglio, nella prima fase l'animatore:

1. Divide i bambini a coppie e li fa sedere uno di fronte all'altro.
2. Racconta ai bambini una storia in cui Heki (il personaggio rappresentato dalla pedina) dovrà compiere una missione, con dei passaggi intermedi prima di raggiungere il suo obiettivo. In questa fase l'animatore introduce il concetto di suddivisione del problema generale in sotto-problemi, facili da risolvere.
3. Consegna a ciascuna coppia il *foglio storia*, un prestampato dal quale i bambini ritaglieranno i disegni che descrivono le tappe dell'avventura di Heki.
4. Consegna ai bambini il *foglio griglia* sul quale è stampata, da ambo i lati, una griglia 6×8 con ai margini laterali rispettivamente una stella e un fiore (specularmente)[6]. I bambini devono incollare i disegni ritagliati in precedenza in diverse caselle della griglia, creando così le tappe dell'avventura di Heki (Figura 2). Se il tempo a disposizione è poco, l'animatore fornisce ai bambini il foglio griglia già riempito.
5. Fa tracciare ai bambini sulla griglia il percorso che Heki dovrà seguire. Questa fase corrisponde a quella di ideazione dell'algoritmo.
6. Consegna a ciascuna coppia un TCB-Laptop, con le tessere raffiguranti le istruzioni.
7. Inserisce nel monitor del TCB-Laptop il foglio griglia con la parte riempita rivolta verso la tastiera, cioè verso il programmatore.
8. Posiziona le due pedine Heki sulla prima casella del percorso, una dalla parte del programmatore e l'altra dalla parte dell'esecutore: quando una viene mossa, l'altra la segue, grazie al magnete di cui entrambe sono dotate.

La fase iniziale dello *storytelling* è di fondamentale importanza: il discorso narrativo rende comprensibile, comunicabile e facilmente memorizzabile il concetto che vogliamo trasmettere,

[6] Poiché i bambini di questa fascia d'età faticano a distinguere la destra dalla sinistra, abbiamo scelto di utilizzare il simbolo di *stella* e *fiore* per indicare gli spostamenti laterali; inoltre essendo le griglie sui due lati del foglio speculari, si risolve il problema di distinguere la posizione rispetto a sé e rispetto all'altro.

nel nostro caso i fondamenti del pensiero computazionale [4]. Coinvolgendo la fantasia dei bambini si possono creare avventure sempre nuove e via via più complesse, offrendo l'occasione di introdurre gradualmente concetti più elaborati: istruzioni, condizioni, sequenze, algoritmi.

La seconda fase dell'attività si svolge come segue:

1. Il programmatore sceglie la tessera istruzione, la posiziona sulla tastiera, e comunica all'esecutore cosa fare.

2. L'esecutore trasla la sua pedina Heki lungo la griglia, rispettando rigorosamente le istruzioni ricevute. Quando il personaggio Heki si trova su una casella speciale (ad esempio un elemento cibo), il programmatore deve utilizzare il tassello comando e l'esecutore deve mimare l'azione richiesta (ad esempio mangiare il cibo).

3. Il programmatore vede il personaggio muoversi lungo il percorso, verificando immediatamente se il comando dato ha avuto l'esito atteso.

4. Se tutto è stato fatto correttamente, si passa al comando successivo (ripartendo da 1), altrimenti il programmatore dovrà identificare e correggere l'errore: abbiamo quindi la possibilità di introdurre il concetto di *debugging*.

Alla fine del percorso, nello spazio tastiera i bambini potranno osservare il codice da loro scritto per far compiere a Heki la sua missione: sono dei veri programmatori!

La durata stimata di una attività con TCB-Laptop è di 2 ore, di cui 75 minuti di gioco, 20 minuti per l'allestimento, 15 minuti per il disallestimento. Dalla nostra esperienza, il buon esito dell'esperienza dipende dalla capacità di organizzare lo spazio e il lavoro in modo che i bambini siano protagonisti e abbiano sempre qualcosa di semplice da fare. Per svolgere i passi delle prima fase, mettiamo a loro disposizione dei materiali che conoscono bene, come forbici, colla e matite, che vengono consegnati e recuperati immediatamente quando non servono più, in modo che i bambini possano restare concentrati sull'attività, senza distrazioni.

3 Conclusioni e spunti futuri

Dal 2019 TCB sta presentando il TCB-Laptop nelle scuole, in centri estivi per ragazzi, in eventi e fiere dedicate alla divulgazione scientifica e delle materie STEM, raggiungendo più di 700 persone, fra bambini e insegnanti, senza contare i genitori. Solo nel 2023 abbiamo svolto 40 laboratori nelle scuole dell'infanzia della provincia di Pordenone, coinvolgendo 160 bambini di età compresa fra i 3 e i 5 anni. Sempre nel 2023 The Coding Box, in collaborazione con SISSA Media Lab[7] di Trieste, ha promosso e realizzato un ciclo di lezioni dedicato agli insegnanti della scuola dell'infanzia, a cui hanno partecipato 20 insegnanti.

Al momento disponiamo di un numero ridotto di TCB-Laptop, quindi possiamo lavorare con gruppi di non più di 20 bambini, ma abbiamo rilevato che in presenza di gruppi più numerosi l'attività diventa troppo rumorosa e dispersiva. Fra le istruzioni introdotte ed utilizzate manca il comando "ruota", perché la difficoltà a distinguere la destra dalla sinistra crea sempre molta confusione, deconcentrando i bambini che così perdono interesse nel gioco. I bambini più grandi intuiscono subito le potenzialità del gioco e spontaneamente chiedono di introdurre ostacoli, deviazioni, condizioni per complicare il percorso di Heki, accogliendo con grande entusiasmo le spiegazioni di istruzioni più complesse. Per come sono progettati e costruiti, i TCB-Laptop sono facilmente espandibili per accomodare esigenze più sofisticate.

[7] https://medialab.sissa.it/

Ringraziamenti

Ringraziamo Silvia Faion, Barbara Razzini e gli altri soci attivi dell'associazione The Coding Box per l'aiuto nella raccolta di informazioni necessarie per la stesura di questo documento.

Riferimenti

[1] Oksana V Anikina and Elena V Yakimenko. Edutainment as a modern technology of education. *Procedia-Social and Behavioral Sciences*, 166:475–479, 2015.

[2] Alessandro Bogliolo. Unplugged language-neutral card games as an inclusive instrument to develop computational thinking skills. In *INTED2015 Proceedings*, pages 7609–7615. IATED, 2015.

[3] Alessandro Bogliolo. *A scuola con CodyRoby. Il coding come gioco di ruolo.* Idee e strumenti. Giunti Scuola, 2020. ISBN 9788809901964.

[4] Quinn Burke and Yasmin B Kafai. The writers' workshop for youth programmers: digital storytelling with scratch in middle school classrooms. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 433–438, 2012.

[5] Julie Hooft Graafland. New technologies and 21st century children: Recent trends and outcomes. 2018.

[6] Michael Lodi, Simone Martini, and Enrico Nardelli. Abbiamo davvero bisogno del pensiero computazionale? *Mondo digitale*, (72):1–15, 2017.

[7] Jean Monnet. Informatics education at school in Europe. 2022.

[8] Enrico Nardelli et al. Informatica nella scuola: disciplina fondamentale e trasversale, ovvero "di cosa parliamo quando parliamo di pensiero computazionale"'. *Scienze e Ricerche Magazine*, 47:36–40, 2017.

[9] Council on Communications, Media, Victor C Strasburger, Marjorie J Hogan, Deborah Ann Mulligan, Nusheen Ameenuddin, Dimitri A Christakis, Corinn Cross, Daniel B Fagbuyi, David L Hill, Alanna Estin Levine, et al. Children, adolescents, and the media. *Pediatrics*, 132(5):958–961, 2013.

[10] Serge Tisseron. *3-6-9-12 Apprivoiser les écrans et grandir.* Erès, 2013.

[11] Burns Tracey, Gottschalk Francesca, et al. *Educational Research and Innovation Education in the Digital Age Healthy and Happy Children: Healthy and Happy Children.* OECD Publishing, 2020.

[12] Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

Videogiocando con Scratch

Daniela Troia
1° Circolo Didattico "G. Marconi"
dnltr71@gmail.com

Abstract

Il racconto descrive un'esperienza realizzata con due classi terze di scuola primaria, finalizzata ad un'introduzione più sistematica dei concetti legati al pensiero computazionale. Il progetto, di natura trasversale, ha previsto l'uso di Scratch 3.0 e si è posto come tentativo di collocare il coding nella didattica scuola primaria riconoscendone le peculiarità legate all'informatica, nell'ambito della più ampia competenza digitale.

1.Introduzione

Nel panorama scolastico italiano, si sono succedute fasi alterne finalizzate all'introduzione di "INFORMATICA", "TECNOLOGIE", "MULTIMEDIALITA'" e "DIGITALE", fino all'avvento del Coding e della Robotica Educativa.

Questi ultimi sono legati all'azione #17 del PNSD che, con i Nuovi Scenari del 2018, ha introdotto nella scuola primaria l'espressione **"Pensiero Computazionale"**, avviando un processo di innovazione metodologico-didattica che ha portato i docenti a barcamenarsi tra corsi e autoformazione, alla ricerca della strada più corretta da seguire.

Contestualmente il Consiglio Europeo ha ribadito il valore della **Competenza Digitale** tra le Competenze Chiave del 2018, mentre il Digicomp Edu ha delineato il framework di riferimento delle competenze digitali in campo educativo.

Come orientarsi in un contesto così variegato? Come mettere in relazione la competenza digitale e la promozione del pensiero computazionale? E quale il ruolo dell'informatica, cui mancano riferimenti per la scuola primaria?

Il racconto dell'esperienza "Videogiocando con Scratch" si propone di dare una risposta a questi interrogativi, nel tentativo di cogliere il senso e il valore del Pensiero Computazionale nella didattica della scuola, e della scuola primaria in particolare.

2.Premessa

L'esperienza ha visto coinvolti 40 alunni di classe terza che sin dalla prima avevano sperimentato attività di coding e robotica, in modalità unplugged attraverso attività su scacchiera, carta e con oggetti programmabili.

Il passaggio dall'unplugged al digitale è stato accelerato dall'emergenza pandemica da Covid-19 che ha favorito l'utilizzo di piattaforme come Code.org o Scratchjr.

Le attività su Code.org hanno consentito di avvicinarli in modo graduale alla programmazione visuale a blocchi, all'interno di percorsi strutturati; quelle su Scratchjr hanno permesso loro di utilizzare le prime acquisizioni in modo creativo.

Le attività relative all'esperienza "Videogiocando con Scratch" sono state realizzate l'anno successivo a distanza, utilizzando Scratch3.0 per un'introduzione più mirata e sistematica dei concetti legati al pensiero computazionale.

3.Percorso

In questa sezione sarà descritto il contesto e le fasi di attuazione.

3.1 Il contesto e l'argomento

L'esperienza ha coinvolto discipline STEM e NON STEM: scienze, geografia, ed. civica, italiano, e indirettamente matematica.

Tema centrale il MARE, parte di un più ampio percorso sull'ACQUA previsto nel PTOF, che ha visto i bambini impegnati nella costruzione di un museo virtuale in Minecraft E.E.

Prendendo ispirazione dal kit didattico "Il mare in 3D", contestualizzato nel racconto "The snail and the whale" di J.Donaldson, è stata realizzata un'esperienza finalizzata alla costruzione di progetti con Scratch 3.0, motivata dal tentativo di affiancare allo sviluppo della competenza digitale, la promozione del pensiero computazionale, in linea con le indicazioni del curricolo digitale di istituto progettato secondo le aree del DigicompEdu ed articolato in 3 aree didattiche: tecnologia digitale, pensiero computazionale e cittadinanza digitale.

3.2 Riferimenti pedagogici e metodologici

All'interno di una cornice pedagogica di stampo costruttivista e costruzionista, ispirata ai principi di Papert, si è scelto un approccio metodologico di tipo induttivo e laboratoriale in cui fosse centrale il problem solving, inteso come strumento privilegiato per favorire la costruzione di processi mentali di tipo algoritmico e logico.

Seguendo la teoria delle 4P di Mitchel Resnick, ciascuna fase è stata improntata alla logica della spirale dell'apprendimento creativo, chiedendo ai bambini di immaginare soluzioni a problemi attraverso la decomposizione in sotto problemi, la progettazione di sequenze di passi ordinati per la loro risoluzione, il test delle stesse tramite un esecutore, la riflessione su quanto realizzato come avvio della fase successiva.

3.3 Ambiente di apprendimento

L'infrastruttura dell'ambiente di apprendimento è stata caratterizzata dall'uso simultaneo di due piattaforme:

- Microsoft Teams, per le attività sincrone e asincrone
- Scratch3.0, per la costruzione dei prodotti digitali

Il cortometraggio "The Snail and the Whale" è stato suddiviso in sequenze corrispondenti alle fasi del progetto

1^ FASE: storytelling su MARE ED EMOZIONI
2^ FASE: videogioco sulla SALINITA' DEL MARE
3^ FASE: quiz su LUCE E VITA NEL MARE
4^ FASE: videogioco multiplayer su LA SALVAGUARDIA DEL MARE

ed ha rappresentato lo scenario funzionale all'introduzione di specifici obiettivi disciplinari e concetti informatici:

- Eventi
- Sequenze di istruzioni
- Cicli
- Variabili
- Sottoprogrammi

Ciascuna fase è stata poi articolata in vari step: partendo da un momento di engage, si è chiesto di investigare su un particolare aspetto relativo al MARE, che sarebbe poi diventato il tema del progetto da realizzare con Scratch.

Sebbene si fosse a distanza, è stato possibile diversificare i setting d'aula "virtuale": i momenti collettivi di engage sono stati realizzati nella stanza virtuale della riunione principale; il lavoro individuale attraverso l'accesso al quaderno digitale precaricato nel TEAM di classe, quello per piccoli gruppi su Scratch attraverso le break out rooms.

3.4 Descrizione delle fasi

Per iniziare, è stata creata una classe su Scratch e i bambini hanno potuto loggarsi come studenti. Per tenere traccia dei progetti, ne è stato creato uno vuoto che ciascuno ha remixato; è stata inoltre creata una galleria di condivisione.

Prima di passare alla 1^ fase del progetto, è stata analizzata la piattaforma Scratch 3.0 nei suoi elementi essenziali: sezione dei blocchi, area di programmazione, stage, area degli sfondi e degli sprite.

1^ FASE: storytelling "Io e il mare"

Concetti da introdurre:

- Eventi
- Sequenze di istruzioni

In questo primo lavoro, il link da remixare per realizzare lo storytelling conteneva le indicazioni da seguire: lo sfondo e il numero di sprite da inserire, le azioni da far compiere a ciascuno.

Partendo dalle prime sequenze del cortometraggio, si è chiesto ai bambini di riflettere sulle emozioni personali legate al mare e di annotarle nel quaderno digitale; quindi di riutilizzare i protagonisti del corto (la balena e la lumaca) per realizzare un semplice storytelling in cui uno degli sprite rappresentasse se stessi e le proprie emozioni.

Si sono presentati primi semplici problemi: (Fig.1)

- utilizzo di eventi diversi, per l'avvio del progetto o per la comunicazione tra gli sprite;
- successioni ordinate di istruzioni funzionali al corretto funzionamento del programma.

Figura 1: eventi ed istruzioni

2^ FASE: videogioco "Salata come il mare"

Concetti da introdurre:

- Eventi
- Istruzioni condizionali
- Sensori
- Operatori
- Variabili
- Cicli

Si è quindi entrati nel vivo delle discipline, attraverso un'esplorazione dei mari utilizzando Google Maps; si è chiesto ai bambini di annotare nel quaderno digitale riflessioni individuali sulle caratteristiche del mare legate ad esperienze personali; queste sono state poi confrontate con approfondimenti e verificate attraverso esperimenti scientifici sulle soluzioni ed infine si è proposto di realizzare un semplice videogioco su quanto appreso.

Questa volta la consegna conteneva solo l'obiettivo finale: realizzare un videogioco che simulasse la salinità del Mar Mediterraneo.

Un problema complesso come quello relativo alla costruzione di un videogioco è stato suddiviso in vari step: (Fig.2)

- usare eventi che consentissero l'interazione con la tastiera;
- introdurre una variabile per incrementare il punteggio o per il cont down del tempo;
- usare i sensori per stabilire un'interazione tra gli sprite;
- introdurre le condizioni per stabilire quando assegnare il punto, quando terminare il gioco definendo lo "stop" per la vittoria in funzione del punteggio o la sconfitta in funzione del tempo.

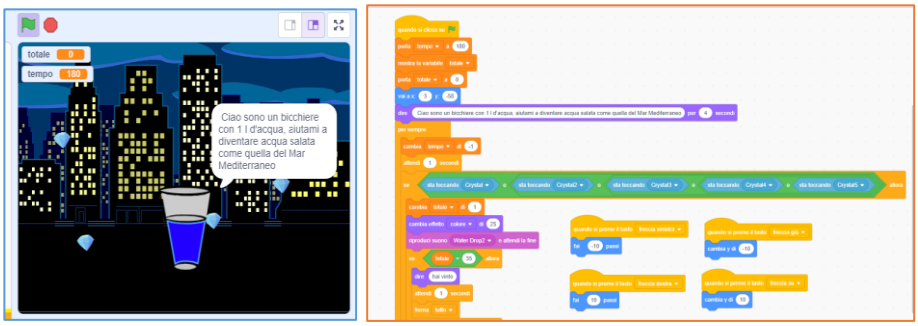


Figura 2: variabili, sensori, istruzioni condizionali, cicli

3^ FASE: quiz "Nel profondo blu"

Concetti da introdurre:

- istruzioni condizionali
- sottoprogrammi

Le attività di scienze questa volta sono state centrate sul rapporto tra luce e mare e sulla vita marina.

Dopo la visione delle sequenze del cortometraggio ambientate nei fondali marini, sono state proposte domande stimolo legate alle esperienze personali confrontate successivamente con approfondimenti e ricerche e consolidate anche attraverso semplici attività sul quaderno digitale.

Obiettivo: programmare un quiz per la verifica delle conoscenze sul mare.

Data la complessità del lavoro, si è preferito suddividerlo in due momenti distinti:

- il mare e la luce
- il mare e la vita sott'acqua

In questa fase è stato necessario ragionare sulla necessità di prevedere feedback diversi in base alla risposta data nonché programmare su due argomenti distinti e quindi: (Fig.3)

- stabilire quale feedback dare e come gestire il punteggio in caso di condizione vera o falsa;

- introdurre sottoprogrammi da richiamare in quello principale.

Figura 3: istruzioni condizionali e sottoprogrammi

4^ FASE: videogioco multiplayer "The whale and the snail"

Quest'ultima fase è stata centrata sugli obiettivi 6 e 14 dell'Agenda 2030 ed è stata intesa come **verifica**, con un reimpiego di quanto appreso precedentemente, ampliando solo la possibilità di giocare con un avversario: la problematizzazione ha perciò riguardato la generalizzazione, ossia la possibilità di riutilizzo delle conoscenze note in un contesto diverso.

In un'ottica formativa, la **valutazione** è stata effettuata dall'insegnante girando nelle break out rooms e fornendo feedback continui ai bambini.

Le attività propedeutiche alla costruzione del progetto sono state centrate su cause, conseguenze e rimedi relativi all'inquinamento del mare per giungere all'analisi degli obiettivi dell'Agenda 2030 e all'ideazione di un videogioco a tema.

I bambini hanno programmato un gioco multiplayer finalizzato alla salvaguardia del mare in cui la balena e la lumaca fossero impegnate nella pulizia rispettivamente delle acque e della spiaggia. (Fig.4)

I problemi da risolvere in questa fase hanno riguardato:

- la gestione della programmazione degli sprite per i due giocatori
- la delimitazione dei "campi di gioco" mare/balena – spiaggia/lumaca.

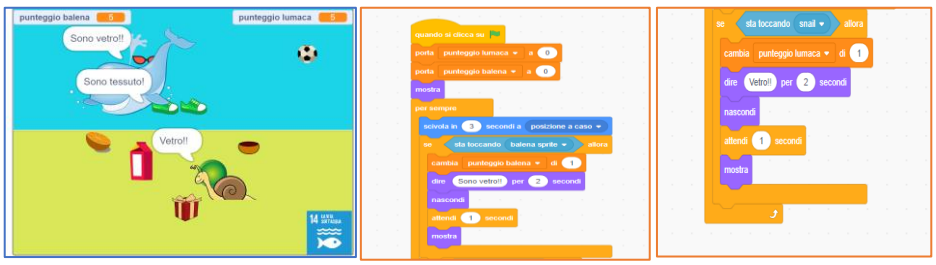


Figura 4: videogioco multiplayer

4. Conclusione

Il percorso è stato complesso, ma ha visto i bambini impegnati in tutte le fasi: dall'analisi del problema alla progettazione ed esecuzione dell'algoritmo, con relativo test e debugging.

Numerosi i momenti di difficoltà legati anche all'approccio centrato sulla scoperta autonoma delle modalità possibili di risoluzione dei problemi e di utilizzo dei blocchi di programmazione, ma l'impegno è stato costante poiché resi protagonisti attivi della costruzione dei propri artefatti, orientanti nel percorso risolutivo dall'insegnante che ha svolto il ruolo di facilitatore.

Sono stati promossi processi mentali di tipo logico ed algoritmico, centrati sulla scomposizione, astrazione e generalizzazione, ma anche pratiche legate al test e alla correzione dell'errore o al riuso di pezzi di codice.

Facendo riferimento alle aree del DigComp Edu su cui è progettato il curricolo digitale, la costruzione dei progetti ha consentito di potenziare le competenze dei bambini in tutte le aree didattiche in cui esso si articola.

Innanzitutto, i bambini hanno dimostrato di conoscere ed utilizzare la tecnologia in un contesto di sviluppo del pensiero computazionale in modo autonomo, creativo e personale, ma anche in termini tecnici di uso di hardware e software e di cittadinanza digitale, promuovendone un uso critico e responsabile.

In particolare, hanno imparato a cercare informazioni provenienti da fonti digitali, nonché a comunicare in ambienti digitali, condividere risorse, collegarsi con gli altri e collaborare utilizzando gli strumenti digitali messi a disposizione. (Informazione e Comunicazione).

Attraverso la creazione di progetti interattivi, hanno sviluppato abilità nella produzione di contenuti multimediali e sono stati introdotti primi cenni relativi alla conoscenza dei diritti di proprietà intellettuale e licenze. (Creazione di Contenuti Digitali)

Il ricorso a piattaforme on – line ha consentito di sensibilizzare i bambini sulle questioni di sicurezza online e sull'importanza di proteggere la propria identità e la privacy durante l'utilizzo di strumenti digitali. (Sicurezza Digitale).

Attraverso la creazione di progetti originali in Scratch, hanno potenziato il pensiero critico e sviluppato la creatività nel risolvere problemi e presentare idee in modo innovativo. (Problem Solving e Pensiero Computazionale).

Le attività svolte, contestualizzate nel curricolo delle diverse discipline, hanno dato un senso ai contenuti disciplinari in quanto funzionali alla realizzazione dei propri artefatti.

Il lavoro di gruppo ha inciso positivamente sulle competenze relazionali e comunicative, favorendo processi di inclusione e team working.

Volendo pertanto provare a dare una risposta all'interrogativo iniziale, sarebbe possibile affermare che una proposta possibile potrebbe essere quella di usare a scuola tecnologia e digitale **per la costruzione** e non solo per la fruizione, in quanto un tale approccio riuscirebbe a tener conto di tutte le dimensioni: da quella più strettamente disciplinare a quella cognitiva legata al pensiero computazionale fino a competenze più trasversali e soft skills di comunicazione, collaborazione, meta cognizione, insegnando a diventare resilienti e creativi.

Tecnologia e digitale, in tal modo, troverebbero una loro legittimazione a fianco del nucleo scientifico rappresentato dai concetti informatici afferenti al pensiero computazionale.

Bibliografia

[1] Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.

[2] Mitchel Resnick, 2018, Come i bambini. Immagina, crea e condividi. Coltivare la creatività con il Lifelong Kindergarten del MIT. Trento. Eickson

[3] Michael Lodi, Simone Martini, Enrico Nardelli. Abbiamo davvero bisogno del pensiero computazionale? Mondo Digitale, 2017, 72, pp.1-15. ffhal-01656340f

[4] Rita Marchignoli, Michael Lodi, 2016 EAS e pensiero computazionale. Fare coding nella scuola primaria. La Scuola

[5] MIUR. (2012). Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione. indicazioni_nazionali_infanzia_primo_ciclo_definitiva (miur.gov.it)

[6] MIUR. (2018). Indicazioni Nazionali e Nuovi Scenari. miur.gov.it/documents/20182/0/Indicazioni+nazionali+e+nuovi+scenari/

[7] RACCOMANDAZIONE DEL CONSIGLIO del 22 maggio 2018 relativa alle competenze chiave per l'apprendimento permanente.

[8] Bocconi, S., Earp, J., and Panesi S. (2018). DigCompEdu. Il quadro di riferimento europeo sulle competenze digitali dei docenti. Istituto per le Tecnologie Didattiche, Consiglio Nazionale delle Ricerche (CNR). DOI: https://doi.org/10.17471/54008
[9] Piano Nazionale Scuola Digitale, La scuola digitale - MIUR (istruzione.it)

Sitografia

[1] Programma il Futuro - Code.org - ProgrammaIlFuturo.it
[2] Scratch - Imagine, Program, Share (mit.edu)
[3] EarlyCode (earlycoders.org)
[4] https://www.raiplay.it/video/2021/10/la-chiocciolina-e-la-balena-48bfd07c-6a9b-4538-88ef-c07d25f82967.html
[5] Il Mare in 3D – Kit didattico - Scuola di Robotica

Coding e ginnastica nella scuola dell'infanzia

Silvia Medici[1], Maria Cecilia Verri[2]

[1] Ssdrl UnDueTre sport presso Istituto Suore Calasanziane
Via delle Centostelle, 9 - Firenze
smedici957@gmail.com
[2] Dipartimento di Statistica, Informatica, Applicazioni
Università degli Studi di Firenze
Viale Morgagni 65 - 50134 Firenze
mariacecilia.verri@unifi.it

Sommario

Nella scuola dell'infanzia una attenzione molto particolare viene data all'educazione al movimento per sviluppare la piena percezione del corpo nello spazio e la coordinazione dei movimenti. La realizzazione di questi obiettivi può conciliarsi con lo sviluppo del pensiero algoritmico quando ogni singolo movimento viene codificato con un simbolo e il movimento viene finalizzato al raggiungimento di un obiettivo. Tramite le attività proposte è stato possibile introdurre concetti informatici come il problem solving, la proceduralizzazione di una soluzione, la descrizione di una procedura tramite simboli ed il confronto di complessità di soluzioni, a bambini e bambine di 5 anni durante le lezioni di educazione fisica.

1 Introduzione

Il progetto *Coding e ginnastica* è nato, quasi per caso, dall'incontro di una insegnante di ginnastica di una scuola per l'infanzia con il mondo del coding e dal suo confronto con una informatica interessata alla didattica della disciplina.

Il mondo della ginnastica e quello dell'informatica sembrano molto distanti e, a prima vista, non si direbbe che abbiano niente in comune. In particolare, lo stereotipo dell'informatico, nerd, asociale e chiuso in sé stesso, poco ha a che fare con quello dell'atleta. Ma lasciarsi guidare dagli sterotipi è un grosso errore e una analisi più attenta e approfondita permette di rivelare affinità insospettate tra questi due mondi.

Partiamo dal Coding. Questo termine, che in italiano può essere tradotto come codificare, indica una attività che consente a bambini e bambine di sviluppare il pensiero computazionale, cioè l'attitudine a risolvere problemi più o meno complessi scomponendoli nei vari aspetti che li caratterizzano, pianificando per ognuno le soluzioni più idonee attraverso piccoli obiettivi intermedi. Il pensiero computazionale aiuta, infatti, a trovare la soluzione migliore, l'algoritmo più efficiente, per raggiungere un traguardo finale attraverso un metodo efficace e divertente che allena la capacità mentale nel risolvere i problemi.

E dove c'è da allenare, come può mancare la ginnastica, quella vera? Tutto questo, non è alla base anche dell'educazione fisica? Si pensi alla definizione di esercizio fisico introdotta da E. Baumann all'inizio del secolo scorso: "atto motorio voluto e precisato" [1]. L'educazione fisica, in particolare nella scuola dell'infanzia, ha proprio l'obiettivo di affinare il movimento, scomporlo in una sequenza di azioni più semplici, renderlo sempre più preciso per il raggiungimento di un traguardo.

A questo punto le affinità tra le due discipline hanno iniziato ad emergere e con le affinità, sono emersi ulteriori interrogativi: come trasformare gli esercizi ginnici in un algoritmo? Quali strumenti sono disponibili per portare la ginnastica nel mondo digitale? E, infine: si potrà veramente fare?

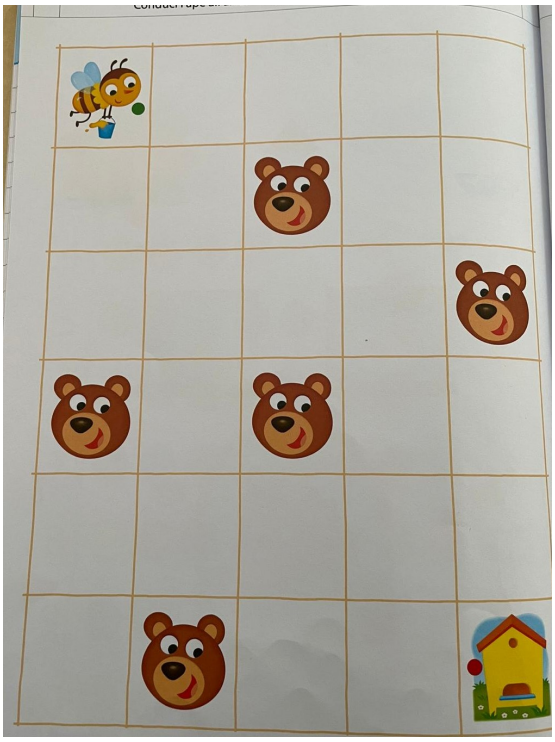


Figura 1: attività di coding proposte alla scuola dell'infanzia

Fondamentali sono stati, a questo scopo l'esperienza nella ginnastica e in particolare in quella per infanti, la curiosità verso il mondo dell'informatica, la voglia di mettersi in gioco e di sperimentare e, grande alleata, una scacchiera gigante.

Le attività di coding che tradizionalmente vengono proposte nella scuola dell'infanzia, riguardano la risoluzione di semplici circuiti come quello riportato in Figura 1: se l'ape deve portare il suo miele all'alveare senza cadere fra le zampe dell'orso, qual è il percorso più sicuro? Non solo, qual è quello più efficace? Queste attività vengono proposte in modo statico, lavorando su schede cartacee, rese accattivanti con l'uso di immagini, ma pur sempre da risolvere seduti ad un tavolino.

Prendendo spunto dalla descrizione di attività presenti on line (si veda, ad esempio, [2] e [3]), è nata l'idea alla base del progetto: trasferire l'attività dalla pagina di un libro ad una enorme scacchiera (4x4 metri) distesa sul pavimento della palestra, in modo da unire l'atto motorio alla capacità di ragionamento (vedi Figura 2).

2 Descrizione del progetto

Il progetto è stato ideato da Silvia Medici, insegnante di educazione fisica presso la scuola dell'infanzia Suore Calasanziane di Firenze, e realizzato con i bambini e le bambine della classe di cinque anni nell'anno 2021, con il supporto, morale e informatico, di M. Cecilia Verri, docente di Informatica all'Università di Firenze.

Figura 2: dalla pagina del libro alla scacchiera gigante

Silvia ha intuito la possibile connessione tra le attività svolte in palestra e finalizzate all'apprendimento della lateralizzazione e del movimento preciso e controllato, con le schede sul coding che i bambini e le bambine compilavano sui loro libri. Parlandone con Cecilia, è emersa la profonda analogia tra le due attività e la possibilità di codificare i movimenti, finalizzarli ad un obiettivo da raggiungere e, addirittura, cercare di ottimizzare il numero di azioni per il raggiungimento del risultato.

Il percorso è durato circa quattro mesi, caratterizzato da prove, tentativi e perché no, qualche fallimento che ha permesso di raddrizzare il tiro.

La programmazione educativa è stata impostata sulla base di quattro obiettivi generali estrapolati dalle indicazioni nazionali. Educazione fisica e informatica sono effettivamente sottogruppi di un unico capitolo: "Gli strumenti culturali per la cittadinanza", che avvicinano così, anche sulla carta, questi due mondi apparentemente lontani.

Gli obiettivi primari sono:

- percezione del proprio corpo nello spazio
- capacità di movimento
- sviluppo della coordinazione
- introduzione alla lateralità

Dei quattro mesi di durata del progetto, almeno tre sono stati impiegati esclusivamente a lavorare sul concetto di esercizio, a cercare quella precisione e quella volontà insita in ogni atto motorio. Nella classe di circa 23 fra bambini e bambine di 5 anni, ultimo anno della scuola dell'infanzia, convivono molte realtà: egocentrici, timidi, iperattivi, insicuri, scoordinati, sportivi, sovrappeso e altro ancora. Ma tutti devono avere le stesse opportunità, la classe deve proseguire il più possibile compatta. Quindi le prime attività si sono concentrate sul movimento fine, lo spostamento tra una casella e l'altra della scacchiera senza rimanere a cavallo di due celle, i movimenti consentiti e quelli proibiti, riconoscere la destra e la sinistra, individuare gli ostacoli e le celle libere.

I bambini e le bambine hanno appreso le regole del gioco, ovvero i vincoli di movimento. Da una posizione su una cella della scacchiera è possibile fare una sola delle tre seguenti azioni:

- avanzare con un passo sulla cella immediatamente davanti, se questa è libera, ovvero non occupata da un ostacolo;
- rimanere nella stessa cella ruotando a destra di 90°,
- rimanere nella stessa cella ruotando a sinistra di 90°.

Ciascuna di queste azioni è considerata un movimento e quindi, nel calcolo della complessità della soluzione, verrà conteggiata come una unità.

Naturalmente, avendo a che fare con bambini e bambine di 5 anni, è stato necessario catturare la loro attenzione e, soprattutto, mantenerla durante tutto lo svolgimento dell'attività: la scacchiera è così diventata un castello, diviso in stanze bianche e nere separate da muri invisibili che possono essere attraversati ma su cui non ci si può fermare. In alcune stanze possono esserci dei fantasmi e quindi vanno evitate, in altre invece c'è un tesoro che va conquistato. La narrazione, che prosegue e si arricchisce di lezione in lezione, ha consentito di mantenere sempre coinvolti i partecipanti nel gioco.

All'interno di queste narrazioni e durante la scoperta del percorso verso il tesoro, si introducono alcune basi del pensiero computazionale:

- istruzione condizionale: *if* nella stanza davanti c'è un fantasma *then* ruota a destra o a sinistra *else* fai un passo avanti;
- backtracking: se hai raggiunto una stanza circondata da fantasmi, prova a tornare sui tuoi passi per cercare un altro percorso;
- ricerca del cammino minimo: quale percorso richiede il minor numero di movimenti per raggiungere il tesoro?

Il passo successivo è stato quello di codificare le azioni svolte, ovvero introdurre una codifica per esprimere i movimenti consentiti. I bambini e le bambine sono stati coinvolti nella costruzione dei simboli: una freccia a diritto per avanzare di una casella, una freccia a sinistra o a destra per ruotare di 90° a sinistra o a destra rimanendo nella stessa casella (vedi Figura 3).

Figura 3: costruzione dei simboli

Tutto questo lavoro ha permesso di passare, nelle ultime lezioni, a lavorare più specificatamente sul coding: rappresentare con un simbolo ogni movimento (vedi Figure 4 e 5), esprimere

Figura 4: rappresentare azioni con simboli: freccia dritta=passo avanti

Figura 5: rappresentare azioni con simboli: freccia a sinistra=girare 90° a sinistra

ed eseguire la sequenza di movimenti, contare il numero di movimenti totali per raggiungere il tesoro.

In questa ultima fase sono state proposte diverse attività. Dapprima l'insegnante ha raccolto in un libretto le istruzioni per raggiungere il tesoro, codificate con i simboli che i bambini e la bambine avevano costruito; i giocatori, muniti del libretto, leggevano la prima istruzione, la interpretavano e la eseguivano come piccoli robot umani, per poi passare alla istruzione

successiva (vedi Figura 6): in questa attività, la cosa più sorprendente è stato il silenzio e la concentrazione con cui i piccoli hanno eseguito il loro compito, scandito dalla lettura del simbolo: "avanti, gira di qua, gira di là, ..." acquisendo così la capacità di riprodurre una sequenza di istruzioni. In un secondo momento, i bambini e bambine sono stati divisi in coppie: uno dei due dava l'istruzione, costruendo così il percorso verso il tesoro, e l'altro la eseguiva, scambiando poi i ruoli nel turno successivo.

Figura 6: esecuzione della procedura

3 Conclusioni

Al termine, per cercare di verificare la validità del progetto, sono state riportate sulla grande scacchiera le schede proposte dal libro adottato dalla classe, come illustrato nella Figura 2. La classe è stata divisa in 2 gruppi: uno ha svolto prima il lavoro sulle schede cartacee e l'altro

ha svolto prima il percorso riprodotto sulla scacchiera. I bambini e le bambine di entrambi i gruppi, ben preparati a muoversi secondo indicazioni precise e in maniera consapevole, hanno dimostrato una notevole capacità di risoluzione dei problemi che venivano loro proposti, ma dal confronto fra i 2 gruppi, è emerso chiaramente che chi ha prima effettuato il percorso fisico, ha avuto molta più facilità nell'affrontare la scheda cartacea di chi ha cominciato invece da quest'ultima. Naturalmente questo test non rispetta i canoni di una verifica sperimentale statistica, che richiederebbe una sperimentazione ben più ampia sia nei test effettuati che nella numerosità e selezione del campione, ma i suoi risultati costituiscono un incoraggiamento ed uno stimolo a proseguire il percorso.

Si può però affermare con sicurezza che Ginnastica e Coding possono coesistere senza snaturare nessuna delle due discipline. Si possono realizzare lezioni di ginnastica idonee per bambini e bambine della scuola dell'infanzia che consentano contemporaneamente di avvicinarli anche ai concetti basilari del problem solving, della sequenzialità procedurale e a concetti anche più avanzati come la ricerca di una soluzione ottima. Il valore aggiunto di questo progetto è stato proprio quello di coinvolgere fisicamente bambini e bambine permettendo loro di apprendere attraverso il movimento consapevole e con il coinvolgimento del loro intero corpo.

Tutto questo, senza dimenticare il divertimento, vero carburante di questa strana avventura!

Riferimenti bibliografici

[1] E. Baumann. Manuale di ginnastica italiana. Roma, 1908.

[2] M. Sabella C. Beltramini. Coding all'infanzia attraverso il corpo. [online]. Accessibile a https://www.raffaelloformazione.it/wp-content/uploads/2019/08/CODING-allinfanzia-attraverso-il-corpo.pdf.

[3] Istituto Comprensivo "Giorgio Perlasca". Primi passi nel coding alla scuola dell'infanzia. [online]. Accessibile a https://www.icsperlasca.edu.it/primi-passi-nel-coding-alla-scuola-dellinfanzia/.

Un progetto pilota per introdurre il pensiero computazionale nella scuola dell'infanzia

Andrea Bonani

Direzione Istruzione e Formazione italiana, via del Ronco 2, Bolzano, Italia
andrea.bonani@scuola.alto-adige.it

Abstract

L'articolo presenta un progetto pilota condotto in undici scuole dell'infanzia per promuovere il pensiero computazionale in età prescolare utilizzando solo oggetti fisici, senza quindi prevedere l'utilizzo di dispositivi digitali a schermo quali smartphone, tablet o computer. Sono state proposte alle scuole una serie di attività in grado di sviluppare processi propri del pensiero computazionale quali: pensiero algoritmico, riconoscimento di pattern, decomposizione. In questa prima fase del progetto si è voluto osservare l'impatto delle attività sui bambini, in particolare verificare l'adeguatezza dei percorsi proposti e dei robot utilizzati, il coinvolgimento dei bambini e se quanto appreso veniva trasferito in altri domini di conoscenza. Nel contempo il progetto è stato l'occasione per realizzare una prima formazione agli insegnanti sui principali aspetti del pensiero computazionale e come essi sono presenti nelle diverse attività.

1 Introduzione

Far parte della trasformazione digitale e sociale in atto nella società in cui viviamo richiede al cittadino di comprendere come funzionano le tecnologie e non solo di saperle utilizzare. Già nel 2015 una circolare del MIUR indicava come *"... un'appropriata educazione al pensiero computazionale, che vada al di là dell'iniziale alfabetizzazione digitale, è infatti essenziale affinché le nuove generazioni siano in grado di affrontare la società del futuro non da consumatori passivi ed ignari di tecnologie e servizi, ma da soggetti consapevoli di tutti gli aspetti in gioco e come attori attivamente partecipi del loro sviluppo."* [10] Diventa quindi indispensabile proporre già nella scuola dell'infanzia attività che promuovono e sviluppano nei bambini le prime competenze di base che, ampliate nei successivi anni di scuola, consentiranno loro di comprendere la logica di funzionamento delle tecnologie digitali che li circondano, come previsto dalla Mozione n.1-00117 del 12 marzo 2019.

Il pensiero computazionale nell'età prescolare non è stato ancora particolarmente esplorato e ci sono pochi studi empirici in questa area [2]. Tuttavia, diversi Paesi hanno incorporato il pensiero computazionale nei loro curricula nelle scuole dell'infanzia [13] e la ricerca con i bambini della scuola dell'infanzia ha visto un deciso aumento negli ultimi anni [17]. I lavori presenti evidenziano come bambini in età prescolare sono in grado di sviluppare abilità legate alle nozioni di base del pensiero algoritimico, dai concetti di sequenza a quelli di successione [3]. Altri lavori indicano che promuovere il pensiero computazionale in età prescolare migliora le capacità analitiche dei bambini e li introduce a nuovi strumenti utili per la risoluzione collaborativa dei problemi [5, 4]. Nel progettare percorsi didattici per bambini in età prescolare non è possibile escludere l'aspetto della manipolazione di materiali, della scoperta e della gestione del proprio corpo e dei propri movimenti nello spazio [11], questa è stata una delle principali motivazioni nel proporre attività utilizzando solo oggetti fisici e non dispositivi digitali a schermo.

2 Il progetto

Il progetto pilota, promosso dalla Direzione provinciale Scuole dell'infanzia in lingua italiana, fa parte di un'azione di sistema portata avanti dalla Direzione Istruzione e Formazione italiana della Provincia di Bolzano che vedrà coinvolto nei prossimi anni l'intero primo ciclo di istruzione delle scuole della provincia in un'ottica di continuità tra i vari cicli di istruzione. Il principale riferimento sono i *"Traguardi ed obiettivi didattici della proposta CINI"* [9].

Il progetto ha visto la partecipazione di 11 scuole dell'infanzia, 15 insegnanti e 110 bambini (48 femmine e 62 maschi) che frequentavano l'ultimo anno. Nell'autunno del 2022 le insegnanti hanno seguito una prima formazione di 12 ore. Nella prima parte hanno appreso i principali concetti che stanno alla base del pensiero computazionale in età prescolare e che forniscono supporto e valore educativo all'attività didattica. Nella seconda parte sono state presentate le tre attività da svolgere con quattro diversi robot educativi.

Da febbraio a maggio 2023 le insegnanti hanno svolto le attività nelle loro scuole con il supporto di un docente esperto esterno che aveva tra l'altro il compito di monitorare lo svolgimento delle varie attività.

Figure 1: Attività con Botley

Lo scopo del progetto era duplice. Da una parte formare il personale insegnante, in quanto fornire agli insegnanti un'adeguata formazione è fondamentale per veicolare in modo consapevole le tematiche affrontate nel progetto [7, 6]. Dall'altra parte si desiderava analizzare la validità del percorso didattico proposto andando ad indagare in particolare tre aspetti: (a) l'adeguatezza delle attività e dei robot educativi proposti; (b) l'impatto sui bambini in termini di coivolgimento emotivo, del livello e durata dell'attenzione, della collaborazione tra pari; (c) le ricadute in altri domini di conoscenza.

Le attività sono state ideate prendendo spunto da recenti studi di alcuni lavori svolti con bambini in età prescolare [1, 12, 14], utilizzando i seguenti robot educativi: Bee Bot (TTS Group), Botley (Learning Resources), Matatalab, Kibo (KinderLab robotics).

La programmazione di un robot non richiede semplicemente di rappresentare il movimento utilizzando i codici delle frecce e i pulsanti, o lo spazio della griglia [16], ma è strettamente intrecciato alle abilità spaziali. Per questo le attività con i robot erano precedute dall'attività zero: il bambino (*l'automa*) si muoveva su una griglia disegnata sul pavimento, seguendo semplici istruzioni di movimento impartite da un altro bambino (*il programmatore*) ed nel contempo un terzo bambino (*il segretario*) posizionava sul pavimento delle frecce seguendo le indicazioni fornite dal bambino-programmatore.

Successivamente all'attività zero le insegnanti iniziavano il percorso svolgendo le attività con il Bee Bot per due settimane, per poi passare al robot successivo (Botley) per altre due settimane e cosi via, concludendo l'intero percorso in poco più di due mesi. L'ordine in cui venivano utilizzati i robot era dettato dal crescente grado di complessità presente nei diversi dispositivi. I bambini, divisi in gruppi di 5 o 6, svolgevano con ogni robot tre diverse attività di difficoltà crescente.

Figure 2: Attività con Kibo

Esse consistevano nel fornire delle istruzioni ad un robot posto su reticolo realizzato e personalizzato dalle insegnanti a seconda del tema che stavano trattando a scuola. Prima di *"programmare"* il robot i bambini dovevano *"scrivere"*, e poi *"leggere"*, le istruzioni tramite delle frecce che andavano a posizionare in sequenza in prossimità del reticolo.

L'obiettivo delle attività era quello di sviluppare nei bambini alcuni dei processi propri del pensiero computazionale: pensiero algoritmico, scomposizione, riconoscimento di pattern [8], da sviluppare attraverso attività che prevedevano la sperimentazione, l'iterazione e la correzione degli errori. L'esperienza didattica doveva far scoprire in modo del tutto spontaneo l'uso degli algoritmi per risolvere problemi, imparare a costruire, scomporre, risolvere e riflettere per arrivare ad un determinato scopo.

3 Risultati

I dati - al momento in fase di elaborazione - sono stati raccolti attraverso: (a) osservazioni dirette da parte dell'autore dell'articolo; (b) compilazione di griglie da parte delle insegnanti al termine di ogni attività; (c) interviste alle singole insegnanti; (d) focus group finale tra tutte le insegnanti impegnate nel progetto.

Nel frattempo è possibile anticipare che le osservazioni evidenziano come quasi tutti i bambini sono stati in grado di terminare i compiti richiesti. Le attività hanno visto un forte coinvolgimento dei bambini, anche in coloro che solitamente dimostrano in altre situazioni una maggiore passività. Una dimostrazione dell'elevato grado di coinvolgimento è stata la durata del livello di concentrazione che si è mantenuto nella maggior parte dei casi elevato fino al termine dell'attività, andando anche oltre le aspettative delle insegnanti. I bambini hanno inoltre evidenziato un elevato grado di collaborazione fornendo consigli ai propri compagni e tentando assieme di correggere eventuali istruzioni sbagliate, svolgendo in tal modo un'attività di debbuging considerata una componente fondamentale [15].

Le insegnanti hanno inoltre osservato un miglioramento nel linguaggio, nell'orientamento spaziale ed infine concordavano nell'affermare che le attività hanno consentito di osservare nei bambini abilità che non sempre emergevano in altre esperienze didattiche.

Le criticità evidenziate riguardavano il numero di bambini, per cui opportuno non superare le quattro, massimo cinque unità e il tempo non sufficiente (due settimane) durante il quale avevano a disposizione il robot. Non sono state osservate differenze di genere, né in termini di capacità di portare a termine la consegna, né dal punto di vista del coinvolgimento e dell'attenzione.

4 Conclusioni e lavori futuri

Il progetto pilota svolto in undici scuole dell'infanzia aveva il duplice scopo di introdurre attività didattiche per promuovere il pensiero computazionale in età prescolare e di fornire una prima

formazione alle insegnanti sul significato e l'importanza dei concetti che stanno alla base del pensiero computazionale.

In futuro si desidera: (a) preparare attività anche per i bambini che frequentano il secondo anno della scuola dell'infanzia utilizzando robot più semplici; (b) ampliare le attività proposte ai bambini del terzo anno utilizzando i sensori in dotazione ai robot per sviluppare altre competenze, come ad esempio le istruzioni condizionali; (c) provare strumenti atti a valutare il grado di apprendimento conseguito dai bambini.

5 Ringraziamenti

Si ringraziano le quindici insegnanti delle nove scuole di Bolzano (Arcobaleno, Bambi, Casa dei Bambini, Casanova, Firmian, Peter Pan, Pollicino, Positano, Raggio di Sole) e delle due di Merano (Coccinella e Girotondo) che hanno partecipato al progetto fornendo preziosi consigli utili per rivedere le attività proposte. Si ringrazia Cinzia Cibin per la gestione del materiale didattico e soprattutto si ringrazia Erika Golin per il fondamentale aiuto nell'organizzazione delle attività nelle varie scuole.

References

[1] Charoula Angeli and Michail Giannakos. Computational thinking education: Issues and challenges, 2020.

[2] Ewelina Bakala, Anaclara Gerosa, Juan Pablo Hourcade, and Gonzalo Tejera. Preschool children, robots, and computational thinking: A systematic review. *International Journal of Child-Computer Interaction*, 29:100337, 2021.

[3] Marina U Bers, Carina González-González, and Mª Belén Armas-Torres. Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education*, 138:130–145, 2019.

[4] Marina Umaschi Bers. *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge, 2020.

[5] Marina Umaschi Bers, Amanda Strawhacker, and Miki Vizner. The design of early childhood makerspaces to support positive technological development: Two case studies. *Library Hi Tech*, 2018.

[6] Andrea Bonani, Rosella Gennari, Alessandra Melonio, and Mehdi Rizvi. Design and computational thinking with iotgo: What teachers think. In *Methodologies and Intelligent Systems for Technology Enhanced Learning, 12th International Conference*, pages 165–174. Springer, 2022.

[7] Isabella Corradini, Michael Lodi, and Enrico Nardelli. Conceptions and misconceptions about computational thinking among italian primary school teachers. In *Proceedings of the 2017 ACM conference on international computing education research*, pages 136–144, 2017.

[8] Michael Lodi. Informatical thinking. *Olympiads in Informatics: An International Journal*, 14:113–132, 2020.

[9] Enrico Nardelli et al. *Coding e oltre: l'informatica nella scuola*. Lisciani Scuola Editore, 2020.

[10] Dipartimento per il sistema educativo di istruzione e formazione del MIUR . Il pensiero computazionale a scuola. https://www.istruzione.it/allegati/2015/prot2187.pdf, 2015.

[11] Direzione provinciale scuole dell'infanzia . Indicazioni provinciali scuole per la scuola dell'infanzia in lingua italiana. https://www.provincia.bz.it/formazione-lingue/scuole-infanzia/downloads/1_Libretto_06072022_BASSA.pdf, 2022.

[12] Alex Pugnali, Amanda Sullivan, and Marina Umashi Bers. The impact of user interface on young children's computational thinking. *Journal of Information Technology Education. Innovations in Practice*, 16:171, 2017.

[13] Peter J Rich, Samuel F Browning, McKay Perkins, Timothy Shoop, Emily Yoshikawa, and Olga M Belikov. Coding in k-8: International trends in teaching elementary/primary computing. *TechTrends*, 63:311–329, 2019.

[14] Evgenia Roussou and Maria Rangoussi. On the use of robotics for the development of computational thinking in kindergarten: Educational intervention and evaluation. In *Robotics in Education: Current Research and Innovations 10*, pages 31–44. Springer, 2020.

[15] Valerie J Shute, Chen Sun, and Jodi Asbell-Clarke. Demystifying computational thinking. *Educational research review*, 22:142–158, 2017.

[16] Deborah Silvis, Victor Lee, Jody Clarke-Midura, Jessica Shumway, and Joseph Kozlowski. Blending everyday movement and representational infrastructure: An interaction analysis of kindergarteners coding robot routes. 2020.

[17] Kai-Yu Tang, Te-Lien Chou, and Chin-Chung Tsai. A content analysis of computational thinking research: An international publication trends and research typology. *The Asia-Pacific Education Researcher*, 29(1):9–19, 2020.

Kit per l'*Empowerment* Computazionale: l'Importanza di Prototipazione Rapida e Riflessione Profonda *

Monica Divitini[1], Rosella Gennari[2], and Alessandra Melonio[3]

[1] NTNU, Trondheim, Norvegia, divitini@ntnu.no
[2] Libera Università di Bolzano, Bolzano, Italia, gennari@unibz.it
[3] Università Ca' Foscari di Venezia, Venezia, Italia, alessandra.melonio@unive.it

Sommario

Il contributo scientifico sviluppa il concetto di *empowerment* computazionale, offrendo strumenti, materiali e pratiche per l'insegnamento dell'informatica in una prospettiva centrata sulla persona. Delinea esperienze in Norvegia ed Italia.

1 Introduzione

Promuovere il pensiero computazionale aiuta a comprendere come analizzare, scomporre ed elaborare soluzioni algoritmiche per problemi computazionali. Di recente, nell'interazione persona-macchina e, nello specifico, nel design partecipativo, la prospettiva si è allargata introducendo il cosiddetto *empowerment* computazionale, definito come il processo in cui i giovani sviluppano "*capacità di riflessione per comprendere la tecnologia digitale e il suo effetto sulla vita e sulla società in generale, e la capacità di impegnarsi in modo critico, curioso e costruttivo con la costruzione e la decostruzione della tecnologia*" [10]. Un obiettivo, dunque, è dare ai giovani opportunità di riflessione mentre sviluppano il pensiero computazionale, anche in relazione all'impatto delle scelte tecnologiche su persone e ambiente. Ciò che attualmente manca è la comprensione di come le competenze, richieste in attività di *empowerment* computazionale, possano essere stimolate trasversalmente a più ordini di scuola e discipline [1]. Per farlo, sono necessari strumenti adeguati, ovvero, adatti ai loro utenti [2].

Il lavoro qui proposto mostra come l'uso di kit di strumenti adeguati possa stimolare la riflessione critica di bambini e giovani sulle tecnologie informatiche in ambiti diversi, soprattutto in relazione al loro effetto sulle persone e sull'ambiente circostante, promuovendo così l'*empowerment* computazionale. In particolare, i kit di strumenti per la prototipazione di oggetti intelligenti, con dispositivi di computazione fisica, promuovono riflessioni su diversi aspetti della vita delle persone, da quelli tangibili a quelli immateriali.

Un oggetto intelligente è, ad esempio, un orologio intelligente: è una versione potenziata dal punto di vista computazionale di un oggetto di uso quotidiano, in grado di svolgere una serie di funzioni tramite dispositivi di computazione fisica, quali microcontrollori come i micro:bit o computer come i Raspberry Pi. Il processo di sviluppo di un simile oggetto intelligente inizia con l'analisi di un contesto, che considera le persone e una situazione problematica per loro, aperta a più soluzioni. Si passa poi all'ideazione, quando si riflette su soluzioni alternative tramite oggetti intelligenti. Concettualizzata un'idea, questa viene elaborata e scomposta seguendo schemi algoritmici, infine programmata; i risultati sono di nuovo oggetto di riflessione.

Questo articolo offre due esempi di kit di questo tipo e dei casi di studio: Tiles, utilizzato in laboratori in Norvegia; il kit IoTgo, utilizzato in ambiti scolastici differenti in Italia.

*Grazie a Maristella Matera (Politecnico di Milano, IT) e Mehdi Rizvi (Heriot Watt University, UK), che hanno contribuito maggiormente ai lavori qui citati.

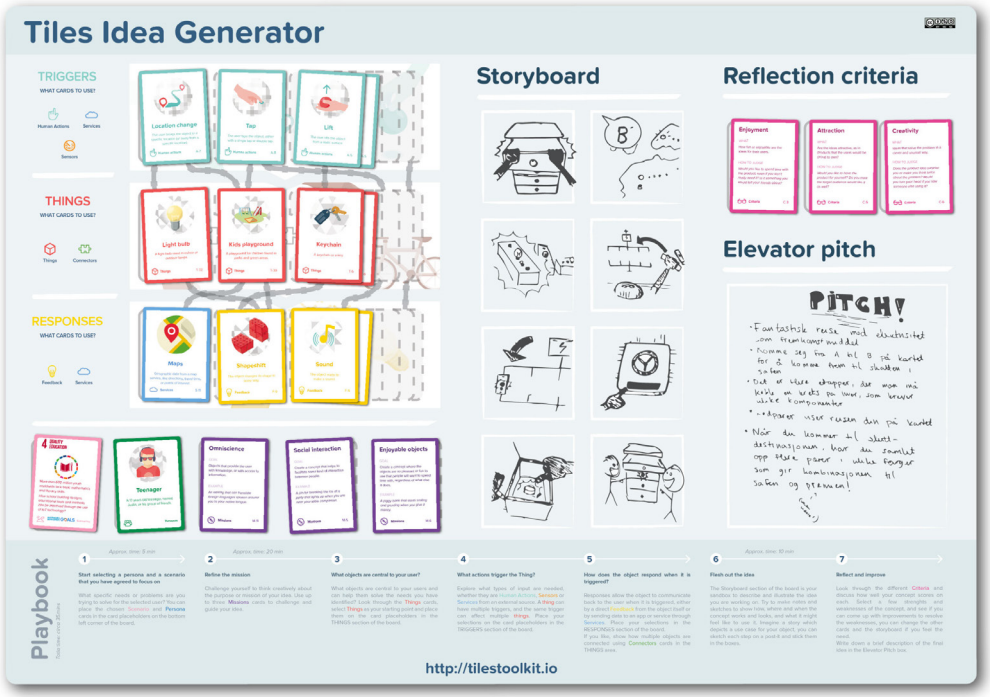


Figura 1: Il tabellone di Tiles inclusa l'aerea per i criteri di riflessione in alto a destra

2 Tiles

Il kit Tiles è dotato di oltre 100 carte e un tabellone di ideazione per coinvolgere i non esperti nella creazione rapida di oggetti intelligenti; consente loro di esplorare combinazioni di metafore di interazione, computazione e oggetti fisici. Il kit è stato inizialmente creato per i workshop di ideazione [12].

Tuttavia, la creazione di prototipi di oggetti intelligenti è un compito complesso per i non esperti. Per questo motivo è stato introdotto RaploT come complemento; si tratta di un kit di strumenti software che facilita la prototipazione di sistemi IoT, fornendo un insieme integrato di tecnologie [9]. In seguito, Tiles è stato ulteriormente ampliato durante diversi laboratori con studenti universitari, studenti delle scuole superiori, decisori, ricercatori e professionisti. In particolare, Tiles offre ora carte scenario, persona e, in generale, di riflessione critica attraverso le quali analizzare e giudicare i risultati. Il kit è disponibile online [13]. Si veda la Figura 1.

Tiles è stato anche utilizzato per consentire agli studenti delle scuole superiori di creare soluzioni per città intelligenti e di riflettere durante il processo. Secondo le analisi dei dati disponibili, il kit Tiles esteso è efficace in questo senso: da combinazioni di carte inesplorate sono emersi concetti innovativi, che i partecipanti hanno perfezionato attraverso riflessioni varie. In particolare, le carte persona e scenario hanno contribuito a inquadrare e contestualizzare il processo, fornendo indicazioni critiche aggiuntive rispetto al toolkit originale [8].

3 IoTgo

Il kit di strumenti IoTgo è evoluto da SNaP; questo a sua volta era ispirato a Tiles, e pensato per l'*empowerment* computazionale dei più giovani [3, 4]. IoTgo combina carte e tabelloni di gioco con un'applicazione digitale e dispositivi per la computazione fisica per i giovani, come micro:bit e relativi dispositivi di input e output. IoTgo si è evoluto per essere adattabile a diversi ordini di scuola e materie. Nel 2021–2022 e di nuovo nel 2022–2023, sono stati organizzati corsi con IoTgo per l'Intendenza Scolastica Italiana della Provincia di Bolzano. I corsi sono per insegnanti in servizio nelle primarie e secondarie, di materie diverse [7]. Il kit è stato poi utilizzato nelle scuole in ambito umanistico-artistico, nonché per insegnare l'informatica e riflettere sull'impatto dell'informatica sulla società. Entrambi i casi sono descritti nel seguito.

3.1 Arte

Nel 2022, un'insegnante di arte di una scuola superiore ha organizzato un'attività STEAM per la classe con il kit IoTgo, combinando l'arte e la computazione fisica [6]. Il percorso con IoTgo è iniziato con l'analisi di problemi non ben definiti relativi agli obiettivi di sostenibilità dell'Agenda 2030 ed alle persone [14]. Lavorando in gruppi di due o tre, gli adolescenti hanno poi elaborato delle prime soluzioni sotto forma di oggetti intelligenti, riflettendo con carte di riflessione predefinite e aperte del kit; si veda la Figura 2. Poi, sempre lavorando in modo collaborativo, hanno usato IoTgo per generare semplici programmi e riflettere sulle loro soluzioni, sempre con le carte di riflessione.

Lo studio ha raccolto dati qualitativi, poi elaborati in una prospettiva olistica che mantenesse l'integrità del caso. L'insegnante ha tenuto un diario delle attività svolte a scuola e ha discusso con i ricercatori la sua esperienza con IoTgo. Anche le voci degli studenti sono state prese in considerazione, considerando ciò che sono riusciti a creare e le riflessioni condotte con IoTgo.

Secondo l'analisi dei dati, nel percorso, tutte le idee iniziali sono maturate per poter essere trasformate in prototipi funzionanti e rilevanti per i problemi analizzati. Secondo l'insegnante, per raggiungere questo obiettivo, sono stati necessari vari cicli di riflessione e revisione; la struttura dei tabelloni di IoTgo, in particolare, ha fatto sì che tutti gli studenti potessero raggiungere questo obiettivo. Ciò trova riscontro nel numero e tipo di carte di riflessione utilizzate in relazione all'usabilità o alla fattibilità, le più frequenti tra tutte quelle utilizzate dagli studenti (50%, cioè 16 su 32).

Tutte le idee finali seguivano schemi simili, probabilmente per via dell'applicazione di IoTgo, che genera programmi seguendo tali schemi. Allo stesso tempo, secondo l'insegnante, l'applicazione e le domande di riflessione hanno permesso agli studenti di implementare e testare rapidamente le loro idee, innescando così i cicli di revisione necessari e portando ad una comprensione profonda del funzionamento dei dispositivi fisico-informatici e di come impiegarli per le loro idee. È interessante notare che, oltre alle riflessioni su usabilità e fattibilità, la maggior parte delle riflessioni degli studenti erano legate agli obiettivi di sostenibilità dell'Agenda 2030 per le persone (31%, cioè 10 su 32).

3.2 Informatica

IoTgo è stato utilizzato anche per insegnare informatica. Ha diversi laboratori, con esempi di *scaffolding*, che specificano in formato infografico una risoluzione algoritmica di un problema. Questa viene tradotta in un programma, attualmente in MakeCode o MicroPython, a seconda del grado scolastico. La Figura 3 mostra lo stesso esempio, una volta per le superiori in inglese e una volta per le primarie e medie in italiano.

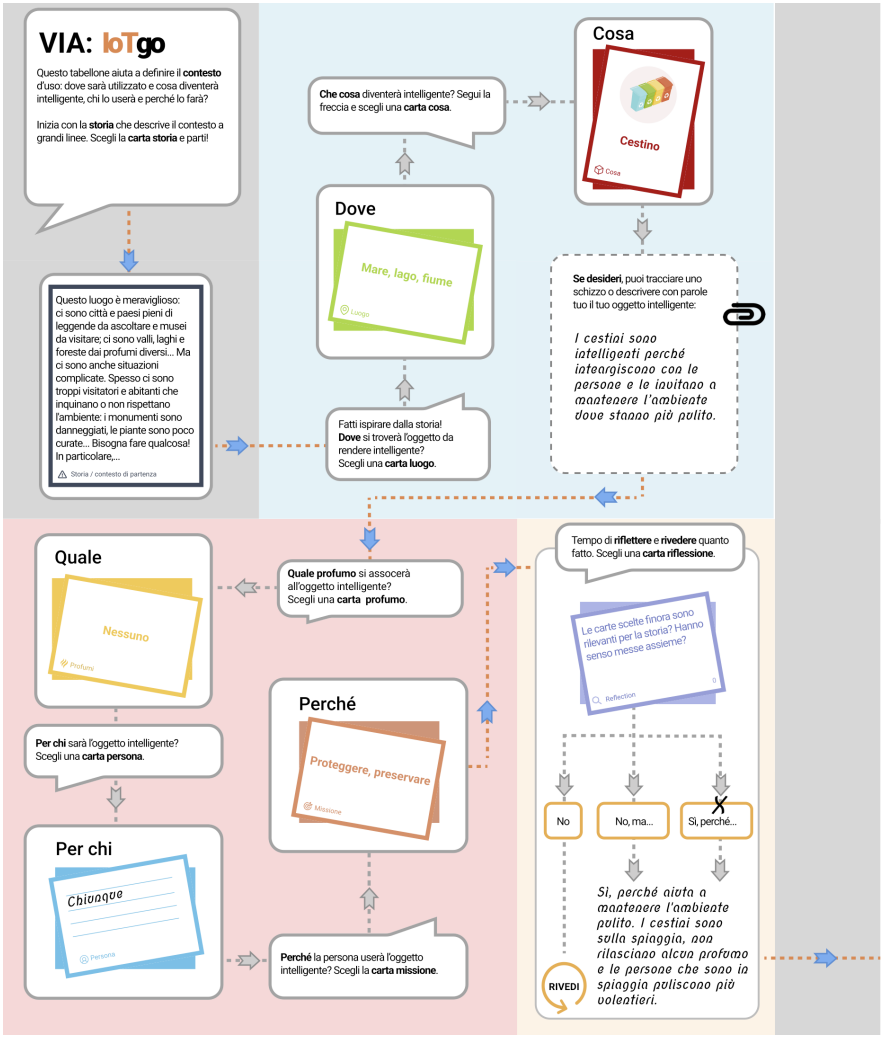


Figura 2: Uno dei tabelloni di IoTgo per elaborare idee e riflessioni per il contesto considerato

IoTgo è stato utilizzato in parte in questo modo in un istituto tecnico. Questo lavoro è ampiamente riportato in [5]. In questo studio, il kit è stato adattato per consentire agli adolescenti di programmare un prototipo di soluzione su piccola scala per la loro scuola e per il benessere sociale digitale, concentrandosi sull'inclusione. Lo studio con 24 studenti ha analizzato le loro riflessioni. Stando ai risultati delle analisi, gli adolescenti sono stati in grado di riflettere criticamente da più prospettive attraverso le carte di IoTgo (rilevanza, sicurezza) e altri criteri di riflessione emersi spontaneamente (accessibilità, usabilità universale, impegno sociale, sostenibilità verde). La capacità dell'applicazione IoTgo di generare programmi, secondo determinati schemi di risoluzione algoritmica, ha di nuovo permesso ai partecipanti di scoprire e applicare rapidamente tali schemi e di avere tempo per riflettere ed elaborare ulteriormente le loro idee. La possibilità di continuare a elaborare i programmi generati è stata accolta da tutti i partecipanti, come documentato in [6].

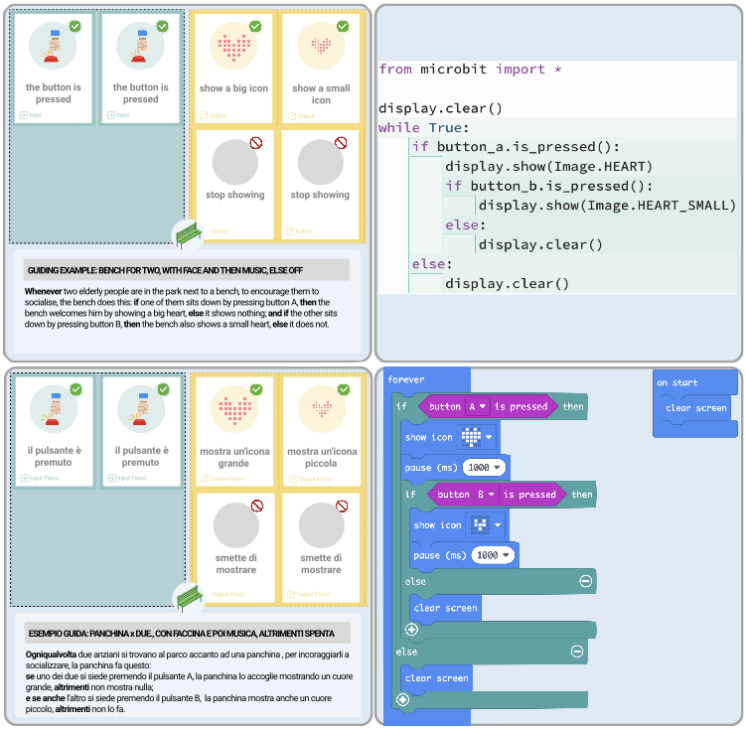


Figura 3: Due laboratori di IoTgo: per le superiori (sopra); per le medie e primarie (sotto)

4 Conclusioni

Seguendo una filosofia di pensiero recente, che allarga l'attenzione dal pensiero computazionale all'*empowerment* computazionale, questo articolo presenta due kit di strumenti e casi di studio che mirano a coinvolgere i non esperti, ed in particolare i giovani, nello sviluppo critico di oggetti intelligenti, sollecitando riflessioni continue sui prototipi proposti [10]. Come in altri lavori recenti, anche questo considera che la didattica dell'informatica tende ad avere pochi spazi di analisi di problemi familiari ai giovani e per i quali elaborare loro soluzioni computazionali, e pochi spazi per la revisione e riflessione profonda; questi sono spazi altrettanto importanti per l'*empowerment* computazionale dei giovani [11]. Entrambi i kit qui presentati invitano ad analizzare un contesto con problemi aperti, rilevanti per le persone. Poi aiutano i giovani a sviluppare rapidamente prototipi di loro soluzioni intelligenti, seguendo schemi di risoluzione algoritmica, e a riflettere continuamente lungo il processo. Nei casi di studio riportati, i kit hanno permesso loro di riflettere profondamente su ciò che stavano prototipando in relazione al contesto analizzato, in vere attività di *empowerment* computazionale.

Riferimenti bibliografici

[1] Charoula Angeli and Michail Giannakos. Computational thinking education: Issues and challenges. *Computers in Human Behavior*, 105:106185, 2020.

[2] Carmelo Ardito, Giuseppe Desolda, Rosa Lanzilotti, Alessio Malizia, Maristella Matera, Paolo Buono, and Antonio Piccinno. User-defined semantics for the design of IoT systems enabling smart interactive experiences. *Personal and Ubiquitous Computing*, 24(6):781–796, 2020.

[3] Rosella Gennari, Maristella Matera, Alessandra Melonio, Mehdi Rizvi, and Eftychia Roumelioti. Reflection and awareness in the design process: children ideating, programming and prototyping smart objects. *Multimedia Tools and Applications*, 80(26):34909–34932, 2021.

[4] Rosella Gennari, Maristella Matera, Alessandra Melonio, and Eftychia Roumelioti. A board-game for co-designing smart nature environments in workshops with children. In Alessio Malizia, Stefano Valtolina, Anders Morch, Alan Serrano, and Andrew Stratton, editors, *End-User Development*, pages 132–148, Cham, 2019. Springer International Publishing.

[5] Rosella Gennari, Maristella Matera, Diego Morra, Alessandra Melonio, and Mehdi Rizvi. Design for social digital well-being with young generations: Engage them and make them reflect. *International Journal of Human-Computer Studies*, 173:103006, 2023.

[6] Rosella Gennari, Maristella Matera, Diego Morra, and Mehdi Rizvi. A phygital toolkit for rapidly designing smart things at school. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*, AVI 2022, New York, NY, USA, 2022. Association for Computing Machinery.

[7] Rosella Gennari, Alessandra Melonio, and Mehdi Rizvi. A tool for guiding teachers and their learners: The case study of an art class. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI EA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[8] Francesco Gianni and Monica Divitini. Designing IoT Applications for Smart Cities: extending the Tiles Ideation Toolkit. *IxD&A*, 2017.

[9] Francesco Gianni, Simone Mora, and Monica Divitini. Rapiot toolkit: Rapid prototyping of collaborative internet of things applications. *Future Generation Computer Systems*, 95:867–879, 2019.

[10] Ole Sejer Iversen, Rachel Charlotte Smith, and Christian Dindler. From Computational Thinking to Computational Empowerment: A 21st Century PD Agenda. In *Proc. of the 15th Participatory Design Conference*, PDC '18, New York, NY, USA, 2018. ACM.

[11] Linda Mannila and Mia Skog. "Look at Our Smart Shoe"—a Scalable Online Concept for Introducing Design as Part of Computational Thinking in Grades 1–6. In *Proceedings of the 22nd Annual ACM Interaction Design and Children Conference*, IDC '23, page 222–232, New York, NY, USA, 2023. Association for Computing Machinery.

[12] Simone Mora, Francesco Gianni, and Monica Divitini. Tiles: A Card-Based Ideation Toolkit for the Internet of Things. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, DIS '17, page 587–598, New York, NY, USA, 2017. Association for Computing Machinery.

[13] Tiles IoT Inventor Toolkit. [Retrieved online], 2023. Available at https://www.tilestoolkit.io.

[14] United Nations. The Sustainability Development Goals, 2022.

SECONDARIA DI SECONDO GRADO

UNIVERSITÀ

Un serious game per imparare a programmare e sensibilizzare alla sostenibilità ambientale

Davide Brescia, Enrica Gentile, Paola Plantamura,
Teresa Roselli, Veronica Rossano
Dipartimento di Informatica, Università degli Studi di Bari
veronica.rossano@uniba.it

Abstract

Negli ultimi anni l'uso dei videogiochi come strumento educativo per favorire l'acquisizione di conoscenze e competenze è cresciuto a dismisura. Il serious game è un videogioco che tramite elementi ludici permette all'utente di acquisire conoscenze e di affinare abilità e competenze in una modalità divertente. I giochi possono offrire un ambiente adatto per acquisire nuove conoscenze e mettere in pratica il problem solving perché, nelle esperienze di gioco, i giocatori sono incoraggiati a pensare in modo critico, analizzare le situazioni, identificare e valutare le opzioni disponibili e prendere decisioni.

Il contributo propone un serious game che promuove sia la sensibilità verso i temi della sostenibilità ambientale sia lo sviluppo del pensiero computazionale e delle capacità di problem solving rivolto a studenti delle prime classi delle scuole superiori di secondo grado. Il primo prototipo di gioco è stato sottoposto a valutazione e i risultati incoraggiano future implementazioni.

1 Introduzione

Nel corso degli anni, le competenze trasversali, o soft skills, stanno diventando sempre più importanti e decisive nella vita di tutti i giorni poiché permettono di affrontare con successo le sfide poste da una società in continua evoluzione. Tra queste abilità ci sono le abilità di problem solving ed il pensiero computazionale [1] che fanno riferimento alle capacità di definire un problema, di determinare la causa del problema e di identificare la soluzione.

In quest'ottica, il quadro europeo delle competenze digitali per i cittadini, noto anche come DigComp[1], aggiornato nel 2016, fornisce un quadro di riferimento per migliorare le competenze digitali dei cittadini. In Italia il Piano Nazionale per la Scuola Digitale (PNSD)[2] ha introdotto il pensiero computazionale come una delle quattro abilità fondamentali che uno studente deve acquisire insieme a scrittura, lettura e calcolo. D'altra parte, in questi anni, un altro tema cruciale, che coinvolge anche molto il mondo della scuola, è quello della sostenibilità. L'Agenda 2030[3] per lo sviluppo sostenibile, un programma d'azione per le persone, il pianeta e la prosperità si concentra su tre aree principali: sostenibilità economica, sostenibilità sociale e sostenibilità ambientale.

In questo contesto la nostra ricerca ha portato alla progettazione e realizzazione di un serious game, ovvero un videogioco creato per un esplicito e ben definito scopo serio e non per scopi di intrattenimento o divertimento [2], con il duplice obiettivo di favorire l'acquisizione di abilità di coding e di sensibilizzare alla sostenibilità ambientale.

[1] https://www.agid.gov.it/sites/default/files/repository_files/digcomp2-1_ita.pdf.
[2] https://scuoladigitale.istruzione.it/pnsd/#PNSD
[3] https://sdgs.un.org/goals

2 *Dorobot* il serious game

Il serious game ***Dorobot*** è un videogioco in cui l'utente deve programmare un robot (un bot) per compiere delle missioni legate al tema della sostenibilità: riciclare i rifiuti, spegnere incendi o piantare degli alberi. L'obiettivo del gioco è migliorare le abilità di problem solving e iniziare ad approcciarsi alla programmazione informatica. Il serious game è stato progettato per ragazzi di età compresa tra i 13 e i 16 anni, che frequentano la scuola superiore e che sono interessati alla programmazione. Per utilizzare il serious game non sono necessarie conoscenze pregresse di programmazione.

Il gioco è un puzzle game con visuale top-down. Un puzzle game è un videogioco in cui le missioni sono enigmi sotto forma di puzzle logici e/o strategici. Il giocatore dovrà guidare ***Dorobot*** (Figura 2), il personaggio principale del gioco, a risolvere problemi di natura ambientale scrivendo un algoritmo in pseudocodice, ovvero scrivendo il programma che il bot dovrà eseguire usando un linguaggio a metà tra il linguaggio naturale e un linguaggio di programmazione. In aiuto del giocatore vi è il ***Dorobook*** (Figura 3), ovvero un manuale che offre un riepilogo di tutte le istruzioni conosciute da ***Dorobot***. Per combinare il tema della sostenibilità, i livelli sono stati disegnati in maniera tale da proporre sfide a tema ambientalistico come la raccolta differenziata, gli incendi boschivi e la riforestazione. La sensibilizzazione alla cura dell'ambiente rappresenta, quindi, l'effetto collaterale del serious game.

2.1 Elementi di design del serious game

Seguendo la Game-Tetrad definita da Shell [3] è possibile descrivere il serious game nelle sue componenti, ovvero, Meccaniche, Dinamiche, Estetica, Storia e Tecnologia.

Meccaniche: sono costituite dagli elementi del gioco che guidano le azioni dei giocatori nel gioco, come ad esempio: sfide, missioni, feedback, ricompense, turni, condizioni di vittoria, ecc. In particolare, in ***Dorobot*** sono stati definiti:

- **Livelli.** Il gioco è strutturato in quattro livelli. La difficoltà dei singoli livelli aumenta in maniera graduale: i primi livelli saranno più semplici e saranno utilizzati per familiarizzare con i comandi, per questo richiedono poche righe di codice. Ogni livello è organizzato in missioni che richiedono di utilizzare le competenze, ovvero le istruzioni, della missione precedente.
- **Feedback**. Per ogni missione l'utente scrive nel ***Dorocode*** lo pseudocodice per risolvere la missione. Un interprete realizzato all'interno del serious game consente al bot di muoversi nella schermata e al termine fornisce il feedback visivo e uditivo che segnala se la missione è stata completata. Anche quando sono rilevati errori di sintassi, per un uso scorretto del linguaggio, viene fornito un messaggio che spiega l'errore (Figura 4).
- **Ricompense**. Insieme al messaggio di missione completata, viene fornito anche un riscontro sulla qualità del codice realizzato usando delle stelle (Figura 1) che si coloreranno in misura proporzionale ad essa. La qualità viene misurata in termini di numero di linee di codice realizzate per portare a termine la missione.

Dinamiche: rappresentano aspetti più astratti che contribuiscono a creare l'esperienza di gioco, come ad esempio i vincoli di gioco, le emozioni che si vogliono far emergere durante il gioco, il senso di progressione, ecc. In ***Dorobot*** il giocatore sarà sempre consapevole dei progressi acquisiti, e i livelli sono costruiti in modo da non far provare un senso di frustrazione dovuto a richieste troppo difficili o noia per richieste troppo semplici. La progressione è visualizzata al giocatore anche nella schermata iniziale del livello in cui può vedere le missioni completate, le stelle guadagnate per ogni missione e le missioni eventualmente sbloccate (Figura 1).

Estetica: è legata all'aspetto del gioco e rappresenta la chiave per costruire un'esperienza interessante ed emozionante. L'estetica di ***Dorobot*** è semplice e intuitiva. Utilizza il colore verde per rimanere in linea con il tema ambientale e ogni livello ha un colore che lo

caratterizza, questo come elemento di orientamento per il giocatore. Nei serious game, infatti, è fondamentale che anche la grafica consenta all'utente di riconoscere in maniera semplice ed immediata il contesto del dominio in cui sta operando. Il ***Dorocode***, la finestra per la scrittura del programma, è simile ad un editor con indicazione del numero della linea di codice (Figura 4).

Storia: rappresenta la sequenza degli eventi che si svolgono nel gioco, come i personaggi interagiscono e come la trama del gioco contribuisce all'esperienza di gioco. ***Dorobot*** è un piccolo robot che vuole aiutare il pianeta, non è stata sviluppata una parte narrativa, ma sono state utilizzate ambientazioni diverse per ogni livello di gioco contestualizzate alle diverse sfide della sostenibilità: raccolta differenziata, prevenzione incendi e riforestazione.

Tecnologia: definisce con quali strumenti ci si può approcciare al gioco e come si può giocare ed interagire. Il gioco, sviluppato in Unity[4], uno dei più utilizzati game engine, è fruibile su un qualsiasi computer e richiede l'uso della tastiera per la scrittura del codice.

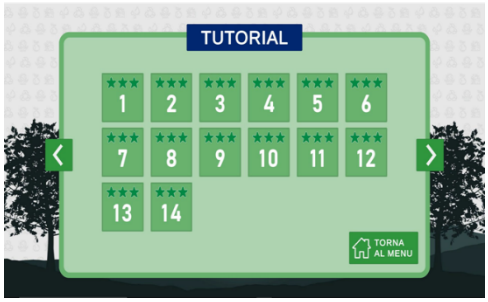


Figura 1: Schermata di inizio livello con tutte le missioni che lo compongono

Figura 2: Comunicazione della missione da svolgere

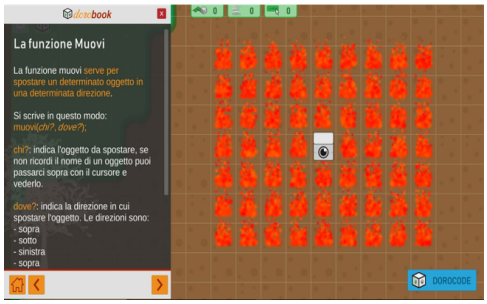


Figura 3: Il *Dorobook* per imparare le istruzioni dello pseudocodice e come utilizzarle

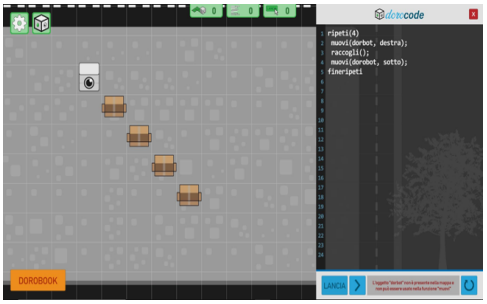


Figura 4: Il *Dorocode* editor e interprete dello pseudocodice con indicazione dell'errore sintattico

3 Test con gli utenti

Per valutare l'efficacia e l'usabilità del serious game ***Dorobot*** è stato condotto un test con 20 studenti, di una prima classe dell'Istituto Tecnico Tecnologico Guglielmo Marconi di Bari, di età compresa tra i 14 e i 16 anni e con poche competenze di informatica. A tutti è stato chiesto di utilizzare il gioco per un periodo di tempo di 60 minuti.

Per misurare l'efficacia i partecipanti sono stati sottoposti ad un pre-test e ad un post-test, composti da 10 domande di tipo teorico/pratico relative alla realizzazione di un programma in pseudocodice. Il **pre-test**, somministrato prima della prova, è servito per misurare le conoscenze informatiche pregresse, il **post-test** è servito per misurare le conoscenze acquisite dopo l'uso del

[4] https://unity.com/

serious game. I due test proponevano semplici esercizi molto simili ai task da affrontare (Figura 5). In particolare, le domande erano di due tipologie:

- Vero o Falso: un'immagine rappresentava lo stato del gioco e il codice, il partecipante doveva indicare se il codice fornito era corretto o errato.
- Scelta multipla: un'immagine rappresentava lo stato del gioco e il partecipante doveva selezionare la risposta corretta tra quattro opzioni.

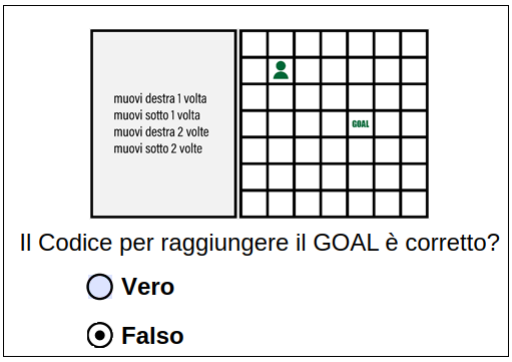


Figura 5. Esempio di esercizio nel pre e post test

Come si nota in Figura 6, quasi tutti gli studenti hanno migliorato la propria conoscenza in merito alla programmazione. Inoltre, poiché non tutti gli studenti hanno utilizzato il gioco per lo stesso tempo, si è notato come l'utilizzo più prolungato del serious game abbia permesso di acquisire più conoscenze relativamente alla programmazione in pseudocodice e più praticità nell'utilizzo del serious game stesso.

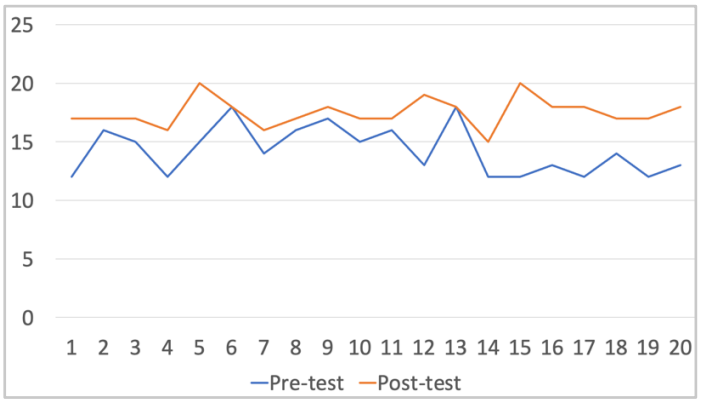


Figura 6: Confronto risultati tra pre-test e post-test

Inoltre, il serious game è stato valutato anche dal punto di vista dell'usabilità utilizzando:

- il **SUS** [4]: System Usability Scale, per valutare la soddisfazione che l'utente prova nell'utilizzare il sistema;
- il **NASA-TLX** [5]: per valutare il carico di lavoro percepito nell'uso del sistema. Indagando 6 dimensioni soggettive: la richiesta mentale, la richiesta fisica, la richiesta temporale, la prestazione, lo sforzo e la frustrazione;
- l'**UES** [6]: User Engagement Scale, per misurare il coinvolgimento dell'utente. Fornisce informazioni sull'attenzione focalizzata, l'usabilità percepita, l'attrattiva estetica e la ricompensa.

Per quanto riguarda i risultati del **SUS**, il punteggio registrato – pari a 77 (Figura 7) - indica che il serious game ha una buona usabilità e che gli utenti sono stati soddisfatti nell'utilizzare il gioco.

I risultati del **NASA-TLX** riportati in Tabella 1, evidenziano che il gioco ha richiesto agli utenti di impegnarsi in maniera importante per poter portare a termine le sfide ma che, allo stesso tempo, gli utenti si sono sentiti soddisfatti delle loro prestazioni. È da notare che il valore della frustrazione, che è abbastanza basso, evidenzia che la strategia di creare livelli a difficoltà crescente ha funzionato.

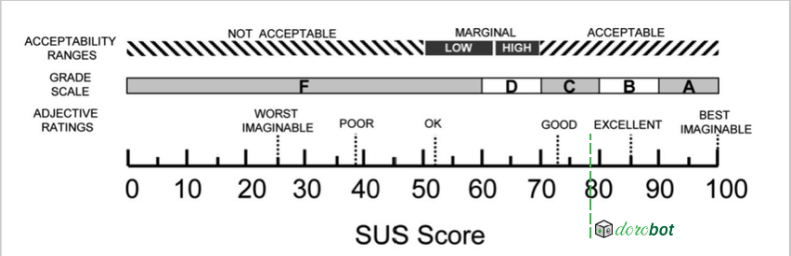


Figura 7: Sondaggio SUS per la valutazione del serious game

	Media (su 10)	Deviazione Standard
Sforzo Mentale	5,3	2,0
Sforzo Fisico	2,5	1,0
Esigenza Temporale	4,2	2,4
Prestazione	7,6	2,3
Impegno	6,2	2,4
Frustrazione	4,6	3,5

Tabella 1:Risultati questionario sul carico cognitivo

	Media (su 5)	Deviazione Standard
Attenzione	3,8	1,1
Usabilità Percepita	4,2	1,0
Estetica	3,7	1,0
Ricompensa	4,2	0,9

Tabella 2: Risultati questionario sul coinvolgimento dell'utente

I risultati del questionario **UES** (Tabella 2) sottolineano come il gioco sia stato giudicato usabile e gli utenti abbiano trovato l'esperienza gratificante. Il gioco, inoltre, ha richiesto all'utente molta attenzione, dettaglio positivo in un serious game che mira ad assorbire il giocatore nell'attività proposta.

4 Conclusioni e sviluppi futuri

È ormai ampiamente riconosciuto come la dimensione del gioco sia utile per incoraggiare i giocatori a pensare in modo critico, analizzare le situazioni, identificare e valutare le opzioni disponibili e prendere decisioni. Tutte queste dimensioni si sposano bene con le competenze richieste dall'informatica. Il serious game presentato è ancora in fase prototipale, un primo test ha consentito di registrare risultati positivi sia in termini di efficacia che di usabilità e coinvolgimento dell'utente. I partecipanti hanno trovato interessante poter imparare la programmazione attraverso l'utilizzo dello pseudocodice e hanno apprezzato molto l'associazione tra i principi dell'informatica e gli aspetti di ecologia per la salvaguardia dell'ambiente.

Ringraziamenti

Il lavoro è stato parzialmente supportato dal fondo di finanziamento ordinario di Ateneo "Computational Thinking: tecniche e strumenti per lo sviluppo del pensiero computazionale", la ricerca in cui si colloca è parte del lavoro svolto dal nodo di Bari del Laboratorio CINI "Informatica e Scuola". Si ringraziano tutti gli studenti dell'Istituto Tecnico Tecnologico Guglielmo Marconi di Bari e del prof. Andreazza Luigi.

Bibliografia

[1] Wing, J.M. (2017). *Computational thinking's influence on research and education for all*. Italian Journal of Educational Technology, 25(2), 7-14. doi: 10.17471/2499-4324/922

[2] Michael, D.R., Chen, S.L. (2005). *Serious Games: Games That Educate, Train, and Inform*. Thomson Course Technology, USA.

[3] Schell, J. (2008). *The art of game design: a book of lenses*. 1nd ed. Burlington, MA, Morgan Kaufmann Publishers.

[4] Brooke, J. (1995). *SUS: A quick and dirty usability scale. Usability Eval*. Ind. 189.

[5] Hart, S. (2006). *Nasa-Task Load Index (NASA-TLX); 20 Years Later*. Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 50(9), pp. 904–908. doi: 10.1177/154193120605000909

[6] O'Brien, H., Cairns, P., Hall, M. (2018). *A Practical Approach to Measuring User Engagement with the Refined User Engagement Scale (UES) and New UES Short Form*. International Journal of Human-Computer Studies. pp. 28-39. doi: 10.1016/j.ijhcs.2018.01.004

Sviluppo di un sondaggio sulla comprensione dei threads tra gli studenti delle scuole superiori

Emanuele Scapin[1] and Nicola Dalla Pozza[1]

ITT G. Chilesotti, 36016 Thiene, VI
{escapin,ndallapozza}@chilesotti.it

Abstract

Le difficoltà degli studenti delle scuole superiori nell'apprendere la programmazione concorrente sono ben note ai docenti di Informatica. Mentre a livello internazionale è ancora aperto il dibattito se tale argomento sia da inserire nei curricula pre-universitari, le linee guida ministeriali del nostro Paese per gli istituti tecnici con specializzazione in Informatica indicano che gli studenti sono tenuti ad acquisire competenze chiave di programmazione concorrente. Con l'obiettivo di studiare la natura delle difficoltà degli studenti, nonché identificare possibili approcci pedagogici che possono essere adottati dagli insegnanti, abbiamo intrapreso un'indagine sulla percezione, competenza e fiducia in sé stessi degli studenti quando si tratta di compiti di concorrenza e sincronizzazione. Questo lavoro vuole presentare come è stato organizzato il questionario proposto agli studenti. Questo strumento d'indagine è stato progettato includendo domande sulla percezione soggettiva da parte degli studenti e piccole attività di comprensione del codice relativo alla sincronizzazione dei thread. Viene inoltre analizzata la fiducia degli studenti in sé stessi in relazione alle loro effettive prestazioni in tali compiti. I risultati preliminari, ottenuti su un campione di 68 studenti di alcune scuole della nostra regione, indicano un basso livello di fiducia in sé stessi in relazione all'argomento. In particolare, i risultati mostrano chiaramente che il concetto di sincronizzazione dei thread è particolarmente difficile da padroneggiare per la maggior parte di essi. Oltre a presentare la metodologia d'indagine, questo lavoro mira a coinvolgere altri educatori per estendere la portata di questa ricerca empirica.

1 Introduzione

Negli ultimi decenni, al fine di realizzare architetture sempre più performanti, le aziende produttrici di micro–processori hanno orientato i propri prodotti verso design multi–core e many–core. Questa tendenza è così ampiamente adottata che oggigiorno tale tipo di hardware è impiegato non solo per scopi industriali e accademici [22], ma è possibile trovare tali macchine anche in laboratori scolastici [2]. Per sfruttare appieno la potenza computazionale di questi elaboratori occorre però realizzare applicazioni che sfruttino i numerosi core con programmi concorrenti.

É quindi diffusa l'esigenza di programmatori esperti che facciano leva su questo paradigma, come riportato anche dalla Task Force ACM/IEEE [10]. La programmazione concorrente è in genere introdotta a livello universitario in corsi avanzati dedicati, in combinazione con argomenti correlati [9, 5, 18], o in moduli introduttivi di programmazione [3, 11]. La discussione su quando e come affrontare l'argomento è ancora aperta, e ancor di più il dibattito sulla fattibilità di coprire la programmazione concorrente nel curriculum delle scuole superiori [1, 12, 20, 2].

Le difficoltà degli studenti ad apprendere la programmazione concorrente sono ben note sia tra gli insegnanti delle scuole superiori sia tra i docenti universitari [8, 12, 2]. Secondo le linee guida ministeriali italiane gli studenti che optano per una specializzazione tecnica in Informatica — negli istituti tecnici industriali — sono tenuti a sviluppare competenze di programmazione parallela e concorrente entro la fine del quarto anno [1].

[1] https://www.gazzettaufficiale.it/eli/gu/2012/03/30/76/so/60/sg/pdf

Manca tuttavia evidenza empirica per valutare se questi obiettivi curricolari sono realistici. In un recente lavoro di Brođanac e al. viene infatti riportato "[i]n the case of teaching parallel programming before university level, the research appears to be scarce" [2].

All'interno del quadro delineato, in questo lavoro presentiamo la progettazione di uno strumento per un'indagine esplorativa sulla competenza e sulla fiducia in sé stessi degli studenti quando si confrontano con compiti di concorrenza e sincronizzazione. Più specificamente, l'indagine è rivolta a trovare alcune risposte preliminari alle seguenti domande di ricerca:

Q1. In che misura gli studenti sono a loro agio con i concetti di base della programmazione concorrente?

Q2. In che misura la percezione della fiducia in sé stessi da parte degli studenti è correlata alle loro prestazioni effettive in semplici compiti di programmazione concorrente?

Q3. Quali sono le maggiori difficoltà nell'apprendimento di questo paradigma?

Per rispondere alle domande di cui sopra, abbiamo progettato un questionario che include domande sulla percezione soggettiva e quattro attività di comprensione del codice che coinvolgono aspetti di base della sincronizzazione dei threads. Nell'indagine preliminare sono stati coinvolti complessivamente 68 studenti frequentanti l'ultimo anno in istituti rappresentativi del nostro territorio. Gli studenti sono stati introdotti alla programmazione concorrente nell'anno scolastico precedente da diversi insegnanti, i quali potrebbero aver utilizzato approcci diversificati.

In aggiunta alla presentazione del sondaggio, uno degli obiettivi principali di questo contributo è quello di invitare gli educatori e i ricercatori interessati a prendere parte al progetto per ampliare il campione di studenti e la portata dello studio empirico.

Il lavoro è organizzato come segue. Nella Sezione 2 riassumiamo brevemente il contesto di questo lavoro. La Sezione 3 riguarda la struttura del sondaggio e la logica alla base del suo progetto. Una breve sintesi dei risultati preliminari della nostra indagine esplorativa è delineata nella Sezione 4. In chiusura, nella Sezione 5, vengono riportate alcune conclusioni e menzionati possibili sviluppi futuri del presente lavoro.

2 Contesto di riferimento

Le difficoltà degli studenti con la programmazione concorrente sono ben note ai docenti di Informatica degli istituti tecnici, a quelli universitari e ai formatori post-laurea. Nonostante ciò, possiamo osservare che a questo argomento non è stata posta adeguata attenzione nel contesto della formazione sulla programmazione. Ad esempio, nello studio sistematico [14], che esamina oltre 700 articoli di ricerca pubblicati tra il 2003 e il 2018, Luxton-Reilly et al. menzionano solo due contributi che affrontano la programmazione parallela o concorrente.

Sono però presenti una serie di lavori sull'apprendimento della programmazione concorrente a livello universitario o pre-universitario che esaminano le prestazioni degli studenti, le loro difficoltà comuni, la loro comprensione e le idee sbagliate sull'argomento. Ad esempio, Choi e Lewis [4] analizzano e classificano le insidie di una raccolta di semplici programmi multithread scritti dagli studenti, Fekete [6] si concentra sulle classi Java thread-safe, discutendone le relative difficoltà pedagogiche e proponendo anche esempi che possano aiutare gli studenti a evitare malintesi comuni. Lönnberg e Berglund [13] indagano i bug presenti nei programmi concorrenti prodotti dagli studenti dal punto di vista dello sviluppo del programma.

I primi tentativi di introdurre la programmazione concorrente nelle scuole superiori risalgono alla metà degli anni '90 [17]. Molto lavoro sull'insegnamento e l'apprendimento di questo paradigma è stato svolto dalla comunità educativa israeliana: Ben-Ari e Kolikant [1] hanno

esplorato l'evoluzione della comprensione di questi concetti da parte degli studenti, e hanno scoperto che alla fine di un modulo dedicato questi erano effettivamente in grado di sviluppare algoritmi paralleli e di dimostrarne la correttezza. Sebbene l'approccio iniziale si sia rivelato impegnativo, gli studenti hanno poi imparato ad apprezzare l'importanza dell'argomento e il suo contributo al miglioramento delle loro capacità cognitive. Successivamente, Kolikant [12] ha osservato che, sebbene gli studenti delle scuole superiori siano in grado di acquisire una piena comprensione dei vari compiti di sincronizzazione, molto spesso le loro soluzioni dei problemi di sincronizzazione sono raggiunte tramite un approccio basato su schemi e scorciatoie che li dispensano dal considerare la dinamica completa dei meccanismi di sincronizzazione, ma allo stesso tempo li porta a sottovalutare e dimenticare questi concetti.

Anche secondo Tobert e al. [20] ci sono ampie prove che il pensiero algoritmico parallelo e il multi-threading possono essere insegnati nella scuola secondaria. Infine, Brođanac et al. [2] hanno recentemente condotto un'indagine su 162 studenti di 3 scuole superiori croate che includono nel curriculum la programmazione parallela. Gli autori riferiscono di aver ricevuto feedback positivi dagli studenti sull'interesse e l'utilità del contenuto appreso, anche se è stato percepito come più difficile rispetto agli altri argomenti. Gli autori concludono che la programmazione parallela può essere insegnata alle scuole superiori almeno come materia facoltativa.

All'interno del panorama educativo italiano della scuola secondaria di II grado, la programmazione concorrente è affrontata negli istituti tecnici industriali con specializzazione in Informatica, seguendo le linee guida ministeriali. Queste linee guida sono articolate in termini di *conoscenze*, *abilità* e *competenze* da raggiungere nel secondo biennio.

In particolare, durante il secondo biennio, gli studenti dovrebbero apprendere concetti come la struttura dei sistemi operativi, le tecniche per la programmazione concorrente e per l'accesso sincronizzato alle risorse condivise. In termini di abilità dovrebbero essere in grado di progettare e sviluppare applicazioni che interagiscono con il sistema operativo, utilizzando ove opportuno strategie di programmazione concorrente. Nell'ultimo anno di scuola superiore, gli studenti dovrebbero acquisire conoscenze sui metodi e sulle tecnologie per la programmazione di rete, nonché sui protocolli e sui linguaggi di comunicazione del livello applicativo. In termini di abilità, dovrebbero riuscire a sviluppare applicazioni che sfruttano la comunicazione di rete, come applicazioni client-server attraverso semplici protocolli di comunicazione.

In questo quadro, pur restando ai docenti la libertà di personalizzare l'organizzazione del corso, la programmazione concorrente è comunque considerata parte essenziale dei curricula delle scuole superiori con specializzazione in Informatica.

3 Strumento d'indagine

In questa sezione presentiamo la struttura dello strumento di indagine e la logica alla base della sua progettazione. L'organizzazione generale, descritta in Figura 1, è in parte tratta da uno strumento simile sviluppato e utilizzato dagli autori per ottenere informazioni sull'apprendimento di altri concetti di programmazione [19]. Il sondaggio completo è reperibile al link `https://forms.gle/W4HxuvCEDJyegjfh9`.

Le domande e i problemi proposti coprono gli argomenti relativi alle domande di ricerca Q1 – Q3 elencate precedentemente. Le domande (sezione 2) e i problemi (sezione 3) sono rivolti a investigare in che misura gli studenti sono a loro agio con alcuni concetti di base della programmazione concorrente (Q1). I problemi (sezione 3) e le relative domande sulla misura della percezione di fiducia (self-confidence) rispetto alle prestazioni degli studenti intendono rispondere alla seconda domanda di ricerca (Q2). Infine, alcune domande specifiche della sezione 2 puntano a investigare quali sono le maggiori difficoltà nell'apprendimento della programmazione

concorrente da parte degli studenti (Q3).

Complessivamente il sondaggio comprende 24 domande, 11 delle quali basate su 4 piccoli compiti — 7 domande sono di comprensione del programma e 4 richiedono valutazioni della fiducia in sé stessi sulle risposte fornite in una scala Likert a 4 gradi.

1. Informazioni generali (2 domande) genere; preferenza soggettiva dei linguaggi di programmazione usati con i thread
2. Concetti generali relativi ai thread (7 domande di percezione soggettiva) difficoltà dell'argomento in generale; self-efficacy in generale; adeguatezza del tempo dedicato all'argomento; adeguatezza degli esempi proposti; difficoltà con concetti/problemi specifici relativi ai thread; difficoltà con specifici strumenti di programmazione relativi ai thread; difficoltà con la sincronizzazione dei thread
3. Task (7 domande di comprensione del programma e 4 sulla valutazione della self-confidence delle risposte fornite) Task1 – mutua esclusione (4 domande); Task2 – schema produttore-consumatore; Task3 – analisi di sincronizzazione; Task4 – analisi di stallo (deadlock)
4. Possibili strumenti di aiuto (3 domande di percezione soggettiva) se le rappresentazioni grafiche siano mai state prese in considerazione; potenziale atteso delle rappresentazioni grafiche; percezione dell'uso di specifiche rappresentazioni grafiche
5. Suggerimenti (1 domanda) suggerimenti per rendere più interessanti e chiare le lezioni sui thread

Figure 1: Struttura generale del sondaggio.

Le domande che affrontano l'apprendimento dei concetti di concorrenza sono ispirate da Choi e Lewis [4]. Gli autori infatti hanno catalogato gli errori che gli studenti tipicamente commettono quando si approcciano a programmi multi-thread, in particolare in relazione a race conditions, deadlock e problemi di sincronizzazione.

I problemi inseriti nel sondaggio sono riportati di seguito. Per adattarsi alle comuni pratiche nelle scuole superiori coinvolte il codice di riferimento è simile a Java. I primi due problemi sono stati ispirati da programmi proposti da Meyer e al. in *"Concurrent Programming with Java Threads"*,[2] e hanno lo scopo di verificare la capacità degli studenti nell'anticipare i risultati di threads simultanei. Il primo problema presenta una classe molto semplice volta a sincronizzare l'accesso ad una risorsa condivisa, in base alla disponibilità dei dati. Sono state quindi presentate quattro sequenze temporali di invocazioni di metodi da parte di due thread concorrenti, con la richiesta di identificare gli esiti risultanti (domande a–d). I frammenti di codice Java per i problemi 1.a–d fanno riferimento alla classe *Counter* definita come segue:

```
public class Counter {
private int count = -1; // a negative value of count is
        interpreted as "undefined"

public synchronized int getCount() {
     while ( count < 0 ) {
          try {
               wait();
          } catch ( Exception e ) {}
     }
     return count;
}

public synchronized void setCounter( int initValue ) {
     if ( initValue >= 0 ) {
          count = initValue;
          notify();
     }
}

public synchronized void increment() {
     while ( count < 0 ) {
          try {
               wait();
          } catch ( Exception e ) {}
     }
     count = count + 1;
}
} // Counter
```

Problema 1.a Analizza l'esecuzione dei frammenti di codice (in Figura 2), relativi a due thread distinti, Thread-1 e Thread-2, che operano su un'istanza condivisa x della classe Counter introdotta sopra. Le operazioni di ciascuno dei due thread sono rappresentate lungo i lati opposti dell'asse

[2] Il materiale del corso è disponibile al link https://se.inf.ethz.ch/courses/2011a_spring/soft_arch/lectures/old/13_softarch_self_study_threads.pdf.

verticale, secondo l'ordine temporale (dall'altro verso il basso) in cui i metodi invocati nelle istruzioni vengono avviati; inoltre, nessuna operazione su x oppure su i è stata omessa nei flussi riportati. Quali sono i valori stampati in output nel corso dell'esecuzione di Thread-1?

Opzioni possibili (una sola opzione selezionabile): i = 1, count = 1; i = 1, count = 5; i = 5, count = 5; i = 6, count = 6; Il risultato non può essere previsto perché ci sono diverse possibilità.

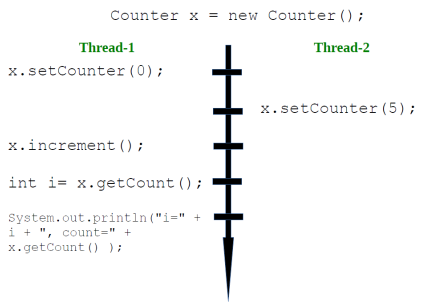


Figure 2: Problema 1.a

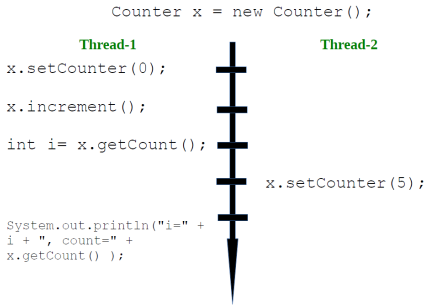


Figure 3: Problema 1.b

Problema 1.b Analizza l'esecuzione dei frammenti di codice (in Figura 3), relativi a due thread distinti, Thread-1 e Thread-2, che operano su un'istanza condivisa x della classe Counter introdotta sopra. Le operazioni di ciascuno dei due thread sono rappresentate come descritto introducendo il quesito Task 1.a. Quali sono i valori stampati in output nel corso dell'esecuzione di Thread-1?

Opzioni possibili (una sola opzione selezionabile): i = 1, count = 1; i = 1, count = 5; i = 5, count = 5; i = 5, count = 6; Il risultato non può essere previsto perché ci sono diverse possibilità.

Problema 1.c Analizza l'esecuzione dei frammenti di codice (in Figura 4), relativi a due thread distinti, Thread-1 e Thread-2, che operano su un'istanza condivisa x della classe Counter introdotta sopra. Le operazioni di ciascuno dei due thread sono rappresentate come descritto introducendo il quesito Task 1.a. Quali sono i valori stampati in output nel corso dell'esecuzione di Thread-1?

Opzioni possibili (una sola opzione selezionabile): i = 0, count = 0; i = 5, count = 0; i = 6, count = 0; i = 6, count = 6; Il risultato non può essere previsto perché ci sono diverse possibilità.

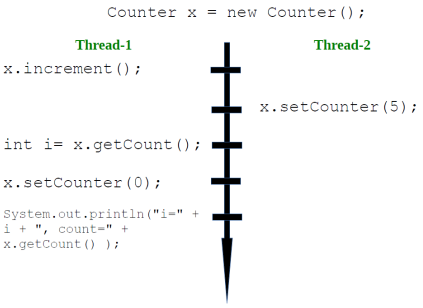


Figure 4: Problema 1.c

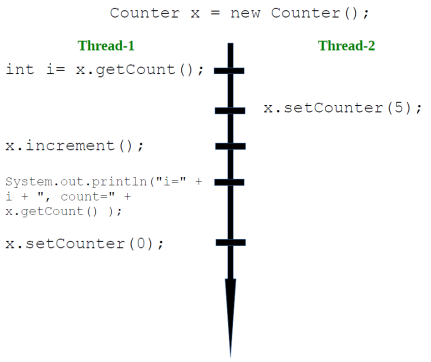


Figure 5: Problema 1.d

Problema 1.d Analizza l'esecuzione dei frammenti di codice (in Figura 5), relativi a due thread distinti, Thread-1 e Thread-2, che operano su un'istanza condivisa x della classe Counter introdotta

sopra. Le operazioni di ciascuno dei due thread sono rappresentate come descritto introducendo il quesito Task 1.a. Quali sono i valori stampati in output nel corso dell'esecuzione di Thread-1?

Opzioni possibili (una sola opzione selezionabile): i = 0, count = 0; i = 5, count = 6; i = 6, count = 6; i = -1, count = 6; Il risultato non può essere previsto perché ci sono diverse possibilità.

Il secondo problema, più complesso, riguardava un'istanza dello schema *produttore-consumatore*; ancora una volta, agli studenti è stato chiesto di identificare l'output corretto.

Problema 2 Considera le classi definite in Figura 6 e immagina di avviare il programma attraverso il metodo main della classe Task2. Quale delle sequenze proposte sarà stampata al termine dell'esecuzione?

Opzioni possibili (una sola opzione selezionabile): P3P7P5; P3PP7PP5P; PP3P5P7; PP3PP7PP5; PPP375; PPPPPP375; Il programma si blocca in una situazione di stallo; Il risultato non può essere previsto perché ci sono diverse possibilità

```
class Adder extends Thread {

  private int loops;
  private Vector<Integer> buffer;   // integer sequence

  public Adder( int loops, Vector<Integer> buffer ) {

    this.loops = loops;
    this.buffer = buffer;
  }

  public void run() {

    for ( int i=0; i<loops; i=i+1 ) {

      synchronized ( buffer ) {
        while ( buffer.size() < 2 ) {
          try {
            buffer.wait();
          } catch ( Exception e ) {}
        }
        // n: adds the first two buffer elements
        int n = buffer.get(0) + buffer.get(1);
        buffer.clear();  // buffer is emptied
        System.out.print( ""+n );
        buffer.notify();
    }}
    System.out.println();
  }

}  // Adder
```

```
class Provider extends Thread {

  private int[] stream;
  private Vector<Integer> buffer;

  public Provider( int[] stream, Vector<Integer> buffer ) {

    this.stream = stream;
    this.buffer = buffer;
  }

  public void run() {

    for ( int i=0; i<stream.length; i=i+1 ) {

      int x = stream[i];
      synchronized ( buffer ) {
        while ( buffer.size() == 2 ) {
          try {
            buffer.wait();
          } catch ( Exception e ) {}
        }
        System.out.print( "P" );
        buffer.add(x);  // new element into buffer
        buffer.notify();
      }
    }
  }

}  // Provider
```

```
public class Task2 {

  public static void main( String[] args ) {

    int[] stream = new int[] { 1, 2, 3, 4, 3, 2 };

    // buffer initially empty
    Vector<Integer> buffer = new Vector<Integer>();
    Adder adder = new Adder( 3, buffer );
    Provider provider = new Provider( stream, buffer );

    adder.start();
    provider.start();
  }

}  // Task2
```

Figure 6: Problema 2

Gli altri due problemi sono stati ispirati dal lavoro di Fekete [6]: il terzo problema ha lo scopo di vedere se gli studenti sono in grado di identificare, tra 5 opzioni, entrambi i frammenti di codice appropriati (equivalenti) per gestire una risorsa condivisa; il quarto problema richiede di riconoscere il codice soggetto a stallo (deadlock) e di impostare le correzioni adeguate.

Nel terzo compito, le (due) implementazioni valide di un semplice schema di sincronizzazione dovevano essere riconosciute tra cinque opzioni.

Problema 3 Quali delle seguenti definizioni di metodi (Figura 7), all'interno di una classe che descrive l'implementazione di una risorsa condivisa, possono aiutare a evitare conflitti nella gestione della risorsa stessa? Sono selezionabili più opzioni.

L'ultimo problema richiede di scegliere una strategia adeguata, descritta informalmente a parole, per risolvere un codice soggetto a deadlock (stallo).

Option 1

```
public synchronized int getValue() {
  return value;
}

public void setValue(int someValue) {
  value = someValue;
}

public void increment() {
  value++
}
```

Option 2

```
public int getValue() {
  return value;
}

public synchronized void
        setValue(int someValue) {
  value = someValue;
}

public void increment() {
  value++
}
```

Option 3

```
public synchronized int getValue() {
  return value;
}

public synchronized void
        setValue(int someValue) {
  value = someValue;
}

public synchronized void increment() {
  value++
}
```

Option 4

```
public int getValue() {
  return value;
}

public void setValue(int someValue) {
  value = someValue;
}

public void increment() {
  value++
}
```

Option 5

```
public int getValue() {
  synchronized (this) {
    return value;
  }
}

public void
  setValue(int someValue) {
  synchronized (this) {
    value = someValue;
  }
}

public void increment() {
  synchronized (this) {
    value++
  }
}
```

Figure 7: Problema 3. Equivalenza: *seleziona tutte le soluzioni applicabili.*

Problema 4 Considera un'istanza della classe *Bouncer* definita qui di seguito. Le modalità di sincronizzazione dei metodi *from1to2* e *from2to1* possono determinare situazioni di deadlock (stallo dell'esecuzione). Quale tra i rimedi suggeriti sotto ritieni possano correggere il codice per evitare che si possa verificare uno stallo (pur garantendo una sincronizzazione appropriata)?

Opzioni possibili (una sola opzione selezionabile): eliminare tutti i synchronized; eliminare i synchronized annidati; eliminare i synchronized da uno dei due metodi; eliminare il synchronized esterno da uno dei metodi e quello annidato dall'altro; trasformare i synchronized annidati in synchronized in sequenza (uno dopo l'altro anziché uno all'interno dell'altro); invertire seq1 e seq2 in tutti i costrutti synchronized; nessuna delle precedenti soluzioni.

```
public class Bouncer {
    private Vector<Integer> seq1;
    private Vector<Integer> seq2;

    public Bouncer( Vector<Integer> seq1, Vector<Integer> seq2 ) {

        this.seq1 = seq1;
        this.seq2 = seq2;
    }

    public void from1to2() {
        synchronized ( seq1 ) {
            if ( seq1.size() == 0 ) {
                try {
                    seq1.wait();
                } catch ( Exception e ) {}
            }
            int item = seq1.elementAt(0);
            seq1.removeElementAt(0);
            synchronized ( seq2 ) {
                seq2.add( item );
                seq2.notify();
            }
        }
    }

    public void from2to1() {
        synchronized ( seq2 ) {
            if ( seq2.size() == 0 ) {
                try {
                    seq2.wait();
                } catch ( Exception e ) {}
            }
            int item = seq2.elementAt(0);
            seq2.removeElementAt(0);
            synchronized ( seq1 ) {
                seq1.add( item );
                seq1.notify();
            }
        }
    }
} // Bouncer
```

Per le soluzioni di ciascuno dei quattro compiti, agli studenti è stato anche chiesto di indicare il livello percepito di fiducia in loro stessi in una scala Likert che va da 1 (per niente fiducioso) a 4 (pienamente fiducioso). Alcuni studi [16, 15] evidenziano l'importanza di valutare la misura in cui la percezione soggettiva della difficoltà da parte degli studenti è correlata alla loro effettiva prestazione in un compito.

Le sezioni conclusive del sondaggio riguardano potenziali strumenti grafici/visivi che potrebbero aiutare a comprendere la programmazione concorrente, così come possibili suggerimenti aggiuntivi per migliorare la pratica didattica. Un particolare interesse per gli strumenti grafici/visivi è motivato da recenti risultati di ricerca che suggeriscono che gli studenti delle discipline STEM hanno maggiori probabilità di esibire uno stile cognitivo visuo-spaziale [21].

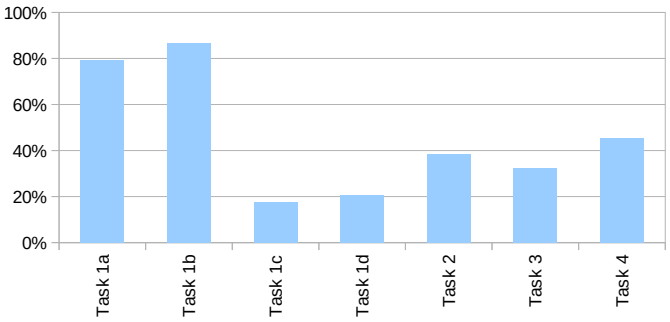


Figure 8: Le prestazioni degli studenti nei compiti proposti.

4 Risultati

Il questionario è stato somministrato a 68 studenti del quinto anno, studenti che hanno frequentano l'indirizzo Informatica in due istituti tecnici della nostra regione. Gli studenti, a cui è stata insegnata la programmazione concorrente nell'anno scolastico precedente, sono stati impegnati nel compito in una situazione controllata, sotto la supervisione dei loro insegnanti. Dovevano completare il questionario entro un'ora.

Le risposte sono state registrate solo in forma anonima e quindi non potevano essere utilizzate per valutazioni formative o sommative.

Una completa ed esauriente analisi dei risultati ottenuti sarà argomento di un successivo lavoro; qui di seguito illustreremo brevemente le indicazioni più significative emerse.

Più di due terzi degli studenti trovano la programmazione di thread difficile (57%) o molto difficile (12%) e dichiarano di essere insoddisfatti (54%) o completamente insoddisfatti (13%) delle loro prestazioni in attività relative ai thread. Comunque, la maggior parte di loro considera adeguati allo scopo sia il tempo impiegato per affrontare l'argomento (67%) sia i problemi di programmazione proposti (63%) dai loro insegnanti.

La *sincronizzazione* e la gestione degli *stati di un thread* sono considerate difficili da imparare rispettivamente dal 77% e dal 74% degli studenti, mentre diversi aspetti relativi all'organizzazione delle classi che implementano i thread sembrano meno critici.

L'ultima domanda della sezione 2 del sondaggio è centrata sulla gestione delle risorse condivise. Tutti questi aspetti sono percepiti come difficili da una percentuale significativa di studenti, che va da circa il 30% (operazioni di lettura/scrittura) a circa il 60% (operazioni di sincronizzazione).

Nella Figura 8 sono riportate le percentuali di risposte (completamente) corrette alle sette domande poste per i compiti 1–4. Come si può vedere, solo la soluzione delle sotto-attività 1a e 1b è corretta per la grande maggioranza degli studenti; in tutti gli altri casi, meno della metà ha avuto successo, con le prestazioni peggiori per le sotto-attività apparentemente facili 1c e 1d. Meno di un terzo degli studenti è più o meno fiducioso delle proprie risposte, il che conferma ancora una volta le loro difficoltà con la programmazione concorrente.

L'ultima sezione di domande a scelta multipla (sezione 4) è centrata su possibili ausili grafici/visivi per l'apprendimento. Il 59% degli studenti ha riferito di aver pensato all'utilizzo di diagrammi grafici e ben l'88% ritiene che una rappresentazione grafica potrebbe essere efficace per migliorare la loro comprensione della programmazione concorrente. Tuttavia, un numero ridotto di studenti ha un'idea chiara su quale tipo di strumento possa essere adatto per affrontare i concetti relativi ai thread. Del resto, gli strumenti più efficaci che forse possono essere utilizzati

non sono molto conosciuti nei contesti scolastici: in particolare, diversi studenti non conoscevano le reti di Petri (74%), i grafici Wait-for/Holt (51%), o gli automi a stati finiti (50%). Pertanto, gli strumenti più votati sono stati i diagrammi a blocchi (44%) e i diagrammi di flusso (51%), anche se non sono i più adatti allo scopo.

5 Conclusioni

In questo lavoro abbiamo presentato la struttura, e in particolare i problemi proposti, del questionario progettato come indagine esplorativa rivolta alla percezione delle difficoltà, alla competenza e alla fiducia in sé stessi degli studenti delle scuole superiori quando si confrontano con compiti di programmazione concorrente. La rilevanza del tema dal punto di vista professionale [7] è indiscutibile, ciononostante gli studenti dimostrano difficoltà con un argomento che implica un alto livello di astrazione ed è difficilmente rappresentabile graficamente, ma di cui comunque il nostro sistema scolastico — a differenza di altri — prevede esplicitamente la trattazione. Il tema prevede un'elevata sfida cognitiva [2] alla luce della maturità degli studenti e del tempo di insegnamento disponibile per sviluppare l'argomento.

Attualmente l'indagine delineata ha coinvolto 68 studenti, ma si prevede di allargare il campione. In particolare, al fine di estendere la portata di questa ricerca empirica, un progetto ambizioso sarebbe riuscire a coinvolgere educatori che operano in contesti diversificati nel territorio nazionale.

References

[1] Mordechai Ben-Ari and Yifat Ben-David Kolikant. Thinking parallel: The process of learning concurrency. *SIGCSE Bull.*, 31(3):13–16, jun 1999.

[2] Predrag Brođanac, Josip Novak, and Ivica Boljat. Has the time come to teach parallel programming to secondary school students? *Heliyon*, 8(1), Jan 2022.

[3] Kim B. Bruce, Andrea Danyluk, and Thomas Murtagh. Introducing concurrency in cs 1. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, pages 224–228, New York, NY, USA, 2010. Association for Computing Machinery.

[4] Sung-Eun Choi and E. Christopher Lewis. A study of common pitfalls in simple multi-threaded programs. *SIGCSE Bull.*, 32(1):325–329, 3 2000.

[5] Davi Jose Conte, Paulo Sergio Lopes de Souza, Guilherme Martins, and Sarita Mazzini Bruschi. Teaching parallel programming for beginners in computer science. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–9, 2020.

[6] Alan D. Fekete. Teaching students to develop thread-safe java classes. *SIGCSE Bull.*, 40(3):119–123, 6 2008.

[7] William B. Gardner. Should we be teaching parallel programming? In *Proceedings of the 22nd Western Canadian Conference on Computing Education*, WCCCE '17, New York, NY, USA, 2017. Association for Computing Machinery.

[8] Stephen J. Hartley. "alfonse, wait here for my signal!". In *The Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '99, pages 58–62, New York, NY, USA, 1999. Association for Computing Machinery.

[9] David J. John and Stan J. Thomas. Parallel and distributed computing across the computer science curriculum. In *2014 IEEE International Parallel & Distributed Processing Symposium Workshops*, pages 1085–1090, 2014.

[10] Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. *Computer Science Curricula 2013: Curriculum Guidelines for Under-*

graduate Degree Programs in Computer Science. Association for Computing Machinery, New York, NY, USA, 2013.

[11] Yousun Ko, Bernd Burgstaller, and Bernhard Scholz. Parallel from the beginning: The case for multicore programming in the computer science undergraduate curriculum. In *SIGCSE 2013 - Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, pages 415–420, 2013.

[12] Yifat Ben-David Kolikant. Learning concurrency: evolution of students' understanding of synchronization. *International Journal of Human-Computer Studies*, 60(2):243–268, 2004.

[13] Jan Lönnberg, Anders Berglund, and Lauri Malmi. How students develop concurrent programs. In *Proceedings of the Eleventh Australasian Conference on Computing Education - Volume 95*, ACE '09, page 129–138, AUS, 2009. Australian Computer Society, Inc.

[14] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. Introductory programming: A systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE 2018 Companion, pages 55–106, New York, NY, USA, 2018. ACM.

[15] Murali Mani and Quamrul Mazumder. Incorporating metacognition into learning. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 53–58, New York, USA, 2013. ACM.

[16] Laurie Murphy and Josh Tenenberg. Do computer science students know what they know? a calibration study of data structure knowledge. In *Proc. of the 10th Annual SIGCSE Conf. on Innovation and Technology in Computer Science Education*, ITiCSE '05, pages 148–152, New York, USA, 2005. ACM.

[17] Adam Rifkin. Teaching parallel programming and software engineering concepts to high school students. In *Proceedings of the Twenty-Fifth SIGCSE Symposium on Computer Science Education*, SIGCSE '94, page 26–30, New York, NY, USA, 1994. Association for Computing Machinery.

[18] Suzanne Rivoire. A breadth-first course in multicore and manycore programming. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, page 214–218, New York, NY, USA, 2010. Association for Computing Machinery.

[19] Emanuele Scapin and Claudio Mirolo. An exploratory study of students' mastery of iteration in the high school. In Külli Kori and Mart Laanpere, editors, *Proceedings of the International Conference on Informatics in School: Situation, Evaluation and Perspectives, Tallinn, Estonia, November 16-18, 2020*, volume 2755 of *CEUR Workshop Proceedings*, pages 43–54. CEUR-WS.org, 2020.

[20] Shane Torbert, Uzi Vishkin, Ron Tzur, and David J. Ellison. Is teaching parallel algorithmic thinking to high school students possible? one teacher's experience. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, pages 290–294, New York, NY, USA, 2010. Association for Computing Machinery.

[21] Jonathan Wai, David Lubinski, and Camilla P Benbow. Spatial ability for stem domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance. *Journal of educational Psychology*, 101(4):817, 2009.

[22] Álvaro Fernández, Camino Fernández, José Ángel Miguel-Dávila, Miguel Ángel Conde, and Vicente Matellán. Supercomputers to improve the performance in higher education: A review of the literature. *Computers & Education*, 128:353–364, 2019.

Python nei Licei: Alcune riflessioni

Maurizio Boscaini[1 2 3], Alberto Montresor[2], e Massimiliano Masetti[4]

[1] Istituto Tecnico-Tecnologico G. Marconi, Verona, Italy
[2] Università di Trento, Trento, Italy
[3] Università di Verona, Verona, Italy
[4] Istituto di Istruzione Superiore L. Cerebotani, Lonato (BS), Italy

1 Introduzione

Tranne per autonome iniziative di singoli istituti, il ruolo dell'informatica nei percorsi liceali italiani è molto limitato e il suo insegnamento può risultare inefficace. Infatti, da un lato, si vorrebbe insegnare la materia con lo stesso rigore di un corso universitario, partendo dalle basi teoriche e dai linguaggi di basso livello. Dall'altro lato, il quadro orario molto limitato rende impossibile raggiungere questo obiettivo e non riesce spesso a coinvolgere e ad appassionare lo studente alla disciplina. Non è raro sentire di studenti lamentarsi che non apprezzano la materia poiché la ritengono "completamente scollegata dalla realtà," quando è ormai chiaro che il loro futuro lavorativo sarà strettamente condizionato dalle conoscenze e competenze informatiche.

In questo articolo, vogliamo promuovere l'idea che la scelta del linguaggio di programmazione utilizzato nell'insegnamento dell'informatica al liceo sia una componente fondamentale per migliorare l'apprezzamento della materia e favorire l'apprendimento di competenze informatiche in maniera inclusiva. In particolare riteniamo che, nell'ottica di una didattica basata sul *learning-by-doing*, Python rappresenti una scelta naturale per la creazione di programmi utili, interessanti e belli.

Nell'articolo, analizzeremo la situazione attuale dell'insegnamento dell'informatica, che risulta molto diverso da quanto proposto nelle Indicazioni Nazionali; esporremo alcuni dati relativi all'insegnamento di Python come primo linguaggio di programmazione e mostreremo vantaggi e svantaggi del suo utilizzo in didattica.

2 L'informatica al liceo

L'inclusione dell'informatica come disciplina indipendente nei programmi di liceo è avvenuta con il Liceo scientifico e tecnologico (Progetto Brocca, 1992-2010). In questo contesto, la materia "Informatica e sistemi" veniva insegnata negli ultimi tre anni per tre ore settimanali, di cui due in laboratorio con la compresenza dell'insegnante teorico e di quello tecnico-pratico [16]. La riforma Gelmini ha sostituito tale indirizzo scolastico con l'introduzione del Liceo scientifico con opzione scienze applicate, nel quale l'informatica viene insegnata due ore alla settimana su tutti e cinque gli anni e non prevede il laboratorio e la compresenza [17].

Alcuni rari Istituti, sfruttando l'autonomia a disposizione, hanno potenziato l'insegnamento dell'informatica, aumentando il numero di ore dedicate alla disciplina. Per esempio l'Istituto Carlo Anti di Villafranca di Verona e il Liceo Volterra di Ciampino prevedono percorsi con 4 ore settimanali di informatica per gli ultimi tre anni e per tutti e cinque gli anni di studio, rispettivamente.

Non prendiamo in considerazione il Liceo scientifico tradizionale. Infatti, se pure tale indirizzo prevederebbe "Matematica con Informatica al primo biennio," nella realtà l'insegnamento dell'informatica rimane quasi sempre soltanto "un asterisco" che rimanda a una nota nei quadri

orari ufficiali. Per accorgersene, è sufficiente guardare i programmi svolti pubblicati nei siti web delle scuole.
Quindi, al momento, ci occupiamo solamente del Liceo Scientifico con opzione scienze applicate (LS-SA), che è l'unico percorso liceale che prevede ore curricolari di informatica.

Notiamo che molti Istituti e siti web dedicati alla scuola affermano [14]: "Proprio lo studio dell'informatica è, in questo liceo, messo in primo piano e sostituisce lo studio del latino, per offrire a tutti gli studenti una preparazione scientifica e tecnologica sempre più completa." Tale affermazione è, però, solo parzialmente vera, dal momento che nel Liceo scientifico tradizionale le ore di Lingua e cultura latina settimanali sono 3 e non 2. Quindi, nei cinque anni si hanno complessivamente 15 ore settimanali di latino, a fronte di sole 10 ore di informatica nel LS-SA.

Inoltre, le Indicazioni Nazionali per il LS-SA [8, p. 38] forniscono un quadro di contenuti estremamente ampio, che spazia su tutte le principali aree dell'informatica: "Dal punto di vista dei contenuti il percorso ruoterà intorno alle seguenti aree tematiche: architettura dei computer (AC), sistemi operativi (SO), algoritmi e linguaggi di programmazione (AL), elaborazione digitale dei documenti (DE), reti di computer (RC), struttura di Internet e servizi (IS), computazione, calcolo numerico e simulazione (CS), basi di dati (BD)."

A parole le Indicazioni Nazionali sembrano comprendere l'importanza della disciplina. Per esempio, in una nota si legge [8, p. 6]: "...la valenza metodologica dell'informatica nella formalizzazione e modellizzazione dei processi complessi e nell'individuazione di procedimenti risolutivi..." Nonostante queste dichiarazioni d'intenti, la situazione sopra illustrata dimostra come nei fatti il quadro normativo scolastico (non solo italiano) sia ancora molto distante dal cogliere:

1. la dimensione, la pervasività e la profondità della rivoluzione tecnico-scientifica che è alla base dell'attuale società della conoscenza (se pure a p. 27 delle Indicazioni nazionali si parli di "rivoluzione informatica");

2. la specificità e la valenza dell'informatica come disciplina in grado di sviluppare la logica e il pensiero computazionale;

3. l'importanza che può avere anche in ambito liceale la sperimentazione laboratoriale come strumento e modalità che favorisce e facilita apprendimenti tecnico-scientifici significativi (il famoso *learning by doing*).

In questo quadro l'insegnamento della programmazione ha un ruolo centrale e di fatto è l'unico ambito informatico che possa essere effettivamente sperimentato dagli studenti. A p. 38 si legge [8]: "Il rapporto fra teoria e pratica va mantenuto su di un piano paritario e i due aspetti vanno strettamente integrati evitando sviluppi paralleli incompatibili con i limiti del tempo a disposizione."

3 Python come linguaggio introduttivo

Il dibattito, a livello universitario ma non solo, riguardo al linguaggio ideale per introdurre la programmazione è incessante. Se il linguaggio Pascal dominava trent'anni fa, intorno al 2000 è stato soppiantato da Java, mentre, dal 2014, Python è emerso come il linguaggio più usato nei corsi introduttivi delle università americane [6].

Nel selezionare un linguaggio per apprendere la programmazione, si dovrebbero considerare due gruppi distinti: studenti non orientati verso una carriera informatica (NonCS) e studenti di informatica (CS). Il nostro interesse ricade sulla prima categoria: per questi studenti, molti studi

nel campo della didattica informatica suggeriscono Python. Questi lavori includono racconti quasi aneddotici di transizioni da linguaggi come C e Java a Python, per lo più con esito positivo, e studi più rigorosi come quelli di Koulori et al. [11], di Wainer e Xavier [15] e di Jayal et al. [9] che evidenziano miglioramenti nel problem solving e una riduzione significativa degli errori sintattici e di compilazione. Esistono anche indicazioni contrarie, come il lavoro di Alzaharani et al. che sembra indicare un più alto livello di difficoltà nella soluzione di problemi per studenti che utilizzano Python rispetto a studenti che utilizzano C++ [1]. Gli autori stessi, tuttavia, esprimono perplessità sulla significatività statistica dei risultati, mancando informazioni sulla popolazione di studenti, e suggeriscono che sono necessarie ulteriori analisi.

Ci sono anche ricerche specificamente dedicate alla programmazione nelle scuole superiori; il primo è il lavoro di Elkner [3], che ha provato ad utilizzare Python con successo già nel 2001, dopo aver abbandonato Pascal e aver provato C++ per due anni con scarsi risultati. Particolarmente importanti i lavori dell'Università di Turku [5, 13], che sostengono Python, e l'articolo di Mannila e de Raadt [12], che considerano Python come uno dei candidati più forti come primo linguaggio di programmazione, elencando varie caratteristiche che un linguaggio introduttivo dovrebbe avere. Tra i criteri più citati vi sono la semplicità della sintassi, la facilità di realizzare input/output, la capacità di fornire feedback immediato attraverso scripting e interfaccia CLI e un ecosistema di librerie di facile installazione e utilizzo.

Per quanto riguarda gli studenti di informatica (CS), il dibattito rimane aperto. Molte università persistono nell'utilizzo di C e Java come linguaggi iniziali, sostenute da corsi lunghi e intensivi. Alcuni articoli sconsigliano l'uso di Python a livello universitario; ad esempio, Hunt critica l'approccio di Python per il suo livello eccessivamente "alto" rispetto alle caratteristiche fisiche della macchina [7].

4 Python nel contesto LS-SA: vantaggi

La scelta del linguaggio di programmazione ha un ruolo fondamentale in vista degli obiettivi didattici dell'apprendimento della programmazione. Python sembra, in questa fase storica e nel contesto del LS-SA, la scelta più idonea. Per motivare questa affermazione, adottiamo alcuni descrittori per valutare, in maniera più che altro qualitativa, il raggiungimento di alcuni obiettivi didattici:

O1 facilità di apprendimento,

O2 livello di approfondimento,

O3 coinvolgimento dello studente,

O4 diffusione nell'ambito della ricerca, sia scientifica che umanistica, e nell'ambito professionale.

Numerose esperienze e ricerche a livello scolastico [2] e universitario [6] mostrano che sul primo punto Python risulta sufficientemente attrezzato grazie a una sintassi semplificata, che lo rende più "inclusivo" di altri linguaggi, in particolare di quelli C-like, che sono più complicati e/o prolissi.

Si confronti nella tabella sotto la scrittura del programma base Hello World in C, C++, Java, Python. Pur essendo banale, l'esempio mostra come Python richieda di scrivere molto meno codice di altri linguaggi. Inoltre, non richiede di anticipare e/o nascondere concetti più avanzati quali l'inclusione di moduli di libreria, il concetto di funzione e di classe, i modificatori di visibilità etc.

```c
#include <stdio.h>

int main()
{
  printf("Hello World!");
}
```

```cpp
#include <iostream.h>

int main()
{
  cout << "Hello World!";
}
```

```java
class Hello
{
  public static void main(String args[])
  {
    System.out.println("Hello World!");
  }
}
```

```python
print("Hello World")
```

Python è molto più vicino al linguaggio naturale e al linguaggio matematico. Per esempio, si consideri un costrutto come il seguente, che stampa tutti i colori di una lista:

```python
colors = ["red", "green", "blue"]
for color in colors:
    print(color)
```

La versione C (ma lo stesso discorso vale per altri linguaggi come, per esempio, Pascal e Java) è molto più complessa e richiede l'introduzione di un elevato numero di concetti:

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
  char *colors[] = { "red", "green", "blue", };
  for (int i = 0; i < 3; ++i) {
    printf("%s\n", colors[i]);
  }
}
```

Con riferimento al secondo (O2) e al terzo punto (O3), osserviamo che il livello di astrazione offerto da Python permette di arrivare a svolgere, nel contesto del corso del LS-SA, programmi che risolvono situazioni non banali, che possono spaziare dalla simulazione alla fisica, dalla linguistica computazionale alla grafica ecc. Infatti, osservando i programmi didattici effettivamente svolti nei corsi del LS-SA, si può affermare che Python permette di arrivare a toccare e sperimentare argomenti che con altri linguaggi non è possibile affrontare.

Nell'ambito scientifico (punto O4), Python è tra i linguaggi più utilizzati, sia in settori prettamente informatici, sia, in particolare, in settori non informatici (fisica, biologia ecc.). L'unica alternativa popolare al momento è R, un linguaggio progettato per l'analisi statistica. Il vantaggio di scegliere Python rispetto a R è quello di avere un unico linguaggio di programmazione che può essere utilizzato in ambiti estremamente diversi.

La nostra opinione è che Python ben si adatta alle scuole superiori, in modo particolare nei LS-SA, perché permette di concentrarsi sui concetti fondamentali della programmazione

(sequenza di comandi, ripetizioni, istruzioni condizionali, modularità, ricorsione) e di porre l'accento sul problem solving piuttosto che sui dettagli sintattici. Inoltre, la possibilità di integrarsi con un insieme amplissimo di librerie permette di estendere il suo uso a casi applicativi reali estremamente diversi, fornendo così agli studenti un senso di "rilevanza" che l'uso di linguaggi di più basso livello non danno, e questo lo rende interessante anche per gli istituti tecnici.

5 Python nel contesto LS-SA: svantaggi

Alcuni sostengono che il fatto che Python sia *dynamically typed* (ovvero il tipo delle variabili non viene dichiarato esplicitamente ma viene determinato a runtime) generi delle incomprensioni per chi inizia a programmare [10]. Si consideri, per esempio, una funzione per il calcolo del fattoriale:

```
def factorial(n):
    tot = 1
    for i in range(2, n + 1):
        tot = tot * i
    return tot
```

Se la funzione viene richiamata con un valore `float` (per esempio, `factorial(5.0)`), il codice restituisce un errore durante l'esecuzione. Il problema si aggrava qualora si arrivi a introdurre strutture dati più complesse ma con sintassi simile, quali liste e dizionari; ad esempio, `A["name"]` ha senso se `A` è un dizionario, ma genera un errore a runtime se `A` è una lista.

Alcuni studi sembrano indicare che questo non comporta particolari problemi durante l'apprendimento del linguaggio. La conferma viene anche da studi che hanno analizzato il passaggio da un corso CS1 basato su Python ad un corso CS2 basato su C++ [4]. Inoltre, nella nostra esperienza di insegnamento di Python nella scuola superiore, iniziata nel 2006, non abbiamo mai riscontrato tale tipologia di problema. Ciò nonostante, è fondamentale fare particolare attenzione nel descrivere il concetto di tipo, per esempio mostrando le funzioni predefinite `type()` e `isistance()` per rendere i tipi "visibili" e più "tangibili."

Il problema sulla distinzione dei tipi di dati viene comunque affrontato in Python quando, per esempio, si deve convertire in numero (`int` o `float`) un dato inserito dall'utente e acquisito con la funzione predefinita `input()`, oppure si devono concatenare, per esempio, una stringa e un numero e si deve utilizzare il cast esplicito, come nel seguente codice:

```
anni = int(input("Quanti anni hai? ")) # Conversione esplicita da stringa a intero
print("Hai "+ str(anni) +" anni") # Conversione esplicita in stringa
```

Inoltre, la mancanza sulla dichiarazione dei tipi di dati è stata parzialmente risolta a partire da Python 3.5 che ha introdotto le annotazioni sul tipo di dato per variabili e funzioni (Figura 1). nel seguente modo:

```
explicit_number: type

def function(explicit_number: type) -> type:
    pass
```

Tale informazione (type annotation) è utile al programmatore e viene utilizzata da strumenti come l'IDE PyCharm (Figura 1) per trovare e segnalare possibili errori di tipo durante l'editing, ma non previene il verificarsi di errori di tipo durante l'esecuzione. Infatti, la dinamicità di Python consente sempre di assegnare valori di tipo diverso alla stessa variabile.

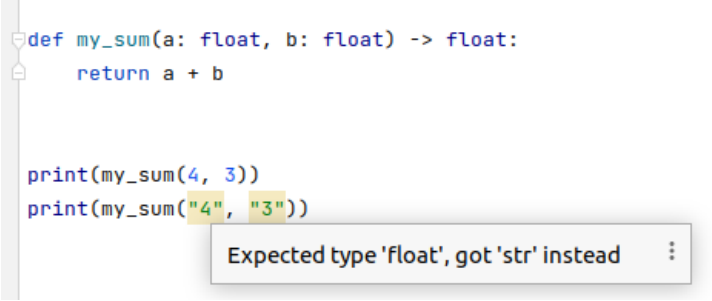


Figura 1: PyCharm segnala la chiamata di funzione con tipi di dati diversi (stringhe) da quelli indicati nella dichiarazione della funzione (numeri float)

6 Conclusioni

In conclusione, Python offre numerosi vantaggi per l'apprendimento della programmazione, come la sintassi semplificata e la vicinanza al linguaggio naturale e a quello matematico, la possibilità di trattare argomenti diversi e complessi e l'ampia applicabilità in diversi settori. Tuttavia, è fondamentale che gli insegnanti siano consapevoli delle limitazioni e adottino le opportune strategie per superarle, come, eventualmente, l'uso delle annotazioni di tipo per una comprensione solida dei concetti di tipo. In questo modo, Python può essere utilizzato efficacemente per insegnare programmazione e problem solving, fornendo agli studenti competenze preziose per il loro futuro accademico e professionale.

Riferimenti bibliografici

[1] Nabeel Alzahrani, Frank Vahid, Alex Edgcomb, Kevin Nguyen, and Roman Lysecky. Python versus C++: An analysis of student struggle on small coding exercises in introductory programming courses. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 86–91, New York, NY, USA, 2018. Association for Computing Machinery.

[2] Maurizio Boscaini. L'informatica e python al liceo: conoscere, pensare, progettare, realizzare. In *Didamatica*, 2009.

[3] Jeffrey Elkner. Using Python in a high school computer science program. In *9th International Python Conference*, 2001.

[4] Richard J. Enbody, William F. Punch, and Mark McCullen. Python cs1 as preparation for c++ cs2. *SIGCSE Bull.*, 41(1):116–120, 2009.

[5] Linda Grandell, Mia Peltomaki, Ralph-Johan Back, and Tapio Salakoski. Why complicate things? introducing programming in high school using python. In *Proc. of the 8th Australasian Conference on Computing Education (ACE'06)*, volume 52, pages 71–80, 2006.

[6] Philip Guo. Python is now the most popular introductory teaching language at top u.s. universities. *Communication of the ACM*, 2014(7), 2014.

[7] John M. Hunt. Python in cs1 - not. *J. Comput. Sci. Coll.*, 31(2):172–179, 2015.

[8] Liceo scientifico opzione scienze applicate. https://www.indire.it/lucabas/lkmw_file/licei2010/indicazioni_nuovo_impaginato/_Liceo%20scientifico%20opzione%20Scienze%20Applicate.pdf. Accessed: 2023-07-02.

[9] Ambikesh Jayal, Stasha Lauria, Allan Tucker, and Stephen Swift. Python for teaching introductory programming: A quantitative evaluation. *Innovation in Teaching and Learning in Information and Computer Sciences*, 10, 02 2011.

[10] Lisa C. Kaczmarczyk, Elizabeth R. Petrick, J. Philip East, and Geoffrey L. Herman. Identifying student misconceptions of programming. In *Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE '10)*, 2010.

[11] Theodora Koulouri, Stanislao Lauria, and Robert D. Macredie. Teaching introductory programming: A quantitative evaluation of different approaches. *Trans. Comput. Educ.*, 14(4):26, 2015.

[12] Linda Mannila and Micahel de Raadt. An objective comparison of languages for teaching introductory programming. In *Proceedings of the 6th Baltic Sea Conference on Computing education research*, 2006.

[13] Linda Mannila, Mia Peltomäki, and Tapio Salakoski. What about a simple language? analyzing the difficulties in learning to program. *Computer Science Education*, 16(3):211–227, 2006.

[14] Liceo scientifico scienze applicate. https://www.studenti.it/liceo_scientifico_scienze_applicate.html. Accessed: 2023-07-02.

[15] Jacques Wainer and Eduardo C. Xavier. A controlled experiment on python vs c for an introductory programming course: Students' outcomes. *ACM Trans. Comput. Educ.*, 18(3):12, 2018.

[16] Liceo scientifico. https://it.wikipedia.org/wiki/Liceo_scientifico. Accessed: 2023-07-02.

[17] Riforma gelmini. https://it.wikipedia.org/wiki/Riforma_Gelmini. Accessed: 2023-07-02.

Python per tutti i gusti: Tre diversi approcci per introdurre neofiti alla programmazione

Daniele Traversaro, Giorgio Delzanno, Giovanna Guerrini, and Davide Ponzini

DIBRIS, Università degli Studi di Genova, Italy
{nome.cognome}@dibris.unige.it

Sommario

Nel presente articolo vengono presentati tre diversi percorsi progettati per introdurre alla programmazione studenti privi di precedente esperienza. I tre percorsi sono accomunati dall'uso del linguaggio Python che viene però introdotto in tre diverse modalità. I percorsi, di durata complessiva di circa 15 ore, sono stati proposti come attività PCTO a studenti del triennio della scuola secondaria di secondo grado.

1 Contesto e motivazione

Il pensiero computazionale è diventato un'abilità fondamentale per tutti, a tal punto da poter essere considerata come la quarta abilità di base oltre al "saper leggere, scrivere e far di conto" [6]. Negli ultimi anni, infatti, si sono diffusi molti movimenti per promuovere il pensiero computazionale attraverso attività di coding rivolte a varie fasce d'età, utilizzando approcci *plugged* o *unplugged* e linguaggi di programmazione visuali o testuali. Tra le strategie più utilizzate per coinvolgere gli studenti nell'informatica, troviamo l'apprendimento basato su progetti, la peer education, la pair programming, la gamification dell'apprendimento e altro ancora.

In questo articolo presentiamo tre brevi percorsi di introduzione alla programmazione in Python rivolti a studenti di scuola secondaria di secondo grado senza precedenti esperienze di programmazione. Queste unità sono state progettate con durata complessiva di circa 15 ore e sono state inserite in percorsi PCTO proposti nell'anno scolastico 2022/2023 a studenti di classi quarte e quinte.

Progettare un percorso di introduzione alla programmazione di durata così limitata per neofiti non è facile. Infatti, per essere efficace in termini di coinvolgimento, motivazione e risultati di apprendimento, deve essere personalizzato tenendo conto di una serie di aspetti, oltre alla fascia d'età, quali il contesto scolastico, la familiarità con la tecnologia, gli interessi e le specifiche esigenze di apprendimento.

I tre percorsi che presentiamo condividono gli stessi obiettivi di apprendimento, ma sono progettati con pedagogie diverse proprio per rispondere alle esigenze e agli interessi specifici delle classi e dei singoli studenti. In particolare, abbiamo sperimentato un approccio tradizionale ma con interfaccia a blocchi, uno di physical computing/tinkering utilizzando BBC micro:bit[1], e uno con una pedagogia "data-centrica" [4] in cui lo studente parte da strutture di dati più complesse ma con cui è abituato a lavorare, come le tabelle, attorno alle quali familiarizza con i vari costrutti di programmazione.

Il primo approccio è quello adottato nei corsi tradizionali di introduzione alla programmazione (CS1), che mettono l'accento sulle strutture di controllo e la programmazione imperativa, ma anche sulle iniziative di coding che enfatizzano la risoluzione dei problemi anziché la sintassi, utilizzando linguaggi visivi a blocchi con un approccio creativo di apprendimento. A questo

[1] BBC micro:bit: https://microbit.org

 ISBN: 978-88-6629-075-9

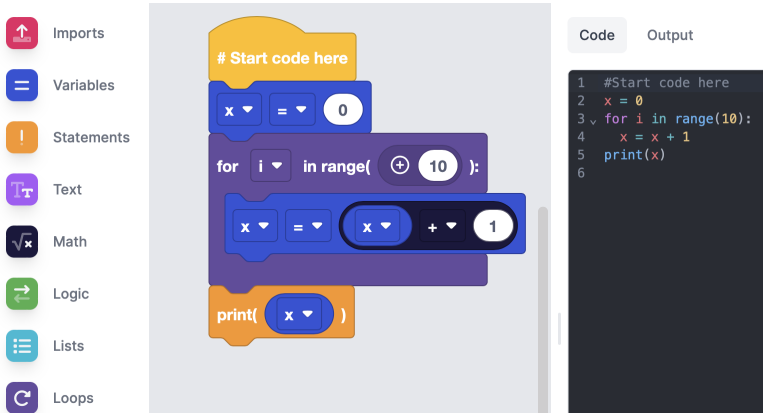


Figura 1: Esempio di programma EduBlocks

scopo, abbiamo utilizzato la piattaforma educativa EduBlocks[2]. Inoltre, per la versione remota del percorso, abbiamo utilizzato anche elementi di *gamification*.

Nel secondo approccio, gli studenti programmano il micro:bit utilizzando Python per interagire con i sensori e il display. Questo permette loro di creare progetti interattivi e vedere l'impatto immediato dei programmi nel mondo reale, aggiungendo un aspetto tangibile alla programmazione.

Nel terzo approccio, gli studenti utilizzano la nostra libreria basata su Pandas per manipolare dataset reali. Questo approccio pratico alla programmazione permette agli studenti di rispondere alle esigenze del mondo reale.

L'obiettivo di questo articolo è duplice: fornire una panoramica dei tre percorsi e discutere e confrontare l'efficacia di questi diversi approcci in termini di risultati di apprendimento, interesse e coinvolgimento nel contesto dell'introduzione degli studenti della scuola secondaria alla programmazione.

Riteniamo che la condivisione delle nostre esperienze con la comunità degli insegnanti di informatica possa essere un'opportunità di discussione e riflessione verso una didattica più accessibile, progettata partendo dalle esigenze e dagli interessi degli studenti.

Nel seguito dell'articolo vengono brevemente presentati i tre percorsi per poi discutere come sono stati accolti dagli studenti.

2 Programmazione Python a blocchi con *EduBlocks*

La seguente unità didattica è stata proposta all'interno del programma PCTO "Stage Informatica 2023", organizzato dal DIBRIS dell'Università di Genova e finalizzato ad aiutare gli studenti delle scuole secondarie di secondo grado a esplorare il campo dell'informatica e a orientarsi nel mondo universitario. Il percorso di coding faceva parte di una proposta più ampia, fornendo una conoscenza di base che sarebbe stata utile per tutte le altre attività dello stage.

In questo percorso, gli studenti vengono introdotti alla programmazione attraverso un approccio di apprendimento attivo che si basa su una pedagogia "tradizionale", la quale enfatizza le strutture di controllo e la programmazione imperativa. Tuttavia, il percorso permette agli

[2]EduBlocks: https://edublocks.org

studenti di concentrarsi sulla risoluzione creativa dei problemi, ponendo l'accento sulle conoscenze concettuali e strategiche invece di quelle sintattiche [1] che possono causare comportamenti imprevisti o errori di esecuzione. Utilizziamo EduBlocks, una piattaforma semplificata che offre un'interfaccia visiva per creare programmi Python tramite il trascinamento e il rilascio di blocchi di codice, vedi Fig. 1. Questo aiuta gli studenti a familiarizzare con la sintassi senza commettere errori e semplifica la comprensione e la progettazione del codice grazie a un set di istruzioni ridotto, facilitando l'apprendimento iniziale.

Il percorso è suddiviso in otto moduli, con brevi sessioni di lezioni interattive che combinano introduzione di nuovi concetti e quiz, mirati a approfondire e consolidare, applicandole su esempi, le nozioni introdotte. Dopo queste sessioni, gli studenti partecipano ad attività pratiche di gruppo, in cui possono mettere in pratica le conoscenze apprese.

Gli argomenti trattati sono i seguenti:

- Algoritmi e diagrammi di flusso: introduzione alla nozione di algoritmo e specifica di semplici algoritmi mediante diagrammi di flusso.
- Variabili: nozione di variabile in Python, assegnamento, casting.
- Tipi di dato di base e I/O: introduzione a interi, float, stringhe e Booleani, semplici operazionni sui valori, comandi `print()` and `input()`.
- Operatori Booleani e di confronto.
- Strutture di controllo condizionale: comandi `if`, `if-else`, e `if-elif-else`.
- Liste: nozione di lista, manipolazione di liste (aggiunta di elementi, accesso a posizioni e porzioni).
- Iterazione: iterazione certa e incerta, comandi `for` e `while`.
- Funzioni: definizione e invocazione di funzioni. Esempi di librerie.

2.1 EduBlocks & Gamification: Percorso a distanza

Durante l'emergenza sanitaria da Covid-19, negli anni accademici 2019/2020 e 2020/2021, abbiamo adattato il precedente percorso per rispondere alle esigenze della didattica a distanza. Un elemento chiave di questa trasformazione è stata l'introduzione della gamification attraverso la piattaforma Smart O.C.A.[3]. Questa piattaforma, una web app di edutainment, ha consentito agli studenti di partecipare al gioco online in tempo reale, sia in modalità individuale che collaborativa.

Il gioco si presentava come un gioco da tavolo con un tabellone composto da varie celle colorate, vedi Fig. 2. In particolare, ogni cella rappresentava un elemento dell'unità di apprendimento, come brevi video lezioni, quiz con feedback istantaneo e task di programmazione in EduBlocks (corrispondenti a quelli dell'unità in presenza). Nel caso di risposte errate ai quiz, agli studenti venivano proposti ulteriori quiz, altrimenti passavano direttamente alla casella contenente il task pratico. Tuttavia, i task finali richiedevano interazioni sincrone e una valutazione manuale da parte degli insegnanti/tutor coinvolti. Infatti, gli studenti dovevano attendere la correzione prima di poter procedere con il gioco. Per la correzione dei task, abbiamo coinvolto

[3]Smart O.C.A. (Online/Offline Challenge Activity): https://www.edutainmentformula.com/web-app/smart-oca/

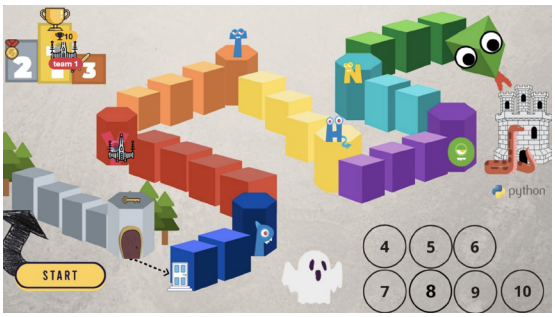


Figura 2: Smart O.C.A. Python.

gli studenti tutor iscritti alla nostra laurea triennale in informatica, promuovendo una cultura di educazione e tutoraggio tra pari.

Tra le varie caratteristiche di gamification, abbiamo introdotto una classifica di gioco in tempo reale, icone, livelli, punti e quiz collaborativi per rendere l'esperienza di apprendimento più coinvolgente e interattiva.

3 Programmazione con *micro:bit*

Il secondo percorso adotta un approccio di physical computing alla programmazione utilizzando il BBC micro:bit [5], un dispositivo programmabile tascabile. Il dispositivo micro:bit è dotato di un microcontrollore, sensori come pulsanti, accelerometro e bussola, un display a LED e connettività wireless. Supporta diversi linguaggi di programmazione, sia visivi che testuali, come Scratch e Python. L'utilizzo del micro:bit consente di sperimentare modelli di insegnamento/apprendimento capaci di mettere gli alunni al centro del processo formativo e di sviluppare il pensiero computazionale in maniera creativa e l'apprendimento STEAM.

In termini di obiettivi di apprendimento e contenuti, il percorso è simile all'unità con EduBlocks, ma l'uso del micro:bit introduce un'esperienza di programmazione guidata dagli eventi.

Dopo ogni breve spiegazione, gli studenti mettono in pratica ciò che hanno appreso utilizzando un ambiente di programmazione testuale per browser[4] appositamente ottimizzato per il micro:bit che include anche una funzione di simulazione del dispositivo.

Gli studenti scrivono semplici programmi per controllarne il comportamento in risposta a eventi specifici, come la pressione di un pulsante o il rilevamento del movimento tramite l'accelerometro. Gli esercizi da implementare alla fine di ogni mini-lezione sono i seguenti:

- Creare un programma che simula un dado. Premendo il tasto A si incrementa una variabile numerica, mentre premendo il tasto B si visualizza il valore della variabile a schermo. Il valore della variabile deve essere compreso tra 1 e 6.
- Modificare il programma precedente per consentire l'incremento e il decremento di 1 del valore del dado premendo i pulsanti A e B. Il valore aggiornato viene visualizzato sullo schermo.
- Visualizzare il simbolo della faccia del dado anziché un numero, utilizzando gli oggetti della classe Image forniti dal micro:bit. Vedi Fig. 3.

[4]Editor Python per microbit: https://python.microbit.org/v/3

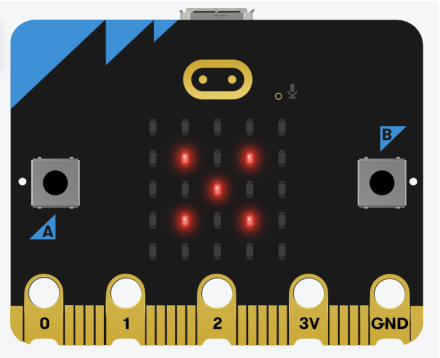


Figura 3: micro:bit che mostra la faccia numero 5 del dado.

- Aggiungere la funzionalità di generare un numero casuale corrispondente alla faccia del dado quando si verifica un evento di movimento rilevato dall'accelerometro.
- Utilizzare una lista per gestire le facce del dado, migliorando l'efficienza e la manutenibilità del codice.
- Implementare una funzione per controllare la parità del numero della faccia del dado.
- Simulare il lancio di due dadi combinando i valori generati da due micro:bit tramite il protocollo radio e paradigma message passing.

Gli esercizi hanno coperto diverse competenze di programmazione, inclusa la gestione degli eventi, le buone pratiche per rendere il codice leggibile, riutilizzabile e manutenibile, fino all'esperienza con i sistemi distribuiti.

4 Programmazione con approccio *Data-Centric*

Il terzo percorso si basa sulla pedagogia *data centric* [4], che è stata sviluppata per ripensare il contenuto dei corsi CS1 adottando un approccio interdisciplinare focalizzato sui dati. Questo metodo mira a coinvolgere e supportare una varietà di studenti, non solo quelli provenienti dal campo dell'informatica. L'approccio è già stato adottato in vari corsi di introduzione all'informatica o alla scienza dei dati che utilizzano linguaggi o librerie personalizzate. Ad esempio, presso la Brown University viene offerto un corso chiamato "CSCI 0111"[5] che segue l'approccio data centrico utilizzando il linguaggio di programmazione Pyret[6] [2]. Presso l'University of California, Berkeley, viene insegnato il corso "Data 8"[7], utilizzando il linguaggio Python ma con la libreria didattica "datascience"[8].

Il percorso è strutturato con brevi sessioni frontali/interattive di introduzione ai concetti alternate ad attività pratiche. Gli argomenti introdotti sono:

- Introduzione alla scienza dei dati e all'informatica, con un focus sulla programmazione.
- Algoritmi, esecutori, programmi e linguaggi di programmazione.

[5] Corso data-centrico "CSCI 0111": https://cs.brown.edu/courses/csci0111/
[6] Linguaggio di programmazione Pyret: https://pyret.org
[7] Corso "Data 8": https://data.berkeley.edu/education/courses/data-8
[8] Libreria datascience, Berkeley: https://github.com/data-8/datascience

```
s1 = series("Bob", "Alice", "Ted", "Carol")
s2 = series(2, 0, 5, 9)
df = concat(s1, s2)
df.show()
```

	0	1
0	Bob	2
1	Alice	0
2	Ted	5
3	Carol	9

Figura 4: Esempio di codice Toypandas.

- Variabili, tipi di dati, tipizzazione dinamica.
- Strutture dati tabellari: Series e Dataframe.
- Operatori di confronto e logici applicati ai dati (ad esempio, maschere e filtri).

Gli studenti, una volta ottenuta una base di conoscenze sufficiente, iniziano ad analizzare un dataset reale. Durante questa attività, i nuovi concetti di programmazione vengono introdotti gradualmente e in modo naturale, solo quando necessari per eseguire azioni specifiche sui dati. Gli argomenti vengono quindi presentati in un ordine diverso rispetto ai due percorsi precedenti. Durante tale fase sono stati affrontati i seguenti argomenti:

- Importazione dei dati (nel nostro caso, il dataset IMDb in formato .csv).
- Tecniche di base per l'esplorazione dei dati.
- Gestione di valori mancanti, duplicati e outlier.
- Interrogare i dati utilizzando le API dei Dataframe, in modo simile ad una interrogazione SQL. Esempio: `movies_df[movies_df['Rating'] >= 8]`
- Trasformazione dei dati utilizzando funzioni definite dall'utente e costrutti condizionali.
- Visualizzazione dei dati tramite grafici e mappe.

In questa sperimentazione abbiamo utilizzato la libreria Toypandas[9], sviluppata da noi e basata su Pandas. Toypandas semplifica la sintassi e rende il codice più espressivo. Ad esempio, utilizza la notazione `.indexlocate[]` per accedere alle righe del Dataframe, rendendo il significato dell'istruzione più chiaro. La libreria semplifica anche molte operazioni sui dati, nascondendo dettagli complessi come l'inizializzazione delle colonne senza richiedere la conoscenza di liste o dizionari. Inoltre, quando un elemento viene rimosso da una colonna, gli indici numerici rimangono in ordine sequenziale. Infine, i dataframe di ToyPandas sono compatibili con i metodi della libreria grafica Matplotlib. In Figura 4, un esempio di codice che utilizza ToyPandas.

[9]toypandas: https://pypi.org/project/toypandas/

5 Discussione

Al termine di ogni percorso sono stati somministrati agli studenti dei post-questionari individuali. Questi questionari includevano diverse domande su scala Likert per valutare la percezione degli studenti del livello di interesse, di difficoltà, di utilità delle attività proposte nel percorso di introduzione alla programmazione. Inoltre, sono state incluse tre domande relative alla programmazione Python, che coprono argomenti comuni ai tre percorsi, in particolare condizioni booleane, iterazione certa e indicizzazione di liste/serie. Il questionario finale chiedeva infine agli studenti di valuare l'efficacia dei percorsi PCTO ai fini dell'orientamento.

I tre percorsi di introduzione alla programmazione con approcci diversi hanno tutti ottenuto buoni risultati. I questionari di valutazione hanno confermato le impressioni in aula, in particolare il percorso con il micro:bit è sembrato il più efficace in termini di coinvolgimento e approccio "learning by making", in pieno spirito costruzionista[10]. D'altra parte, la pedagogia data centrica presenta una maggiore difficoltà iniziale che potrebbe scoraggiare gli studenti. Tuttavia, ben presto diventa coinvolgente grazie alla manipolazione di un dataset reale. In questa pedagogia, l'iterazione è implicita, simile ai linguaggi "dichiarativi" come SQL [3], semplificando la macchina nozionale sottostante. La curva di apprendimento inizia con una salita ripida per poi scendere rapidamente, rendendo l'approccio ancora più accessibile rispetto ad altri. Infine, l'approccio tradizionale a blocchi si è dimostrato in generale meno coinvolgente, proprio a causa della mancanza di un elemento di partenza concreto come un dataset o un artefatto tangibile come il micro:bit. Tuttavia, la versione gamificata ha ottenuto risultati paragonabili agli altri percorsi, indicando che gli elementi di gamification hanno efficacemente mitigato le sfide dell'ambiente di apprendimento a distanza, che a volte può risultare meno coinvolgente.

L'analisi dei risultati relativi alle tre domande di programmazione proposte nel questionario finale ha rilevato risultati comparabili per i tre diversi percorsi. Questo risultato è abbastanza inaspettato, poiché avevamo previsto che i diversi approcci, offrendo diverse prospettive sugli argomenti, avrebbero potuto introdurre una leggera distorsione nei risultati dell'apprendimento. Tuttavia, così non è stato, suggerendo che questi percorsi possono essere considerati equivalenti in termini di risultati di apprendimento e possono quindi essere utilizzati in contesti di apprendimento diversi. Ad esempio, gli studenti che hanno partecipato al percorso basato sulla pedagogia incentrata sui dati non hanno ottenuto risultati migliori degli altri nelle domande sugli operatori booleani e di indicizzazione, nonostante l'uso estensivo di questi concetti nella manipolazione dei Dataframe. Allo stesso modo, gli studenti hanno ottenuto risultati paragonabili con gli altri nella domanda sull'iterazione certa (ciclo `for`), anche se hanno utilizzato un'iterazione più implicita.

Dal punto di vista dell'efficacia ai fini dell'orientamento, infine, tutte e tre le attività PCTO si sono rivelate utili: alcuni studenti hanno scoperto attraverso la partecipazione al percorso che l'informatica, contrariamente a quanto pensavano prima della partecipazione, non era la loro vocazione, mentre altri, inizialmente convinti che l'informatica non li interessasse, dopo la partecipazione al percorso hanno dichiarato di avere cambiato opinione e di considerarla per i loro studi futuri.

[10] Il costruzionismo è una teoria dell'apprendimento, basata sulla teoria del costruttivismo, sviluppata da Seymour Papert.

Riferimenti bibliografici

[1] Piraye Bayman and Richard E Mayer. Using conceptual models to teach basic computer programming. *Journal of Educational Psychology*, 80(3):291, 1988.

[2] Kathi Fisler. Data-centricity: Rethinking introductory computing to support data science. In *1st International Workshop on Data Systems Education*, pages 1–3, 2022.

[3] Shriram Krishnamurthi and Kathi Fisler. 13 programming paradigms and beyond. *The Cambridge handbook of computing education research*, page 377, 2019.

[4] Shriram Krishnamurthi and Kathi Fisler. Data-centricity: a challenge and opportunity for computing education. *Communications of the ACM*, 63(8):24–26, 2020.

[5] Sue Sentance, Jane Waite, Lucy Yeomans, and Emily MacLeod. Teaching with physical computing devices: the bbc micro: bit initiative. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, pages 87–96, 2017.

[6] Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

Sonic TBL: Un Percorso Sonico da Creatività a Didattica dell'Informatica

Giorgio Delzanno, Giovanna Guerrini, and Daniele Traversaro

DIBRIS, Università degli Studi di Genova, Italy
{name.surname}@unige.it

Sommario

Nel presente articolo vengono presentate alcune attività basate su Sonic Pi, nato per live music coding performance, e sul metodo didattico Team-Based Learning (TBL) per condurre esperimenti didattici sull'introduzione di concorrenza e multithreading in vari momenti del percorso della Laurea Triennale in Informatica della nostra Università.

1 Contesto e Motivazioni

Imparare la programmazione è considerato universalmente un processo di apprendimento difficile e faticoso in quanto strettamente legato alle capacità di ragionamento logico, problem solving e pensiero creativo. Le numerose iniziative nazionali ed internazionali volte ad anticipare l'introduzione dei concetti computazionali nella scuola primaria e secondaria sono un chiaro esempio del tentativi in atto per migliorare la preparazione di base della disciplina. Nonostante questi sforzi e nonostante l'enorme gap con richiesta del mercato professionale, i corsi di laurea in Informatica soffrono di un elevato tasso di abbandono nei primi anni. Da recenti statistiche relative ai corsi di laurea italiani, risulta che uno studente su quattro iscritto ad Informatica passa ad un'altro corso di laurea o lascia l'università tra il primo e secondo anno. Una delle cause di questo alto tasso di abbandoni è sicuramente legata alle difficoltà iniziali nell'affrontare corsi di programmazione offerti a livello universitario. Per questo motivo la necessità di proporre metodi didattici innovativi, ad esempio basato sull'individuazione di Notional Machine adatte al modello mentale di studenti provenienti da diversi tipi di scuole, è diventato un obiettivo della maggior parte dei corsi di laurea italiani.

Il corso di laurea in Informatica della nostra Università attira negli ultimi anni più di 300 matricole con background eterogeneo. Gli studenti sono motivati principalmente dalle opportunità di lavoro offerte dalla laurea, e spesso solo una piccola parte di loro ha motivazioni e attitudini favorevoli all'apprendimento. Le lacune nelle conoscenze e nelle competenze di base sono spesso complementari. Ad esempio, chi ha un background di programmazione di solito non ha una solida base matematica e viceversa. Gli studenti con poche competenze di base possono quindi essere scoraggiati dall'osservare compagni di corso che raggiungono gli obiettivi dei corsi introduttivi con meno sforzo. Gli studenti con un background adeguato, invece, spesso non ricevono sufficienti incentivi per approfondire gli argomenti forzatamente introduttivi. Molti dei nostri studenti inoltre hanno difficoltà nel lavorare in gruppo con carenze generali nel relazionarsi a compagni e docenti. Il problema persiste anche negli anni successivi, in cui molti studenti sembrano non aver acquisito né metodologie adeguate né la necessaria autonomia. A partire dall'anno accademico 2019/20 il corso di laurea ha avviato una serie di azioni di innovazione didattica e alcuni esperimenti per introdurre metodi didattici attivi strettamente integrati con i corsi dei primi anni. In questo contesto sono stati introdotti metodi didattici non tradizionali come flipped class, team-based learning, think-pair-share, debate e argomenti aggiuntivi ai contenuti tradizionali dei corsi per creare collegamenti, stimoli, puntatori ad argomenti avanzati

e gruppi di lavoro che potessero coinvolgere studenti di diversi anni (es studenti di dottorato e magistrale come mentor per studenti della triennale).

Nel presente articolo, esaminiamo una serie di attività basate sull'uso combinato di un domain specific language chiamato Sonic Pi[1], nato per live music coding performance, e del metodo didattico Team-Based Learning (TBL) per condurre esperimenti didattici sull'introduzione di concorrenza e multithreading in vari momenti del percorso della laurea in Informatica.

Sonic Pi è un domain-specific language basato su un server Supecollider e supportato da un editor di facile utilizzo per la creazione di basi musicali. Il sistema permette di introdurre sia concetti computazionali di base (strutture dati, costrutti, funzioni, ecc) sia concetti che giocano un ruolo sempre più importante nella programmazione come multi-threading e hot code swap [1, 2, 10, 3]. La semplicità dello strumento fornisce un'esperienza di apprendimento naturale e coinvolgente, in particolare attraverso il live coding (basato appunto su hot code swapping) e la ricca resa sonora (multi-strumentale grazie all'uso di sample di sintetizzatori) dei corrisponenti programmi. Il nostro esperimento didattico è stato ripetuto in tre diversi anni accademici, pre e post pandemia, esplorando formati e programmando interventi sia al primo che al terzo anno, anche per valutare, in momenti diversi del percorso di studio, la percezione degli studenti verso questo tipo di attività. In questo articolo presenteremo brevemente le varie edizioni, esplorando obiettivi di apprendimento, risultati preliminari ed implicazioni degli esperimenti facendo riferimento a quattro insegnamenti: Introduzione alla Programmazione (IP) primo semestre del primo anno, Architettura dei Calcolatori (ADC) annuale al primo anno, Programmazione Concorrente e Algoritmi Distribuiti (PCAD) primo semestre del terzo anno, Informatica per Creatività, Didattica e Divulgazione (ICDD) insegnamento a scelta al terzo anno.

2 Fase 1: Programmazione Creativa (2019/20)

Durante l'anno accademico 2019/2020, abbiamo avviato un progetto pilota che ha utilizzato Sonic Pi in combinazione con la metodologia didattica del Team-Based Learning (TBL). Questo progetto era rivolto alle matricole del primo anno iscritte al corso di laurea in informatica. La sperimentazione si è posta due obiettivi principali. Da un lato, volevamo fornire agli studenti competenze trasversali altamente richieste nel mondo del lavoro, come la creatività, la capacità di lavorare in squadra e di comunicare efficacemente con gli altri, nonché la capacità di mediare le proprie idee con quelle degli altri. L'intento era abbattere lo stereotipo dell'informatico come persona asociale e favorire il lavoro in un ambiente interdisciplinare, collaborando con persone provenienti da diverse esperienze professionali. Per raggiungere questo obiettivo, abbiamo creato gruppi di lavoro omogeneamente eterogenei, utilizzando variabili come la scuola di provenienza, il genere, il voto di maturità, eventuali esperienze lavorative e altri fattori. Il secondo obiettivo era legato alla motivazione all'apprendimento degli studenti, al fine di contrastare l'alto tasso di abbandono accademico. Il corso di introduzione alla programmazione è un corso semestrale incentrato sulla programmazione imperativa in C++. Tuttavia, per gli studenti provenienti da istituti scolastici in cui l'informatica è materia curricolare, il corso può offrire inizialmente pochi stimoli e sembrare limitato nella sua visione della programmazione. D'altra parte, per gli studenti provenienti da percorsi scolastici che non prevedono lo studio dell'informatica (come ad esempio un liceo scientifico tradizionale), il corso può risultare più difficile e, di conseguenza, demotivante poiché i colleghi con un background informatico riescono a raggiungere gli stessi risultati con meno tempo e fatica. A tal fine, abbiamo progettato

[1] https://sonic-pi.net

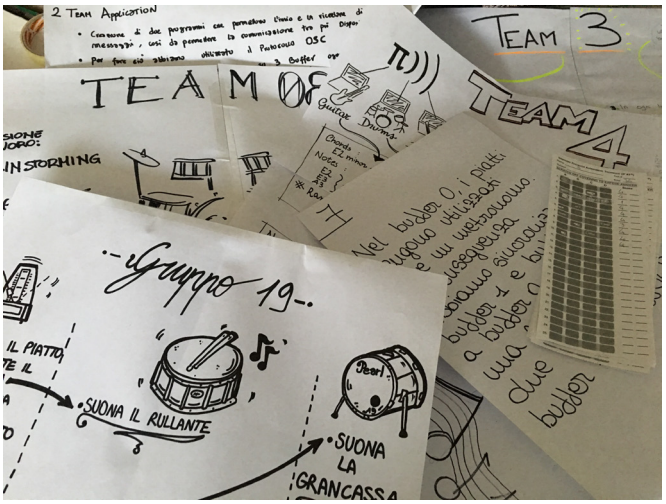


Figura 1: Creatività e lavoro di grupppo a servizio dell'apprendimento della programmazione.

un'attività basata sull'introduzione di concetti avanzati di programmazione come concorrenza, temporizzazione e sincronizzazione in un loro contesto naturale di applicazione come la creazione di un brano musicale multistrumentale. L'analisi dell'esperienza e dei dati raccolti è discussa in dettaglio in [14].

La Fig. 6 in Appendice mostra un esempio degli esercizi di warm-up su Sonic Pi proposti in questa attività.

3 Fase 2: Knowledge Transfer (2021/22)

Basandoci sull'esperienza acquisita nel primo anno dell'esperimento, abbiamo ulteriormente perfezionato e adattato il formato didattico nell'anno accademico 2021/2022 per gli studenti del terzo anno iscritti ai corsi di PCAD e ICDD. Questi corsi hanno fornito una piattaforma ideale per esplorare la concorrenza e la sua applicazione in contesti creativi. L'obiettivo principale di apprendimento era introdurre gli studenti al concetto di multithreading in Python. Tuttavia, abbiamo affrontato questo obiettivo partendo dalla concorrenza in Sonic Pi e poi passando a Python utilizzando la libreria `python-sonic`. Questa interfaccia ha permesso agli studenti di combinare le capacità di multithreading di Python con le funzionalità di Sonic Pi all'interno dell'ambiente di sviluppo Python. La libreria consente specificamente di incorporare il codice musicale di Sonic Pi all'interno dei thread di Python. Ciò permette una transizione graduale da un dominio all'altro, permettendo agli studenti di tradurre la logica del programma concorrente da Sonic Pi a Python utilizzando la libreria `threading`. Per gli studenti iscritti al corso ICDD, l'esperimento ha permesso anche di esaminare l'attività dal punto di vista dell'instructional design, analizzando gli elementi pedagogici dell'esperimento stesso, come gli obiettivi di apprendimento, la scelta del linguaggio di programmazione per la metodologia didattica, ecc. In queste due edizioni delle attività l'enfasi si è quindi spostata su concorrenza e multithreading mantenendo sempre un forte contatto sul dominio specifico degli strumenti adottati (live music coding) e gli argomenti dei corsi.

4 Fase 3: Misconcezioni nella Concorrenza (2022/23)

Per l'anno accademico 2022/23, ci siamo concentrati sull'obiettivo di introdurre gli studenti al concetto di concorrenza attraverso una serie di brevi compiti pratici, ognuno progettato per affrontare specifiche misconcezioni sulla concorrenza, vedi ad esempio [9, 12, 13], e concetti di base della programmazione. In questo modo, viene consentito agli studenti di costruire gradualmente la loro comprensione affrontando ad esempio le diverse dimensioni dello scope delle variabili in presenza dei thread (globali, automatiche, thread-local), chiamate di funzioni e passaggio di parametri in presenza di thread, sincronizzazione e race condition, correttezza dei programmi multithreaded. Alcuni task si sono concentrati sulla comprensione del programma, mentre altri richiedevano la scrittura del codice. I task di comprensione [8] rappresentano un approccio costruttivista all'insegnamento della programmazione, in cui lo studente interagisce con un artefatto che rappresenta il programma, ad esempio un pezzo di codice. Attraverso questa interazione, lo studente viene stimolato a costruire e affinare il proprio modello mentale della macchina nozionale (*notional machine)* sottostante [4, 7, 11]. La Figura 2 mostra un esempio di task, incluso nella Team App di TBL, collegato a riformulazioni in Sonic Pi di misconcezioni sulla concorrenza con attenzione su scope delle dichiarazioni (es. globale, automatico, thread local) e data race. La Figura 3 mostra invece un esempio di task dove il suono viene usato per "sentire" (es. differenza tra suonare un accordo o una sequenza di note) possibili anomalie legate all'esecuzione out of order di un programma.

Un modulo con aspetti di base è stato proposto al termine del corso ADC nel contesto di alcune ore dedicate alle architetture multicore e alle GPU. Un modulo con task avanzati (es. strutture dati concorrenti e sincronizzazione) è stato proposto a PCAD anche per valutare la diversa percezione di studenti che avevano già svolto attività con Sonic Pi maggiormente orientate all'uso creativo della programmazione. L'analisi dell'esperienza e dei dati raccolti è discussa nel rapporto tecnico [6].

5 Discussione

La sperimentazione sul campo nel corso delle diverse edizioni dell'attività ha permesso di sintetizzare un formato, ribattezzato Sonic TBL [6], che combina le caratteristiche migliori del metodo TBL e di strumenti come Sonic Pi. Riteniamo che, a seguito dei diversi raffinamenti descritti in questo articolo, il formato abbia raggiunto nel corrente anno accademico forma e sostanza adeguati al livello universitario come testimoniato da una vasta partecipazione, circa 200 studenti, alle attività proposte durante il corso dell'anno accademico. Dai dati collezionati nell'anno accademico 2022/23, vedi [5], i risultati ottenuti nei vari task enfatizzano la diffusa presenza di misconcenzioni tra gli studenti sia su concetti relativi a nozioni di base di concorrenza, sia su concetti generali di programmazione calati nel contesto della programmazione concorrente (es. scoping a livello di funzione e thread). In generale l'approccio combinato TBL e Sonic Pi consente di offrire agli studenti un'esperienza di apprendimento che integra attività pratiche di programmazione anche con aspetti avanzati rispetto al tipico programma del primo anno, applicazioni su un dominio concreto come la creazione di musica e apprendimento collaborativo. Inoltre il metodo ha permesso la graduale introduzione della programmazione multithreaded al termine del primo anno, un aspetto ormai fondamentale visto il sempre crescente interesse ad esempio per computazione ad alte prestazioni basata su multicore e GPU.

Per quanto riguarda il corso di Architetture dei Calcolatori del primo anno, 23 studenti su 130 hanno compilato il questionario finale. Nella domanda A, "Penso che Sonic Pi sia utile per capire la concorrenza", il valore mediano è risultato 4 su scala di Likert da 1 a 5. Per la

A

```
in_thread do
  use_synth :piano
  x = 40
  10.times do
    x += 4
    sleep 0.5
    play x
  end
end

in_thread do
 use_synth :kalimba
  x = 40
  10.times do
    x -= 4
    sleep 0.5
    play x
  end
end
```

B

```
x = 40
in_thread do
  use_synth :piano
  10.times do
    x += 4
    sleep 0.5
    play x
  end
end

in_thread do
  use_synth :kalimba
  10.times do
    x -= 4
    sleep 0.5
    play x
  end
end
```

C

```
x = 40
in_thread do
  use_synth :piano
  10.times do
    x += 4
    sleep 0.5
    play x
  end
end

x = 60
in_thread do
  use_synth :kalimba
  10.times do
    x -= 4
    sleep 0.5
    play x
  end
end
```

Voting cards: Which answer is true?

- In B and C, different threads operate on a common resource and the result depends on the order in which the different threads execute their instructions;
- In A and C, different threads operate on a common resource and the result depends on the order in which the different threads execute their instructions;
- In all three programs, there are no resources shared between threads;
- In A and C, there are no resources shared between threads. The result depends on the order in which the instructions of the different threads are executed.

Figura 2: Team Apps - Misconcezione concorrenza in Sonic Pi.

domanda B, "Penso che l'attività sia stata efficace per introdurre e approfondure concetti della programmazione concorrente", il valore mediano è risultato 3 su scala di Likert. La Figura 4 mostra la distribuzione delle risposte ottenute. Il 74% dei partecipanti ha trovato l'attività di difficoltà adeguata, il 22% ha ritenuto l'attività difficile, e il rimanenre 4% facile. Il 22% degli studenti ha trovato il lavoro in team molto utile, il 26% si è dimostrato neutrale, il 13% lo ha ritenuto poco utile e il 9% per nulla utile.

Per quanto riguarda il corso di Programmazione Concorrente e Algoritmi Distribuiti del terzo anno, 16 studenti su 54 hanno compilato il questionario finale. Nella domanda A, "Penso che Sonic Pi sia utile per capire la concorrenza", il valore mediano è risultato 4 su scala di Likert da 1 a 5. Per la domanda B, "Penso che l'attività sia stata efficace per introdurre e approfondure concetti della programmazione concorrente", il valore mediano è risultato 4 su scala di Likert. La Figura 5 mostra la distribuzione delle risposte ottenute. L'80% dei partecipanti ha trovato l'attività di difficoltà adeguata, il 18% ha ritenuto l'attività difficile, e il rimanenre 4% facile. Il 25% degli studenti ha trovato il lavoro in team molto utile e il

```
use_synth :piano

note=[52,55,59,40]
i=0

define :foo do |x|
  in_thread do
     play note[x]
  end
end

3.times do
  foo i
  i+=1
end
```

Voting cards: Which answer is true?

(1) The program has no race conditions.
(2) The program creates only one thread.
(3) The program plays the notes in the "`note`" list in sequence.
(4) The program plays the E minor chord.

Gallery walk:
Describe in detail the behavior of the script with particular pay attention to the changes to the value of the variable "`i`" and to the possible sequences of notes play.

Figura 3: Team App - Sentire gli interleaving con Sonic Pi.

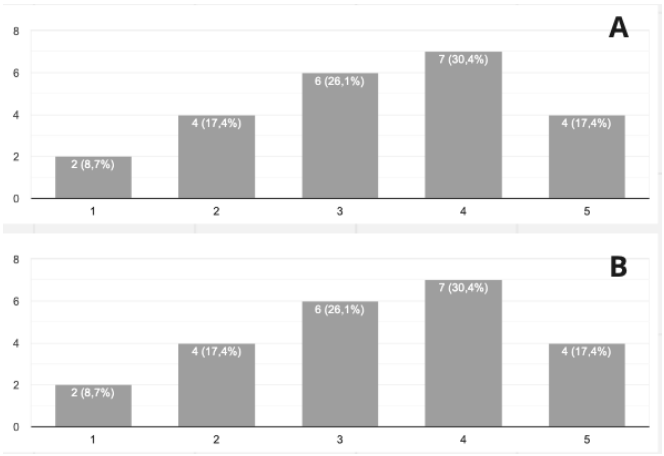


Figura 4: Distribuzione risposte al questionario finale.

75% sufficientemente utile. Da questi dati si può notare come affrontare le misconcezioni più comuni in aspetti di base e avanzati di programmazione sia ritenuto utile da studenti sia del primo anno che del terzo. Inoltre gli studenti del terzo anno, forse anche grazie all'esperienza acquisita affrontando progetti ed esami, tende a dare molto più valore al lavoro in team rispetto ai novizi.

Riferimenti bibliografici

[1] Samuel Aaron, Alan F. Blackwell, and Pamela Burnard. The development of sonic pi and its use in educational partnerships: Co-creating pedagogies for learning computer programming. *Journal of Music Technology and Education*, 9(1):75–94, 3 2016.

[2] Samuel Aaron, Dominic A. Orchard, and Alan F. Blackwell. Temporal semantics for a live coding language. In Alex McLean, Michael Sperber, and Henrik Nilsson, editors, *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design, FARM@ICFP 2014, September 1-3, 2014*, pages 37–47, Gothenburg, Sweden, 2014. ACM.

[3] Roberto Agostini, Leo Izzo, and Giovanni Nulli. Spiegare il "caso" e l'array nel live coding musicale. In *Proc. Didamatica 2022*, page 410–419, Milan, Italy, 2022. AICA.

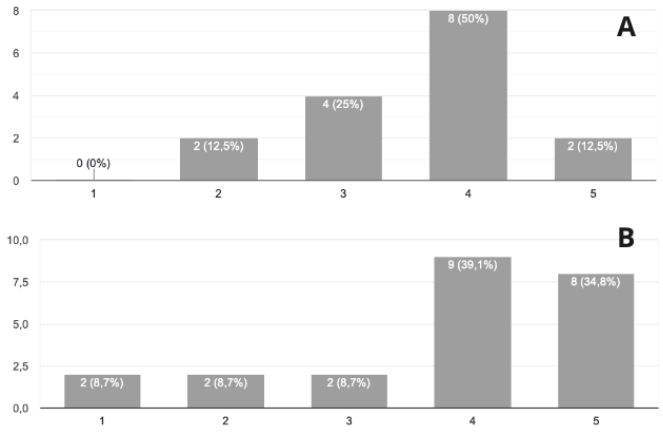


Figura 5: Distribuzione risposte al questionario finale.

[4] Benedict Du Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73, 1986.

[5] Giorgio Delzanno, Giovanna Guerrini, and Daniele Traversaro. Concurrency education with sonic pi: Hear and play mistakes and misconceptions in multithreaded programs, technical report. Technical report, DIBRIS, Università deli Studi di Genova, 2023.

[6] Giorgio Delzanno, Giovanna Guerrini, and Daniele Traversaro. Concurrency Education with Sonic Pi: "Hear" and "Play" Mistakes and Misconceptions in Multithreaded Programs. Under review, available on request. Technical report, DIBRIS, University of Genoa, July 2023.

[7] Benedict du Boulay, Tim O'Shea, and John Monk. The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies*, 14(3):237–249, 1981.

[8] Cruz Izu, Carsten Schulte, Ashish Aggarwal, Quintin Cutts, Rodrigo Duran, Mirela Gutica, Birte Heinemann, Eileen Kraemer, Violetta Lonati, Claudio Mirolo, et al. Fostering program comprehension in novice programmers-learning activities and learning trajectories. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, ITiCSE-WGR '19*, pages 27–52. ACM, Aberdeen, Scotland, UK, 2019.

[9] Jan Lönnberg. Student errors in concurrent programming assignments. In *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006*, Baltic Sea '06, page 145–146, New York, NY, USA, 2006. Association for Computing Machinery.

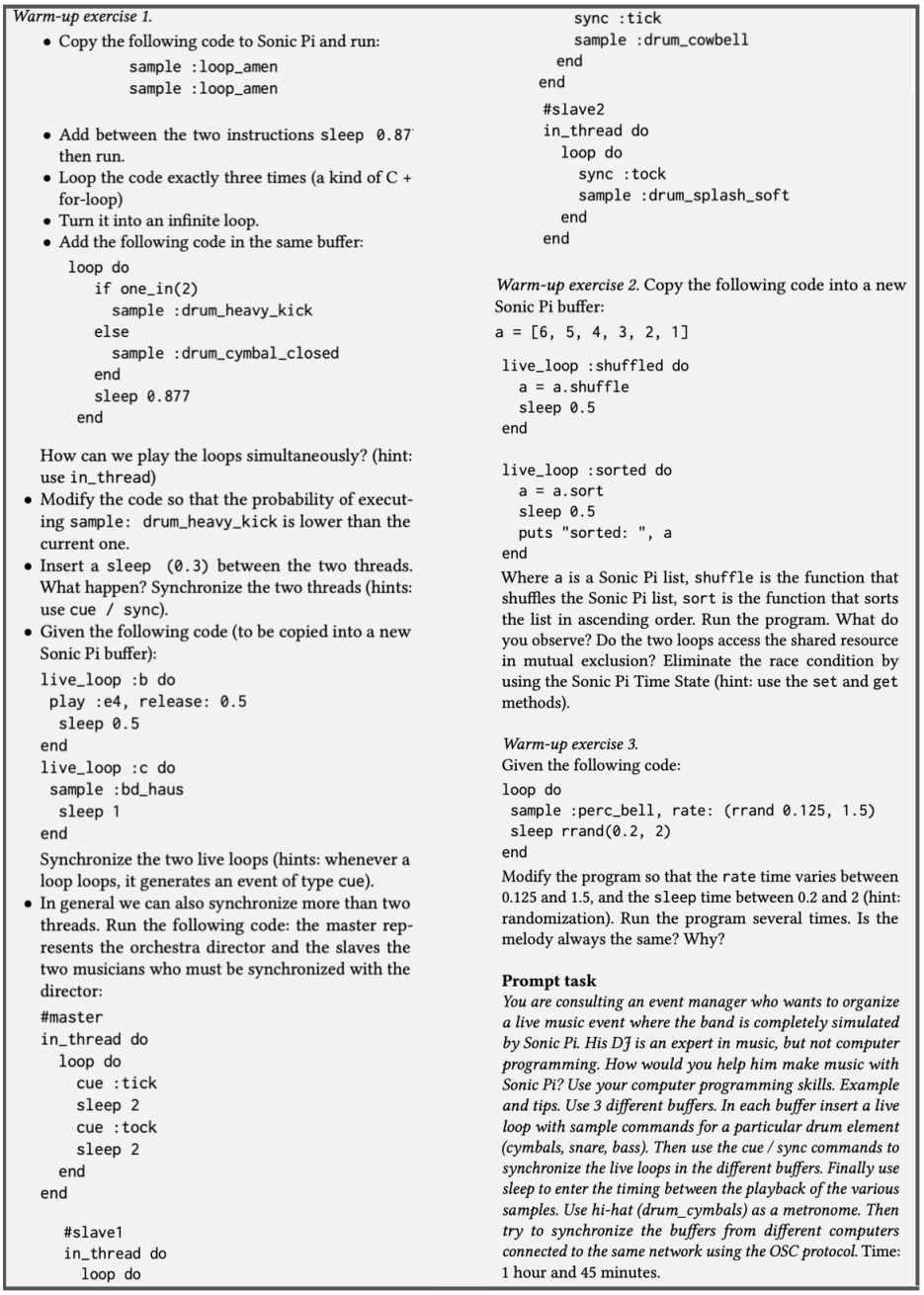
[10] Christopher Petrie. Programming music with sonic pi promotes positive attitudes for beginners. *Computers & Education*, 179:104409, 2022.

[11] Juha Sorva. Notional machines and introductory programming education. *ACM Trans. Comput. Educ.*, 13(2), jul 2013.

[12] Filip Strömbäck. *Teaching and Learning Concurrent Programming in the Shared Memory Model*. PhD thesis, Linköping University, Sweden, 2023.

[13] Filip Strömbäck, Linda Mannila, Mikael Asplund, and Mariam Kamkar. A student's view of concurrency - a study of common mistakes in introductory courses on concurrency. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, ICER '19, page 229–237, New York, NY, USA, 2019. Association for Computing Machinery.

[14] Daniele Traversaro, Giovanna Guerrini, and Giorgio Delzanno. Sonic Pi for TBL Teaching Units in an Introductory Programming Course. In *Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '20 Adjunct, page 143–150, New York,



Warm-up exercise 1.

- Copy the following code to Sonic Pi and run:

```
sample :loop_amen
sample :loop_amen
```

- Add between the two instructions `sleep 0.87` then run.
- Loop the code exactly three times (a kind of C + for-loop)
- Turn it into an infinite loop.
- Add the following code in the same buffer:

```
loop do
  if one_in(2)
    sample :drum_heavy_kick
  else
    sample :drum_cymbal_closed
  end
  sleep 0.877
end
```

How can we play the loops simultaneously? (hint: use `in_thread`)

- Modify the code so that the probability of executing `sample: drum_heavy_kick` is lower than the current one.
- Insert a `sleep (0.3)` between the two threads. What happen? Synchronize the two threads (hints: use `cue / sync`).
- Given the following code (to be copied into a new Sonic Pi buffer):

```
live_loop :b do
 play :e4, release: 0.5
  sleep 0.5
end
live_loop :c do
 sample :bd_haus
  sleep 1
end
```

Synchronize the two live loops (hints: whenever a loop loops, it generates an event of type cue).

- In general we can also synchronize more than two threads. Run the following code: the master represents the orchestra director and the slaves the two musicians who must be synchronized with the director:

```
#master
in_thread do
  loop do
    cue :tick
    sleep 2
    cue :tock
    sleep 2
  end
end

  #slave1
  in_thread do
    loop do
      sync :tick
      sample :drum_cowbell
    end
  end

  #slave2
  in_thread do
    loop do
      sync :tock
      sample :drum_splash_soft
    end
  end
```

Warm-up exercise 2. Copy the following code into a new Sonic Pi buffer:

```
a = [6, 5, 4, 3, 2, 1]

live_loop :shuffled do
  a = a.shuffle
  sleep 0.5
end

live_loop :sorted do
  a = a.sort
  sleep 0.5
  puts "sorted: ", a
end
```

Where `a` is a Sonic Pi list, `shuffle` is the function that shuffles the Sonic Pi list, `sort` is the function that sorts the list in ascending order. Run the program. What do you observe? Do the two loops access the shared resource in mutual exclusion? Eliminate the race condition by using the Sonic Pi Time State (hint: use the `set` and `get` methods).

Warm-up exercise 3.
Given the following code:

```
loop do
 sample :perc_bell, rate: (rrand 0.125, 1.5)
 sleep rrand(0.2, 2)
end
```

Modify the program so that the `rate` time varies between 0.125 and 1.5, and the `sleep` time between 0.2 and 2 (hint: randomization). Run the program several times. Is the melody always the same? Why?

Prompt task
You are consulting an event manager who wants to organize a live music event where the band is completely simulated by Sonic Pi. His DJ is an expert in music, but not computer programming. How would you help him make music with Sonic Pi? Use your computer programming skills. Example and tips. Use 3 different buffers. In each buffer insert a live loop with sample commands for a particular drum element (cymbals, snare, bass). Then use the cue / sync commands to synchronize the live loops in the different buffers. Finally use sleep to enter the timing between the playback of the various samples. Use hi-hat (drum_cymbals) as a metronome. Then try to synchronize the buffers from different computers connected to the same network using the OSC protocol. Time: 1 hour and 45 minutes.

Figura 6: Esercizi di riscaldamento su Sonic Pi.

NY, USA, 2020. Association for Computing Machinery.

Dalla tartaruga alla ricorsione: LOGO, linguaggi formali e frattali per introdurre i sistemi ricorsivi

Gaetano Impoco[1]

I.T.T. "Emauele Morselli", Gela (CL)
gaetano@impoco.it

Sommario

In questo contributo racconteremo un breve percorso didattico per l'apprendimento del concetto di ricorsione. Partendo dalla navigazione di un robot, tramite uno strumento di programmazione visuale, si introducono elementi relativi a linguaggi formali per la rappresentazione procedurale di frattali. L'uso dei linguaggi formali è poi arricchito di elementi che, da una semplice stringa di simboli, condurranno a grammatiche *context-free* con produzioni autoreferenziali e sviluppi secondari che ricordano le chiamate di funzione.

Introduzione

La ricorsione è tradizionalmente ritenuta un argomento inerentemente difficile [1]. In molti testi, manuali scolastici ma anche monografie professionali o dedicate all'istruzione terziaria, è trattata come argomento ancillare o "avanzato" [10]. Tuttavia, la ricorsione ricalca un aspetto basilare del *pensiero computazionale* [11]: la decomposizione di un problema in sotto-problemi e, particolarmente, lo schema *divide-et-impera* [4, 3]. La presunta marginalità della ricorsione, forse non esplicitamente professata ma di fatto praticata nella scuola, è smentita anche dall'ampiezza del dibattito di cui è oggetto nell'ambito della formazione terziaria [7, 6].

Sulla presunta difficoltà intrinseca della ricorsione Turbak e colleghi propongono un punto di vista originale [10]. La difficoltà nel comprendere la ricorsione, ipotizzano, dipende dal fatto che essa sia introdotta dopo il concetto di ciclo. E' evidente che la ricorsione segue lo schema *divide-conquer-glue* (DCG) nel quale il problema viene suddiviso in sotto-problemi che sono istanze più semplici del problema originario (*divide*) e che vengono risolti separatamente (*conquer*) per poi essere combinati al fine di ottenere una soluzione per il problema di partenza (*glue*). Ciò che è meno palese è il fatto che anche il ciclo, come la ricorsione, possa può essere considerato un problema autoreferenziale che è un'istanza dello schema *divide-conquer-glue* (DCG) ma dove la fase di fusione (*glue*) delle soluzioni dei sotto-problemi non è necessaria. Secondo gli autori gli studenti, che percepirebbero la struttura concettuale sottostante il ciclo, sono portati a pensare che la fase di fusione sia una peculiarità della ricorsione e non una caratteristica dello schema concettuale DCG. Per non cadere in quella che chiamano la "trappola dell'iterazione", suggeriscono che la ricorsione vada presentata prima dell'iterazione.

Sembrano portare alla stessa conclusione i dati raccolti da Lee e Lehrer [5] che dimostrano come gli studenti con precedente esperienza di programmazione (in BASIC) commettono maggiori errori nella programmazione di sistemi ricorsivi rispetto a chi non ha nessuna esperienza. In questo lavoro, tuttavia, appare chiaro come i falsi concetti derivino dall'utilizzo dell'istruzione GOTO.

Senza addentraci ulteriormente in questo dibattito, intendiamo rimarcare che la ricorsione è uno schema fondamentale del pensiero computazionale e, quindi, non solo uno strumento per programmatori ma un modello di ragionamento dal valore universale [11]. Diventa dunque

 ISBN: 978-88-6629-075-9

cogente trovare pratiche didattiche che facilitino l'avvicinamento degli studenti a questo schema concettuale. Crediamo che questa riflessione abbia bisogno della partecipazione attiva della comunità scolastica, oltre che di quella scientifica. Questo contributo nasce proprio da questa esigenza.

Nel seguito, quindi, racconteremo una semplice esperienza svolta in classe, senza la pretesa di assegnarle un valore di universalità che le è estraneo. Prima però vale la pena di citare un ultimo contributo che viene dal mondo della scuola italiana, presentato da Cozza e colleghi [2]: un percorso didattico in cui si cerca di ancorare il concetto di ricorsione ad elementi "estrinseci" quali foto di frattali in natura, l'autoreferenzialità nell'arte, filastrocche, indovinelli, *web quest*. A margine dello studio dello schema ricorsivo, poi, gli autori esplicitano in classe i collegamenti con altre discipline. Percorso certamente interessante ma, a nostro giudizio, con due limiti: amplia lo sguardo dal punto di vista culturale ma approfondisce poco quello tecnico; richiede sforzo organizzativo per la preparazione delle attività multidisciplinari. Nell'ambito dell'istruzione secondaria, poi, crediamo che la natura astratta e profondamente speculativa della ricorsione possa essere colta più immediatamente con un ragionamento di carattere simbolico, più che operativo, soprattutto dal secondo biennio. La nostra esperienza va in questa direzione.

Percorso didattico

Il percorso didattico che raccontiamo in queste pagine è stato realizzato nelle classi terze di un istituto tecnico tecnologico ad indirizzo informatico. Diversi studenti conoscevano già linguaggi di programmazione visuali. Fermo restando questo prerequisito, la nostra proposta può essere facilmente adattata ad altri corsi di studio e offerta anche a studenti del primo biennio o del primo grado. Nelle nostre classi, questo percorso è stato proposto prima dell'introduzione della ricorsione nella programmazione in C, segnatamente per l'implementazione dell'algoritmo *merge sort*.

Il nostro percorso non si discosta dalla tradizionale introduzione attraverso il linguaggio Logo [9]. Tuttavia, anziché lavorare sulla programmazione visuale, le attività sono prevalentemente di tipo *unplugged*. Siamo ben coscienti che questo modo di operare possa apparire in controtendenza rispetto alle metodologie che oggi vanno per la maggiore, fondate sull'esperienza manipolativa. Crediamo tuttavia, incoraggiati dall'esperienza in aula, che le attività proposte siano intellettualmente stimolanti e creino quel giusto stimolo alla sfida e alla curiosità che favorisce l'apprendimento. Essendo proposto come una serie di brevi attività a carattere ludico, il percorso è inclusivo nel senso che riesce a raggiungere studenti con stili di apprendimento diversi[1].

Il percorso didattico è così strutturato.

Linguaggi formali Giocando con la tartaruga (in termini più attuali, un robot rover) gli studenti seguono i percorsi proposti e ne inventano di nuovi in una sfida collaborativa con i compagni. Scoprono che le stringhe prodotte sono parole di un linguaggio il cui alfabeto $\Sigma = \{a, s, d\}$ può essere interpretato come movimenti e rotazioni ma può avere anche altre interpretazioni. Si comincia ad intravedere la distinzione tra sintassi e semantica di un linguaggio.

Regole di semplificazione Dopo le prime esperienze con la descrizione di tracciati, proponendo figure sempre più complesse e con marcate regolarità, il docente fa nascere il bisogno

[1] Nell'A.S. 2022/23 il percorso è stato proposto con successo anche ad uno studente con BES facendo leva sulle sue spiccate abilità nel disegno geometrico.

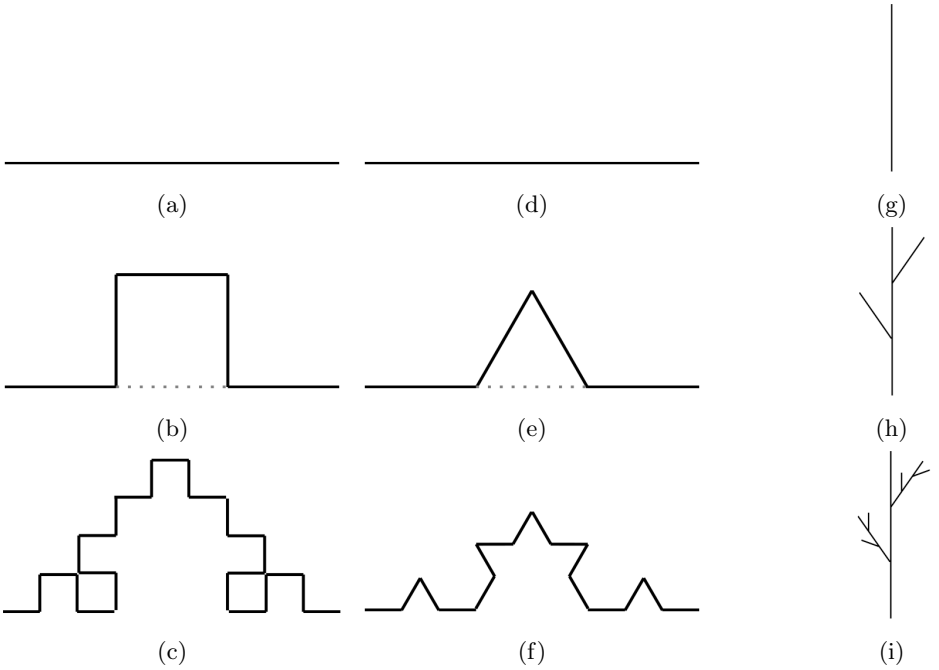


Figura 1: Frattali usati nel percorso didattico. Da 1a a 1c curva di Koch quadratica di tipo 1 (generatore $A \rightarrow AsAdAdAsA$). Da 1d a 1f curva di Koch (generatore $A \rightarrow AsAddAsA$). Da 1g a 1i esempio di frattale per la rappresentazione di una (parte di) pianta (generatore $R \rightarrow a[sR]a[dR]a$).
Le figure 1a, 1d e 1g mostrano gli iniziatori mentre i generatori sono mostrati nelle figure 1b, 1e e 1h. Le figure 1c, 1f e 1i mostrano il risultato ottenuto dopo l'applicazione della sequenza di regole **1,1,2** (due iterazioni).

di catturare tali regolarità all'interno delle strutture sintattiche del linguaggio; stimola quindi il dibattito sull'utilità di inventare una simbologia per descrivere in maniera più compatta i percorsi ripetitivi. Nascono così strutture sintattiche per la ripetizione come $(ad)^4$ che corrisponde allo sviluppo *adadadad*.

Grammatiche *context-free* A questo punto il docente introduce altri livelli di regolarità, per esempio chiedendo agli studenti di rappresentare tre quadrati uno accanto all'altro. Con le nuove strutture sintattiche, possiamo scrivere ad esempio $[(ad)^4a]^3$. Si suggerisce, quindi, di assegnare "nomi" ad alcune figure tipiche, per esempio utilizzando la notazione $Q \rightarrow (ad)^4$ per il quadrato di lato 1. La nostra sequenza di tre quadrati si può dunque scrivere come $[Qa]^3$. Il docente sottolinea come i nomi siano simboli che non fanno parte dell'alfabeto e che, quindi, non sono immediatamente comprensibili dalla tartaruga. Si introduce in questo modo la distinzione tra simboli *terminali* e *non-terminali*.

Autoreferenzialità Dopo aver proposto il tracciato $A \rightarrow asadadasa$ (Figura 1b), con un piccolo gioco di prestigio il docente chiede cosa accade se sostituiamo tutte le 'a' con 'A' in questo modo $A \rightarrow AsAdAdAsA$ (**regola 1**). La risposta immediata è che la stringa non può essere eseguita dal robot. Sostituire i simboli non terminali con la stessa regola porta però ad una regressione all'infinito. Il docente suggerisce quindi di aggiungere la regola

$A \to a$ (**regola 2**) che può essere usata in alternativa alla prima, invitando gli studenti a generare figure diverse scegliendo arbitrariamente, ad ogni iterazione, la **regola 1** o la **2**. Gli studenti notano immediatamente che la **regola 2** "chiude" la regressione e permette di ottenere una stringa con soli simboli terminali, mentre la **regola 1** lascia la stringa "aperta". Ogni percorso potrà dunque essere descritto dalla sequenza di regole da applicare. Per esempio, la stringa da cui siamo partiti, $A \to asadadasa$, si può scrivere come **1,2**.

Frattali L'ultimo passaggio di questa esperienza prevede che gli studenti creino figure frattali utilizzando le grammatiche *context-free*. Si comincia riproponendo la grammatica

1. $A \to AsAdAdAsA$ (**generatore**)
2. $A \to a$ (**iniziatore**)

per chiedere agli studenti di calcolare la lunghezza della curva "zoomando" man mano che si aggiungono nuove iterazioni (**regola 1**) alla figura. Partendo da un linea di lunghezza 1 (l'**iniziatore** in Figura 1a - **regola 2**), si chiede di applicare una sola volta la **regola 1** (**generatore**) e di calcolare la nuova lunghezza. Gli studenti, opportunamente stimolati, noteranno che la linea può essere divisa in tre parti uguali (Figura 1b) e, di conseguenza, la lunghezza della nuova 'a' è $1/3$. Poiché nella **regola 1** ci sono cinque 'a' la lunghezza è dunque pari a $5/3$. Proseguendo nella regressione il docente condurrà gli studenti a derivare la corrispondenza tra il numero n di passi della regressione (quante volte applichiamo la **regola 1**) e l'esponente che appare nella formula $(5/3)^n$ che descrive la lunghezza della curva risultante. Lo stesso calcolo si chiede di fare per la curva di Koch

1. $A \to AsAddAsA$
2. $A \to a$

sottolineando che qui la rotazione non è di 90° ma di 60°.

Piante Chiusa l'introduzione alla ricorsione ci troviamo spesso di fronte a richieste di approfondimento sul tema affascinante dei frattali. Proponiamo quindi una semplice attività in cui introduciamo il concetto di chiamata a funzione usando un semplice espediente che riflette l'idea delle operazioni PUSH-POP su uno stack. Usando la grammatica

1. $R \to a[sR]a[dR]a$
2. $R \to a$

aggiungiamo al nostro "menu" sintattico le parentesi quadre che ci consentono di sollevare la penna. Ovvero, alla chiusura della parentesi la tartaruga ritorna nelle condizioni (posizione o orientamento) in cui era prima dell'apertura della perentesi. Si tratta chiaramente di *L-Systems* presentati come un semplice "ampliamento" delle grammatiche *context-free*. Se poi aggiungiamo un elemento aleatorio nell'applicazione delle parentesi quadre, ad ogni esecuzione otteniamo piante simili ma non uguali. Creatività e strumenti per il disegno procedurale [8] consentono di familiarizzare con ricorsione e gestione delle chiamate sullo stack.

Conclusioni

Abbiamo presentato un breve percorso per introdurre la ricorsione con attività curiose e stimolanti che usano ingredienti quali linguaggi formali, grammatiche *context-free*, autoreferenzialità, frattali e il disegno delle piante. Vissute dagli allievi come attività creative, aprono lo sguardo su temi che trattati in maniera convenzionale risultano complessi e distanti dai loro interessi. La nostra esperienza degli ultimi anni suggerisce che la proposta è efficace. Nel futuro lavoreremo alla costruzione di un legame più evidente tra il percorso e la sintassi della ricorsione nei linguaggi di alto livello.

Riferimenti bibliografici

[1] John R. Anderson, Peter Pirolli, and Robert Farrell. Learning to program recursive functions. *The nature of expertise*, pages 153–184, 1988.

[2] Pasquale Cozza, Viviana Andreotti, and Vittoria Cozza. Percorso didattico sulla ricorsione dalla natura al coding. In *Atti del convegno DIDAMATICA 2016*, 19–21 Aprile 2016.

[3] Dennis Komm. Teaching recursion in high school – a constructive approach. In Erik Barendsen and Christos Chytas, editors, *Proceedings of the 14th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (ISSEP 2021)*, volume 13057 of *Lecture Notes in Computer Science*, pages 69–80. Springer, 2021.

[4] Dennis Komm, Ulrich Hauser, Bernhard Matter, Jacqueline Staub, and Nicole Trachsler. Computational thinking in small packages. In *Proceedings of the 13th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives (ISSEP 2020)*, pages 170–181. Springer, 16–18 Novembre 2020.

[5] Okhwa Lee and Richard Lehrer. Conjectures concerning the origins of misconceptions in Logo. *Journal of Educational Computing Research*, 4(1):87–105, 1988.

[6] Renée McCauley, Scott Grissom, Sue Fitzgerald, and Laurie Murphy. Teaching and learning recursive programming: a review of the research literature. *Computer Science Education*, 25(1):37–66, 2015.

[7] Christian Rinderknecht. A survey on teaching and learning recursive programming. *Informatics in Education*, 13(1):87–119, 2014.

[8] Kevin Roast. L-Systems turtle graphics renderer. https://www.kevs3d.co.uk/dev/lsystems/.

[9] Cynthia Solomon, Brian Harvey, Ken Kahn, Henry Lieberman, Mark L. Miller, Margaret Minsky, Artemis Papert, and Brian Silverman. History of logo. *Proceedings of ACM on Programming Languages*, 4(HOPL):79:1–79:66, 2020.

[10] Franklyn A. Turbak, Constance S. Royden, Jennifer L. Stephan, and Jean Herbst. Teaching recursion before loops in CS1. *Journal of Computing in Small Colleges*, 14(4):86–101, 1999.

[11] Jeannette M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, Marzo 2006.

Appendice: Piano didattico

Per favorire un comodo utilizzo in classe da parte dei docenti, il piano didattico presentato in questo contributo è organizzato in sette lezioni da un'ora ciascuna. La Tabella 1 riporta in maggiore dettaglio il piano delle lezioni. È possibile reperire il materiale didattico di riferimento all'indirizzo `https://impoco.it/didattica/itadinfo2003`.

argomento	tempo (min)	descrizione
linguaggi formali #1	10	tartaruga: intro
	20	esercizio #1 - scrivere la stringa che rappresenta alcune figure semplici
	30	esercizio #2 - sfida collaborativa a coppie: ciascuno inventa una figura e ne scrive la sequenza, poi la passa al compagno; il compagno prova a ricostruire la figura dalla stringa
linguaggi formali #2	10	formalizzazione: simboli, alfabeto, regole
	10	simboli e interpretazione: esempi con espressioni algebriche
	15	angoli diversi da 90° - esempio (triangolo equilatero): discussione in coppia, discussione con la classe
	5	angoli diversi da 90° - recap del docente
	20	esercizio #3 - attività (in coppia): produrre una figura di fantasia con angoli da 60° e scrivere la parola che la rappresenta
regole di semplificazione	10	stimolo iniziale: stringa per il quadrato → regolarità
	15	esercizio #4 - attività (in coppia): trovare le regolarità nel quadrato e inventare meccanismi (sintattici) per scrivere il quadrato in maniera più compatta
	15	restituzione e discussione sull'attività
	20	esercizio #1bis - esercizi si "semplificazione"
grammatiche *context-free*	10	percorsi complessi e regolarità: assegnare "nomi" a sottostringhe
	30	esercizio #5 - attività (in coppia): disegnare un percorso complesso fatto di elementi semplici, a cui sono assegnati "nomi" (simboli non terminali), e trascriverne la stringa usando i simboli non terminali
	10	simboli terminali e non terminali
	10	grammatiche *context-free*
autoreferenzialità	10	autoreferenzialità
	10	esercizio #6 - attività (in coppia): osservare la produzione $A \rightarrow AsAdAdAsA$ e spiegare di che si tratta, nel contesto dei linguaggi formali come trattati sino a questo momento

argomento	tempo (min)	descrizione
	10	discussione collettiva e introduzione della produzione $A \rightarrow a$ nella grammatica
	10	esercizio #7 - cosa cambia? come fa questa nuova regola a risolvere il problema della regressione all'infinito?
	5	mettere tutto insieme: autoreferenzialità e ricorsione
	10	esercizio #8 - disegnare il frattale prodotto dalla grammatica $A \rightarrow AsAddAsA$ / $A \rightarrow a$
	5	discussione conclusiva
frattali	10	richiamo alla lezione precedente
	15	esercizio #9 - web quest: i frattali in natura
	10	proprietà dei frattali
	15	esercizio #10 -attività (in coppia): calcolare la lunghezza dei due frattali di esempio
	10	lunghezza frattale
piante (l-systems)	10	introduzione del "meccanismo" **PUSH-POP**
	15	esercizio #11 - attività (in coppia): disegnare un sistema dato contenente due meccanismi PUSH-POP
	10	svelare l'arcano: condivisione dei risultati e spiegazione del docente
	15	esercizio #12 -attività (in coppia): disegnare un sistema dato, espandendo ricorsivamente
	10	casualità e probabilità di "ramificazione" - note finali

Tabella 1: Piano delle lezioni.

Una Macchina Nozionale per Architetture dei Calcolatori come possibile collegamento tra gli insegnamenti del primo anno della laurea in Informatica

Giorgio Delzanno, Daniele D'Agostino, Giovanna Guerrini e Daniele Traversaro

DIBRIS, Università degli Studi di Genova, Italy
{name.surname}@unige.it

Sommario

In questo articolo presentiamo un esperimento basato sull'adozione del modello RAM di Cook e Reckhow come macchina nozionale per introdurre il linguaggio macchina RISC-V e di un linguaggio visuale a blocchi per avvicinare i novizi ai concetti di base dell'assembler e delle architetture di calcolo. L'esperimento si è tenuto nella prima parte del corso di Architetture dei Calcolatori nell'anno accademico 2022/23 con 320 studenti con conoscenze molto eterogenee. La scelta di combinare RISC-V, modello RAM e linguaggio a blocchi è stata dettata dal tentativo di adottare una macchina nozionale in grado di rappresentare i concetti di base di architetture load-store anche per novizi e dalla quale poter ricostruire i costrutti che nel corso di Introduzione alla Programmazione venivano nel frattempo presentati attraverso il C++.

1 Contesto e Motivazioni

Il primo semestre del primo anno del Corso di Laurea in Informatica dell'Università di Genova è organizzato intorno a tre insegnamenti fondamentali: Introduzione alla Programmazione (IP), Architetture dei Calcolatori (AC), Algebra e Logica per Informatica (ALI). I tre insegnamenti condividono un nucleo comune legato ai principi della programmazione (basso ed alto livello) e della struttura degli elaboratori. Il primo anno è da alcuni anni oggetto di studio ed applicazione di metodi didattici innovativi nel tentativo di ridurre gli abbandoni e comunque stimolare beginner senza farli scoraggiare alle prime difficoltà della materia. Tra i vari tentativi, sono state definite alcune attività ponte (per collegare ad esempio IP ad ALI). In questo ambito principi e strumenti del Pensiero Computazionale (PC) si sono rivelati molto utili in moduli integrativi per studenti alle prime armi con la programmazione. Questo articolo descrive brevemente un esperimento che cade nell'intersezione $AC \cap IP \cap ALI \cap PC$, partendo dal punto di vista di AC, vedi Fig. 1.

A partire dall'edizione 2022/23 il programma di AC segue l'approccio proposto da Patterson e Hennessy in [9] adottando RISC-V [15] come architettura di riferimento, un open standard ISA (Instruction Set Architecture), basato sul principio RISC (Reduced Instruction Set Computer) [11, 12]. In [8] Hennessy e Patterson hanno indicato RISC-V come una delle maggiori opportunità della nuova golden age delle architettture insieme a multicore e acceleratori domain specific come GPU e TPU. Questo è testimoniato anche dal fatto che MIPS Technologies, l'azienda celebre per aver ideato e sviluppato per quasi quattro decenni la famiglia di architetture MIPS, di cui detiene la proprietà intellettuale, ha annunciato l'intenzione di utilizzare RISC-V anziché MIPS, come ISA di alcune nuove CPU [10].

Dal punto di vista tecnico, RISC-V è una architettura load–store con istruzioni per accedere alla memoria (load e store tra memoria e registri) e istruzioni aritmetico-logiche che operano solo su registri. Lo schema logico utilizzato da RISC-V poggia su un array composto da registri

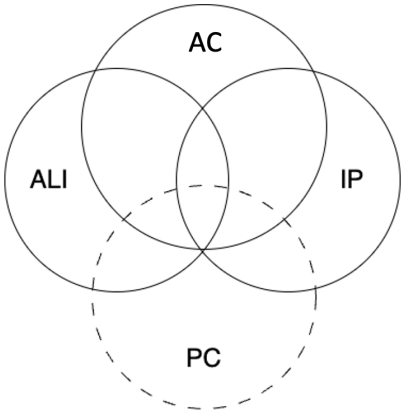


Figura 1: Relazione tra i corsi del primo anno.

per numeri interi, float e registri di controllo e stato per le modalità privilegiate che contengono informazioni sullo stato operativo del processore, i dati che vengono utilizzati in ogni istante e i meccanismi legati alla loro gestione. L'array è abbastanza ampio da ridurre al minimo la necessità di accedere alla memoria esterna per molte attività di base svolte dal processore: in questo modo si può ridurre il consumo energetico aumentando contemporaneamente le prestazioni. I progetti per i chip RISC-V sono stati per il momento ideati nelle versioni a 32, 64 e 128 bit. Il set di istruzioni è molto semplice ma allo stesso tempo estremamente modulare.

2 Assembler e Macchina Nozionale

Negli insegnamenti dedicati alle architetture dei calcolatori uno degli argomenti ritenuto ostico, in particolare da studenti con poco background informatico, è proprio quello della programmazione in linguaggio macchina. La cosa diventa ancora più evidente per modelli RISC. Infatti ridurre al minimo l'instruction set porta vantaggi in termine di efficienza e consumi ma rende molto complesso codificare anche algoritmi banali (pensate alla complessità dei programmi per le Macchine di Turing, una macchina ideale con una sola istruzione). Affrontare i concetti computazionali dovendo gestire in maniera esplicita indirizzamento e trasferimento di dati tra memoria e registri ed avendo a disposizione solo istruzioni di base (quali semplici istruzioni sia di salto che per le operazioni aritmetico-logiche che necessitano di operand già caricati nei registri) può risultare molto indigesto al primo approccio al mondo dell'informatica. Come nel caso dei linguaggi di programmazione [13, 1, 5], un modo per superare queste difficoltà iniziali può essere quello di accompagnare il discente nella costruzione e nell'affinamento del modello mentale dell'elaboratore (macchina nozionale) attraverso esperienze di apprendimento mirate a far emergere eventuali misconcezioni, cioè specifiche convinzioni scorrette responsabili di errori che mostrano l'inadeguatezza del modello mentale [13]. Una macchina nozionale, vedi Fig 2, in inglese *notional machine* (NM), è un computer idealizzato "le cui proprietà sono implicite nei costrutti del linguaggio di programmazione utilizzato" [4], ma che può anche essere reso esplicito nell'insegnamento [6]. In particolare, la NM al nostro livello di astrazione può essere vista come un'insieme di regole, non necessariamente formalizzate, per descrivere una versione astratta della vera architettura che possa aiutare uno studente a ad apprendere un modello

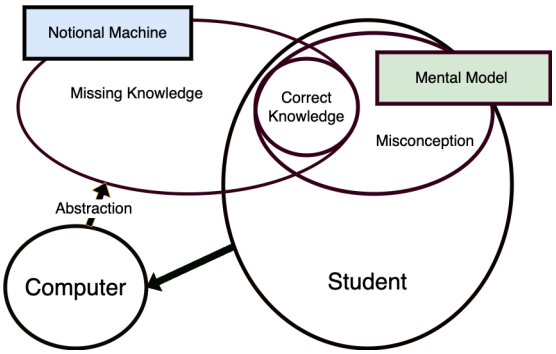


Figura 2: Macchina Nozionale e modello mentale dei discenti.

mentale più vicino possibile al suo funzionamento, minizzando ad esempio misconcezioni ed errori comuni.

3 Modello RAM e Macchina Nozionale a Blocchi

In questo breve articolo descriveremo un esperimento tenuto nella prima parte del corso AC 2022/23. Nelle prime settimane la classe era composta da più di 320 studenti con conoscenze molto eterogenee. Il rischio di scoraggiare (in maniera più o meno consapevole) molti degli indecisi presenti all'inizio del semestre è sembrato subito molto alto in particolare pensando di iniziare a parlare di assembly o di progettazione di architetture RISC. La scelta, forse inusuale, è nata sia per necessità sia dai seguenti requisiti. (1) Si è voluto adottare una NM in grado di rappresentare i concetti di base di architetture load-store e relativi reduced instruction set, partendo dalla quale poter ricostruire i costrutti che nel corso di Introduzione alla Programmazione venivano nel frattempo presentati attraverso il C++. In alternativa a modelli basati su regole o diagrammi, come quelli proposti per Python in [3], si è pensato ad un approccio, ispirato a Pensiero Computazionale e coding, in cui (2) la NM fosse esplicita ed eseguibile tramite un simulatore, (3) il simulatore stesso fosse facilmente manipolabile anche dagli studenti allo scopo di permettere di adattarlo al proprio modello mentale, confrontarlo con quello proposto dai docenti, e fare sperimentazioni su di esso.

La scelta è caduta in maniera molto naturale sulla combinazione di due mondi solo apparentemente lontani tra loro:

(M1) un'idealizzazione di un computer RISC basato sul modello delle Random Access Machine introdotte da Cook e Reckhow in [2];

(M2) la descrizione di tale macchina non tramite linguaggi formali (ad esempio transition system o sistemi induttivi) bensi tramite un linguaggio di programmazione a blocchi come Scratch[1] con semantica intuitiva e descrizione delle istruzioni anche in italiano (o altre lingue).

[1] https://scratch.mit.edu/

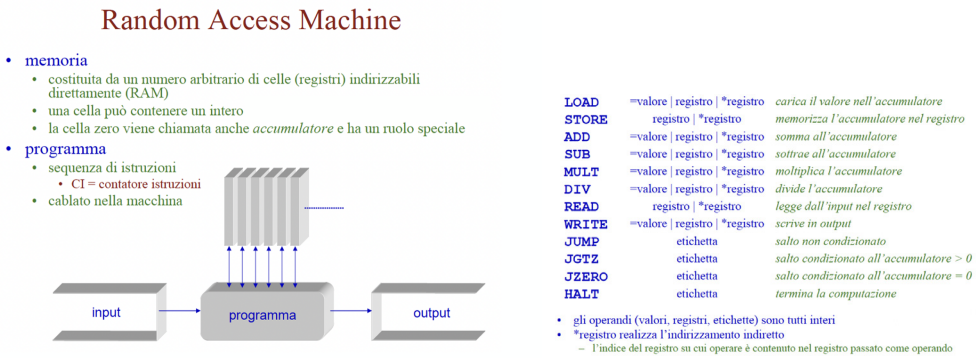


Figura 3: Modello e istruzioni RAM

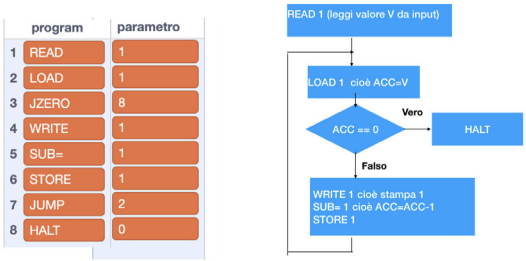


Figura 4: Esempio di programma

4 Lezioni e Compiti a Casa

Dopo una prima settimana di lezione con concetti generali sulle componenti di un elaboratore, abbiamo quindi introdotto in una lezione frontale modello e linguaggio della RAM, vedi Fig. 3, con spiegazione informale ed esempi di programmi utili in generale per capire ciclo di fetch, gestione di registri, e operazioni.

Una seconda lezione è stata quindi dedicata alla definizione in classe della macchina nozionale attraverso il linguaggio a blocchi di Scratch. In questa fase abbiamo decomposto la macchina nozionale nelle seguenti componenti: *programma*, *stato*, *ciclo di fetch*, *decodifica*, ed *esecuzione*. Un programma è stato rappresentato con due liste Scratch con istruzioni e parametri come in Fig. 4 dove abbiamo mostriamo un diagramma di flusso usato come ulteriore aiuto per comprende il collegamento con costrutti ad più alto livello. Per rappresentare i registri di stato (es. program counter e accumulatore) e i registri di memoria abbiamo usato rispettivamente variabili e liste Scratch. L'editor ha l'opzione per visualizzare lo stato di tali oggetti che naturalmente si adatta a rappresentare graficamente lo stato corrente di un programma, come in Fig. 5 in alto a sinistra, e il programma tramite liste di stringhe e parametri, vedi Fig. 5 in basso. La nozione di esecuzione è stata descritta tramite regole per definire il significato del ciclo di fetch, vedi Fig. 6 a sinistra, e regole di decodifica per le singole istruzioni, Fig. 6 a destra. Per queste regole abbiamo sfruttato i costrutti di Scratch, molto vicini a regole in linguaggio naturale ma tuttavia eseguibili come programmi imperativi. Per rappresentare l'esecuzione ci siamo basati quindi sull'interprete sottostante con la possibilità di eseguire passo per passo un programma, funzionalità programmata a sua volta con messaggi ed eventi in Scratch.

Figura 5: Stato

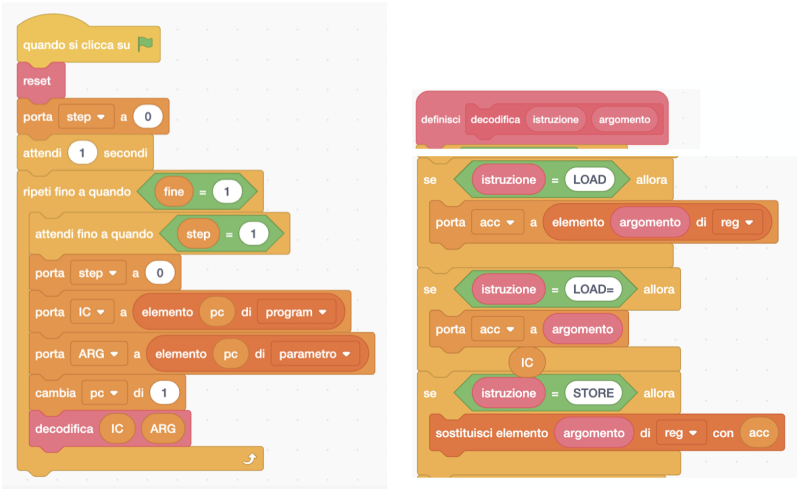


Figura 6: Rappresentazione del ciclo di fetch e regole di decodifica

Il programma risultante è stato costruito insieme agli studenti in classe illustrando principalmente programma, stato, ciclo di fetch e alcuni esempi di decodifica. In questo modo si è la possibilità ai singoli studenti di elaborare il proprio modello mentale dell'instruction set completo ed eventualmente completarlo creando un cosidetto remix del progetto Scratch. Una bozza di interprete in Scratch è stata comunque condivisa sul repository pubblico di Scratch[2] proponendo agli studenti di risolvere come esercizi asincroni, di cui due con consegna, una serie di esercizi di programmazione di difficoltà crescente alcuni con la richiesta di aggiungere nuove istruzioni (ad esempio indirizzamento indiretto) e quindi le corrispondenti regole alla notional machine per ragionare anche sul proprio modello mentale, vedi 7.

[2]https://bit.ly/ramsim

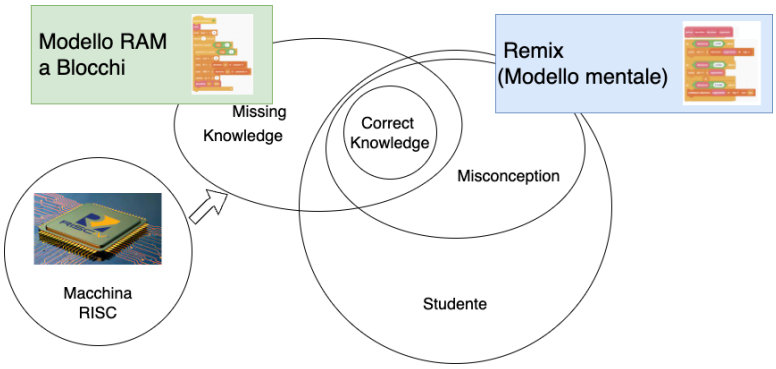


Figura 7: Macchina nozionale e modello RAM

5 Implicazioni e Discussione

Il vantaggio di questo approccio, dettato dall'esigenza di formare un terreno comune sul quale poi costruire la parte rimanente del corso, è stato molteplice. Gli studenti inesperti hanno potuto cimentarsi subito (quasi inconsciamente) con un interprete e comunque con una artefatto da usare per modellare il proprio modello mentale di calcolatore RISC. Questo a permesso di explorare misconception legate ad esempio al funzionamento di alcuni registri (accumulatore e program counter) e al funzionamento delle operazioni (ad es. eventuale reset di celle sorgenti di una load, funzionamento indirizzamento indiretto anche con celle di memoria, ecc). Il legame tra RAM e RISC-V è stato poi ulteriormente evidenziato nel corso del semestre proponendo gli stessi esercizi proposti con la macchina RAM anche in RISC-V. In questo caso è stato evidenziato come la logica risolutiva sia simile ma l'ìmplementazione per certi versi semplificata dalla possibilità di utilizzare un instruction set più articolato, per altri complicata dalla anche se ovviamente con complicazioni legate alla necessità di gestire formato, dimensione istruzioni, indirizzamento registri e memoria secondo lo standard RISC-V.

L'esperimento è risultato utile per studenti alle prime armi con programmazione e architetture anche perchè ha permesso di stabilire un ponte tra i diversi insegnamenti del primo anno incluso il corso di Algebra e Logica grazie ad esempio al concetto di macchina astratta. In linea di principio si potrebbe ulteriormente sviluppare questa idea sia provando a dedicare una lezione di ALI per la definizione formale (attraverso insiemi e relazioni) della semantica operativa di una versione core della RAM e usare formule logiche e definizione induttive induzione per specificare sue proprietà. Un approccio di questo tipo potrebbe essere di interesse anche per studenti esperti di programmazione che potrebbero cimentarsi invece sull'applicazione di concetti di algebra e logica in un dominio a loro consueto (programmazione).

Esistono esempi di linguaggi a blocchi customizzabili, ad esempio tramite la libreria Google Blockly[3], e strumenti low code che potrebbero essere utilizzati per evitare di confondere gli studenti con strumenti come Scratch nati con altri obiettivi formativi e per livelli di istruzione più bassi.

Gli studenti già esperti hanno comunque trovato stimolante questo approccio trasformando l'esperimento proposto in un progettino per la creazione di un interprete per la RAM con linguaggio di sviluppo scelto a piacere (es. Python, Rust) trovando stimoli ulteriori rispetto agli esercizi tradizionali proposti nelle prime settimane di IP.

[3] https://developers.google.com/blockly

Visto il ridotto set di istruzioni fornito dalla RAM, per tutti gli studenti, esperti e non, la programmazione di soluzioni di problemi apparentemente semplici si è rivelato un problema concettualmente interessante. Un esempio per tutti è il problema di effettuare operazioni aggregate su K numeri letti da input dove K è a sua volta un input. Questo problema, riproposto poi anche in assempler RISC-V alla fine del semestre, è uno spunto per confrontarsi con problematiche di gestione dinamica della memoria (ad esempio attraverso indirizzamento indiretto).

Nel prossimo anno accademico prevediamo di estendere tali attività per creare un ponte con insegnamenti di anni succeddivi, specificamente Programmazione Concorrente ed Algoritmi Distribuiti ed High Performance Computing. Il focus del primo anno di un corso di studio in Informatica dev'essere necessariamente quello di dotare gli studenti degli strumenti e conoscenze necessarie a scrivere programmi corretti. Questo ovviamente non è sufficiente, ma dev'essere accompagnato dall econoscenze e tecniche necessarie a scrivere programmi corretti e perfomanti.

Il modello sequenziale è stato superato dal 2006, con l'avvento dei primi processori dual core. Il corso di AC rappresenta il luogo naturale in cui insegnare come questa evoluzione condizioni la scrittura di programmi efficienti in quanto in grado di sfruttare efficacemente la cache, le unità di vettorizzazione e la presenza di più core [14]. Tale approccio richiederà sia il passaggio dal modello RAM al modello MP-RAM [7], sia l'introduzione di esercitazioni mirate alla valutazione approfondita delle prestazioni utilizzando opportuni strumenti di profiling [4].

Riferimenti bibliografici

[1] Benedict Du Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73, 1986.

[2] Stephen A. Cook and Robert A. Reckhow. Time-bounded random access machines. In Patrick C. Fischer, H. Paul Zeiger, Jeffrey D. Ullman, and Arnold L. Rosenberg, editors, *Proceedings of the 4th Annual ACM Symposium on Theory of Computing, May 1-3, 1972, Denver, Colorado, USA*, pages 73–80. ACM, 1972.

[3] Paul E. Dickson, Tim D. Richards, and Brett A. Becker. Experiences implementing and utilizing a notional machine in the classroom. In Larry Merkle, Maureen Doyle, Judithe Sheard, Leen-Kiat Soh, and Brian Dorn, editors, *SIGCSE 2022: The 53rd ACM Technical Symposium on Computer Science Education, Providence, RI, USA, March 3-5, 2022, Volume 1*, pages 850–856. ACM, 2022.

[4] Benedict Du Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73, 1986.

[5] Benedict du Boulay, Tim O'Shea, and John Monk. The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies*, 14(3):237–249, 1981.

[6] Benedict Du Boulay, Tim O'Shea, and John Monk. The black box inside the glass box: presenting computing concepts to novices. *International Journal of man-machine studies*, 14(3):237–249, 1981.

[7] Steven Fortune and James Wyllie. Parallelism in random access machines. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 114–118, 1978.

[8] John L. Hennessy and David A. Patterson. A new golden age for computer architecture. *Commun. ACM*, 62(2):48–60, 2019.

[9] John L. Hennessy and David A. Patterson. *Struttura e progetto dei calcolatori. Progettare con RISC-V*. Zanichelli, 2023.

[10] Steven Leibson. Mips joins the risc-v gang. *Forbes*, 2022.

[4] https://book.easyperf.net/perf_book

[11] David A. Patterson. Reduced instruction set computers then and now. *Computer*, 50(12):10–12, 2017.

[12] David A. Patterson. 50 years of computer architecture: From the mainframe CPU to the domain-specific tpu and the open RISC-V instruction set. In *2018 IEEE International Solid-State Circuits Conference, ISSCC 2018, San Francisco, CA, USA, February 11-15, 2018*, pages 27–31. IEEE, 2018.

[13] Juha Sorva. Notional machines and introductory programming education. *ACM Trans. Comput. Educ.*, 13(2), jul 2013.

[14] Herb Sutter et al. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's journal*, 30(3):202–210, 2005.

[15] Andrew Waterman, Yunsup Lee, Rimas Avizienis, Henry Cook, David A. Patterson, and Krste Asanovic. The RISC-V instruction set. In *2013 IEEE Hot Chips 25 Symposium (HCS), Stanford University, CA, USA, August 25-27, 2013*, page 1. IEEE, 2013.

Pensiero cooperativo per una didattica agile dell'Informatica

Paolo Ciancarini[1], Marcello Missiroli[12], and Daniel Russo[3]

[1] DISI - Università di Bologna,
{paolo.ciancarini,marcello.missiroli}@unibo.it
[2] IIS Corni Liceo e Tecnico, Modena
m.missiroli@istitutocorni.it
[3] Aalborg Univeristy, Aalborg, Danimarca
daniel.russo@cs.aau.dk

Sommario

Descriviamo la nozione didattico-educativa di Pensiero Cooperativo, che estende il Pensiero Computazionale con i valori di collaborazione alla base del Manifesto Agile. Presentiamo quindi alcune esperienze didattiche volte a stimolare il Pensiero Cooperativo tra gli studenti medi e universitari in ambito informatico. Infine, descriviamo brevemente i risultati di tali esperienze.

1 Introduzione

Secondo il World Economic Forum[11], le tecnologie informatiche quali l'Internet mobile, il cloud, Big Data, robotica e, più recentemente, l'intelligenza artificiale, aprono nuove prospettive tanto per gli utenti quanto per gli sviluppatori. Tutte queste tecnologie sono software-intensive. È quindi necessario sviluppare una diversa struttura culturale e didattica per risolvere i problemi in un mondo in cui i sistemi software diventano sempre più complessi.

Il sistema educativo ha già iniziato a ridefinire i curricula scolastici includendo il Pensiero Computazionale sin dai primi anni di scuola primaria (pensiamo ad esempio all'ambiente Scratch)- Riteniamo però che tale ripensamento non sia sufficiente. Il Pensiero Computazionale si limita a esprimere la capacità **individuale** di produrre codice e andrebbe affiancato alla capacità **sociale** di interagire positivamente all'interno e all'esterno del gruppo di lavoro. L'aumento della complessità richiede un team coordinato e funzionale in grado di integrare diverse abilità e competenze, ancora più di singoli individui iperspecializzati.

L'introduzione dei metodi Agili ha introdotto nello sviluppo del software l'analisi dei dati, il pensiero critico, l'apprendimento cooperativo e continuo, il feedback tempestivo, il team integrato. Questi sono tutti elementi considerati fondamentali dalle moderne teorie pedagogiche ma che sono raramente affrontati organicamente a livello scolastico, specie a partire dalla scuola superiore. Pensiamo ad esempio come il lavoro di gruppo sia percepito in modo fondamentalmente diverso (e inferiore) rispetto alle prestazioni individuali.

Serve una competenza di livello superiore, che chiameremo del **Pensiero Cooperativo** *PCoop*, [10]), definito come **la capacità di descrivere, riconoscere, scomporre i problemi e risolverli computazionalmente in team in modo socialmente sostenibile**.

La novità del Pensiero Cooperativo, come mostrato dalla fig. 1, risiede nell'integrazione di diverse competenze: il Pensiero Computazionale, che è la fondamentale abilità di comprendere i problemi e applicare, valutare e produrre una soluzione sotto forma di algoritmo, e la competenza di lavoro in gruppo - mutuata dal mondo Agile -, che permette al team di comunicare, auto-organizzarsi e collaborare efficientemente.

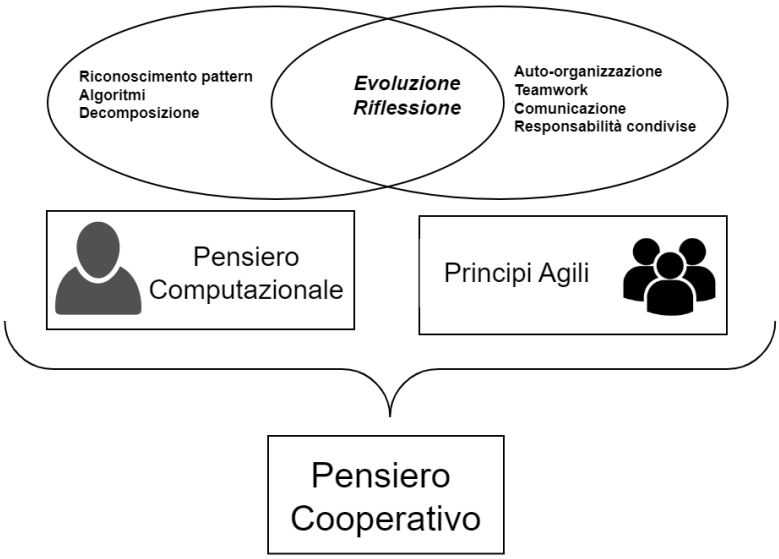


Figura 1: Elementi costitutivi del Pensiero Cooperativo

Educare gli studenti a trattare responsabilmente i rapporti interpersonali significa renderli consapevoli che un'organizzazione lavorativa socialmente sostenibile è fondamentale per risolvere problemi complessi, cosa che permetterà loro di affrontare le future sfide nel campo dell'informatica e più in generale nel mondo del lavoro, non solo tecnologico.

Il seguito di questo articolo ha la seguente struttura: nel paragrafo 2 descriviamo il contesto delle nostre esperienze; nel paragrafo 3 riassumiamo le basi del nostro metodo; il paragrafo 4 descrive come introduciamo alcune pratiche specifiche di collaborazione agile in diversi contesti educativi; il paragrafo 5 discute i risultati ottenuti; infine, il paragrafo 6 propone alcuni sviluppi per il prossimo futuro.

2 Contesto

Nel 2006 Jeannette Wing[12] introdusse il concetto di Pensiero Computazionale, descrivendolo come una competenza fondamentale in tutti i settori, non solo nell'Informatica. È un modo per affrontare problemi complessi, suddividerli in problemi più piccoli (decomposizione), tenere conto di come problemi simili sono stati risolti in passato (riconoscimento di pattern), ignorare informazioni irrilevanti (astrazione) e produrre una soluzione generale e deterministica (algoritmo). Tale idea è oggi universalmente riconosciuta e integrata nei programmi scolastici in tutto il mondo, a partire dalla scuola primaria.

Tuttavia, sempre più studiosi sostengono che il concetto di Pensiero Computazionale sia troppo vago per avere un effetto reale, specie considerando che la continua complessità dei problemi del mondo reale richiede sempre più soluzioni realizzati da team piuttosto che da singoli individui. Purtroppo, le competenze informatiche sono tuttora insegnate a livello individuale. Le ragioni possono essere molteplici: in primis la tendenza a perseguire le modalità di insegnamento più comuni e consolidate, seguite dall'ignoranza di metodologie alternative. La

più importante è probabilmente la difficoltà di valutazione degli studenti: è molto più difficile tracciare le competenze acquisite e il contributo dei singoli quando si lavora in gruppo.

Alcuni studi hanno affrontato l'idea che le competenze hard dovrebbero essere integrate con le competenze *soft*, introducendo possibilmente l'apprendimento attivo e cooperativo in informatica. In particolare, noi ci siamo occupati di questo problema [2] mettendo in luce in luce come il Pensiero computazione e i valori Agili si rafforzino reciprocamente per sostenere il nuovo costrutto del Pensiero Cooperativo. Successivamente abbiamo analizzato i risultati delle valutazioni dei nostri corsi [3], rilevando come i gruppi che avevano conseguito una migliore competenza nell'applicazione delle pratiche Agili risultavano essere anche quelli con le valutazioni migliori - elemento molto interessante dal punto di vista didattico.

3 Metodo

Quello che segue riassume i nostri progressi nell'introduzione del Pensiero Cooperativo nei nostri corsi a livello di istruzione secondaria e terziaria. Ispirato al pensiero costruzionista di Jonassen[6] e alle idee alla base del lavoro di gruppo secondo Johnson et al.[5], si basa su quattro principi cardine, applicabili sia alla scuola superiore sia all'università.

1. Didattica progettuale "costruttivamente autentica" ([12], [6]). Aspetti decisivi sono quelli di richiedere la realizzazione di un progetto impegnativo, complesso, con la necessità di interagire con uno stakeholder esterno ([7]). In tal modo si promuove lo sviluppo di abilità capacità adeguate per affrontare questioni stratificate, negoziare compromessi, trovare un modo sostenibile per ideare una soluzione di gruppo al problema.

2. Centralità del team. È il team, non l'individuo, a dover risolvere il problema. In questo modo si stimola l'apprendimento continuo, l'auto-organizzazione e l'adattabilità sociale e la responsabilità individuale. Per realizzarlo si mettono in atto diverse strategie di *teamforming* (guidato e non) e di *teambuilding* (Lego Serious Play, Scrum Lego Game, Scrumble). L'ultimo aspetto si ottiene basando la valutazione sui risultati di gruppo con correttivi individuali, quali osservazione diretta, relazioni personali e autovalutazioni.

3. Metodologia di sviluppo. Si introducono varie pratiche Agili, quali pair programming, test automatici, rilasci incrementali, specifiche variabili, proprietà condivisa, spesso supportati da tool specifici (es: *Git*, *Taiga*[4]). L'obiettivo è quello di stimolare lo sviluppo dei principi Agili[1], che riteniamo fondamentali per uno sviluppatore al passo coi tempi.

4. Feedback continuo: questo si applica sia a livello "verticale" (docenti-studenti), sotto forma di osservazioni continue e valutazioni formative fornite al termine di ogni iterazione di sviluppo, sia a livello "orizzontale" (studenti-studenti). Lo scopo è quello di tenere sotto controllo il processo e migliorarlo.

Più in dettaglio, il nostro metodo prevede i seguenti elementi in ordine temporale:

- Presentazione sintetica degli elementi di sviluppo innovativi (incluso gli strumenti digitali a supporto dello sviluppo collaborativo);

- Formazione dei team e conseguente costruzione dell'identità di gruppo mediante attività ludiche di teambuilding;

- Presentazione del prodotto da sviluppare in modalità Agile *Scrum-like*. Elementi di rilievo sono: complessità realizzativa superiore alle capacità del singolo, presenza di elementi tecnici **non noti**, coinvolgimento di stakeholder esterni all'istituzione didattica con cui rapportarsi;
- Fase di sviluppo intensiva con struttura iterativa a sprint con timeboxing rigido e frequenti momenti di confronto e riflessione;
- Presentazione e discussione finale del progetto;
- Valutazione incentrata più sul processo che sul prodotto (ispirate alle *rubric* di Lepida Scuola[13]).

La precisa declinazione degli elementi varia a seconda del tipo di istituzione scolastica, e spesso di anno in anno, poiché, come accennato, si tratta di un metodo didattico-educativo in continua evoluzione.

4 Le pratiche di collaborazione agile

In questa sezione vogliamo presentare in maggiore dettaglio alcune pratiche messe in atto nei nostri corsi che riteniamo utili a promuovere tra gli studenti l'idea del pensiero cooperativo. Si tratta di metodi in continua evoluzione e che necessitano sempre di opportuni aggiustamenti dovuti al mutevole contesto didattico.

4.1 Scuole medie superiori

L'esempio qui presentato si riferisce alle classi quarte di un triennio tecnico ad indirizzo informatico. Tali classi sono dotate di una grande quantità di ore nel ramo tecnico; riteniamo possibile adattare la metodologia anche ad altre realtà, come ad esempio la specializzazione SIA degli ITES o i licei LSSA. In ogni caso, il prerequisito necessario è quello di ottenere la collaborazione della dirigenza e del consiglio di classe, dato che il progetto richiede anche l'utilizzo delle ore di altre materie.

La pratica è inquadrata all'interno di un percorso di PCTO, in modo da avere la libertà di manovra che questa modalità consente. Nel nostro caso, essa si svolge nell'arco del 4° anno, ma potrebbe essere anticipato al 3° se le classi dimostrano una buona capacità di programmazione. Ulteriori elementi si possono recuperare da [8], anche se alcuni dettagli si sono gradualmente modificati nel corso degli anni.

4.1.1 Fase di definizione

La fase di preparazione si effettua durante i primi mesi dell'anno e richiede in primo luogo di identificare un "cliente" per il prodotto da realizzare. Si consiglia di trovare un professionista esterno al mondo scolastico, in linea con l'idea del PCTO; se ciò non fosse possibile, il referente potrebbe essere un docente. In questa fase occorre anche delineare:

- Il prodotto che il cliente desidera. L'obiettivo deve tenere presente le capacità e le competenze della classe, ma deve comunque essere **ambizioso** e percepito come **difficile e complesso**. Nel corso della nostra esperienza, abbiamo proposto ad esempio un gestore di fatture, un videogame, un appello elettronico tramite *bluetooth*, un analizzatore dell'andamento della borsa in real time. L'importante è che sia un prodotto potenzialmente utilizzabile dall'utente finale.

- I prerequisiti e le competenze richieste. Sulla base del prodotto richiesto, occorre fornire agli studenti le competenze base per poter affrontare il lavoro, pur lasciando alcune parti inesplorate in modo da instillare l'idea che la formazione avviene anche durante lo sviluppo.

- Le pratiche Agili di sviluppo che saranno richieste agli studenti. Alcune di queste sono semplici da spiegare ed attuare (come ad esempio il timeboxing, e il pair programming), mentre altre richiedono una certa pratica ed esperienza (user stories, tdd, retrospettive, git). Si consiglia ai colleghi di non introdurre tutto in una sola volta, ma iniziare dapprima con alcune pratiche e introdurne di nuove nelle iterazioni successive del progetto.

- I modi e i tempi di esecuzione. Il periodo di sviluppo del progetto deve essere definito in anticipo e con una durata ben precisa; dovrebbe essere inoltre scisso in due o più iterazioniper ottenere feedback dal cliente, in modo analogo a Scrum. In particolare, occorre programmare modalità e durata le fasi di preparazione, di esecuzione e di valutazione, che potrebbero richiedere l'aggiustamento di programmi e orari anche di più materie, non solo quelle tecniche. Ad esempio, il cliente potrebbe essere (o simulare di essere) straniero, per cui tutte le interazioni e i risultati saranno prodotti in lingua inglese. Di particolare importanza è il momento di presentazione al cliente, che risulta spesso in un momento di emozioni intense.

- Le modalità di valutazione. In questo tipo di esperienza didattica è opportuno che si tenga conto di tre fattori: la valutazione del cliente finale (che valuta quanto il team è riuscito a capire delle sue esigenze), quella dei docenti (che valuteranno l'impegno di gruppo ed individuale, la qualità del processo e la qualità del codice prodotto) e quella degli studenti stessi (che si auto-valuteranno per impegno individuale e prestazioni di gruppo). Il mix preciso di tali componenti può variare anche considerevolmente, ma in generale è importante che vi sia un bilanciamento tra valutazione di gruppo e del singolo - in modo da favorire tanto la responsabilità individuale quanto quella individuale. Riteniamo opportuno che tali criteri siano stabiliti in anticipo e condivisi con gli studenti.

4.1.2 Fase di preparazione

In questa fase si introduce la modalità di lavoro che il gruppo dovrà applicare durante l'esperienza.

Per prima cosa occorre approntare i team. Nella maggior parte dei casi i migliori risultati si sono ottenuti con gruppi misti, quindi con un mix di abilità e competenze (tecniche e non). Tuttavia, in alcuni casi si è optato per gruppi omogenei – i cosiddetti "supergruppi" nel caso degli studenti migliori – anche se più di una volta i risultati sono stati inferiori alle attese. Fatto questo, è opportuno effettuare attività di teambuilding (tra quelle citate in precedenza) nonché effettuare il "rodaggio" del gruppo su un semplice esercizio risolvibile in un paio d'ore. Questo permetterà di affinare il gruppo e - se necessario - effettuare qualche modifica alla composizione.

Successivamente, occorre istruire i ragazzi sulle pratiche di sviluppo Agile che si chiederà di utilizzare. Dalla nostra esperienza, le due pratiche più difficili da introiettare risultano la *shared code responsibility* (tramite git e github/gitlab) e la *test driven development*, tanto nella variante *test first* quanto *test last* (tramite JUnit o equivalente). In entrambi i casi si consiglia di iniziare i ragazzi a tali tecniche prima possibile e, se possibile, integrarle nelle normali attività didattiche; in caso contrario, si rischia che cadano nel vuoto per mesi e, al lato pratico, debbano essere rispiegate quasi daccapo.

Altre tecniche come le *user story* e il *timeboxing* risultano spesso di facile comprensione teorica, ma di scarsa qualità, anche perché i ragazzi sono restii ad abbandonare le proprie abitudini e non sono abituati a gestire progetti che richiedano una effettiva pianificazione e strutturazione; in questa fase le indicazioni da parte dei docenti sono vitali. Si consiglia quindi di effettuare qualche attività (come ad esempio Elephant Carpaccio[1]) e esercizi di decomposizione/raffinamento di user story per evitare di cadere negli errori più comuni: scrivere user story dal punto di vista del programmatore, assenza di condizioni di validazione, omettere la stima della durata, ecc.

Il *pair programming* (anche nella sua variante *mob programming*) risulta la pratica più popolare, spesso effettuata anche in remoto e in completa autonomia.

4.1.3 Fase progettuale

In questa prima fase si chiederà ai ragazzi di rapportarsi con il cliente e di realizzare una progettazione di massima, idealmente con diagrammi UML, studi di fattibilità, strutturazione in user story. Si tratta di una sorta di Sprint preparatorio (Sprint-0), che è un po' in contrasto con l'approccio Agile, ma è giustificato dalla necessità didattica di guidare gli studenti attraverso le novità metodologiche e tecniche. Tipicamente, la durata è di 10 ore, e si conclude con la presentazione del progetto a cliente, docenti e colleghi che esprimono un giudizio puramente indicativo.

È prevedibile che il risultato di questa fase sia di qualità piuttosto bassa, ma è importante che gli studenti affrontino questi problemi e tentino di risolverli come team e non individualmente.

4.1.4 Fase di codifica

La fase di codifica è quella che, ovviamente, coinvolge in maniera più completa le classi. Si consiglia di strutturare lo sviluppo in Sprint di durata fissa (almeno due) e al termine di ognuna di esse si effettua un momento di valutazione (sprint review, che culmina con la presentazione di un prototipo parziale ma funzionante - MVP) seguito da un fase di riflessione (sprint retrospective). È consigliato guidare gli studenti indicando alcuni obiettivi "minimi" da raggiungere al termine di ogni iterazione. Ogni Sprint dovrebbe prevedere 10-30 ore di sviluppo, escluso il lavoro a casa.

La fase termina con un momento collegiale (spesso festoso) i cui tutti i gruppi presentano i propri risultati al cliente come team - tutti devono partecipare e contribuire.

4.1.5 Fase di valutazione

La valutazione si esprime attraverso diversi fattori. In primo luogo, il cliente valuta in modo insindacabile il prodotto finale, ovvero quanto il risultato è conforme alle sue richieste e aspettative. Non è necessario esprimere voti, può essere sufficiente identificare i lavori migliori, quelli accettabili e quelli non soddisfacenti. Tale valutazione è *sempre* di squadra. A seguire i docenti esprimono una loro valutazione sia sul prodotto (basandosi sul codice e gli artefatti realizzati durante lo sviluppo) sia sul metodo seguito (tramite osservazione diretta). Infine, gli studenti valutano il proprio prodotto, la metodologia seguita e i colleghi di gruppo, tramite una relazione di team (pubblica) ed una personale (privata). I docenti quindi, tenendo conto di tutti questi fattori, esprimeranno uno o più voti da applicare in Informatica e nelle altre materie coinvolte nel progetto.

[1] https://sites.google.com/view/carpacciodielefante/home

4.2 Università

In ambito universitario le difficoltà da affrontare sono un po' diverse. Per esempio, non è necessario confrontarsi con i colleghi, ma si è costretti a convivere con tempi serrati e la sovrapposizione di impegni degli studenti. Presentiamo qui la nostra esperienza relativa ai corsi del 3°anno della laurea triennale in Scienze dell'informazione, corso non obbligatorio.

4.2.1 Fase di preparazione

Per prima cosa occorre decidere se il progetto e le lezioni saranno sequenziali o intercalate. Nel primo caso, il corso prevede una parte di lezioni frontali corredate di esercitazioni, seguite dalla fase di sviluppo. Nel secondo caso, lo sviluppo è concorrente alle spiegazioni, e ogni sprint"aggiunge" richieste metodologiche. A livello italiano ed internazionale ritroviamo entrambe le modalità, a seconda delle necessità e dello stile didattico. Il punto fondamentale che al termine di ogni sprint ci sia un momento di riflessione su processo e prodotto.

In ambito universitario è comunque necessario indicare preventivamente e tempestivamente le modalità di valutazione che saranno applicate (team/gruppo, modalità di superamento alternative, possibilità di ripetere l'esame ecc.),

Per la scelta del progetto, trattandosi di un livello molto più elevato, si può puntare su obiettivi articolati. Nel corso degli anni si è spaziato da sistemi di gestione del gioco degli scacchi online a sistemi di allarme per eventi catastrofici, passando per la sentiment analysis dei tweet relativi ad un evento. Il docente dovrà ricoprire il ruolo di "cliente incompetente", anche se è possibile avere un cliente esterno al mondo universitario.

La costruzione dei team sarà responsabilità degli studenti stessi. Un punto chiave è quello di avere gruppi numericamente corposi (5-7 persone), in modo da stimolare l'aspetto di collaborazione e sviluppo delle soft skills, peraltro molto richieste nel mercato del lavoro. Riteniamo essenziale l'esigere lo svolgimento di almeno un paio di attività di teambuilding.

4.2.2 Fase di sviluppo e feedback

Lo sviluppo vero e proprio avverrà ovviamente in gran parte al di fuori delle lezioni, spesso in modalità remota. È però fondamentale organizzare momenti di feedback con il cliente dopo ogni retrospettiva, in modo da fornire tempestivo feedback ai team. Si consiglia inoltre di monitorare (eventualmente partecipando) almeno alla prima sprint review dei gruppi: si tratta di una pratica nuova che, se mal impostata, può risultare in una inutile perdita di tempo dedicata ad auto-incensazione o, peggio, ad accuse reciproche. Lo scopo invece è indicare cosa non ha funzionato in modo da migliorare il processo in futuro.

4.2.3 Fase valutazione

Anche in questo caso il docente dovrà esprimere una valutazione relativa alla qualità del processo e del prodotto, idealmente sbilanciata verso la prima voce. Gli elementi di valutazione sono ovviamente il livello di soddisfazione del cliente, l'aderenza al processo e le prestazioni del team. Questi ultimi due elementi sono valutate tramite relazione di gruppo ed individuale. È evidente come la maggior parte (se non tutta) della valutazione impatti il gruppo nel suo complesso, dato che le differenze individuali possono risultare solo dalle relazioni individuali e dal comportamento durante la presentazione finale del progetto.

5 Risultati e lezioni apprese

In primo luogo, osserviamo come le abilità di lavoro in team promosse nel campo informatico siano risultate benefiche per tutti gli studenti, ma in particolare per quelli di abilità medio-basse, che riescono così ad apprendere ed operare in modo più costruttivo ed efficiente, ciascuno ricavandosi il ruolo che meglio si adatta alle proprie capacità. Questo è stato verificato con esperimenti e tramite valutazione dei risultati[9, 3]. Possiamo inoltre aggiungere ai vantaggi un notevole sviluppo della capacità di riflessione e autovalutazione incentrata sugli aspetti tecnici e psicologici.

Inoltre, riteniamo che l'introduzione dei principi e dei metodi agili a livello didattico sia un valore aggiunto per gli studenti. Dalle conversazioni con i nostri studenti laureati e diplomati emerge che tali tecniche, considerate ancora una novità nel nostro paese, si stanno rapidamente diffondendo e ogni esperienza nel campo è molto gradita in ambito professionale.

Terzo, limitatamente al lato universitario per il quale il corso è opzionale, si è rilevato un miglioramento delle valutazioni medie e una decisiva diminuzione del numero di respinti agli appelli; gli studenti hanno anche il vantaggio di superare l'esame *durante* il corso, avendo così più tempo a disposizione per gli altri esami della sessione.

Tra gli elementi di criticità di questo nuovo approccio rileviamo un aumento del carico di lavoro da parte degli studenti universitari - significativo in particolar modo per studenti-lavoratori, dato ricavato dai sondaggi post-corso. Tuttavia, si tratta di un aumento *percepito*, perché stando alle stime di impegno orario da loro stessi fornito si ottiene un valore perfettamente in linea con il valore in crediti fornito dal corso. Tra gli studenti abbiamo invece notato come non sia semplice gestire gli aspetti psicologici ed emotivi, nonché motivare gli individui meno versati dal punto di vista tecnico.

6 Prospettive

In questo articolo abbiamo esposto succintamente la nostra proposta di introduzione del Pensiero Cooperativo nei corsi di informatica in ambito di istruzione secondaria e terziaria. Riteniamo che l'azione abbia ottenuto risultati generalmente positivi. Intendiamo approfondire e raffinarne l'applicazione. Per i dettagli si rimanda alle pubblicazioni specifiche in bibliografia, rimanendo disponibili a contatti diretti.

Riferimenti bibliografici

[1] Agile Alliance. The 12 principles behind the agile manifesto. https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/.

[2] Paolo Ciancarini, Marcello Missiroli, and Daniel Russo. Cooperative thinking: Analyzing a new framework for software engineering education. *Journal of Systems and Software*, 157:110401, 2019.

[3] Paolo Ciancarini, Marcello Missiroli, and Sofia Zani. Empirical evaluation of agile teamwork. In *Quality of Information and Communications Technology: 14th International Conference, QUATIC 2021, Algarve, Portugal, September 8–11, 2021, Proceedings*, pages 141–155. Springer, 2021.

[4] Taiga community. Taiga: The free and open-source project management tool. https://taiga.io/.

[5] David W Johnson and Roger T Johnson. *Learning together and alone: Cooperative, competitive, and individualistic learning*. Prentice-Hall, Inc, 1987.

[6] David H Jonassen and Lucia Rohrer-Murphy. Activity theory as a framework for designing constructivist learning environments. *Educational technology research and development*, 47(1):61–79, 1999.

[7] Maisa Mielikäinen. Towards blended learning: Stakeholders' perspectives on a project-based integrated curriculum in ICT engineering education. *Industry and Higher Education*, 36(1):74–85, 2022.

[8] Marcello Missiroli, P Ciancarini, D Russo, and Paolo Torricelli. Developers' week: alternanza scuola lavoro rovesciata. *Mondo Digitale*, 18(81), 2019.

[9] Marcello Missiroli, Daniel Russo, and Paolo Ciancarini. Learning agile software development in high school: an investigation. In *Proceedings of the 38th international conference on software engineering companion*, pages 293–302, 2016.

[10] Marcello Missiroli, Daniel Russo, and Paolo Ciancarini. Cooperative thinking, or: computational thinking meets agile. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, pages 187–191. IEEE, 2017.

[11] WEF. *The Future of Jobs: Employment, Skills and Workforce Strategy for the Fourth Industrial Revolution*. World Economic Forum, Jan 2016.

[12] Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

[13] Enzo Zecchi. Project Based Learning (PBL) activities using the" Lepida Scuola" method. https://enzozecchi.com/project-based-learning-pbl-activities-using-the-lepida-scuola-method-prova-di-aggiunta-note/, 2012.

$Informatica \times Gioco = Fantasia + Regole$

Rosario Culmone[1], Nicola Del Giudice[1],
Alessandro Marcelletti[1] and Barbara Re[1]

Università degli Studi di Camerino, Sezione di Informatica, Camerino, Italia
{rosario.culmone, nicola.delgiudice, alessand.marcelletti, barbara.re}@unicam.it

1 Introduzione

Il continuo sviluppo dell'Informatica nella società sta portando a nuovi progetti sempre più complessi, portando ad una richiesta crescente di figure professionali in tutti i settori collegati all'Informatica. Tuttavia, c'è una carenza di figure professionali, dovuta sia alla mancanza di orientamento universitario sia alla tipologia di didattica scolastica. Questi fattori portano ad avere studenti e studentesse con poca consapevolezza di cosa sia l'Informatica e dei settori ad essa collegati. C'è quindi una forte necessità di organizzare attività di orientamento che facciano comprendere il significato di un percorso in Informatica, enfatizzando sia la programmazione sia la fantasia e creatività, aspetti oggigiorno fondamentali. Allo stesso tempo, è altrettanto necessario sviluppare nuove metodologie di insegnamento portando così allo sviluppo di tutte le competenze fondamentali[1] per formare i professionisti del futuro.

In questo contesto la Sezione di Informatica dell'Università degli Studi di Camerino in collaborazione con Italian Video Game Program (IVIPRO) ha bandito per quattro edizioni il contest online "$Informatica \times Gioco = Fantasia + Regole$"[2]. Il contest è stato finanziato dal Piano Lauree Scientifiche ed ha come obiettivo la realizzazione di un videogioco tramite la piattaforma Gamefroot[3] con la finalità di valorizzare le capacità progettuali e realizzative di studentesse e studenti delle scuole secondarie di secondo grado.

L'utilizzo di un linguaggio di programmazione visuale basato su blocchi rende facilmente accessibili tutte le terminologie ed i costrutti della programmazione anche a contesti didattici non tecnici. Realizzare un videogioco richiede inoltre fantasia, rigore e capacità di governare la complessità nelle varie situazioni del gioco, qualità che sono indispensabili per affrontare con successo qualsiasi attività in ambito di studio o professionale[4]. Il lavoro di gruppo, inoltre, stimola le capacità relazionali degli sviluppatori rispetto al lavoro per il raggiungimento di un obiettivo comune [1, 2]. Particolare risalto è stato dato alla realizzazione di videogiochi con lo scopo di divulgare e promuovere il territorio locale. I videogiochi sono, infatti, un potenziale veicolo per riscoprire il mondo che ci circonda come patrimonio storico, architettonico e urbanistico.

Il progetto attivo da quattro edizioni ha visto oltre millecinquecento studenti confrontarsi nello sviluppo, rendendo il videogioco e la sua ideazione un utile strumento di orientamento verso corsi di laurea in informatica come quelli attivi presso l'Università di Camerino. Non trascurabile è anche il ruolo che il game design ha nell'avvicinare studentesse all'Informatica[5], ne sono testimonianze la percentuale di ragazze che hanno partecipato al contest che poi si sono ribaltate in un incremento complessivo delle iscritte ai corsi di laurea in informatica di UNICAM oltre la media nazionale.

[1] Competenze identificate nel "Framework for 21st Century Learning" disponibile al seguente link
[2] Sito dell'iniziativa disponibile qui
[3] https://make.Gamefroot.com/
[4] Aspetto evidenziato nel documento del Dipartimento per la trasformazione digitale al seguente link
[5] Nel 2022 il 42,0% dei videogiocatori sono donne come riportato nel report annuale IIDEA al seguente link

 ISBN: 978-88-6629-075-9

Il resto dell'articolo è suddiviso come segue. Sezione 2 introduce la struttura del contest e le varie fasi del progetto. Sezione 3 mostra i dati complessivi del contest, con una messa a fuoco sulla partecipazione e sui giochi vincitori. Infine, Sezione 4 riassume i punti salienti concludendo l'articolo.

2 $Informatica \times Gioco = Fantasia + Regole$

Organizzazione dell'Iniziativa. Il contest "$Informatica \times Gioco = Fantasia + Regole$" è rivolto a gruppi composti da due a sei tra studentesse e studenti delle scuole secondarie di secondo grado[6]. La partecipazione prevede la realizzazione di videogiochi originali utilizzando la piattaforma Gamefroot. E' richiesta la realizzazione di almeno tre livelli di gioco di qualsiasi tipo purché non siano violenti, offendano razze o religioni, discriminatori per genere, appartenenza politica, e confessione religiosa. Tutti i gruppi interessati fruiscono di una sessione di formazione online.

Terminata la formazione, ogni gruppo ha tre mesi di tempo per progettare e portare a termine lo sviluppo del videogioco nonché per predisporre la documentazione composta da (i) Game Design Document, (ii) link al gioco pubblicato online e (iii) un video di presentazione. In questa fase i partecipanti hanno la possibilità di usufruire del supporto di tutor utili a risolvere eventuali criticità tecniche e ricevere dei consigli. Raggiunta la scadenza di consegna i progetti vengono quindi valutati considerando i seguenti aspetti: (i) *Originalità*, ovvero la presenza di inventiva e novità rispetto agli esempi forniti nonché la presenza di personalizzazioni grafiche e sonore (1 - 5 pt) (ii), *Complessità*, ovvero quanto il gioco è articolato e intrigante (1 - 5 pt), (iii) *Correttezza*, ovvero l'assenza di situazioni impreviste, stallo o errori (1 - 5 pt), (iv) *Accuratezza* della documentazione presentata (1 - 5 pt). A questi parametri viene aggiunto il plus storico (0 - 3 pt) assegnato in base alla presenza di ricerca storica espressa tramite storie, grafiche e meccaniche di gioco.

Concluse le valutazioni, nel mese di Maggio viene organizzata una giornata finale di presentazione dei progetti dove vengono annunciati i migliori dieci. Questi hanno la possibilità di presentare i propri videogiochi davanti a tutta la giuria e agli altri partecipanti al contest. Infine vengono annunciati i migliori due giochi, primo e secondo posto, che si aggiudicano rispettivamente il premio di 1000 e 500 euro.

Ruolo della formazione. Al fine di accompagnare i partecipanti nell'attività di progettazione e prototipazione è prevista una sessione di formazione di dieci ore interamente online. La prima parte della formazione si incentra sulla progettazione del videogioco come mezzo di valorizzazione del territorio. Durante la prima lezione, vengono infatti forniti esempi reali di videogiochi che hanno incentivato il turismo in alcuni luoghi d'Italia enfatizzandone gli elementi caratteristici. Il secondo seminario si concentra invece sulla tematica del game design, introducendo le tecniche utilizzate dagli sviluppatori in fase di progettazione per delineare al meglio gli aspetti ludici e tecnici dell'elaborato. Ciò viene fatto mostrando nel dettaglio tutti i documenti necessari per rappresentare storie, personaggi e componenti del videogioco supportati da esempi reali come il Game Design Document e il Pitch. Inoltre, particolare attenzione viene posta alla presentazione del gioco e la definizione di tutta la documentazione utile per la presentazione finale del contest. la seconda parte della formazione mira a fornire tutte le competenze necessarie per sviluppare un videogioco tramite la programmazione a blocchi basata su eventi.

[6] Il bando disponibile al seguente link è pubblicato nel mese di Ottobre di ogni anno.

A tal fine, un primo seminario si concentra sull'introduzione dell'Informatica e della programmazione ad eventi basata su Scratch. Durante un secondo seminario invece, viene spiegata la piattaforma Gamefroot tramite lo sviluppo passo passo di un videogioco di esempio. In questo modo, viene fornito ai vari gruppi un primo prototipo con tutte le componenti necessarie per poter continuare in autonomia.

Piattaforma di sviluppo. determinanti nella scelta dello strumento. Per la realizzazione del prototipo finale, si è scelta la piattaforma gratuita online Gamefroot, basata su una programmazione a blocchi orientata ai videogiochi. L'editor permette di personalizzare grafiche, animazioni, suoni ed altro ancora oltre alla possibilità di salvare lo stato corrente del progetto rendono la condivisione del codice estremamente facile. Terminato il videogioco, Gamefroot ne permette la condivisione e pubblicazione online in modo gratuito. Infine, la possibilità di lavorare interamente online, senza la necessità di particolari requisiti hardware, consente a chiunque di potersi approcciare alla programmazione di videogiochi.

3 Dati delle Quattro Edizioni

Nel corso degli anni, è stato riscontrato un interesse crescente riguardo il tema del game design da parte di scuole, docenti e studenti, incidendo così sulla riuscita del contest.

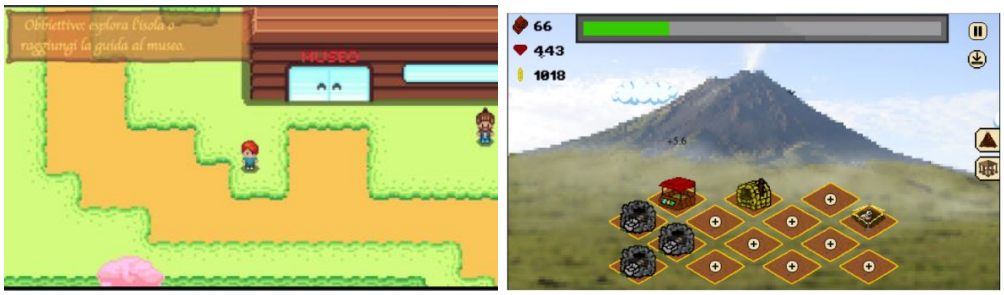
In Tabella 1 sono riportati per ciascuna edizione il numero di scuole partecipanti, studenti e studentesse iscritte, le regioni coinvolte e il numero di progetti consegnati. Si può notare come negli anni sia fortemente aumentato il numero di studenti, arrivato a 800 nell'ultima edizione. Di conseguenza è aumentato anche il numero di progetti e delle scuole coinvolte, distribuite in modo capillare in tutte le regioni d'Italia.

Edizione	Scuole	Partecipanti	Progetti	Regioni
2022/23	50	791 femmine 21% maschi 79%	125	13 (Abruzzo, Basilicata, Calabria, Emilia-Romagna, Lazio, Lombardia, Marche, Piemonte, Puglia, Sardegna, Toscana, Umbria, Veneto)
2021/22	21	786 femmine 30% maschi 70%	83	12 (Marche, Basilicata, Emilia-Romagna, Lazio, Lombardia, Piemonte, Puglia, Sardegna, Sicilia, Toscana, Umbria, Veneto)
2020/21	11	94 femmine 31% maschi 69%	19	3 (Abruzzo, Marche, Umbria)
2019/20	3	15 femmine 40% maschi 60%	3	1 (Marche)

Table 1: Andamento edizioni contest 2019-2023.

Al fine di garantire un monitoraggio continuo dell'iniziativa per le ultime due edizioni è stata posta particolare attenzione all'analisi della qualità della docenza. Al termine del percorso di formazione, è stato infatti chiesto di valutare su una scala da 1 a 5 ciascun modulo formativo. Nell'edizione 2022/2023 su 237 risposte, la media è stata di 3.9 per ogni seminario, mentre nell'edizione 2021/2022 su 172 risposte, la media complessiva è stata di 4.1 dato positivo e che attesta la soddisfazione da parte dei partecipanti rispetto alla proposta formativa[7].

[7] I risultati dei questionari sono disponibili al seguente link



(a) Akragas - Gioco vincitore 2021/2022. (b) Pompei 2079 - Gioco vincitore 2022/2023.

Figure 1: Giochi vincitori edizioni 2022/2023 e 2021/2022.

I giochi vincitori delle ultime due edizioni del contest sono mostrati in Figura 1. Figura 1a mostra il gioco *Akragas* della scuola *IIS F. Re Capriata* di Licata. Akragas è ambientato in un'isola i cui scenari sono riconducibili all'antica polis di Falaride e alla città greca di Finziade. Inoltre, i luoghi riprodotti rappresentano diversi tempi e infrastrutture del tempo. Protagonista è un bambino che, desideroso di visitare i siti antichi dell'isola affronterà diverse sfide ottenendo come ricompense degli artefatti storici.

Figura 1b mostra invece il gioco *Pompei 2079* della scuola *ITIS A. Rossi* di Vicenza. Il gioco è un gestionale ambientato nell'area sottostante al Vesuvio nel 2079 a.c dove gli abitanti del tempo, protagonisti del gioco, devono costruire un tappo per impedire l'eruzione del vulcano. Per fare ciò, è possibile costruire diverse infrastrutture che aiuteranno il giocatore nel raggiungere l'obbiettivo finale.

Entrambi i vincitori hanno mostrato un' eccellente originalità nella storia, nelle grafiche e nelle meccaniche di gioco introdotte. Inoltre, la corretta realizzazione, quindi l'assenza di errori o situazioni impreviste, ha fortemente inciso rispetto l'esperienza di gioco. Fornendo una profonda e dettagliata documentazione, i partecipanti hanno inoltre dimostrato una grande cura nella fase di analisi e progettazione del videogioco.

4 Conclusioni

Il continuo sviluppo dei settori informatici sta aumentando la domanda di figure professionali ad essi collegati, oggi carenti. Per favorirne lo sviluppo, è necessario intraprendere percorsi di orientamento universitari e nuove metodologie di insegnamento nei percorsi scolastici. l'Università degli Studi di Camerino da quattro anni porta avanti l'iniziativa "$Informatica \times Gioco = Fantasia + Regole$" che ha avvicinato ragazzi e ragazze di tutta Italia all'Informatica grazie all'ideazione e programmazione di videogiochi. I risultati mostrati hanno confermato il forte interesse per l'attività e per la tematica del game design favorendo anche le iscrizioni nei nostri corsi di laurea di studenti e studentesse. Inoltre, il continuo contatto con i docenti ha permesso la realizzazione di una comunità interessata alle nuove metodologie di insegnamento.

Ringraziamenti

Ringraziamo IVIPRO quale partner dell'iniziativa, i docenti UNICAM e IVIPRO per essersi messi a disposizione per la formazione, le scuole, i dirigenti, i docenti, ma sopratutto gli studenti

e le studentesse che hanno partecipato all'iniziativa.

References

[1] Zenobia Ismail. Benefits of STEM education. 2018.

[2] Yu Xie, Michael Fang, and Kimberlee Shauman. STEM education. *Annual review of sociology*, 41:331–357, 2015.

L'apprendimento cinestetico degli argoritmi di ordinamento e ricerca su un vettore attraverso la danza

Luca Pinet[1], Laura Frasson[2]
[1] Istituzione Scolastica di Istruzione Liceale, Tecnica e Professionale (I.S.I.L.T.P)
l.pinet@mail.scuole.vda.it
[2] Istituzione Scolastica di Istruzione Liceale, Tecnica e Professionale (I.S.I.L.T.P)
l.frasson@mail.scuole.vda.it

Abstract

Insegnare gli algoritmi di ordinamento del vettore e di ricerca nel vettore attraverso la danza, con un approccio cinestetico alla didattica, riduce i tempi di apprendimento degli stessi per l'intero gruppo classe rispetto a metodi più tradizionali, oltre a favorire l'inclusione di alunni con bisogni educativi speciali.

1 Introduzione

Tra gli argomenti presenti nelle "Linee guida" per gli istituti tecnici con settore tecnologico, in particolare per il secondo biennio dell'indirizzo "Informatica e telecomunicazioni" con articolazione "Informatica", con riferimento alla disciplina "Informatica", emanate attraverso la Direttiva ministeriale n° 4 del 6.1.2012, uno dei più complessi è sicuramente "Principali strutture dati e loro implementazione". Progettando un'unità di apprendimento riguardante questo tema, non si può non fare un'analisi approfondita della struttura dati vettore e dei principali algoritmi notevoli ed essa connessi. In particolare, tra gli algoritmi notevoli che gli studenti hanno maggiore difficoltà ad apprendere vi sono quelli di ordinamento di un vettore e quelli di ricerca all'interno del vettore.

Il Dipartimento di informatica dell'Istituzione scolastica di istruzione liceale, tecnica e professionale di Verrès, in Valle d'Aosta (AO), sperimenta da ormai tre anni scolastici l'apprendimento di suddetti algoritmi di ordinamento e ricerca attraverso la danza. Il risultato che ne deriva è la notevole riduzione dei tempi di apprendimento a parità di risultati nelle valutazioni.

2 Il modello VARK

Il modello VARK, teorizzato da Neil Fleming, definisce l'insieme delle modalità sensoriali che ognuno studente privilegia durante la fase di apprendimento. Il modello VARK prevede la classificazione degli stili di apprendimento in quattro principali categorie:

- Apprendimento visuale. Un Visual learner (un apprendente visivo) preferisce osservare immagini e diagrammi per apprendere al meglio. Possiede un'intelligenza visuo-spaziale molto sviluppata. Egli impara in modo semplice e diretto quando l'informazione viene presentata in modo visivo. Gli apprendenti visivi prediligono l'informazione testuale organizzata in rappresentazioni grafiche, immagini, mappe concettuali, diagrammi di flusso, diagrammi gerarchici, linee temporali. Possono essere utili le presentazioni con slide, i testi con numerose illustrazioni e gli audiovisivi. Questa tipologia di apprendente

ha bisogno di avere un contratto visivo con il docente quando questi spiega: vedere i suoi gesti, le sue espressioni facciali, infatti occupa i primi banchi. Risultano molto utili delle dispense di appunti che riassumono le lezioni svolte con schemi e diagrammi. Negli appunti spesso introducono frecce, simboli, riquadri e altri contrassegni grafici. Possono essere facilmente distratti da movimenti o azioni che avvengono nell'aula, e viceversa non sono infastiditi dai rumori. Buona pratica è evidenziare, sottolineare o colorare le parole importanti. Organizzare l'informazione in punti elenco, che tendono a riassumere i concetti.

- Apprendimento uditivo. Un Aural learner (un apprendente uditivo) è avvantaggiato in prevalenza dall'ascolto. Impara in modo efficace attraverso l'ascolto, pertanto la sua lezione ideale è di tipo verbale, caratterizzata da discussioni e confronti tra docente e studenti. La sua particolare preferenza per il canale comunicativo di tipo uditivo lo porta ad essere sensibile al meglio alle informazioni trasmesse dal tono della voce, dalla cadenza e dal suo ritmo. Pone attenzione alla velocità con cui l'insegnante parla e alle sue pause durante l'esposizione. L'attenzione degli apprendenti uditivi è rivolta soprattutto alle parole che vengono pronunciate dal docente o dagli altri studenti quando sono chiamati a spiegare qualcosa. Durante le lezioni, di solito, gli apprendenti uditivi non prendono appunti dettagliati, perché preferiscono concentrarsi sulle parole ascoltate. Quando leggono, sono soliti farlo ad alta voce oppure bisbigliando, per favorire la loro inclinazione a recepire l'informazione mediante il canale uditivo. Traggono beneficio dalla lettura in classe da parte di terzi. Risultano utili anche registrazioni audio delle lezioni del docente. La loro attenzione cala drasticamente in un ambiente rumoroso. L'apprendimento si consolida quando spiegano in prima persona oralmente ad altre persone. Sono inclini all'apprendimento delle lingue straniere, possiedono un buon vocabolario.
- Apprendimento testuale. Un reader/writer (un lettore/scrittore) riconosce nel testo (letto o scritto) il migliore veicolo di approccio alla conoscenza. La caratteristica principale degli apprendenti lettori/scrittori è la loro naturale vocazione ad apprendere dal testo scritto. Tale aspetto favorisce molto questi studenti, in quanto il testo scritto è ancora un veicolo fondamentale di diffusione del sapere. Sono soliti prendere appunti dettagliati in classe, perché ciò favorisce il loro apprendimento a casa. Possiedono ottime abilità nella realizzazione di testi scritti. La lettura è il processo favorito in input, la scrittura è il processo favorito in output. Riescono ad apprendere facilmente dai libri di testo, appunti e dizionari. Usano in modo efficace i programmi di videoscrittura e di presentazione e sono abili nella ricerca di informazioni testuali nel web. Inoltre, sono particolarmente bravi nel riorganizzare gli appunti che hanno scritto durante le lezioni, di sintetizzarli ed estrarne i contenuti pregnanti. Il docente deve fissare dei concetti spiegati oralmente anche con del testo scritto alla lavagna; preparare delle dispense testuali da distribuire agli alunni; dare agli studenti del tempo per riorganizzare i loro appunti.
- Apprendimento cinestetico. Un Kinesthetic learner (un apprendente cinestetico) trova naturale apprendere mediante l'esperienza diretta e la pratica sul campo. Gli studenti che prediligono questo stile non privilegiano un senso in particolare durante l'apprendimento, piuttosto usano i sensi in modo olistico e integrato. Gli apprendenti cinestetici impegnano una grande energia nelle loro attività. La loro necessità di apprendere mediante azioni concrete li porta spesso a muoversi nella classe e a cercare interazioni con l'ambiente circostante. Una lezione basata sulla trasmissione di informazioni mediante canale visivo e uditivo può penalizzarli molto, tanto da farli risultare spesso distratti dal loro desiderio di muoversi, oppure passivi e poco interessati. Sono molto attivi e apprendono velocemente le abilità di tipo fisico: buone performance atletiche, buona coordinazione, senso del ritmo che li fa eccellere nelle pratiche coreutiche e musicali. Possono apprendere anche materiale astratto, teorico e concettualizzato, purché sia presentato loro con le necessarie analogie, con esempi legati alla vita comune e con le opportune metafore. Il docente deve limitare le spiegazioni al tema centrale e all'idea

principale, coinvolgere gli studenti con attività partiche (esperienze di laboratorio, visite guidate), fare pause frequenti.

3 Apprendimento cinestetico e @AlgoRythmics

Tra le esperienze che rientrano nella definizione di processo di insegnamento/apprendimento cinestetico riportata nel capitolo precedente vi è quindi sicuramente quella della danza, che prevedere il movimento all'interno dell'ambiente di apprendimento. L'approccio cinsestetico è da favorire, quando possibile, rispetto ad altre modalità di stili di insegnamento poiché semplifica i processi cognitivi di apprendimento, oltre che per la classe intera, soprattutto a quegli studenti che presentano bisogni educativi speciali, quali per esempio il disturbo da deficit di attenzione e iperattività, i disturbi specifici dell'apprendimento (poiché i contenuti non vengono trasmessi in forma testuale), ecc.

Ad aiutarci nel combinare il processo di insegnamento/apprendimento cinsestetico (ovvero, nel nostro caso, la danza) con gli algoritmi di ordinamento del vettore e di ricerca nel vettore, realizzando un anomalo connubio tra arte e scienza, vi sono i numerosi video del canale YouTube **@AlgoRythmics**. L'idea, molto originale, di spiegare gli algoritmi di ordinamento (selection sort, bubble sort, insertion sort, quick sort, merge sort, shell sort) attraverso la danza è nata da una università, la Sapientia University, situata a Tirgu Mures (Marosvásárhely), in Romania. Cosa c'è di meglio di far "vedere" agli studenti come funzionano questi algoritmi? In poche parole, i danzatori presenti nei vari video, si sono occupati di coreografare gli algoritmi.

Una volta visionato il video contenente la danza, è poi possibile riprodurlo in classe, predisponendo l'ambiente di apprendimento (che deve essere mobile) in maniera adeguata.

4 Apprendimento visuale e VisuAlgo

Come spiegato nella sezione 2, tra gli stili di apprendimento del modello VARK, è presente, oltre all'apprendimento cinestetico, anche l'apprendimento visuale. Abbiamo detto che un apprendente visivo preferisce osservare rappresentazioni grafiche, immagini e diagrammi per apprendere al meglio.

Per quanto riguarda il processo di insegnamento/apprendimento visuale, uno degli strumenti più efficaci per l'insegnamento degli algoritmi di ordinamento su di un vettore è sicuramente la piattaforma online **VisuAlgo**, che permette agli studenti di visualizzare l'algoritmo in questione in forma "animata". L'utilizzo di questa piattaforma permette al docente di evitare la trattazione teorica degli algoritmi con un approccio tradizionale, attraverso la classica lunga speigazione teorica, tipica della lezione frontale. Partendo da un'animazione, saranno infatti gli studenti stessi a capire la logica di funzionamento dell'algoritmo, senza una spiegazione teorica e dettagliata del docente. Dopo aver "estrapolato" dall'animazione la logica di funzionamento dell'algoritmo in questione, gli studenti saranno poi in grado di formalizzarla secondo determinate fasi illustrate nelle sezioni successive.

5 Profili di apprendimento

Secondo Neil Flaming, vi sono diversi profili di apprendimento che variano in base allo studente preso in considerazione:

- Preferenza singola. Il discente con preferenza singola predilige un solo stile di apprendimento del modello VARK, e con alunni con questo profilo si deve lavorare in maniera molto mirata e individualizzata.

- Preferenza doppia. Il discente con preferenza doppia predilige soltanto due tra i quattro stili di apprendimento proposti dal modello VARK. Si possono quindi utilizzare i due stili in maniera individuale o combinata per raggiungere gli obiettivi di apprendimento.
- Preferenza tripla. Il discente con preferenza tripla predilige soltanto tre tra i quattro stili di apprendimento proposti dal modello VARK. Si possono quindi utilizzare i tre stili in maniera individuale o combinata per raggiungere gli obiettivi di apprendimento.
- Profilo VARK-I. In questo caso, il discente è in grado di apprendere attraverso tutti e quattro gli stili di apprendimento del modello VARK, tuttavia ne seleziona soltanto uno alla volta in base alla situazione in cui si trova e all'argomento specifico che gli viene presentato e al quale è soggetto a stimolo.
- Profilo VARK-II. Anche in questo caso, il discente è in grado di apprendere attraverso tutti e quattro gli stili di apprendimento del modello VARK, tuttavia, perché l'apprendimento sia significativo, i quattro stili devono essere combinati e quindi ha bisogno di tutte le modalità.

Questa premessa serve per dire che, pur essendo il processo di insegnamento/apprendimento cinestetico estremamente efficace e induttivo, pur essendo anche il processo di insegnamento/apprendimento visuale immediato, soprattutto se applicati a concetti complessi come l'ordimentanto e la ricerca di un vettore, spesso da soli non sono sufficienti. Per poter raggiungere la più ampia percentuale di studenti si deve spaziare tra i diversi stili di apprendimento e progettare una lezione dinamica e strutturata.

6 Apprendimento induttivo

L'apprendimento deduttivo, quello tradizionalmente utilizzato nella didattica, prevede principalmente tre fasi, eseguite in maniera sequenziale: presentazione della regola teorica generale; applicazione formale e pratica della regola generale teorica; applicazione della regola in un caso reale particolare.

Al contrario, l'apprendimento induttivo, prevede che le suddette fasi vengano eseguite in ordine inverso. Ovvero, prima si parte dall'applicazione della regola in un caso reale particolare, e solo in seguito si arriva all'applicazione formale e pratica della regola generale teorica, per poi infine arrivare a stabilire la regola teorica generale.

Tra le varie tipologie di apprendimento induttivo vi è quello per scoperta. L'apprendimento per scoperta (discovery learning), teorizzato da Jerome Seymour Bruner, ha come idea quella di ricondurre l'attività di apprendimento in classe a una vera e propria dinamica di scoperta, come quella che viene vissuta dallo scienziato durante le sue ricerche e i suoi studi. Agli studenti viene presentato uno stimolo iniziale che può essere rappresentato da una domanda a cui fornire una risposta, da un problema da risolvere, da un insieme di dati da interpretare. In modo completamente autonomo, gli studenti devono affrontare lo stimolo proposto. Il docente non guida in alcun modo gli studenti. Egli può intervenire fornendo un feedback solo se richiesto dagli alunni. Questa metodologia posa le sue radici nel Costruttivismo, infatti gli studenti sono chiamati in prima persona a costruire le proprie conoscenze.

Il concetto di apprendimento induttivo è lo stesso con cui opera anche l'intelligenza artificiale attraverso il machine learning: vengono forniti alla macchina numerosi esempi di modo che essa possa raggiungere la formulazione di una teoria generale.

L'approccio alla programmazione è sicuramente più significativo e produttivo quando prevede un apprendimento di tipo induttivo, per scoperta, che mette di studenti nella condizione di dover ragionare, di dover risolvere un problema partendo da un semplice stimolo iniziale. Gli studenti devono diventare scienziati, scoprire loro una soluzione. L'utilizzo della danza di @AlgoRythmics e degli algoritmi animati di VisuAlgo permettono al docente di adottare una didattica attiva, che preveda un apprendimento induttivo, poiché, partendo dal ballo e dalla visualizzazione dell'algoritmi, saranno gli studenti che, in autonomia, dovranno stabilire la teoria, la logica, il funzionamento dell'algoritmo.

7 Una lezione-tipo

Per ciascun algoritmo dei sopracitati i tempi richiesti per l'apprendimento attraverso una didattica deduttiva, testuale e verbale, sono all'incirca tre o quattro moduli orari da 50 minuti ciascuno, per un totale di 150 o 200 minuti.

L'utilizzo della seguente lezione tipo, che fonde assieme più stili di apprendimento, permette agli studenti di assimilare i contenuti fondamentali in maniera permanente in circa due moduli orari di lezione da 50 minuti ciascuno, per un totale di 100 minuti. Come si può constatare, quindi, i tempi dedicati all'apprendimento di ciascun algoritmo sono pressoché dimezzati. Le fasi in cui si articola la lezione-tipo, applicabile a ciascun tipo di algoritmo, sono quindi le seguenti:

1. Predisposizione del setting d'aula (5 minuti).
2. Visione ed emulazione, attravedrso la danza, del video disponibile all'interno del canale YouTube **@AlgoRythmics** corrispondente all'algoritmo di ordinamento o di ricerca nel vettore preso in considerazione (Apprendimento cinestetico) (15 minuti).
3. Prime considerazioni riguardo la teoria di funzionamento dell'algoritmo (5 minuti). Gli studenti, in maniera collaborativa (Apprendimento collaborativo), come intero gruppo classe, provano a fare una prima descrizione teorica, astratta, formale e verbale dell'algoritmo (Apprendimento uditivo).
4. Visione dell'animazione dell'algoritmo di ordinamento o ricerca nel vettore presente sulla piattaforma online **VisuAlgo** (Apprendimento visuale) (15 minuti).
5. Conferma o rielaborazione delle precedenti considerazioni riguardo la teoria di funzionamento dell'algoritmo (10 minuti). Gli studenti, in maniera collaborativa (Apprendimento collaborativo), come intero gruppo classe, provano a dare una descrizione teorica, astratta, formale e verbale più accurata dell'algoritmo (Apprendimento uditivo).
6. **Scrittura del codice** (Apprendimento testuale) nel linguaggio di programmazione prescelto che risolve l'algoritmo di ordinamento o ricerca preso in considerazione (25 minuti). Gli studenti, nel laboratorio di informatica (Apprendimento esperienziale) attraverso la metodologia di sviluppo pair programming (ovvero attraverso un processo di sviluppo collaborativo) implementano il codice, lo testano e correggono eventuali errori.
7. Realizzazione del **diagramma di flusso** relativo all'algoritmo appena implementato (15 minuti). Gli studenti realizzano lo schema grafico (Apprendimento visuale) del diagramma di flusso per poter raggiungere quella che è la regola generale dell'algoritmo di ordinamento o ricerca preso in considerazione (Apprendimento induttivo).
8. **Analisi a parole** dell'algoritmo preso in considerazione (10 minuti). Scrittura di un breve testo (Apprendimento testuale) che riassume e descrive il funzionamento e la logica dell'algoritmo in maniera astratta e formale, utilizzando il linguaggio specifico di settore e ricapitolando quanto appreso durante la lezione.

8 Conclusioni

In conclusione, in base all'esperienza didattica significativa riguardante l'apprendimento cinestetico degli algoritmi di ordinamento e ricerca sugli algoritmi attraverso l'uso della danza, oggetto della presente relazione, dopo una sperimentazione di ben tre anni scolastici in tal senso, ci sentiamo di poter dire che, in classi a medio-alta complessità, il processo di insegnamento/apprendimento cinestetico sia davvero uno strumento efficace per poter rendere la didattica accessibile a tutti gli studenti, soprattutto se combinato ad una moltitudine di altri stili di apprendimento e insegnamento. Inoltre, tale approccio alla didattica ci ha dimostrato di dimezzare i tempi dedicati dagli studenti ad apprendere significativamente ogni singolo algoritmo.

E ora si guarda alle sperimentazioni future riguardanti l'approccio cinestetico. Tra le idee c'è, ad esempio, quella di spiegare le topologie fisiche di una rete locale facendo diventare gli studenti

i dispositivi fisici (router, switch o host) e di collegarli attraverso dello spago. Dovranno comunicare con i loro compagni come se si trovassero tutti dentro una vera rete di calcolatori. Altra idea può essere quella di spiegare le matrici simulando una scacchiera umana.

Bibliografia

[1] Linee guida del secondo biennio e quinto anno relative agli istituti tecnici del settore tecnologico – Decreto ministeriale n° 4 del 6/01/2012

[2] Avvertenze Generali per tutte le classi di concorso. Parte generale dei programmi concursuali per l'accesso ai ruoli del personale docente – E. Barbuto, G. Mariani – Edizione EdiSES – V/2020.

[3] 24 CFU per l'accesso a concorsi a cattedra e percorsi FIT - Ambiti disciplinari: Pedagogia, Psicologia, Antropologia, Metodologie e tecnologie didattiche – E. Barbuto, M. La Rana, G. Pianura, M. De Martino, M. Salvatrice Elia, A. Marciano, L. Santoro, A. Schiedi, E. Visconti - Edizione EdiSES – V/2018

Storytelling e Learning by Doing nelle discipline STEM: il caso del "Laboratorio di Informatica" ai Licei Faes di Milano

Fabio Sartori[1], Elisabetta Zanichelli[2], Miriam Nobile[2]
[1] Università degli Studi di Milano-Bicocca
fabio.sartori@unimib.it
[2] FAES Milano
{elisabetta.zanichelli, miriam.nobile}@faesmilano.it

Abstract

Questo articolo presenta l'esperienza del progetto di "Laboratorio di Informatica" svolto come collaborazione tra l'Università degli Studi di Milano-Bicocca e i Licei Faes di Milano. Il progetto ha avuto come obiettivo l'introduzione all'Informatica per gli studenti delle classi prime e seconde dei Licei che lo sceglievano come progetto di potenziamento didattico per il proprio curriculum di studi e si è basato su una metodologia didattica che ha posto al centro dell'attenzione i principi base delle Comunità di Pratica, del learning by doing e dello storytelling. Gli obiettivi raggiunti sono stati l'insegnamento dei fondamenti della programmazione imperativa e a oggetti a studenti completamente digiuni della disciplina e le creazione di nuove collaborazioni future tra due importanti realtà nell'educazione sul territorio milanese.

1 Introduzione

Le "Comunità di Pratica" (CoP) sono gruppi di persone che condividono pratiche ed esperienze per mezzo di relazioni informali [1]. Ogni membro di una CoP mette a disposizione le proprie competenze a beneficio della comunità, per creare una conoscenza globale che ognuno sia in grado di utilizzare e accrescere liberamente. Quindi, le esperienze di ciascuno diventano fondamentali e le relazioni sociali con gli altri membri della comunità promuovono un positivo senso di appartenenza che spinge ciascuno a contribuire al mantenimento e alla crescita della conoscenza globale della CoP [2]. La *legittima partecipazione periferica* (LPP) è, senza dubbio, il più importante aspetto delle CoP nell'assicurare il loro mantenimento e sviluppo, soprattutto grazie alla definizione di opportune metriche atte a misurare quantitativamente il grado di partecipazione di ogni membro alla comunità [3]. In più, la possibilità di implementare efficacemente strategie di *learning by doing*, motivando gli studenti ad apprendere attraverso esempi concreti piuttosto che tramite didattica "tradizionale", rende le CoP ottimi strumenti per la definizione di metodologie di

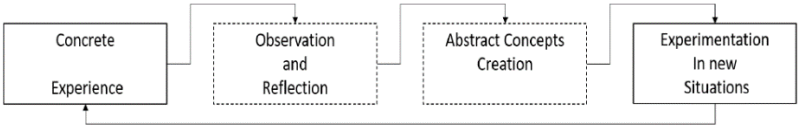


Figura 1: Fasi del Learning by doing

apprendimento basate su modelli di tipo competence-based [4], particolarmente adatto nelle discipline STEM.

La Figura 1 mostra le fasi del ciclo di learning by doing come previste nel modello di riferimento [5]:

1. Concrete experience: la prima fase del ciclo di apprendimento, in cui il docente introduce gli studenti ai contenuti del corso attraverso attività prevalentemente pratiche;
2. Observation and reflection: fase teorica, in cui l'esempio introdotto in precedenza viene analizzato in maniera più formale, estraendone la struttura tramite opportuni strumenti metodologici;
3. Abstract concepts creation, in cui le regole necessarie alla comprensione dell'esempio proposto vengono assimilate e i concetti teorici acquisiti;
4. Experimentation in new situations: ulteriore fase pratica, in cui gli studenti diventano capaci di applicare la conoscenza appena acquisita in contesti nuovi, simili ai precedenti, contribuendo a estendere la conoscenza globale della CoP.

In questo framework concettuale, lo sviluppo metodologico di corsi in ambito STEM risulta particolarmente arricchente e coinvolgente, sia per il docente che per il discente, come dimostrato in letteratura [6] e recentemente sperimentato nell'ambito del progetto "Laboratorio di Informatica per le classi prime e seconde" svolto in collaborazione con i Licei FAES di Milano.

2 Il Fremework Concettuale: Obiettivi e Ruoli

Lo scopo finale di un processo educativo, che sia quello scolastico o quello promosso dalle CoP, è la crescita dell'individuo nella sua totalità. L'obiettivo principale delle CoP è la formazione dei partecipanti in maniera condivisa, mettendo al centro le competenze e l'esperienza di ogni membro. L'apprendimento avviene attraverso lo sviluppo dell'individualità e dell'identità del singolo, con lo scopo della creazione di un sapere comune. Il senso di appartenenza alla comunità che viene sviluppato è strettamente collegato con l'acquisizione di contenuti dal sistema. Di conseguenza, l'apprendimento diventa appunto il risultato di una pratica nella comunità. Gli aspetti più importanti da considerare nell'implementazione di una CoP sono tre [2]:

- la rimozione di barriere per la partecipazione individuale, in modo da facilitare l'accesso e le attività agli utenti aventi difficoltà;

  la centralità dell'unicità di ogni individuo, sostenendola e arricchendola nel contesto della comunità: è importante che ognuno apporti il proprio contributo personale e trasmetta le proprie conoscenze;

- collegare l'unicità degli elementi allo scopo della comunità, il contributo e le particolarità di ogni partecipante devono trovare spazio nel sistema per arricchirlo.

2.1 Ruoli principali all'interno di una CoP

All'interno di una CoP emergono tre tipologie di utenti: *knowledge contributors*, *knowledge seekers* e *lurkers*. I primi sono coloro che contribuiscono in maniera attiva alla crescita della comunità condividendo la propria conoscenza all'interno del sistema. L'azione di condivisione può avvenire sia saltuariamente che in maniera ripetuta. Questa figura svolge un ruolo importante per lo sviluppo, l'incremento e il mantenimento di un *Knowledge Management System* (KMS). I

knowledge seekers sono coloro che ricercano all'interno del sistema le informazioni di cui hanno bisogno, analizzando poi il materiale recuperato. Agiscono in maniera occasionale e passiva sul sistema, entrandoci solo per necessità e non apportando aggiunte o modifiche, come nel caso dei contributors. Infine i lurkers [7] sono degli utenti passivi esterni al KMS che si astengono dalla partecipazione alle attività principali a supporto della costruzione della CoP [7]. Questi ruoli non sono fissati a prescindere, in ogni momento i partecipanti possono evolvere nel sistema passando da un ruolo all'altro.

2.2 Attività e Legittima Partecipazione Periferica

Le CoP sono delle comunità attive che possono evolvere nel tempo, il cui fine ultimo è il miglioramento collettivo attraverso la condivisione di conoscenza. Il concetto base su cui sono formate è la partecipazione attiva degli utenti, che garantisce la crescita del KMS sottostante. Tuttavia, la capacità di contribuire in maniera attiva non è insita nei partecipanti, spesso è difficile da ottenere e necessita di metodologie concrete per essere migliorata e per incrementare il coinvolgimento. Un esempio di questi atteggiamenti esiste nell'uso dei social media e in questo caso la CoP di riferimento si specializza in VCoP (Virtual Community of Practice): la maggior parte degli utenti li utilizza a livello di lurker, visualizzando i post e scorrendo la bacheca con limitate o nulle interazioni; altri utenti invece si comportano da knowledge seeker, ricercando nel social post di interesse, partecipando per esempio con l'aggiunta di commenti; per finire ci sono utenti che partecipano molto attivamente condividendo post oppure, come gli influencer, sfruttando i social con fini personali o commerciali, questi ricoprono il ruolo dei knowledge contributors. Per misurare il grado di coinvolgimento dei partecipanti all'interno di una CoP, sia essa in presenza o virtuale, si possono utilizzare metriche opportune, come il *Sense Of Virtual Community* (SOVC) [3]. Lo sviluppo delle CoP, sia in presenza che virtuali è fondato sul concetto della Legittima Partecipazione Periferica (LPP): questa riguarda la possibilità del singolo di incrementare il livello di sapere dell'intera comunità [1]. Essendo cruciale l'apporto di ogni utente, è necessario che i partecipanti non restino soltanto dei lurkers passivi, ma che operino in maniera attiva nel sistema. Spesso però i ruoli all'interno di una CoP sono abbastanza stabili, perciò risulta difficile soddisfare la LPP. Tuttavia è possibile evolvere da un ruolo all'altro, acquisendo maggiore responsabilità e influenza sul sistema: questo passaggio deve essere consentito dalla struttura stessa della CoP, incentivando le attività e la partecipazione.

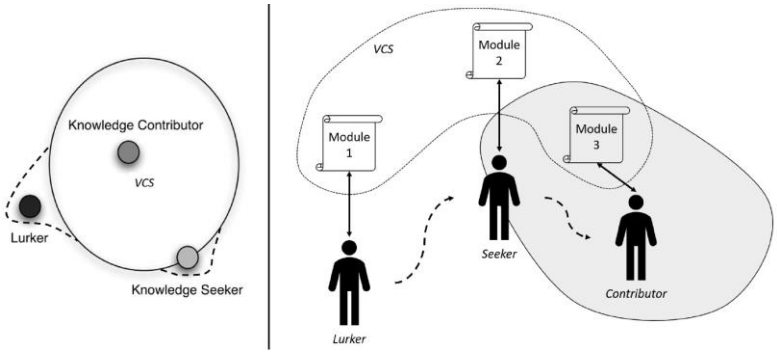


Figura 2: Ruoli all'interno di una VCoP sotto il punto di vista della Legittima Partecipazione Periferica

Per quanto riguarda la LPP dei membri alla Virtual Community, si può fare riferimento alla Figura 2. Come mostrato dalla parte sinistra, i knowledge contributors sono pienamente inseriti nel sistema e cooperano al suo funzionamento e mantenimento. Essi sono, per esempio, gli sviluppatori delle piattaforme e degli strumenti tecnologici necessari al funzionamento della Virtual Community. I lurkers sono esterni alla comunità, vi partecipano saltuariamente in caso di necessità. Per ultimi, i knowledge seekers appartengono alla frontiera, sono una via di mezzo tra gli altri due ruoli, possono comportarsi sia come lurker che da contributor, a seconda dell'esigenza. Mentre i contributors contribuiscono alla crescita della Virtual Community e possono operare su di essa, sia seekers che lurkers partecipano in maniera passiva, essi sono strettamente dipendenti dalle caratteristiche della VCoP in funzione delle proprie esigenze: le loro interazioni col sistema si limitano quindi alla propria area di competenza. Le linee tratteggiate indicano la possibilità dei lurkers e seekers di evolvere entrando a far parte della comunità; solitamente grazie a supporti in grado di destare i loro interessi. Nella parte destra della Figura 2 si può notare come tutte le tipologie di utenti abbiano nel sistema dei moduli appositi a cui poter accedere.

3 Metodologia: Storytelling e Learning by Doing per l'apprendimento dell'Informatica

La collaborazione tra il Dipartimento di Informatica, Sistemistica e Comunicazione e i Licei Faes di Milano si è concentrata sulla progettazione di una serie di moduli didattici che permettessero agli studenti interessati di apprendere gli aspetti fondamentali dell'informatica in maniera proficua e coinvolgente. Da un lato, i requisiti posti dai dirigenti della scuola secondaria in base ai propri ordinamenti risultavano abbastanza complicati da soddisfare pienamente:

- almeno quattro moduli didattici indipendenti e senza pre-requisiti, da sottoporre agli studenti ad anni alterni;
- possibilità per gli studenti di iscriversi a tutti i moduli a sotto-insiemi di essi, avendo la garanzia di ottenerne beneficio nel proprio percorso di crescita personale e nessuna penalizzazione dal non seguito i precedenti.

Dall'altro lato, il principio che muoveva il mondo accademico era la salvaguardia del filo comune alla base dell'idea progettuale, ovvero una comunità che si muoveva attraverso le varie fasi di sviluppo di un progetto software complesso, affrontandone le varie parti in una parte di corso che necessariamente dovesse avere delle basi di partenza dalle precedenti.
Il compromesso è stato raggiunto nei termini rappresentati in Figura 3: dato un dominio di riferimento rappresentato in maniera informale come una *storia,* l'idea di fondo del progetto è far capire ai ragazzi come formalizzarla in maniera opportuna, in modo che sia comprensibile al calcolatore, permettendo loro di affrontare il problema da quattro punti di vista diversi:

- il livello concettuale, ovvero come modellare opportunamente le scelte progettuali in maniera indipendente da quelle che saranno le scelte implementative, utilizzando linguaggi di modellazione specifici, come UML [8];
- il livello dei dati, ovvero come rappresentare opportunamente in maniera formale, attraverso linguaggi e strumenti opportuni, come DBMS sia SQL che NoSQL, [9] le informazioni necessarie alla rappresentazione corretta e completa dello stato degli attori coinvolti nella storia e della loro evoluzione;
- il livello degli algoritmi, ovvero come rappresentare opportunamente, in maniera formale, il comportamento degli attori e le relazioni tra essi, attraverso linguaggi di programmazione, come ad esempio Java [10] che permettano la codifica e l'esecuzione di programmi, riprendendo i concetti precedentemente formalizzati attraverso un linguaggio di modellazione a livello concettuale e, possibilmente, facendo riferimento alle relazioni esistenti con il livello dei dati;

- il livello delle interfacce, ovvero come rappresentare opportunamente, in maniera formale, la relazione tra l'utente e il sistema realizzato, attraverso linguaggi di modellazione e di programmazione adatti, che permettano di considerare la vista sui cosiddetti "requisiti non funzionali", ma di fondamentale importanza per assicurare un livello di fruizione adeguato da parte degli osservatori esterni alla CoP che ha realizzato il sistema.

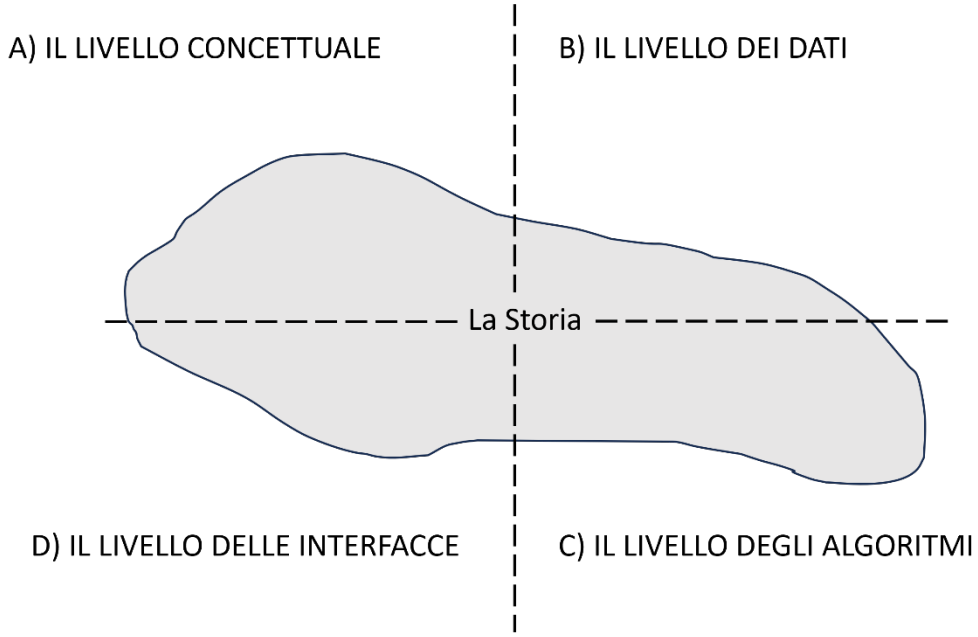


Figura 3: la suddivisione in moduli del progetto

Nel primo anno di progetto si è realizzato un corso pilota, che ha coinvolto le classi prime e seconde dei licei, creando una CoP di studenti coinvolta in particolare sui livelli A) e C) del framework metodologico presentato, e in maniera parziale sul livello B).

Il progetto è stato pensato come una collezione di unità didattiche in cui gli studenti fossero invitati a riflettere in maniera pratica su come modellare un problema in maniera comprensibile a un calcolatore: per fare questo, si è partiti dall'analisi di un problema affrontato con tecniche di narrazione tipiche dello storytelling. Gli esempi necessari sono stati tratti dal "Piccolo Principe", celebre racconto scritto da Antoine de Saint-Exupéry: ad ogni incontro, la CoP di studenti è stata invitata a leggere, comprendere e analizzare un capitolo nuovo, individuandone gli *attori* principali, le loro *caratteristiche*, le *relazioni* stabilite tra essi e il proprio *comportamento*. Dal punto di vista del ciclo rappresentato in Figura 1, questo primo passo dell'incontro ha coperto idealmente la fase *di concrete experience.*

Il lavoro di analisi è stato poi formalizzato in un diagramma delle classi UML, trasformando gli attori in *classi,* le caratteristiche in *attributi,* le relazioni in *associazioni tra classi,* i comportamenti in *operazioni*: durante questa realizzazione della fase di *observation and reflection* del ciclo in Figura 1, gli studenti hanno potuto comprendere come la fase di progettazione alla base di un qualsiasi progetto software sia fondamentale, utilizzando un linguaggio di modellazione di livello già "professionale" ma in un contesto a loro noto, riuscendo ad apprendere in maniera "informale" concetti propri della programmazione strutturata e a oggetti, come *variabile, metodo, attributo, classe, oggetto.*

Quindi, la fase di *abstract concept creation* del ciclo di learning by doing è stata realizzata attraverso l'implementazione del diagramma prodotto nel codice sorgente di un programma in un linguaggio di programmazione reale, ovvero Java.

Infine, la fase di *experimentation in new situations* è stata pensata come lo sviluppo del progetto di esame, in cui agli studenti è stato chiesto di mettere alla prova le proprie conoscenze.

4 Risultati e Conclusioni

La fase di experimentation è stata pensata come un progetto a gruppi in cui ogni CoP ha analizzato un capitolo esistente o creato un nuovo capitolo del "Piccolo Principe", secondo le linee apprese a lezione: analisi e comprensione del dominio di riferimento, personaggi e relazioni tra essi (la "storia"), rappresentazione nel linguaggio di modellazione UML, in termini di classi, attributi (denotanti lo stato degli oggetti), metodi (denotanti il comportamento degli oggetti) e relazioni tra essi e implementazione nel linguaggio di programmazione JAVA, con presentazione alla classe del lavoro svolto attraverso una breve serie di slide (opzionale) realizzata in PowerPoint, Canvas o programmi simili.

Gli studenti hanno formato in maniera autonoma i gruppi (tranne in un paio di situazioni, in cui studenti rimasti isolati sono stati uniti dal docente a gruppi già formati) e in maniera autonoma hanno scelto tematica (il capitolo su cui lavorare) e le strategie operative. Ogni gruppo ha quindi deciso se sviluppare il progetto come unità a sé stante o inquadrarlo nel lavoro svolto durante le sessioni di docenza. Il progetto è stato svolto prevalentemente in classe, durante due delle ultime tre lezioni, con presentazioni svolte durante la penultima e ultima lezione a seconda delle disponibilità dei gruppi e delle assenze dichiarate dagli studenti, in modo da permettere l'interazione col docente, che ha supervisionato le scelte senza però influenzarle, garantendo a ogni gruppo di manifestare la propria autonomia creativa e agire come una vera e propria comunità di pratica, sperimentando i diversi ruoli all'interno del gruppo a seconda delle proprie capacità e peculiarità, portando ciascuno il proprio apporto alla riuscita finale del progetto e accrescendo ognuno il proprio livello di competenze generali, grazie alla condivisione delle scelte operative e al confronto coi compagni e col docente. In Figura 4 è riportato un esempio di progetto svolto da uno dei gruppi, in relazione al ciclo di learning by doing alla base del framework concettuale di riferimento.

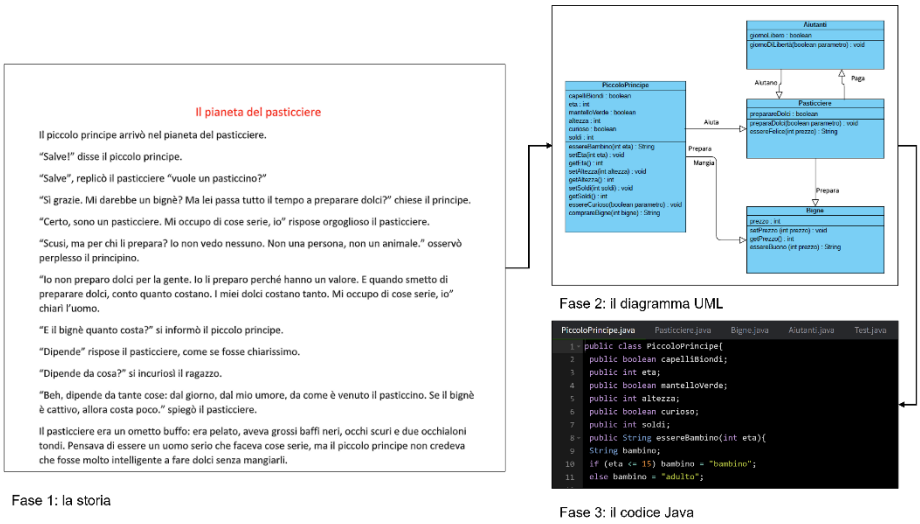


Figura 4: la fase di Experimentation realizzata durante lo sviluppo del progetto da ogni CoP di studenti

Complessivamente, sono stati coinvolti nel progetto 42 ragazze/i (20 di prima e 22 di seconda superiore), che hanno formato 13 gruppi. Tutti i gruppi hanno dimostrato impegno e dedizione nello svolgimento del progetto assegnato, riuscendo a sperimentare con successo quanto visto a lezione in un contesto nuovo. Pur essendo per la maggior parte ragazzi completamente inesperti dal punto di vista dell'Informatica, il gruppo classe si è dimostrato capace di apprendere i concetti trasmessi e replicarli in maniera originale nell'attività progettuale, riuscendo a produrre dei programmi funzionanti e in grado di rappresentare in maniera coerente storie prese dal "Piccolo Principe". Alcuni studenti hanno raggiunto un livello di particolare eccellenza dal punto di vista tecnico

(aggiungendo dettagli implementativi molto fini e addirittura non spiegati durante le lezioni), altri hanno sviluppato progetti molto complessi dal punto di vista organizzativo, riuscendo a mantenere un eccellente livello di coerenza progettuale e implementativa, nonché di presentazione; tutti i gruppi e gli studenti al loro interno hanno dimostrato di aver pienamente raggiunto gli obiettivi del corso.

Per l'AS 2023/2024 si è deciso di partire con il progetto completo: nel primo quadrimestre, da ottobre a dicembre, verrà attivato il corso a livello B) per i ragazzi del triennio, mentre al secondo quadrimestre, da febbraio a maggio, verrà attivato il livello A) per il ragazzi del biennio; nel 2024/2025 si dovrebbe proseguire poi con il livello C) per il biennio e il livello D) per il triennio, in modo da portare a regime su base quadriennale la proposta dei corsi complessivi per gli studenti delle superiori.

I risultati ottenuti saranno sicuramente valutati nell'ottica di future collaborazioni tra Scuola e Università: da un lato, gli studenti hanno sicuramente potuto sperimentare un approccio nuovo all'insegnamento della disciplina dell'Informatica, decisamente distante da quello a cui sono stati abituati finora e più vicino all'idea realistica di progettazione software all'interno di un "team di lavoro"; dall'altro, i docenti coinvolti nel progetto, sia di scuola superiore che universitari, hanno potuto verificare come la metodologia utilizzata abbia favorito un maggior coinvolgimento dei ragazzi nell'apprendimento dei concetti presentati rispetto a un approccio di tipo più tradizionale, traendo utili "lessons learned" per la definizione di possibili scenari di collaborazioni future.

Bibliografia

[1] D. Brown, «Knowledge and organization: A social-practice perspective,» *Organization science,* 2001.

[2] C. Y. Wang, H. Y. Yang e T. C. Seng-cho, «Using peer-to-peer technology for knowledge sharing in communities of practices,» *Decision Support Systems,* vol. 45, n. 3, pp. 528-540., 2008.

[3] M. Blanchard, « The experienced sense of a virtual community: Characteristics and processes,» *ACM Sigmis Database,* 2004.

[4] R. Wesselink, C. D. Jong e H. J. Biemans, «Aspects of competence-based education as footholds to improve the connectivity between learning in school and in the workplace,» *Vocations and Learning 3.1,* vol. 12, n. 7, pp. 19-38, 2010.

[5] F. Sartori, C. Maga, B. Tosi e A. Varallo, «Lurking Reduction at School Through Virtual Communities of Practice: The Binario 9 ¾ Project,» *International Conference on Computational Science and Its Applications. Springer, Cham,* pp. 535-543, 2020.

[6] H. Jho, O. Hong e J. & Song, «An analysis of STEM/STEAM teacher education in Korea with a case study of two schools from a community of practice perspective,» *Eurasia Journal of Mathematics, Science and Technology Education,* vol. 12, n. 7, pp. 1843-1862, 2016.

[7] A. Yeow, S. Johnson e S. & Faraj, «Lurking: legitimate or illegitimate peripheral participation?,» *ICIS 2006 Proceedings,* p. 62, 2006.

[8] B. Unhelkar, Software engineering with uml., CRC Press, 2017.

[9] A. Meier e M. Kaufmann, SQL & NoSQL databases, . Berlin/Heidelberg, Germany: Springer Fachmedien Wiesbaden, 2019.

[10] W. Savitch, Java: An Introduction to Problem Solving and Programming, Student Value Edition Plus MyProgrammingLab with Pearson eText-Access Card Package., Pearson, 2017.

"A Primer on Big & Open Data" (Un'introduzione all'uso dei Big Data in modalità Open)

Francesco Picca
I.I.S.S. "Augusto Righi" – 74100 Taranto, Italy
f.picca@righitaranto.edu.it

Abstract

L'azione #15 del Piano Nazionale della Scuola Digitale (PNSD) aveva previsto la realizzazione di Curricoli Digitali, messi a disposizione di tutto il sistema scolastico per lo sviluppo di competenze digitali.

Così, a seguito del Decreto del Ministro dell'Istruzione, dell'Università e della Ricerca del 2016, è stato proposto la realizzazione di **Curricoli Digitali** da parte delle istituzioni scolastiche ed educative statali (secondarie di 2° grado), *favorendo esperienze di progettazione partecipata, al fine di creare, sperimentare e mettere a disposizione di tutte le scuole nuovi Curricoli Didattici innovativi, strutturati, aperti e in grado di coinvolgere la comunità scolastica allargata.* [1]

Per l'attuazione della sperimentazione, il Ministero (con bando nazionale) ha affidato ad una rete di scuole, capofila il Liceo Marie Curie di Giulianova (TE) [2], lo sviluppo di un curricolo digitale sui Big e Open Data, denominato "A Primer on Big & Open Data".

Partner del progetto sono stati il Laboratorio Nazionale "Informatica e scuola" del CINI, Consorzio Interuniversitario Nazionale per l'Informatica, l'Università dell'Aquila, l'Università di Camerino, il Centro Ricerche Themis.

A seguito di un Bando Pubblico, il Progetto ha visto coinvolti 20 Docenti a livello nazionale, in un percorso di formazione biennale, con lo scopo di discutere e definire il Curricolo Digitale Sperimentale; lo scrivente in rappresentanza *dell'I.I.S.S. Augusto Righi* di Taranto (Istituto di Titolarità).

1 Introduzione

"Leggere, scrivere e far di conto", era il detto di una volta, che caratterizzava le competenze minime di chi, terminata la scuola, si affacciava nel modo del lavoro, o si apprestava ad essere cittadino attivo e consapevole.

Oggi, invece, quella consapevolezza è *attaccata* quotidianamente dall'uso improprio delle tecnologie: bambini e ragazzi *prigionieri* dei loro dispositivi ed inconsapevoli dei processi sottostanti, che permettano loro di scambiare messaggi, giocare in rete, socializzare, usare streaming audio/video, spesso anche annoiarsi, alla ricerca di un qualcosa che sfugge alla logica della loro età.

Occorre una nuova e consolidata consapevolezza: il Digitale come strumento attivo di sviluppo produttivo di competenze e non solo come momento di svago improduttivo.

 ISBN: 978-88-6629-075-9

La scuola italiana ha bisogno di investire con coraggio sull'innovazione per dare risposte che siano inclusive e non divisive; risposte che permettano nei giovani di sviluppare il senso critico, il problem solving, la cittadinanza attiva e digitale, la curiosità ed il voler fare, il saper essere Cittadino Digitale.

Il progetto "A Primer on Big & Open Data", dal punto di vista dello sviluppo del digitale, ha permesso di *vedere* ad una scuola nuova, con la definizione di un curriculum biennale (66 ore totali), in cui affrontare i temi dei dati e dei big data, la definizione di oggetti digitali, la progettazione di proprie semplici App.

2 La Formazione dei Docenti

La Formazione dei Docenti, tenutasi online dall'Università dell'Aquila, dall'Università di Camerino, con il supporto del CINI e del Centro Ricerche Themis, ha permesso di evidenziare i nuclei fondamentali che lo studente della scuola secondaria di secondo grado dovrebbe conoscere/dovrebbe saper fare, e in particolare

- studio dei modi con cui mediante gli algoritmi si estrae in modo efficace informazione da dati massivi ed eterogenei (Big Data)
- studio dei modi con cui mediante ambienti di sviluppo si creano applicazioni avanzate, interattive ed interoperabili
- realizzazione di un'esperienza di elaborazione e pubblicazione di dati aperti (Open Data)
- riflessione sull'impatto dei dati nella società digitale

Il gruppo di progetto ha redatto un Curricolo Digitale sui Principi di Informatica: programmazione e dati [3], attraverso:

1. La programmazione delle singole attività, che il docente può sviluppare in classe, con il dettaglio degli incontri, dei contenuti, dei tempi, delle fasi di sviluppo
2. Le attività del materiale originale del corso CSP di Code.org, per la formazione della Classe Virtuale degli alunni, per consentire le esercitazioni pratiche
3. Le versioni in italiano dei piani di lavoro, che prevedono:
    a. Le lezioni e tutto il materiale didattico per gli studenti, che il docente può usufruire effettuando il login con credenziali di tipologia "insegnante"
    b. Le traduzioni delle esercitazioni interattive, che gli studenti continuano a svolgere direttamente sul sito di fruizione di Code.org.

Il Curricolo è stato presentato in occasione del convegno "*Didattica dell'informatica: sfide e strategie*", tenutosi a Milano il 27 maggio 2022 (Figura 1).

Figura 1: Presentazione del Curricolo

3 La Formazione degli Alunni: La *"Cassetta degli Attrezzi"* dell'Informatico

Parallelamente al confronto tra i docenti e il gruppo di progetto, è stata avviata la Formazione degli Alunni.

Ogni docente in formazione ha creato nel proprio istituto un gruppo classe, prevalentemente con alunni il cui piano di studi non prevedesse lo studio dell'Informatica o discipline affini.

Lo scrivente ha costituito un gruppo di 15 ragazzi provenienti dall'indirizzo Liceo Sportivo (di terza e quarta classe) e condotto le attività di formazione, secondo un calendario che prevedeva incontri pomeridiani da 2 ore.

Ogni incontro ha previsto brevi momenti, per l'introduzione e la discussione dei temi di natura teorica, per poi soffermarsi sulle attività pratiche della Piattaforma, corso CSP Code.org, (Figura 2).

Agli alunni è stata proposta l'analogia del *tecnico specializzato* che, per portare a termine il suo intervento e *riparare l'elettrodomestico rotto o sostituire un rubinetto che perde,* si serve di strumenti, che caratterizzano la sua professione.

Via via, durante lo svolgimento delle lezioni, i ragazzi sono stati coinvolti nella progettazione di attività pratiche, varcando così il confine che li vedeva semplici *smanettatori* di APP già pronte all'uso, e diventare ideatori, progettisti e programmatori delle proprie APP.

Tutti devono poter manipolare gli attrezzi principali dell'Informatica; i più appassionati, visionari, sognatori, faranno di quella curiosità iniziale una professione che li accompagnerà per tutta la vita!

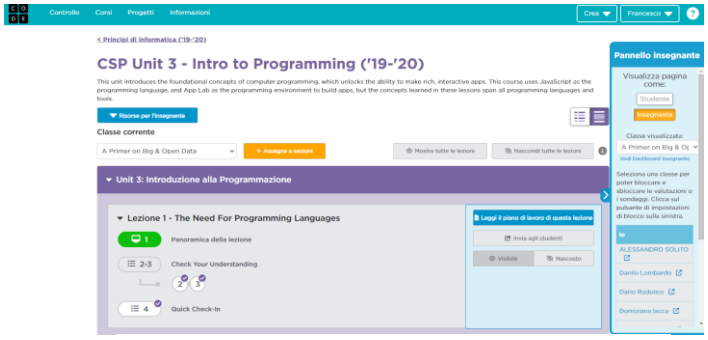


Figura 2: Piattaforma Code.org per le Esercitazioni

3.1 La cassetta degli attrezzi dell'Informatico: Nozioni di Base

Gli strumenti di base della nostra cassetta degli attrezzi sono i tipici concetti, seppure trattati in maniera semplificata (considerata la pianificazione oraria del curricolo), ai quali uno studente di Informatica al primo/secondo anno si avvicina, per comprendere:

- Algoritmi
- Programmazione in coppia
- Efficienza nella programmazione
- Costrutti di base: sequenza, selezione e iterazione
- Tipi di dati ed array, le Stringhe
- Numeri casuali
- Funzioni con parametri e con valori di ritorno

3.2 La cassetta degli attrezzi dell'Informatico: Semplici GUI

Dopo la Logica di funzionamento di un Algoritmo/Programma (back end), il passo successivo è stato l'approfondimento del front end, ed in particolare:

- Programmazione guidata dagli eventi
- Interfacce grafiche e componenti
- Oggetti java script: attributi e metodi
- Azioni sui componenti dell'Interfaccia

Per consentire maggiore autonomia di sviluppo e svincolarsi, seppur per una breve parentesi, dal corso CSP Code.org, è stato chiesto al gruppo classe di esplorare AppLab.

Tutti hanno riconosciuto un ambiente simile alla piattaforma sino ad allora utilizzata e apprezzato la possibilità di poter operare in libertà, utilizzando screen, componenti e codice a blocchi.

Considerato che gli alunni coinvolti sono iscritti al Liceo Sportivo, è stato proposto loro di sviluppare una APP per il "Calcolo dell'Indice di Massa Corporea". (Figura 3) [4].

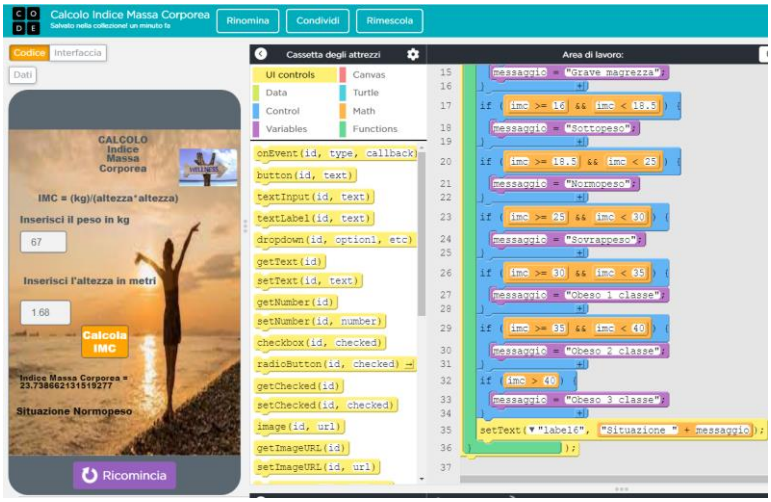


Figura 3: Calcolo dell'IMC con AppLab

3.3 La cassetta degli attrezzi dell'Informatico: Applicazioni e Database

Traendo spunto dall'attività sopra descritta, ci si è avvicinati al bisogno di memorizzare in maniera persistente i dati, introducendo così:

- Database
- Tabelle e record
- Persistenza dei dati
- Accesso alle informazioni in lettura e scrittura

Non avendo il tempo necessario per un approfondimento, ma comunque coltivando la speranza che qualche alunno avrebbe poi sperimentato autonomamente, è stato fatto notare come anche in AppLab esiste la possibilità di memorizzare i dati (Figura 4).

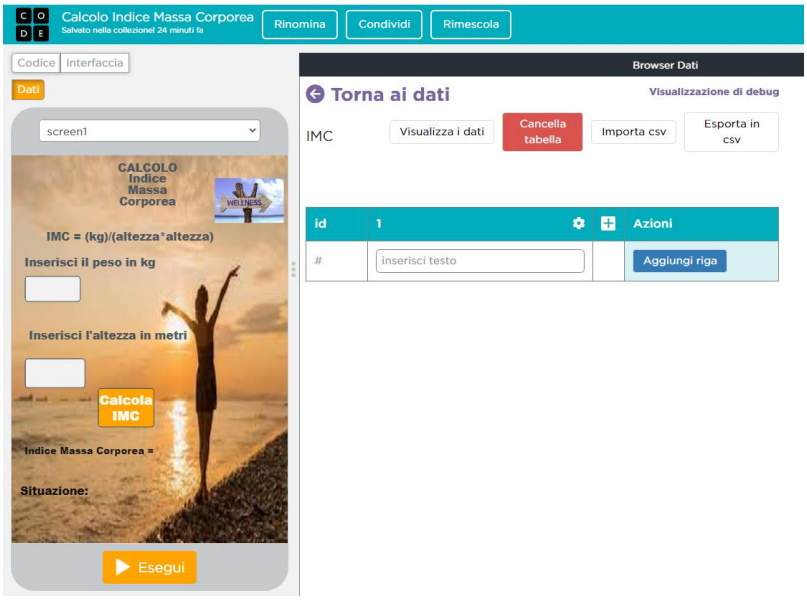


Figura 4: Persistenza dei dati con AppLab

3.4 La cassetta degli attrezzi dell'Informatico: Big Data e Privacy, Crittografia

Per il tema successivo, le domande stimolo, che hanno dato origine ad una attenta ed animata discussione con gli alunni, sono state: I nostri dati sono sicuri? Perché le transazioni di acquisto non possono essere intercettate?

I temi affrontati sono stati:

- Riflessioni sui big data e sulla Privacy
- Crittografia semplice, con chiavi e password e a chiave pubblica
- Cifrario di Cesare e Vigenère
- Crimini Informatici: phishing, DDoS, Virus informatici
- Attacchi informatici
- Open data

Il relativo Modulo del corso CSP di Code.org è ricco di materiali (documenti, video, simulazioni), che avvicinano lo studente alla matematica (Figura 5), che è alla base dei sistemi di crittografia a chiave privata e pubblica (Figura 6).

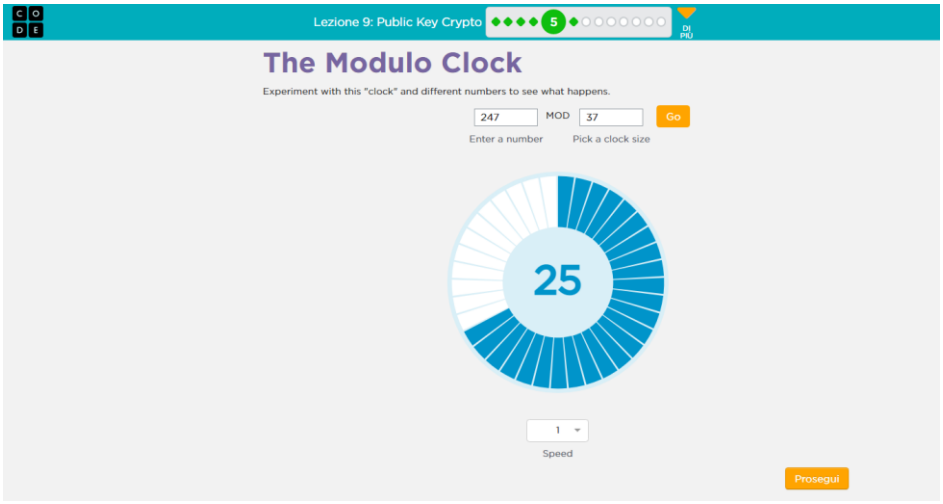


Figura 5: Simulatore di Aritmetica Modulare

Figura 6: VIDEO Crittografia a Chiave Pubblica

4 Sviluppi Futuri

A conclusione del percorso biennale, come ulteriore elemento formativo, lo scrivente ha suggerito agli studenti coinvolti di approfondire ulteriori sezioni del vasto programma offerto da Code.org (Programma il Futuro); in particolare ha chiesto loro, in autonomia, di studiare la sezione relativa alle tecnologie legate all'Intelligenza Artificiale, web app l'Intelligenza Artificiale ed il mare, come funziona l'IA e AI ed Etica. (Figura 7) [5]

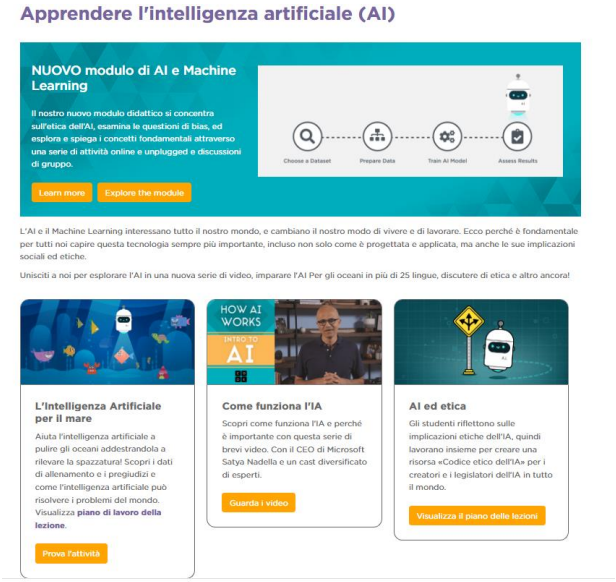


Figura 7: Apprendere l'Intelligenza Artificiale (AI)

5 Conclusioni

Quanto descritto in questo contributo rende solo minimamente l'idea della completezza e della varietà dei materiali del Curricolo Digitale.

Il gruppo classe coinvolto ha sempre partecipato alle attività proposte, con grande entusiasmo, spirito di scoperta e voglia di sperimentare gli strumenti loro forniti.

I Docenti coinvolti nella sperimentazione, in un continuo confronto, concordano nella necessità di fornire le scuole di strumenti di supporto, per l'attuazione di un percorso curriculare dell'Informatica.

In aggiunta ai Riferimenti già indicati, i docenti interessati ad approfondire l'intero Curricolo Digitale posso consultare "Principi di Informatica - Vademecum per i Docenti. [6]

Sitografia

[1] [Online]. Available: https://www.istruzione.it/scuola_digitale/curricoli_digitali.shtml

[2] [Online]. Available: https://www.liceomariecuriegiulianova.it/big-open-data

[3] [Online]. Available: https://programmailfuturo.it/big-open-data

[4] App Calcolo IMC. [Online]. Available: https://tinyurl.com/2uevmhap

[5] Apprendere l'Intelligenza Artificiale (AI). [Online]. Available: https://code.org/ai

[6] Vademecum. [Online]. Available: https://tinyurl.com/5h5u53uk

Insegnare Answer Set Programming nelle Scuole Superiori

Kristian Reale[1]
[1] Istituto d'Istruzione Superiore "Pietro Mazzone" – Roccella Ionica (RC)
Dipartimento di Matematica e Informatica – Università della Calabria – Rende (CS)
kristian.reale@yahoo.it

Abstract

Questo articolo presenta due esperienze didattiche inerenti l'insegnamento dell'Answer Set Programming (ASP) in due Istituti d'Istruzione Superiori situati in provincia di Reggio Calabria. L'ASP è un paradigma di programmazione dichiarativo proposto nell'ambito dell'Intelligenza Artificiale, del Ragionamento non-monotòno e della Programmazione Logica per risolvere problemi computazionali complessi. Nonostante il linguaggio ASP sia ampiamente utilizzato e studiato nella comunità scientifica e comunemente insegnato in prestigiose Università, l'introduzione dello stesso come argomento di insegnamento negli Istituti Scolastici Superiori ha ricevuto scarsa attenzione. Le due esperienze presentate in questo articolo riguardano la sperimentazione dell'insegnamento dell'ASP nel contesto delle scuole superiori, essendo questo un argomento complesso che di solito non è incluso nei relativi programmi di studio standard.

1 Introduzione

L'esperienza didattica ha riguardato la sperimentazione dell'insegnamento della Programmazione Logica, con un focus specifico sull'Answer Set Programming (ASP) [6] aggiornato ai più recenti risultati di ricerca. L'ASP è un paradigma di programmazione logica di tipo espressivo e versatile introdotto nel campo del ragionamento non-monotono che consente di definire problemi computazionali complessi in modo chiaro e completamente dichiarativo. La natura dichiarativa e la potenza espressiva di ASP hanno portato allo sviluppo, nella comunità scientifica, di sistemi di supporto e di esecutori efficienti quali DLV [5]. Ciò ha consentito di sfruttare l'ASP in una vasta gamma di applicazioni in settori quali la pianificazione, i flussi di lavoro, i problemi di ottimizzazione e altro ancora.

Nonostante l'ASP sia ampiamente insegnato in corsi universitari, quali ad esempio quelli offerti da prestigiose istituzioni come il *Politecnico di Vienna (Austria)*, l'*Università di Calabria (Italia)*, il *Texas Tech University (USA)* e l'*Università di Potsdam (Germania)*, la sua integrazione nei contesti delle scuole superiori è limitata. Inoltre, nonostante siano state proposte alcune esperienze di insegnamento, nonché strumenti avanzati per assistere gli studenti nell'apprendimento di ASP [1, 2, 3, 4], questo ha ricevuto scarsa attenzione nelle scuole superiori, specialmente tra gli adolescenti che si concentrano principalmente sui metodi di studio tradizionali tipici degli istituti superiori. Di conseguenza, questa esperienza didattica rappresenta un approccio innovativo che può portare a risultati interessanti; ad esempio, potrebbe portare alla scoperta di nuovi metodi di studio per insegnare la logica avanzata e le sue applicazioni agli studenti delle scuole superiori.

L'esperienza didattica è stata realizzata grazie a due progetti italiani finanziati a livello nazionale. Il primo progetto, noto come "*PON Smart Village: un Paese nel Digitale*" (abbreviato in PON), si è svolto presso l'Istituto di *Istruzione Superiore "La Cava" di Bovalino (RC),* il quale

offre quattro indirizzi di studio: Istituto Professionale per i Servizi Commerciali, Liceo Classico, Liceo Scientifico, e Liceo Linguistico. Il secondo progetto, denominato "*La logica dell'Intelligenza Artificiale*", è stato condotto presso l'*Istituto di Istruzione Superiore "Mazzone" di Roccella Ionica (RC)*, il quale offre vari indirizzi di studio, inclusi il Liceo Scientifico, il Liceo Linguistico e l'Istituto Tecnico Tecnologico: tale progetto è parte del Piano Lauree Scientifiche (abbreviato in PLS), un progetto volto a suscitare interesse ed entusiasmo per le scienze. Entrambi i progetti mirano ad approfondire argomenti che di solito non sono inclusi nei programmi di studio standard delle scuole superiori e a esplorare nuovi argomenti scientifici avanzati che generalmente sono riservati ai corsi universitari o basati su risultati di ricerca scientifica.

2 Descrizione delle esperienze didattiche

Di seguito, vengono illustrate in dettaglio le due esperienze svolte presso i due Istituti.

In particolare, presso l'Istituto d'Istruzione Superiore "IIS La Cava", hanno partecipato a questa esperienza formativa 18 studenti provenienti da diversi indirizzi di studio. Gli studenti più grandi (17-18 anni) provenivano dal Liceo Classico e hanno rappresentato la maggioranza degli studenti, mentre altri studenti, anche di età più bassa (15-18 anni), provenivano dal Liceo Scientifico e dall'Istituto Professionale per i Servizi Commerciali. Tra questi, solo gli studenti del Liceo Scientifico avevano competenze informatiche avanzate, in particolare nella programmazione in C++. Gli studenti dell'Istituto Professionale per i Servizi Commerciali avevano competenze informatiche di base, mentre gli studenti del Liceo Classico non avevano conoscenze informatiche a causa dell'assenza di questa materia nel loro curriculum.

Presso l'Istituto d'Istruzione Superiore "IIS Mazzone", invece, hanno partecipato a questa esperienza formativa 14 studenti del Liceo Scientifico di età compresa tra i 16 ed i 18 anni. Tutti possedevano competenze di programmazione, in particolare in C++, e alcuni di loro conoscevano Java.

"PON Smart Village: un paese nel digitale: Pianificare i Percorsi del pensiero" (PON). Questo progetto, durato 30 ore complessive, è iniziato con un'introduzione all'Intelligenza Artificiale dando particolare enfasi all'importanza di dotare gli Smart Device, quali i dispositivi mobili come gli Smartphone e i sistemi IoT come Raspberry, di Intelligenza Artificiale. L'attenzione è poi passata all'attività di laboratorio che ha riguardato applicazioni pratiche dell'Answer Set Programming, in cui gli studenti, con la mia guida, hanno collaborativamente costruito una *App* per *Android* che consiste in un navigatore intelligente per assistere gli utenti nella navigazione della città di Bovalino (RC). In particolare, il navigatore aveva lo scopo di fornire agli utenti il percorso più breve da un punto A ad un punto B all'interno della città stessa, tenendo conto di interruzioni stradali e deviazioni. Durante la sessione di laboratorio, è stata inizialmente introdotto il concetto di algoritmo, accompagnato da esercizi utilizzando lo strumento *Scratch*. L'introduzione di *Scratch* ha fornito una base preziosa per comprendere lo strumento *App Inventor*, che consente la creazione di App Android reali utilizzando una metodologia simile a *Scratch*. L'attenzione si è poi spostata alla programmazione in Java e infine all'utilizzo del linguaggio ASP per la fase di sviluppo dell'algoritmo di navigazione. Agli studenti sono stati inoltre somministrati due questionari: un questionario intermedio e un questionario finale. I questionari sono consistiti in domande a risposta multipla che riguardavano tutti gli argomenti affrontati durante il percorso: dalla parte introduttiva legata agli Smart Device e ai sistemi IoT, a finire con domande tecniche sul concetto di algoritmo, sul ragionamento logico e sul linguaggio ASP. Sui 30 punti massimi che era possibile raggiungere la media delle risposte corrette è stata superiore al 75%.

"**La logica dell'Intelligenza Artificiale**" (PLS). Questo progetto, durato 12 ore complessive, ha coinvolto due fasi. Nella prima fase del progetto è stata effettuata un'introduzione al concetto di Intelligenza Artificiale, ponendo particolare enfasi sulla distinzione tra *Ragionamento Deduttivo (AI Simbolica)* e *Ragionamento Induttivo (Apprendimento Automatico/Deep Learning)*. Poiché l'obiettivo principale dell'attività era focalizzato sull'utilizzo della Logica nell'Intelligenza

Artificiale, si è data maggiore enfasi sull'approccio del Ragionamento Deduttivo, che tipicamente utilizza l'Answer Set Programming (ASP). L'attività è stata suddivisa in due parti: una parte teorica e una sessione di laboratorio. Durante la parte teorica, sono state presentate alcune slide agli studenti con lo scopo di fornire una panoramica dell'Intelligenza Artificiale in campo scientifico, con particolare attenzione alle applicazioni della logica nell'IA. In particolare, gli argomenti hanno riguardato la Logica Proposizionale e cenni di Logica del Primo Ordine. Successivamente, è stato introdotto l'ASP, spiegando la differenza tra linguaggi di programmazione procedurali e linguaggi di programmazione dichiarativi. La sessione di laboratorio è consistita nel suddividere gli studenti in gruppi e assegnare loro dei compiti da implementare con il linguaggio ASP. L'obiettivo era creare un'Intelligenza Artificiale in grado di giocare ad un gioco di carte a loro scelta. Prima di far lavorare loro, è stata effettuata una dimostrazione pratica per spiegare agli studenti come creare un sistema di Intelligenza Artificiale per il gioco di carte "Sette e Mezzo". Per il loro progetto, alcuni studenti hanno scelto di sviluppare un'IA per il gioco della "Briscola" ottenendo un programma ASP finale che è riuscito a battere a briscola alcuni di noi. Agli studenti è stato inoltre somministrato un questionario finale che è consistito in domande a risposta multipla che riguardavano la logica in generale e il linguaggio ASP. Sui 25 punti massimi che era possibile raggiungere la media delle risposte corrette è stata superiore al 75%.

3 Conclusioni

I risultati di entrambi i progetti (PON e PLS) sono stati estremamente soddisfacenti, poiché gli studenti hanno partecipato con entusiasmo ottenendo risultati notevoli. Hanno compreso rapidamente i principi del ragionamento logico grazie alla natura auto-documentante di ASP, il che ha reso più facile per loro comprendere e applicare il linguaggio in maniera agevole. Le due esperienze hanno inoltre permesso agli studenti di comprendere le differenze tra i paradigmi di programmazione imperativa classica e i linguaggi dichiarativi basati sulla logica, potenziando le loro capacità di pensiero computazionale. La mia supervisione fornita durante la fase iniziale di laboratorio è gradualmente diminuita in quanto gli studenti diventavano, in poche ore, gradualmente sempre più abili nel pensiero logico. Questo è risultato particolarmente evidente nel progetto PON dove c'erano studenti senza precedenti competenze in informatica che sono stati comunque in grado di comprendere il pensiero logico utilizzando ASP. I questionari finali somministrati agli studenti hanno evidenziato riscontri positivi. Si noti inoltre come, nonostante il numero di ore dedicate ai progetti fosse esiguo (30 ore per il PON e solo 12 ore per il PLS), gli studenti sono comunque riusciti a raggiungere gli obiettivi di apprendimento prefissati e a costruire in entrambi i casi qualcosa di tangibile.

In conclusione, l'utilizzo dell'ASP per sviluppare un navigatore automatico intelligente (progetto PON) e un agente intelligente in grado di giocare a un gioco di carte (progetto PLS) ha aiutato gli studenti a comprendere l'importanza degli approcci logici deduttivi nel progresso dell'Intelligenza Artificiale, rappresentando un'alternativa ai metodi di Intelligenza Artificiale induttiva tradizionali e più complessi basati sulle reti neurali. Questo significa che argomenti che inizialmente possono sembrare difficili per gli studenti delle scuole superiori, possono diventare più accessibili se affrontati con attività pratiche e concrete. Applicazioni reali di argomenti difficili ha rappresentato un potente incentivo per il coinvolgimento e l'apprendimento da parte degli studenti, consentendo loro di apprendere concetti complessi creando attivamente prodotti tangibili. Infine, la capacità degli studenti di comprendere rapidamente un linguaggio logico dichiarativo come ASP indica la sua potenziale inclusione come argomento di Informatica nelle scuole superiori.

Per quanto riguarda future esperienze di insegnamento di ASP nelle scuole superiori, ho intenzione di includere più esercitazioni pratiche in cui gli studenti collaborano nella creazione di programmi ASP. L'approccio pratico migliora il loro processo di apprendimento e fornisce una guida preziosa durante lo sviluppo del programma.

Ringraziamenti. Le esperienze scolastiche sono state parzialmente supportate da: (i) Progetto PON "Smart Village: Un Paese Nel Digitale" Modulo "Pianifichiamo i Percorsi del Pensiero" codice identificativo "10.2.2A-FSEPON-CL-2018-535"; (ii) Piano Lauree Scientifiche 2017/18 previsto dal D.M. 1047/2017 con l'"Accordo di Partenariato" tra l'Università degli Studi di Pavia/Università di Milano e l'Università della Calabria.

Altri ringraziamenti: Caterina Autelitano, Dirigente Scolastica dell'Istituto "La Cava" di Bovalino, che ha svolto un ruolo cruciale nel permettere il progetto PON; Francesco Bonaparte, vice-preside dell'Istituto "La Cava" di Bovalino, il quale ha fornito supporto nella definizione e nell'espletamento del progetto; Rosita Fiorenza, Dirigente Scolastica dell'Istituto "Mazzone" di Roccella Ionica, che ha reso possibile la collaborazione con l'Università della Calabria per il progetto PLS.

Bibliografia

[1] Dovier, A. and Benoli, P. and Brocato, M. C. and Dereani L. and Tabacco F. (2016) Reasoning in high schools: Do it with ASP!. CEUR Workshop Proceedings, pp. 205—213

[2] Nguyen, V. and Zhang, Y. and Jung, K. and Xing, W. and Dang, T. (2020) VRASP: A Virtual Reality Environment for Learning Answer Set Programming. In: Komendantskaya, E., Liu, Y. (eds) Practical Aspects of Declarative Languages. PADL 2020. Lecture Notes in Computer Science(), vol 12007. Springer

[3] Febbraro, O. and Reale, K. and Ricca, F. (2011) ASPIDE: Integrated Development Environment for Answer Set Programming. In: LPNMR 2011: Logic Programming and Nonmonotonic Reasoning pp 317–330

[4] Marcopoulos, E. and Zhang, Y. (2019) OnlineSPARC: A programming environment for answer set programming. Theory and Practice of Logic Programming, 19(2), 262-289.

[5] Alviano, M. and Calimeri, F. and Dodaro, C. and Fuscà, D. and Leone, N. and Perri, S. and Ricca, F. and Veltri, P. and Zangari, J. (2017) The ASP System DLV2. In 524 Proceedings of the LPNMR. Springer, Vol. 10377, Lecture Notes in Computer Science, pp. 215–221.

[6] Gelfond, M. and Lifschitz, V. (1991) Classical Negation in Logic Programs and Disjunctive Databases. New Gener. Comput. 1991, 9, 365–386. 516

L'esperienza di Ragazze Digitali: come rendere l'informatica attrattiva per le ragazze

Francesco Faenza, Lisa Fregni, Claudia Canali
Università di Modena e Reggio Emilia
{claudia.canali,francesco.faenza}@unimore.it,
lisa.fregni@gmail.com

Abstract

L'ultimo rapporto del Digital Economy and Society Index (DESI) sulle prestazioni digitali degli Stati membri dell'UE mostra che gran parte della popolazione dell'UE non dispone ancora di competenze digitali di base, anche se la maggior parte dei lavori richiede tali competenze. Il rapporto evidenzia inoltre i tassi estremamente bassi di donne iscritte ai corsi accademici di informatica e ingegneria dell'informazione, che si traducono non solo in una massiccia perdita di talenti per le aziende e le economie, ma anche nel perpetuare i divari nella disuguaglianza di genere nei campi relativi alle TIC. Questo racconto di esperienze sul campo ha l'obiettivo di illustrare l'esperienza di Ragazze Digitali, un summer camp innovativo organizzato annualmente a partire dal 2014 dall'Università di Modena e Reggio Emilia, in collaborazione dal 2018 con l'Università di Bologna e dal 2022 con Regione Emilia Romagna e gli altri Atenei dell'Emilia Romagna, con l'obiettivo di avvicinare all'informatica le studentesse delle classi terze e quarte delle scuole superiori.

1 Introduzione

Recenti statistiche sulle condizioni europee e mondiali indicano che la presenza delle donne nelle professioni informatiche e nei relativi programmi universitari non è migliorata significativamente nell'ultimo decennio e le donne sono ancora fortemente sottorappresentate in questi campi (Eurostat Statistics, 2017-2019). Sia la letteratura che i dati reali mostrano che le differenze di genere rispetto agli interessi, al senso di appartenenza, alla fiducia in se stessi e all'impegno nei confronti delle STEM e dell'informatica sono già presenti in tenera età (Spieler et al., 2020)(Davaki, 2018). Raggiungere una completa comprensione delle ragioni di questo divario di genere è un compito complesso, ma (almeno alcune delle) motivazioni principali sembrano essere legate a questioni sociali e culturali, come gli stereotipi di genere nel campo dell'informatica (Master, 2021). Gli stereotipi, in generale, influenzano le persone e producono travisamenti: l'informatica è tipicamente associata al ruolo maschile (Smeding, 2018), ma gli stereotipi possono riguardare anche l'aspetto fisico, il tipo di personalità e l'abilità digitale proiettati sulle giovani donne, influenzando negativamente le loro decisioni accademiche e le scelte professionali (Berg, 2018)(Heilman, 2001).

Il divario di genere esistente nel campo dell'informatica unito alla crescente esigenza di competenze in questo settore, ha spinto istituzioni e scuole a dare vita a numerose attività extracurricolari come summer camp e iniziative analoghe, volte ad avvicinare giovani studentesse ai concetti scientifici di base dell'informatica e all'uso consapevole della tecnologia digitale.

Nato nel 2014, il progetto Ragazze Digitali (Faenza, 2021 e 2021b) offre alle ragazze un'esperienza di apprendimento innovativo volto a scardinare gli stereotipi di genere nell'informatica. Per la sua natura, la caratteristica di essere completamente gratuito per le partecipanti e per la sua lunga durata (inizialmente 4 settimane), il summer camp Digital Girls

rappresenta un'esperienza unica in Italia e, per quanto ne sappiamo, nel mondo. A testimonianza di questo, menzioniamo gli importanti riconoscimenti internazionali ricevuti dal progetto. Dopo essere stato citato nel report She Figures 2021 della Commissione Europea come best practice italiana per avvicinare le ragazze verso lo studio delle materie ICT, Ragazze Digitali è stato selezionato tra i vincitori della Call for Innovations di OPSI – Observatory of Public Sector Innovation, che ha ricevuto 1084 candidature da 94 Paesi del Mondo. Il progetto è quindi stato inserito nella Biblioteca dei casi di studio OPSI, l'osservatorio della OECD (Organization for Economic Co-operation and Development) che si occupa di analizzare e condividere le buone pratiche per l'innovazione per la Pubblica Amministrazione dei paesi che ne fanno parte.

In questo racconto di esperienze sul campo, analizzeremo l'esperienza del campo estivo nelle sue diverse edizioni, evidenziando l'impatto di tale attività sugli atteggiamenti delle ragazze e sui loro progetti di studi e carriere future. La disponibilità di dati su diverse edizioni del summer camp ci consente di evidenziare pro e contro di diversi approcci nello svolgimento di attività extracurriculari per ridurre il divario di genere nell'educazione ICT.

In questo contributo illustreremo le principali tecniche, apprese e affinate sul campo, per progettare e realizzare un'attività formativa in grado di favorire la partecipazione e l'inclusione femminile nel campo dell'informatica. Presenteremo inoltre uno strumento di valutazione appositamente progettato per misurare l'impatto delle attività extrascolastiche e del background delle partecipanti sulle loro intenzioni future relativamente a scelte di studio o di carriera. Lo strumento proposto include metodi di misurazione sia qualitativi che quantitativi atti anche ad una valutazione della self-efficacy percepita dalle studentesse nei campi digitale e tecnologico (Lewis 2011).

2 Il progetto Ragazze Digitali e la sua evoluzione

Nato nel 2014 a Modena da una collaborazione tra il Dipartimento di Ingegneria 'Enzo Ferrari' dell'Università di Modena e Reggio Emilia e l'associazione femminile EWMD (European Women Management Development), il progetto Ragazze Digitali ha rappresentato fin dall'inizio un esempio di sinergia positiva tra enti del territorio. Prima di tutto, ha previsto il coinvolgimento delle scuole superiori del territorio e dei loro insegnanti, invitati direttamente a partecipare agli incontri di promozione e presentazione dei summer camp, organizzati annualmente presso l'università o presso gli stessi istituti scolastici. Il progetto ha poi ricevuto negli anni un fondamentale supporto finanziario da parte di enti e istituzioni locali, tra cui meritano menzione i Comuni delle città di Modena e Reggio Emilia, sedi di Unimore, alcune Fondazioni bancarie, tra cui ricordiamo il contributo principale di Fondazione di Modena, ed aziende private, tra cui ad esempio Iren, che, unitamente al lavoro pro bono di professori e ricercatori accademici, hanno permesso di realizzare i summer camp consentendone la partecipazione completamente gratuita alle studentesse.

Questo esempio virtuoso di comunità educante ha consentito la realizzazione del progetto a Modena a partire dal 2014 e la sua replica a partire dal 2018 a Reggio Emilia e a Cesena, con la collaborazione del Dipartimento di Informatica - Scienza e Ingegneria dell'Università di Bologna. L'esperienza dei summer camp si è basata su un approccio learn-by-doing fondato sul lavoro di squadra e rivolto ad un apprendimento del coding contestualizzato ad ambiti applicativi che fossero creativi ed innovativi, come la programmazione di videogiochi in Python o la realizzazione di robot controllati da Arduino. Le attività hanno sempre previsto anche interventi e testimonianze di donne, esperte ed imprenditrici aventi ruoli tecnici all'interno di imprese operanti nel settore dell'informatica, che potessero rappresentare modelli di ruolo femminili in grado di contrastare gli stereotipi di genere imperanti nel settore.

Nel 2020 e nel 2021, la pandemia da COVID-19 e le conseguenti misure di distanziamento sociale hanno portato all'impossibilità di svolgere il summer camp Raagzze Digitali in presenza. Le edizioni di quegli anni sono state organizzate per svolgersi completamente online e sono stati necessari alcuni adattamenti in termini di durata e attività disponibili. Si sono realizzati camp di tre settimane, basati sulla progettazione di siti Web, app mobili e semplici videogiochi in Python,

affiancando il tradizionale percorso project-based all'erogazione di interventi e seminari online su approfondimenti di tematiche informatiche. Le edizioni online hanno registrato un successo di partecipazione superiore alle attese iniziali ed hanno consentito di mettere in luce l'attrattività che il progetto ha al di fuori dei territori locali di tradizionale svolgimento, attirando numerose partecipanti da altre città e regioni d'Italia.

A partire dal 2022 il progetto Ragazze Digitali è stato adottato e finanziato dalla Regione Emilia Romagna, divenendo 'Ragazze Digitali ER'[1]. Questo supporto ha permesso di estendere il progetto a tutti gli altri atenei dell'Emilia Romagna, coinvolgendo centinaia di ragazze e diventando un fiore all'occhiello della formazione della Regione ER per la sua capacità di promuovere l'educazione digitale tra le giovani donne. L'edizione 2023 del progetto ha visto l'attivazione di ben 18 summer camp gratuiti, a carattere laboratoriale, per studentesse del terzo e quarto anno delle scuole superiori dell'Emilia-Romagna, sui temi del digitale e delle sue applicazioni. La collaborazione ha coinvolto tutte le Università presenti sul territorio e diversi enti di formazione ed ha permesso di attivare summer camp in tutti i comuni capoluogo (Modena, Reggio Emilia, Bologna, Parma, Ferrara, Cesena, Forlì, Rimini, Ravenna e Piacenza) e raggiungere anche altri comuni di più piccole dimensioni come Cento (FE), Imola (BO), Lugo (RA), Mirandola (MO), Riccione (RN) e San Lazzaro di Savena (BO).

3 Meccanismi di valutazione

3.1 Stato dell'arte

Come affermato in (Faenza, 2021), negli ultimi anni sono state realizzate molte iniziative extracurricolari volte a ridurre il divario di genere nel campo legato all'informatica. Tuttavia, queste iniziative di solito non sono supportate da adeguati strumenti di valutazione che consentano a ricercatori e professionisti di indagare il reale beneficio delle iniziative e il relativo impatto sui partecipanti e sulle loro scelte future. Solitamente, infatti, vengono utilizzati questionari e sondaggi principalmente rivolti a valutare la soddisfazione dei partecipanti.

Sembra mancare nel panorama attuale uno strumento di valutazione che possa essere direttamente riutilizzato o facilmente adattato per valutare l'impatto di attività extracurricolari volte a ridurre il divari odi genere nell'informatica. Analogamente, in letteratura mancano indicazioni o linee guida per la progettazione e lo sviluppo di tale strumento. In alcuni casi, la struttura e la metodologia di indagine utilizzate in alcuni studi scientifici sono fornite in appendice. Un esempio è il caso di (Danoff M., 2017), che fornisce la metodologia utilizzata per valutare le barriere di genere nei confronti dell'informatica ad Harvard. Tuttavia, lo studio si concentra sugli studenti universitari e l'indagine è adattata al contesto della facoltà di Harvard. In altri casi, lo scopo principale dello studio è fornire suggerimenti e linee guida per realizzare le attività in campi estivi STEM senza però fornire informazioni adeguate sulla fase di valutazione (Davis et al., 2013; Mohr-Schroeder et al., 2014).

Alcuni studi si concentrano sull'analisi dei divari di genere nell'informatica e sull'identificazione dei principali fattori che impattano sulle scelte delle ragazze in termini di studi futuri e carriera. L'indagine in (Spieler et al., 2020) ha incluso l'analisi di 28 articoli sottoposti a peer review su tale argomento evidenziando in particolare come diversi fattori possano influire sulla decisione di scegliere un corso di laurea in informatica. Indagano anche la self-perception dei partecipanti riguardo alle proprie capacità nini campo informatici e le somiglianze e le differenze tra la loro identità e l'identità percepita di un esperto ICT (Lewis, Anderson & Yasuhara, 2011). Inoltre, vengono studiate le correlazioni tra l'uso dei videogiochi e l'atteggiamento nei confronti delle discipline informatiche (Davies et al., 2014). Infine, l'influenza degli insegnanti e dei genitori sulle aspirazioni dei partecipanti viene analizzata in (Wong &

1 https://digitale.regione.emilia-romagna.it/ragazze-digitali

Kemp, 2018). Abbiamo preso in considerazione questi risultati per progettare il nostro strumento di valutazione, come descritto nella sezione successiva.

3.2 Il tool di valutazione

In questa sezione descriviamo lo strumento di valutazione realizzato, specificamente progettato per valutare l'impatto sulle partecipanti di attività extracurricolari volte a favorire la partecipazione e l'inclusione femminile. Lo strumento di valutazione proposto è stato realizzato utilizzando focus group per la validazione dello strumento, che è composto da un questionario principale, che include domande classificabili in cinque categorie principali: informazioni di base, percezione dell'informatica, scelte future, stereotipi di genere e soddisfazione rilevata.

Focus group

Focus group sono stati utilizzati per validare il questionario di valutazione. In seguito ad una call for participation rivolta alle partecipanti delle precedenti edizioni di Ragazze Digitali, ne sono state selezionate 15, di cui 5 hanno deciso di proseguire gli studi in corsi di laurea di Informatica o Ingegneria Informatica. Inoltre, abbiamo selezionato 5 studentesse del corso di Informatica con sede a Modena che non avevano avuto l'opportunità di partecipare ad alcuna iniziativa extracurricolare relativa all'informatica durante le scuole superiori. Abbiamo realizzato 4 sessioni con 5 partecipanti ognuna per massimizzare l'efficacia dei focus group. Infine, abbiamo definito gli stimoli da utilizzare durante ogni sessione. Consistevano in quattro domande e frasi principali, ciascuna con due o tre sottostimoli da elaborare in modo più approfondito in caso di conversazione in stallo. Inoltre, gli stimoli erano leggermente diversi per coloro che non avevano partecipato ad alcuna iniziativa extracurricolare prima di scegliere di iscriversi a un corso di laurea in ambito informatico. Nello specifico:

- Riflettiamo sul momento in cui hai scelto di iscriverti a [Ragazze Digitali | Laurea in Informatica]
  - spiega quali erano le tue riflessioni in merito
  - descrivi cosa ti aspettavi di trovare
  - descrivi cosa hai trovato
- Qual era la tua idea dell'informatica prima di quel momento?
  - spiega come si è formata questa idea
  - descrivi prima la tua esperienza con l'informatica
  - descrivi se e come iscriversi al [camp | laurea] ha cambiato questa idea
- (se partecipante al camp) descrivi se e come la partecipazione al camp ha modificato la tua scelta futura
- (se studente di laurea in informatica) descrivi il tuo possibile lavoro futuro e la tua ambizione nel campo
- Individua quattro parole chiave per descrivere la tua esperienza con il mondo informatico fino ad oggi

L'analisi dei risultati dei focus group hanno consentito di affinare lo strumento di valutazione.

Questionari

Lo strumento di valutazione consiste in due questionari, uno presentato all'inizio e uno alla fine delle attività del camp. Alcune domande sono state ripetute sia prima che dopo il camp per cogliere l'eventuale cambiamento nelle conoscenze, percezioni e atteggiamenti delle partecipanti a seguito dell'esperienza. Nel resto di questa sezione viene presentato un elenco di categorie di domande per comprendere meglio la struttura proposta.

La prima categoria di domande è legata al *background delle partecipanti*. Le domande di questa categoria riguardano informazioni generali come nazionalità, tipologia della scuola frequentata, luogo di nascita ed età ma anche informazioni più specifiche quali:

- lavoro dei genitori;
- se e come i genitori sono coinvolti o appassionati di informatica o di elettronica;
- se il partecipante gioca regolarmente ai videogiochi e quale in particolare.

Fanno parte di questa categoria anche domande sull'eventualità che abbiano già avuto esperienze di coding in precedenza e domande più approfondite sulla tipologia di esperienza (es. tipo di linguaggio di programmazione utilizzato).

La seconda categoria di domande è dedicata a comprendere le *percezioni delle partecipanti sull'informatica e sui professionisti del settore*, prima e dopo il camp. In particolare, abbiamo chiesto loro di scegliere alcuni aggettivi per descrivere un professionista informatico e poi alcuni aggettivi per descrivere se stessi, con l'obiettivo di confrontare il risultato prima e dopo il camp e analizzare come evolvono tali descrizioni. Inoltre, prima del campo, abbiamo chiesto quale fosse la loro idea riguardo all'informatica e quanto ne sapessero, mentre dopo il camp, abbiamo chiesto se la loro percezione fosse cambiata e, in caso affermativo, come.

La terza ed essenziale categorie di domande è dedicata alle *scelte future* delle partecipanti. Questa sezione del questionario viene ripetuta prima e dopo l'attività del camp per capire meglio se questo ha avuto un impatto sulle intenzioni future delle partecipanti. Chiediamo loro se sono intenzionate a continuare a studiare o a cercare lavoro e quale sarà il loro campo di studio o di occupazione. Poi, nel sondaggio finale, chiediamo se la loro idea è cambiata e perché; nel sondaggio iniziale chiediamo loro se hanno parlato delle possibili scelte con i genitori o con gli insegnanti ed eventualmente quale è la loro opinione.

Un'intera ulteriore sezione è dedicata agli *stereotipi di genere*; in questo caso l'obiettivo è capire se la partecipante crede o meno che essere donna o avere una famiglia possa rappresentare un ostacolo per la scelta futura o per intraprendere una carriera in ambito informatico o non informatico. Inoltre, queste domande sono state poste prima e dopo il camp poiché, nel corso degli anni, abbiamo notato che è sorta una maggiore consapevolezza riguardo agli stereotipi di genere in ambito di studio e di lavoro.

Una sezione finale inclusa solo nel questionario post-camp analizza la *soddisfazione generale* e il gradimento specifico di singoli aspetti del camp, come le attività svolte, il lavoro di squadra, gli insegnanti e altro ancora. Inoltre, chiediamo alle partecipanti di evidenziare la parte migliore e quella peggiore dell'esperienza e di suggerire possibili miglioramenti.

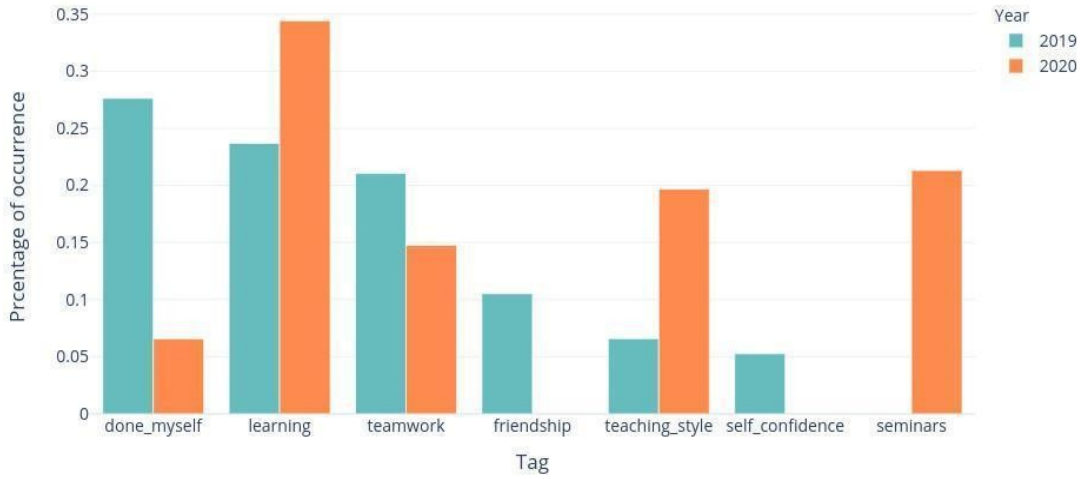
4 Caso di studio

In questa sezione si riportano alcuni risultati legati alle edizioni passate dei summer camp Ragazze Digitali. Un'analisi approfondita dei risultati e delle relazioni tra le caratteristiche del camp e gli impatti ottenuti è molto complessa. Per motivi di spazio si riportano soltanto alcune delle analisi effettuate, per evidenziare alcuni aspetti interessanti che è stato possibile rilevare per mezzo del questionario di valutazione sviluppato.

Per esempio, un risultato interessante è stato quella relativo all'analisi delle principali differenze tra le edizioni in presenza e online del camp. A tale proposito, riportiamo alcuni dati relativi alle valutazioni delle edizioni 2019 (in presenza) e 2020 (online), in particolare considerando le risposte a due domande libere incluse nell'ultima sezione del questionario post-camp: (a) "descrivi cosa ti è piaciuto di più del campo"; (b) "descrivi cosa ti è piaciuto di più del tuo progetto". Nello specifico, a ciascuna risposta sono stati assegnati uno o più tag significativi per caratterizzarla. Ad esempio, la risposta "Mi è piaciuto lavorare in gruppo" è stata taggata con il tag singolo "teamwork", mentre la risposta "Mi è piaciuta la collaborazione all'interno del mio gruppo e la cordialità dell'insegnante" è stata taggata con i seguenti tag: "teamwork" e "teaching_style". Successivamente, sono state contate le occorrenze dei tag per identificare le tendenze principali. La Figura 1 mostra la percentuale di occorrenze dei tag nel corso delle due

edizioni considerate. Una colonna mancante per un tag indica zero occorrenze di quel tag specifico.

Figura 1: analisi dei tag di gradimento



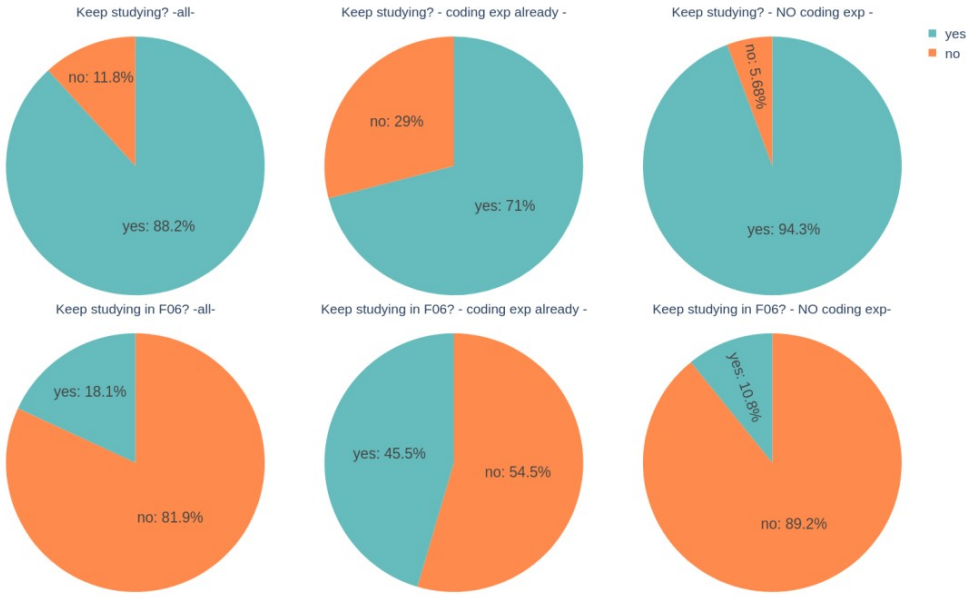
Una delle principali osservazioni è l'assenza del tag "friendship" nei commenti all'edizione online 2020: questo tag è stato assegnato a tutti i commenti che includevano l'esperienza di conoscere qualcun altro, che sono stati inclusi nei commenti dell'edizione 2019, in affermazioni del tipo "mi è piaciuto incontrare nuove persone" o "mi è piaciuto interagire con nuove persone e fare nuove amicizie". L'assenza di questo tag evidenzia la mancata interazione sociale tra le ragazze e la difficoltà ad instaurare relazioni e nuove amicizie durante le attività online. Inoltre, osserviamo che la frequenza del tag "teamwork" è leggermente diminuita per l'edizione online rispetto all'edizione in presenza, sottolineando la maggiore difficoltà di collaborare e interagire con gli altri partecipanti durante la fase di sviluppo del progetto. Notiamo inoltre che il tag "seminar" è presente solo nell'edizione online 2020: ciò è motivato dal fatto che, quando sono impegnate in attività di laboratorio e nella realizzazione di un progetto complesso (come nell'edizione in presenza), le ragazze esprimono meno apprezzamento per i seminari, perché sentono il bisogno di finalizzare i loro progetti.

È emersa un'osservazione molto interessante riguardo al tag "self-confidence", assente nell'edizione online: le attività in presenza sembrano essere più efficaci nell'aumentare la fiducia in sè stesse delle ragazze in ambito informatico. Lo stesso concetto emerge con il tag "done_myself", molto più presente nei risultati relativi all'edizione in presenza. Queste osservazioni rivelano come un elemento fondamentale per contrastare il divario di genere sia più facilmente raggiungibile attraverso attività laboratoriali in presenza. Le studentesse sono generalmente molto meno sicure delle proprie competenze informatiche rispetto agli studenti maschi. Pertanto, migliorare la fiducia in se stesse delle ragazze nei campi delle discipline informatiche può costituire un elemento chiave per coinvolgere più donne nell'informatica.

Veniamo ora ad analizzare qualche risultato relativo al principale obiettivo dell'attività, ossia ridurre il divario di genere nel campo dell'informatica. Per valutare in tal senso l'impatto del camp, abbiamo provato a misurare il cambiamento nelle intenzioni sulle scelte di studio future come conseguenza della partecipazione alle attività previste da Ragazze Digitali. Nei questionari presentati prima e dopo il campo abbiamo inserito due domande specifiche: (1) "Intendi proseguire gli studi all'università?", e in caso di risposta positiva (2) "In quale ambito intendi continuare gli studi?". Per la seconda domanda, consentiamo scelte multiple, che abbiamo convertito nella Classificazione Internazionale Standard dell'Istruzione (ISCED), inclusa l'opzione relativa al campo F06 - Tecnologie dell'informazione e della comunicazione. In particolare, abbiamo ritenuto interessante valutare la connessione tra la futura scelta degli studi accademici e

l'eventuale precedente esperienza delle partecipanti relativamente ad attività di coding. Per questi risultati mostriamo i dati relativi all'edizione 2021, che sono piuttosto simili alle precedenti 2 edizioni. Dal questionario pre-camp emerge che solo il 26,2% dei partecipanti aveva già provato a programmare prima di partecipare al campo estivo. Quindi, abbiamo analizzato la differenza nelle scelte universitarie, considerando i dati raggruppati sulle precedenti esperienze di coding. La Figura 2 mostra le intenzioni relative alle scelte accademiche future, disaggregate sulla base delle precedenti esperienze di coding, come emerse prima del campo.

Figura 2: Analisi scelte accademiche future nelle risposte al questionario pre-camp

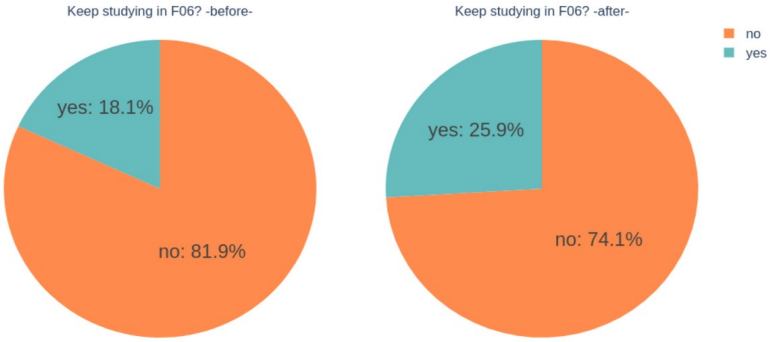


La Figura evidenzia una differenza significativa tra le ragazze che hanno già sperimentato la programmazione e le loro coetanee senza la stessa esperienza. Da un lato, una percentuale maggiore di ragazze con precedenti esperienze di coding esprime l'intenzione di non proseguire gli studi: ciò si può spiegare con il fatto che sono consapevoli di possedere una competenza, molto apprezzata e valorizzata nell'attuale mercato del lavoro, che consente loro di trovare facilmente lavoro. D'altro canto, tra le ragazze che intendono proseguire gli studi, è evidente che chi ha già sperimentato il coding ha molte più probabilità rispetto alle coetanee di proseguire gli studi nel settore F06 (45,5% vs 10,8%). Questo risultato è molto significativo in quanto evidenzia l'importanza di sperimentare la programmazione a scuola e come tale esperienza possa avere un impatto positivo e significativo sulla volontà degli studenti di scegliere l'informatica come campo di studio.

Valutiamo ora l'impatto del campo estivo sulla futura scelta delle ragazze di studiare in un campo informatico. A tal fine confrontiamo le risposte delle ragazze prima e dopo l'esperienza del campo estivo. La Figura 3 mette a confronto le risposte fornite prima (parte sinistra del grafico) e dopo (parte destra del grafico) il camp: i risultati mostrano che la percentuale di ragazze motivate a proseguire gli studi negli ambiti F06 aumenta dal 18,1% al 25,9%. Dai risultati emerge che l'esperienza del camp sembra influenzare positivamente la disponibilità delle ragazze a scegliere una disciplina informatica. Notiamo inoltre che la percentuale di studentesse intenzionate a continuare a studiare in ambito F06 è molto più alta della percentuale media a livello europeo. Infatti, come mostrano i dati Eurostat sull'iscrizione degli studenti all'istruzione terziaria, la percentuale di donne iscritte all'istruzione terziaria in Europa (UE28) nel settore ICT corrispondente all'1,7%. I nostri risultati possono essere spiegati dal fatto che il summer camp Ragazze Digitali attira probabilmente una percentuale più alta di ragazze interessate a proseguire gli studi nelle discipline ICT rispetto alla media delle ragazze del stessa età. Inoltre, va

considerato che le ragazze esprimono un'intenzione, mentre i dati Eurostat riportano la percentuale effettiva di studentesse iscritte al settore F06.

Figura 3: Analisi risultati pre- e post-camp



5 Conclusioni

In questo articolo abbiamo descritto in parte la nostra esperienza nell'ambito del progetto Ragazze Digitali, che è arrivato quest'anno alla sua decima edizione. Nel corso degli anni, abbiamo realizzato uno strumento di valutazione per misurare l'impatto delle attività sulle partecipanti, con l'obiettivo di comprendere meglio quali aspetti possono essere più efficaci per ridurre il divario di genere nelle discipline informatiche. Lo strumento è stato testato nel contesto delle ultime edizioni del summer camp Ragazze Digitali, permettendoci anche di paragonare le attività svoltesi in presenza e quelle svoltesi solo online durante gli anni della pandemia da COVID-19. Oltre a descrivere l'origine e l'evoluzione storica del progetto Ragazze Digitali, questo articolo ha presentato una descrizione della struttura dello strumento di valutazione e la presentazione di alcuni risultati derivanti dalla sua applicazione. Tali risultati ci hanno permesso di individuazione e discutere alcuni elementi ritenuti importanti nell'ottica di offrire alle studentesse un'esperienza che abbia un reale impatto sulle future scelte delle giovani partecipanti.

Bibliografia

Berg, T. and Sharpe, A. and Aitkin, E. (2018) Females in computing: Understanding stereotypes through collaborative picturing, Computers and Education, Volume 126, Pages 105-114. https://doi.org/10.1016/j.compedu.2018.07.007

Danoff, M. (2017). Gender and Computer Science at Harvard (Doctoral dissertation). http://nrs.harvard.edu/urn-3:HUL.InstRepos:38811504

Davaki, K. (2018) The underlying causes of the digital gender gap and possible solutions for enhanced digital inclusion of women and girls, European Parliament, Policy Dept. For Citizen's right and constitutional affairs, http://www.europarl.europa.eu/RegData/etudes/STUD/2018/604940/IPOL_STU(2018)604940_EN.pdf

Davies, J. J., & Hemingway, T. J. (2014). Guitar hero or zero? Fantasy, self-esteem, and deficient self-regulation in rhythm-based music video games. Journal of Media Psychology: Theories, Methods, and Applications, 26(4), 189. https://doi.org/10.1027/1864-1105/a000125

Davis, K. E. B., & Hardin, S. E. (2013). Making STEM fun: How to organize a STEM camp. Teaching Exceptional Children, 45(4), 60-67. https://doi.org/10.1177/00400599130450040

European Commission (2018) Women in the digital age, European Commission, DG Communications Networks, Content & Technology https://op.europa.eu/en/publication-detail/-/publication/84bd6dea-2351-11e8-ac73-01aa75ed71a1

European Eurostat Statistics (2017). Girls and women under-represented in ICT https://ec.europa.eu/eurostat/de/web/products-eurostat-news/-/EDN-20170426-1

European Statistics Eurostat (2019). ICT specialists are predominantly male

https://ec.europa.eu/eurostat/de/web/products-eurostat-news/-/DDN-20190513-1

Faenza, F. and Canali, C. and Colajanni, M. and Carbonaro, A. (2021) "The digital girls response to pandemic: Impacts of in presence and online extracurricular activities on girls future academic choices", Education Science Journal, Vol. 11. https://doi.org/10.3390/educsci11110715

Faenza, F. and Canali, C. and Carbonaro, A. (2021b) "ICT Extra-curricular Activities: The "Digital Girls" Case Study for the Development of Human Capital", Research and Innovation Forum, Rii Forum, Online Event. 10.1007/978-3-030-84311-3_18

Lewis, C. M., Yasuhara, K., & Anderson, R. E. (2011). Deciding to major in computer science: a grounded theory of students' self-assessment of ability. In *Proceedings of the seventh international workshop on Computing education research* (pp. 3-10). https://doi.org/10.1145/2016911.2016915

Master, A. (2021), Gender Stereotypes Influence Children's STEM Motivation. Child Dev Perspect, 15: 203-210. https://doi.org/10.1111/cdep.12424

Mohr-Schroeder, M. J., Jackson, C., Miller, M., Walcott, B., Little, D. L., Speler, L., ... & Schroeder, D. C. (2014). Developing Middle School Students' Interests in STEM via Summer Learning Experiences: S ee B lue STEM C amp. School Science and Mathematics, 114(6), 291-301. https://doi.org/10.1111/ssm.12079

Moss-Racusin, Corinne & Dovidio, & Brescoll, Victoria & Graham, Mark & Handelsman, Jo. (2012). Science Faculty's Subtle Gender Biases Favor Male Students. Proceedings of the National Academy of Sciences. 109. 16474-16479. 10.1073/pnas.1211286109.

Smeding, A. (2012). Women in science, technology, engineering, and mathematics (STEM): An investigation of their implicit gender stereotypes and stereotypes' connectedness to math performance. *Sex roles*, *67*(11), 617-629. https://doi.org/10.1007/s11199-012-0209-4

Spieler, B. and Oates-Indruchova,L. and Slany, W. (2020) Female Students in Computer Science Education: Understanding Stereotypes, Negative Impacts, and Positive Motivation, Journal of Women and Minorities in Science and Engineering, Volume 26, Issue 5, 2020, pp. 473-510. https://doi.org/10.48550/arXiv.1903.01190

Heilman, M. E. (2001). Description and prescription: How gender stereotypes prevent women's ascent up the organisational ladder. *Journal of Social Issues*, *57*(4), 657–674. https://doi.org/10.1111/0022-4537.00234

Wong, B., & Kemp, P. E. (2018). Technical boys and creative girls: the career aspirations of digitally skilled youths. Cambridge Journal of Education, 48(3), 301-316. https://doi.org/10.1080/0305764X.2017.1325443

Sperimentazione del metodo PRIMM per l'insegnamento della programmazione*

Giulia Peserico[1], Maria Serafini[1], Francesca Voltolini[1], Federica Picasso[2], Daniele Agostini[2], Francesca Fiore[2], Anna Serbati[2], e Alberto Montresor[2]

[1] Liceo Scientifico "Leonardo da Vinci", Trento, Italy
{nome.cognome}@liceodavincitn.it
[2] Università di Trento, Trento, Italy
{nome.cognome}@unitn.it

1 Introduzione

Il progetto "Algoritmicamente: dal problem solving all'informatica" ha svolto un'attività di ricerca-azione nell'a.s. 2022-23 con l'obiettivo di sperimentare la metodologia PRIMM [6, 7] per l'insegnamento della programmazione nella disciplina Informatica dell'opzione "Scienze Applicate" del Liceo Scientifico. Il progetto è stato proposto dal Liceo "Leonardo da Vinci" di Trento, in collaborazione con l'Università di Trento e l'Associazione Culturale Glow, ed è stato finanziato dalla Fondazione Caritro.

L'idea di base della metodologia PRIMM (Predict, Run, Investigate, Modify, Make) parte dall'osservazione che l'insegnamento "tradizionale" della programmazione richiede fin da subito la scrittura di codice da parte dello studente. Questo differisce dalle metodologie utilizzate nell'apprendimento delle lingue (compresa la madrelingua), dove l'attività di produzione è preceduta dalla lettura e dalla comprensione del testo.

Partendo dall'osservazione che il carico cognitivo necessario per arrivare alla scrittura del codice è molto elevato, la metodologia PRIMM inverte la sequenza tradizionale, partendo prima dalla comprensione del testo e solo successivamente arrivando alla scrittura di codice vero e proprio. In particolare, l'approccio proposto è suddiviso in cinque attività:

- **Predict**: gli studenti leggono un segmento di codice, scritto in Python, e fanno previsioni su cosa farà il codice quando viene eseguito;
- **Run**: gli studenti eseguono il codice proposto nell'attività precedente, confrontando il comportamento effettivo rispetto alla previsione;
- **Investigate**: agli studenti è richiesto di analizzare più in profondità il codice o sue varianti, utilizzando varie tipologie di esercizi, come per esempio la correzioni di bug, l'annotazione del codice, l'uso di Parson Puzzle, domande esplorative, etc;
- **Modify**: agli studenti è chiesto di modificare il codice, partendo da variazioni molto semplici e poi con modifiche sempre più complesse;
- **Make**: infine, gli studenti creano un programma totalmente nuovo, ispirandosi al codice visto in precedenza, ma realizzando una nuova funzionalità o risolvendo un problema diverso.

*I materiali sono liberamente scaricabili (previa registrazione email) a questo indirizzo: https://fablab.unitn.it/sperimentazione-primm/

Dal punto di vista teorico, PRIMM si colloca nell'intersezione fra un approccio più strutturato basato sulla scoperta guidata e sull'istruzione diretta, e un approccio più costruzionista [5] basato sulla scoperta pura e su problemi aperti. Seguendo i suggerimenti di Grover et al., le cinque fasi partono da attività fortemente guidate per arrivare ad attività completamente libere, riducendo mano a mano il livello di scaffolding [3].

Per quanto possibile, molte delle attività si svolgono favorendo il lavoro di gruppo e lo scambio fra pari, seguendo l'idea che l'articolazione verbale delle soluzioni favorisce in generale l'apprendimento, in particolar modo nell'ambito della programmazione dove la componente linguistica è fondamentale.

Nel resto dell'articolo descriveremo il contesto in cui abbiamo operato, le attività svolte e i risultati della sperimentazione. Concluderemo con una riflessione sulla validità dell'approccio e su possibili miglioramenti da applicare negli anni successivi.

2 Descrizione del contesto

Il Liceo Scientifico "Leonardo Da Vinci" è composto da 75 classi per un totale di 1.530 alunni. Ci sono 42 classi per l'indirizzo scienze applicate e 33 per l'ordinamentale, con una media di 20 studenti per classe. L'informatica viene insegnata nell'opzione Scienze Applicate per due ore a settimana, dalla prima alla quinta classe; tuttavia, è dalla classe seconda che si inizia ad affrontare la programmazione utilizzando il linguaggio Python.

La sperimentazione didattica è stata applicata in cinque classi seconde del liceo (102 studenti), su un totale di nove, sotto la guida delle professoresse Giulia Peserico, Maria Serafini e Francesca Voltolini. Nelle altre quattro classi seconde (86 studenti), che fungono da classi di controllo, sono state adottate le metodologie didattiche tradizionali. La Tabella 1 contiene la lista delle classi coinvolte e di controllo, l'insegnante responsabile, il numero di studenti con la suddivisione di genere.

Per quattro delle classi coinvolte ciascuna attività aveva una durata di 100 minuti , mentre per la quinta classe le attività si svolgevano in due sessioni di 50 minuti ciascuna.

Nome classe	**Docente**	**Orario**	**N° alunni**	**M**	**F**
Primm1	Docente1	100'	23	13	10
Primm2	Docente1	100'	20	8	12
Primm3	Docente1	100'	25	14	11
Primm4	Docente2	100'	19	15	4
Primm5	Docente3	50'+50'	18	13	5
Controllo1	Docente4	50'+50'	21	16	5
Controllo2	Docente4	50'+50'	20	11	9
Controllo3	Docente5	50'+50'	21	13	8
Controllo4	Docente6	50'+50'	22	13	9

Tabella 1: Lista delle classi coinvolte, con docenti, orario e dimensione.

3 Descrizione delle attività

Per le cinque classi coinvolte nella sperimentazione sono state realizzate delle schede didattiche per introdurre i concetti di base della programmazione in linguaggio Python (variabili, tipi, operatori booleani, selezione, iterazione, liste, stringhe, uso della libreria Turtle).

La maggior parte delle attività proposte era strutturata in lavoro a coppie, spesso organizzate per gruppi di livello, e comprendeva:

- 3-4 proposte di *Predict*, che proponevano un nuovo concetto del quale si chiedeva di prevedere il funzionamento, seguite ognuna da una fase di *Run*, nella quale gli studenti copiavano ed eseguivano il codice per verificare la propria predizione;
- alcune domande esplorative, nelle quali si chiedeva di spiegare a parole quanto capito del nuovo concetto, confrontandolo anche con concetti precedentemente appresi;
- un codice all'interno del quale trovare alcuni errori (sintassi, esecuzione, semantica);
- un Parson's Puzzle;
- alcuni esercizi di modifica del codice (*Modify*);
- alcuni esercizi di scrittura del codice (*Make*).

Tutte le lezioni sono state svolte in laboratorio di informatica e ciascuno studente aveva un pc (con interprete Thonny [2]) a disposizione per il lavoro. Al termine di ciascuna attività venivano assegnati dei compiti a casa per studiare e ripassare quanto appreso in classe e alcune attività di Modify e Make.

Dopo le prime lezioni, dove sono stati introdotti i concetti di base, le schede sono state strutturate riducendo la parte *Predict-Run*, togliendo i Parson's Puzzle e aumentando le attività di *Modify* e *Make* durante le attività in coppia. È stato inoltre deciso di introdurre all'inizio di ogni scheda una breve spiegazione dei nuovi concetti e una sessione collettiva per riprendere quanto introdotto, in quanto si è rilevato che il lavoro a casa non sempre era svolto con la dovuta attenzione.

Durante l'anno si sono svolte delle verifiche sia di tipo strutturato (con domande a risposta multipla e aperta, domande simili alle attività di *Predict*, rilevazione dell'errore), sia di tipo pratico che consistevano nello scrivere 3-4 programmi per la soluzione di altrettanti problemi.

In fondo all'articolo riportiamo, come esempio, la scheda utilizzata per la spiegazione del costrutto *if semplice (a una via)*. Nella prima fase della scheda sono stati proposti tre esercizi di *Predict*, chiamati *"Cosa fa il codice?"*, ognuno con le relative domande di approfondimento (Figure 1, 2, 3). É stato quindi inserito un semplice esercizio di ricerca degli errori (Figura 4) seguito da un Parson's Puzzle (Figura 5 e la sua soluzione 6). Infine, dopo aver schematizzato e rivisto con gli studenti i termini chiave (Figura 7), si è passati alle fasi *Modify* e *Make* (Figure 8 e 9).

4 Risultati

Per verificare l'efficacia del metodo PRIMM, sono stati raccolti tre tipi di dati sulle cinque classi sperimentali e le quattro classi di controllo:

- Il primo tipo di dati riguarda la motivazione dello studente a svolgere questo tipo di attività. Lo strumento utilizzato per raccogliere questo dato è stato l'Intrinsic Motivation Inventory (IMI) [4], tradotto ed adattato, utilizzando solo i moduli che si sono ritenuti utili per questa sperimentazione, ovvero: Interesse/appagamento, Sforzo/importanza, Percezione di competenza, Valore/Utilità, Pressione/Tensione.
- Il secondo tipo di dati riguarda l'apprendimento di conoscenze e abilità riguardanti la programmazione ed è stato rilevato tramite un test accuratamente preparato.

- Il terzo tipo di dato include la classe di appartenenza, il genere e i voti finali nelle varie discipline scolastiche.

I primi due tipi di dato sono stati rilevati secondo un disegno quasi-sperimentale Pre-Post, cioè sono stati rilevati prima dell'inizio della sperimentazione e dopo la fine per rilevare gli eventuali effetti e le differenze tra gruppo sperimentale e di controllo. Gli esiti dell'analisi, per la quale è stata utilizzata una regressione lineare con errori standard robusti per tenere in conto l'eterogeneità dei gruppi, evidenziano i seguenti risultati:

- Non esiste una correlazione statisticamente significativa tra l'appartenenza al gruppo sperimentale e un incremento del punteggio del test. Nonostante questo, gli incrementi dei punteggi dei test del gruppo sperimentale sono lievemente più alti di quelli di controllo.
- Esiste una correlazione statisticamente significativa tra l'appartenenza al gruppo sperimentale e un incremento del punteggio IMI Sforzo/importanza dell'attività. La metodologia PRIMM quindi potrebbe promuovere l'impegno degli studenti nelle attività e la comprensione della loro importanza.
- Esiste una correlazione fortemente significativa tra l'appartenenza al gruppo sperimentale e un incremento del punteggio IMI riguardante la Percezione di competenza. La metodologia PRIMM quindi potrebbe favorire la Percezione della propria competenza e la sicurezza degli studenti nei compiti di programmazione. Gli studenti potrebbero inoltre essere in grado di autovalutarsi in modo più preciso.
- Gli studenti con il miglioramento più marcato del punteggio del test finale rispetto a quello iniziale hanno ottenuto voti più alti in informatica.
- L'appartenenza al gruppo sperimentale e quindi la metodologia PRIMM potrebbe avere un impatto positivo sui voti di informatica; il risultato dell'analisi si avvicina molto alla significatività statistica.
- È stata inoltre esplorata la differenza di genere nei risultati. Nonostante non ci siano risultati conclusivi, l'analisi rileva, nel gruppo di controllo, uno sbilanciamento maggiore nell'interesse da parte degli studenti maschi rispetto alle studentesse, suggerendo che gli studenti maschi nel gruppo di controllo potrebbero essere stati più interessati dalle attività didattiche rispetto alle studentesse. Questo indicherebbe che la metodologia PRIMM potrebbe essere più inclusiva rispetto a quella tradizionale.

5 Riflessioni e conclusioni

La prima iterazione di questa ricerca-azione ha prodotto risultati promettenti, ma suggerisce anche la necessità di perfezionare il campione e alcune metodologie e procedure. Per esempio, alcuni item del test sembrano necessitare di una verifica perché non danno risultati consistenti con tutti gli altri indicatori. Inoltre, molte attività sono state svolte con gruppi suddivisi per livello dello studente. Questo potrebbe avere influito sul fatto che gli studenti con maggiore miglioramento nel punteggio del test abbiano ottenuto anche voti più alti in informatica. Nelle future iterazioni i gruppi verranno suddivisi in un'ottica di peer tutoring [1]. In generale, l'adozione del metodo PRIMM sembra promettente, e già da ora è chiaro che ha un notevole e positivo impatto sull'approccio degli studenti alla programmazione, in particolare sul proprio senso di competenza, sull'impegno e l'importanza attribuita alle attività proposte, ma sono necessarie ulteriori ricerche e ripetizioni per ottimizzare la sua efficacia.

Alla fine dell'anno abbiamo chiesto agli studenti dei commenti sulla modalità di lavoro PRIMM sperimentata. Un folto gruppo di studenti ha lamentato la mancanza di lezioni frontali "classiche" con spiegazione dell'insegnante prima di affrontare attività di laboratorio; possiamo ipotizzare che questo dipenda da un percorso scolastico (italiano) molto incentrato sulla lezione frontale e che gli studenti non abbiano dimestichezza con approcci innovativi. Alcuni studenti inoltre hanno affermato che il lavoro a coppie per livello non è stato produttivo e avrebbero desiderato essere a coppie con compagni più bravi, che li aiutassero maggiormente nel lavoro. Infine, un gran numero di studenti ha comunque rilevato che questa modalità di lavoro ha permesso loro di "andare al proprio passo" svolgendo le schede assegnate senza la fretta del lavoro collegiale, e potendole poi ripercorrere a casa. Tutto sommato il lavoro è stato apprezzato dagli studenti e verrà riproposto anche in futuro.

Riferimenti bibliografici

[1] F. J. Alegre Ansuategui and L. Moliner Miravet. Emotional and cognitive effects of peer tutoring among secondary school mathematics students. *International Journal of mathematical education in science and technology*, 48(8):1185–1205, 2017.

[2] Aivar Annamaa. Thonny: A Python IDE for learning programming. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '15, page 343. ACM, 2015.

[3] S. Grover, R. Pea, and S. Cooper. Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, 25(2):199–237, 2015.

[4] E. McAuley, T. Duncan, and V. V. Tammen. Psychometric properties of the intrinsic motivation inventory in a competitive sport setting: A confirmatory factor analysis. *Research quarterly for exercise and sport*, 60(1):48–58, 1989.

[5] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., USA, 1980.

[6] Sue Sentance, Jane Waite, and Maria Kallia. Teachers' experiences of using primm to teach programming in school. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, page 476–482, Minneapolis, USA, 2019. ACM.

[7] Sue Sentance, Jane Waite, and Maria Kallia. Teaching computer programming with primm: a sociocultural perspective. *Computer Science Education*, 29(2-3):136–176, 2019.

Lavoro in coppia

Di seguito c'è del codice scritto in Python:

```
print("Benvenuto nel mio programma di conversazione")
print()
print("Ti piace andare in bicicletta? Rispondi si o no")
risposta = input()
if risposta == "si":
    print("Molto bene! Ti terrai in forma.")
print("Ciao ciao.")
```

Secondo voi, che cosa appare sul monitor all'esecuzione?
Ora scrivete il codice in Thonny e controllate. Fa quello che vi aspettavate? Se non lo fa, descrivete nel dettaglio cosa accade in modo diverso.

Rispondete alle seguenti domande:

1. Cosa succede se si digita "Si" invece di "si" quando si esegue il programma? (provate e verificate)
2. Qual è la differenza tra = e == ? In questo programma vengono utilizzati entrambi?

Figura 1: Predict - 1

Lavoro in coppia

Di seguito c'è del codice scritto in Python:

```
print("Come è andata la verifica?")
voto = float(input("Scrivi il voto che hai preso: "))
if voto > 6:
    print("Bene. Hai preso di più della sufficienza!")

print("Alla prossima!")
```

Secondo voi, che cosa appare sul monitor all'esecuzione?
Ora scrivete il codice in Thonny e controllate. Fa quello che ti aspettavate? Se non lo fa, descrivete nel dettaglio cosa accade in modo diverso.

Rispondete alle seguenti domande:

3. Cosa accadrebbe se usassimo il controllo `if voto >= 6` ?
4. Cosa accadrebbe se usassimo il controllo `if voto < 6` ?
5. Cosa accadrebbe se usassimo il controllo `if voto != 6` ?

Figura 2: Predict - 2

Lavoro in coppia

Di seguito c'è del codice scritto in Python:

```
print("Come è andata la verifica?")
voto = float(input("Scrivi il voto che hai preso: "))
if voto > 6:
    print("Bene. Hai preso più della sufficienza!")
    risposta = input("Sei contento del voto? Rispondi si o no")
    if risposta=="si":
        print("Ottimo! Anch'io sono contento per te!")
```

Secondo voi, che cosa appare sul monitor all'esecuzione?
Ora scrivete il codice in Thonny e controllate. Fa quello che ti aspettavate? Se non lo fa, descrivete nel dettaglio cosa accade in modo diverso.

Rispondete alle seguenti domande

6. Avete notato che il secondo `if` è spostato più internamente? Cosa vuol dire?
7. Se noi avessimo messo il secondo `if` allineato al primo, quale sarebbe stato l'output? (prova a modificare il codice e vedere su Thonny il risultato.)

Figura 3: Predict - 3

Lavoro in coppia

In questo programma ci sono **2 errori**, riesci ad individuarli? Evidenziali con un colore. *(Se non riesci a trovare alcuni errori puoi aiutarti copiando il codice in Thonny e verificandone il funzionamento)*
print("Ciao! sono un po' curioso.") risposta = input("Ti piace la pizza?") if risposta > "si"   print("Bene. Allora possiamo fare una serata pizza e cinema.") print("A presto.")

Figura 4: Trova gli errori

```
if (somma >= 10):
print("la somma e' compresa tra 10 e 15")
addendo1 = int(input("Inserisci il primo addendo: "))
somma = addendo1 + addendo2
print("la somma e' maggiore o uguale a 10")
addendo2 = int(input("Inserisci il secondo addendo: "))
print("Ciao! Facciamo il controllo della somma.")
if (somma <= 15):
print("Ciao, alla prossima!")
```

Figura 5: Parson's Puzzle

```
print("Ciao! Facciamo il controllo della somma.")
addendo1 = int(input("Inserisci il primo addendo: "))
addendo2 = int(input("Inserisci il secondo addendo: "))
somma = addendo1 + addendo2
if (somma >= 10):
    print("la somma e' maggiore o uguale a 10")
    if (somma <= 15):
        print("la somma e' compresa tra 10 e 15")
print("Ciao, alla prossima!")
```

Figura 6: Parson's Puzzle - soluzione

TERMINE	SIGNIFICATO	PYTHON
Operatore relazionale	Un simbolo che serve per fare un confronto tra due dati.	>, >=, <, <=, ==, != sono gli operatori relazionali di Python
Condizione	Il test che inseriamo nel costrutto *if*. Si costruisce inserendo un *operatore relazionale* tra due dati. Il risultato della condizione deve essere vero (**True**) o falso (**False**)	risposta == "sì" a > 5 b != 7 10 >= x b == a
Selezione	Quando nel codice c'è un punto in cui viene effettuata una scelta e viene utilizzato il costrutto *if* per creare percorsi alternativi	`if a > 10:` `  print("ciao")`

Indentare	Scrivere una o più istruzioni "rientrate" (di 4 spazi) rispetto alle istruzioni precedenti/successive	`if a > 10:`
		`  print("ciao")`
		L'istruzione print è **indentata** rispetto a if

Annidare	Scrivere una selezione dentro un'altra selezione	`if numero > 5:`
		`  if numero < 10:`
		`  print("numero tra 5 e 10")`

Figura 7: Termini chiave

1. Cosa stampa il seguente codice?

```
numero = int(input("inserisci un numero"))
if numero%3==0:
    print("il numero va bene")
```

   Ricopia il codice per accertarti di aver risposto correttamente e modificalo in modo tale che venga controllato se un numero è divisibile per 5.

2. Modifica il codice dell'esercizio precedente per verificare se dati due numeri (interi) inseriti dall'utente, il primo è multiplo del secondo.

3. Cosa stampa il seguente codice?

```
lato1  = float(input("inserisci la misura del primo lato"))
lato2  = float(input("inserisci la misura del secondo lato"))
if lato1==lato2:
    print("il triangolo è isoscele")
```

   Ricopia il codice per accertarti di aver risposto correttamente e modificalo in modo tale che venga verificato se il triangolo è equilatero.

4. Cosa stampa il seguente codice?

```
prezzo  = float(input("inserisci il prezzo del biglietto"))
numero  = int(input("inserisci il numero di biglietti acquistati"))
if numero > 20:
    numero = numero-1
totale = numero * prezzo
print("il totale da pagare è", totale)
```

   Ricopia il codice per accertarti di aver risposto correttamente e modificalo in modo tale che venga regalato un biglietto omaggio **ogni** 20 biglietti (se si acquistano 30 biglietti se ne pagano 29, acquistandone 50 se ne pagano 48, ecc).

Figura 8: Modify

5. Calcolare il totale speso per acquistare delle pesche sapendo che se si superano i 10 euro si ha il 20% di sconto.

6. Calcolare il costo della bolletta di consumo del gas (espresso in metri cubi) ottenuto dalla somma delle seguenti voci:
   - quota fissa pari a 20€
   - 0,575€ al metro cubo per i primi 500 metri cubi
   - 0,783€ al metro cubo per ogni metro cubo eccedente i primi 500

7. Comperando un'automobile, se spendo più di 20.000 € ho uno sconto del 10%. Indipendentemente dal prezzo, se pago in contanti uno sconto di 1.000. Visualizzare quanto si spende.

Figura 9: Make

Apprendimento della programmazione guidato dalla necessità: il Necessity Learning Design

Marco Sbaraglia, Michael Lodi, Simone Martini

Dipartimento di Informatica - Scienza e Ingegneria
Alma Mater Studiorum - Università di Bologna
Bologna, Italia
{marco.sbaraglia, michael.lodi, simone.martini}@unibo.it

Abstract

Descriviamo una metodologia didattica originale, il Necessity Learning Design, in cui gli studenti cercano di risolvere un problema per il quale serve un concetto di programmazione che ancora non conoscono, sentendone quindi la necessità. Le fasi di spiegazione e di soluzione seguono. Descriviamo i primi risultati di una sperimentazione di questa metodologia in un istituto tecnico informatico.

1 Il momento giusto per insegnare

Alcune osservazioni di Boaler [1, pp. 66-69] sono illuminanti: vengono posti problemi di matematica (es. recintare l'area maggiore possibile con 36 pezzi di staccionata, trovare il volume di un limone) a studenti che ancora non hanno tutti gli strumenti necessari a risolverli (es. trigonometria, integrali). Solo dopo che gli studenti hanno sperimentato, fatto ipotesi, tentato strategie di soluzione, gli insegnanti hanno spiegato i concetti necessari, ricevendo grande interesse e attenzione (Boaler racconta di un ragazzino che spiega euforico al suo gruppo "una cosa molto *cool* che ha imparato dall'insegnante": come usare la funzione trigonometrica seno). In altre classi Boaler aveva osservato lezioni tradizionali sugli stessi concetti, bollate dagli studenti come noiose.

In didattica della matematica e delle scienze si parla in questi casi di approcci dove il Problem Solving precede l'Insegnamento diretto (PS-I) [10]. Il titolo di uno dei lavori su questo tema è evocativo: *A time for telling* [15], che potremmo tradurre come "Il momento giusto di spiegare". Sperimentare difficoltà e fallire può stimolare la motivazione e preparare gli studenti all'apprendimento (es. tramite attivazione della loro conoscenza pregressa, favorendo una presa di consapevolezza delle lacune o il riconoscimento delle caratteristiche fondamentali di un problema) [10, 11]. L'approccio più conosciuto e sperimentato è quello del *Productive failure* [5, 6], in cui agli studenti viene chiesto di risolvere un problema complesso, mal strutturato, con dettagli inutili o sovrabbondanti, che ostacolano una soluzione "canonica". Segue una fase di insegnamento diretto, basata sul confronto delle soluzioni sub-ottime realizzate dagli studenti.

La maggior parte della ricerca educativa sostiene che le metodologie attive – specialmente quando gli studenti esplorano e costruiscono attivamente la conoscenza – favoriscono l'apprendimento [3, 12]. D'altra parte, così come in altre discipline, anche nell'introduzione alla programmazione è difficile far scoprire specifici concetti tecnici tramite l'esplorazione libera [4]. Di conseguenza, nei corsi introduttivi, resta comune l'insegnamento diretto degli elementi del linguaggio, seguito dalla loro applicazione in esercizi di programmazione. L'insegnamento diretto non sembra però ideale per i novizi: potrebbero annoiarsi e non cogliere in concreto l'utilità dei concetti [2], con conseguente bassa motivazione e scarsi risultati.

Abbiamo applicato le idee del PS-I al contesto specifico dell'introduzione alla programmazione, formalizzando l'idea generale del "meccanismo di necessità" e progettando una metodologia didattica specifica per l'Informatica che abbiamo chiamato *Necessity Learning Design* (NLD) [14].

Riassumiamo qui gli aspetti fondamentali del nostro modello [14], descrivendo poi i primi risultati di una sua sperimentazione alla scuola secondaria.

2 La necessità

2.1 Il meccanismo di necessità

Il *meccanismo di necessità* consiste nell'assegnare un problema progettato in modo che sia percepito come familiare perché simile (nelle parole, forma, richiesta) a molti già svolti in precedenza e quindi padroneggiati dagli studenti. Gli studenti capiscono facilmente il problema e le sue richieste, e credono di saperlo risolvere. Tuttavia, il problema è costruito in modo che manchi esattamente un solo elemento per poterlo risolvere: il concetto che si vuole insegnare (*concetto target*). Si tratta di un meccanismo generale, utilizzabile e adattabile (come abbiamo fatto ad esempio per la crittografia [7, 8, 9]) ad altri contesti o discipline.

Per fare un esempio, dopo che gli studenti hanno appreso costrutti per la scansione di sequenze (`foreach`) e l'iterazione determinata (`for` con indice), si può assegnare un problema risolvibile solo[1] con l'iterazione indeterminata (`while`), che essi ancora non conoscono, ad esempio:

> contare dopo quante generazioni di un numero random esce il numero 42.

2.2 Il Necessity Learning Design

Una sequenza di apprendimento progettata tramite il Necessity Learning Desing, la nostra metodologia didattica specifica per l'introduzione della programmazione che fa uso del meccanismo di necessità, è costituita da tre fasi successive: P!S, I, PS (Figura 1).

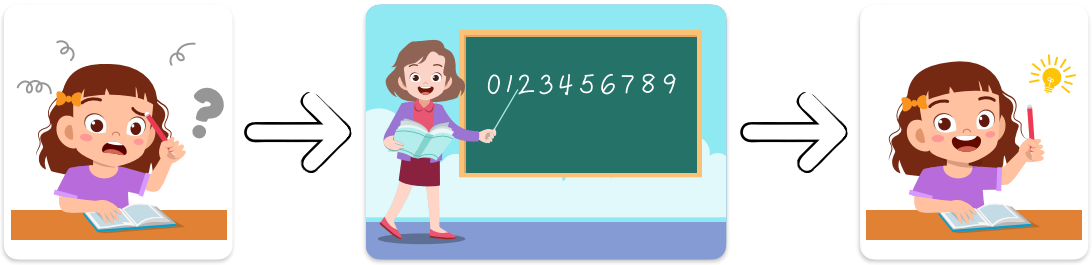


Figura 1: Le fasi di una sequenza di apprendimento progettata con il Necessity Learning Design.

Le immagini sono state acquistate con licenza *Shutterstock Standard Image License* (`https://www.shutterstock.com/license`).

P!S Si assegna un semplice esercizio, apparentemente familiare, che lo studente non sa risolvere senza il concetto target. Poiché il contesto e la difficoltà del problema sono simili a quelli che lo studente ha già affrontato, sentirà la necessità di qualcosa che ancora non conosce, ma di cui può intuire caratteristiche

[1] Pensiamo a linguaggi come Python in cui le iterazioni con il `for` sono sostanzialmente realmente *determinate* - solo con stratagemmi non ovvi per i novizi si può sfruttare il `for` per iterazioni indeterminate. Diversa la situazione per linguaggi quali il C, in cui un `for` è di fatto un'abbreviazione per un `while`. Ovviamente supponiamo anche che gli studenti non conoscano ancora la ricorsione.

e utilità. Poiché i programmi sono oggetti eseguibili, lo studente potrà verificare autonomamente la correttezza del proprio tentativo di soluzione.

I L'insegnante spiega direttamente il concetto target, fornisce esempi semplici e introduttivi, ma senza riferirsi all'esercizio della fase P!S, per non sprecare il potenziale di apprendimento insito nella necessità sperimentata in precedenza dallo studente.

PS Lo studente soddisfa la propria necessità applicando il concetto target alla risoluzione dell'esercizio iniziale. Avendolo già esplorato ampiamente, può giovarsi dei ragionamenti fatti durante i tentativi iniziali. Può inoltre comprendere più facilmente l'importanza del concetto target e la sua applicazione in uno specifico contesto.

2.2.1 Quando usare il NLD

Questa metodologia è stata ideata per introdurre (fare "bootstrap") in modo motivante e significativo nuovi concetti fondamentali (es. nuovi costrutti della programmazione). Non è adatta ad essere utilizzata in tutte le fasi dell'apprendimento, in cui vanno utilizzate altre metodologie per favorire lo sviluppo della padronanza.

Riteniamo utile usare NLD nei momenti in cui cambia il livello di astraizone all'interno di un linguaggio, ad esempio quando si introduce un costrutto più astratto (es. array invece di variabili scollegate) o uno meno astratto (es. `while` dopo il `for`).

2.2.2 Esempio: gli array

Una possibile sequenza P!S-I-PS, descritta in [14, §4.4.3], viene preceduta da una fase di avvicinamento in cui gli studenti, usando variabili semplici, ciclo `for` e numeri casuali, maturano padronanza su esercizi del tipo:

- contare teste e croci in 10.000 lanci casuali di una moneta;
- contare quante volte esce ciascuna faccia in 1 milione di lanci di un dado a 6 facce.

Nella fase P!S, poi, viene chiesto agli studenti di

- contare quante volte esce ciascun numero del Lotto (da 0 a 89) in 1 milione di estrazioni.

Sebbene questo esercizio abbia una soluzione sub-ottima che fa uso di 90 variabili (es. `estratto1`, ... `estratto90`) mediante un'altrettanto complessa struttura condizionale, gli studenti dovrebbero sentire la necessità di "una struttura di dati per raccogliere la frequenza con cui esce ogni numero, accedendo ai valori di tale struttura a runtime sulla base del numero estratto"[2].

Nella fase I, l'insegnante introduce gli array e la loro scansione con il `for` con indice.

Nella fase di PS, gli studenti possono risolvere il problema utilizzando un array di 90 interi, in cui ogni cella conterà la frequenza di estrazioni del numero corrispondente all'indice di tale cella nell'array.

[2] Ovviamente ci aspettiamo che gli studenti abbiano un'idea molto meno formale e intuitiva, ma comunque coerente, di ciò di cui hanno bisogno, come vedremo nella prossima sezione.

3 La sperimentazione

Abbiamo sperimentato la sequenza per introdurre gli array (vedi 2.2.2) in un Istituto Tecnico Tecnologico bolognese, articolazione Informatica. Il quasi-esperimento ha coinvolto due classi terze (stessi insegnante e ITP) di studenti alle prime armi con la programmazione. La classe sperimentale ha seguito la sequenza P!S-I-PS di NLD, la classe di controllo una classica sequenza I-PS.

Riportiamo qui le analisi qualitative delle esperienze di insegnanti, studenti (classe sperimentale) e ricercatori. Per altre analisi più dettagliate si veda [13, capitoli 9 e 10]. Ulteriori analisi, qualitative e quantitative, sono in corso.

La classe sperimentale (più fragile della classe di controllo) mostrava storicamente scarsi interesse e risultati e ha mantenuto un atteggiamento disinteressato e passivo durante gli esercizi di avvicinamento alla fase P!S (vedi 2.2.2). Gli studenti si sono animati nella fase P!S (problema del Lotto), formando spontaneamente gruppi per cercare soluzioni. Tutti si sono impegnati sul problema, chi prendendolo come sfida, chi cercando online, chi procedendo in modo "cieco" cercando – senza riuscirci per le difficoltà sintattiche di questa strategia – di risolverlo senza usare gli array. Tutti gli studenti hanno sentito la *necessità* di qualcosa in più (es., cambiare dinamicamente il nome di una variabile); alcuni hanno provato frustrazione, in particolare quelli abituati a buoni risultati.

Nella fase I l'insegnante ha introdotto gli array in modo generale e gli studenti sono rimasti focalizzati. Alcuni, però, cercando di applicare immediatamente gli array al problema del Lotto, hanno perso passaggi chiave della spiegazione. Un setting più adatto alla lezione frontale (lontano dai computer) potrebbe favorire l'attenzione nella fase I.

Anche nella fase PS gli studenti sono rimasti focalizzati sul problema da risolvere (comportamento insolito, a detta dell'insegnante). Tuttavia, la maggioranza non è riuscita a risolverlo pur usando gli array. Alcuni, come detto, avevano perso dettagli fondamentali della spiegazione, altri sono stati ostacolati dall'ulteriore difficoltà di usare il numero estratto come indice dell'array. Questo sottolinea che il design di una sequenza NLD, destinata alla programmazione introduttiva, deve prevedere un solo nuovo concetto/costrutto/pattern.

Anche se da confermare con analisi quantitative, la classe sperimentale ha lievemente migliorato i propri risultati di apprendimento rispetto agli argomenti precedenti.

In generale, la maggioranza degli studenti ha apprezzato la sfida e ne ha compreso il potenziale educativo. Alcuni, tuttavia, si sono sentiti ingannati o troppo frustrati.

È fondamentale che l'insegnante gestisca bene i tempi, evitando che la fase P!S duri troppo, sfruttando invece il "momento giusto per insegnare". Inoltre, per evitare che gli studenti "mangino la foglia", è bene ricorrere a NLD poche volte, solo per introdurre nuovi concetti importanti o difficili. Una delle maggiori difficoltà per l'insegnante è quella di "mantenere il segreto" del concetto target (prima e durante la fase P!S), senza però frustrare eventuali studenti che dovessero già conoscerlo o scoprirlo.

Fondi

Il lavoro di M. Lodi è supporrato dallo Spoke 1 "FutureHPC & BigData" del Centro Nazionale di Ricerca in "High Performance Computing, Big Data and Quantum Computing" (ICSC) finanziato da MUR Missione 4 Componente 2 Investimento 1.4: Potenziamento strutture di ricerca e creazione di campioni nazionali di R&S (M4C2-19) - Next Generation EU (NGEU).

Bibliografia

[1] Jo Boaler. *Mathematical mindsets: unleashing students' potential through creative math, inspiring messages, and innovative teaching*. San Francisco, CA: Jossey-Bass & Pfeiffer Imprints, 2016. ISBN: 9781118418277 9781118415535.

[2] Michael E. Caspersen. «Teaching Programming». In: *Computer science education: perspectives on teaching and learning in school*. A cura di Sue Sentance, Erik Barendsen e Carsten Schulte. London-New York: Bloomsbury Academic, 2018, pp. 109–130. ISBN: 9781350057111 9781350057104.

[3] Scott Freeman et al. «Active learning increases student performance in science, engineering, and mathematics». In: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8410–8415. DOI: 10.1073/pnas.1319030111.

[4] Mark Guzdial. «Balancing Teaching CS Efficiently with Motivating Students». In: *Commun. ACM* 60.6 (mag. 2017), pp. 10–11. ISSN: 0001-0782. DOI: 10.1145/3077227.

[5] Manu Kapur. «Examining Productive Failure, Productive Success, Unproductive Failure, and Unproductive Success in Learning». In: *Educational Psychologist* 51.2 (2016), pp. 289–299. DOI: 10.1080/00461520.2016.1155457.

[6] Manu Kapur e Katerine Bielaczyc. «Designing for Productive Failure». In: *Journal of the Learning Sciences* 21.1 (2012), pp. 45–83. DOI: 10.1080/10508406.2011.591717.

[7] Michael Lodi, Simone Martini e Marco Sbaraglia. «Crittografia a blocchi al Liceo Matematico». In: *Cryptography and Coding Theory Conference 2021*. Vol. 3. Collectio Ciphrarum. Roma, Italy: Aracne, 2022, pp. 103–104. DOI: 10.53136/979125994981340. URL: https://www.aracneeditrice.eu/it/pubblicazioni/estratti/10.53136/979125994981340-crittografia-a-blocchi-al-liceo-matematico-estratto.html.

[8] Michael Lodi, Simone Martini e Marco Sbaraglia. «Programmare per imparare la crittografia al Liceo Matematico». Italian. In: *Rendiconti del Seminario Matematico* 80.2 (2022). URL: http://www.seminariomatematico.polito.it/rendiconti/80-2/Lodi.pdf.

[9] Michael Lodi, Marco Sbaraglia e Simone Martini. «Cryptography in Grade 10: Core Ideas with Snap! and Unplugged». In: *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1*. ITiCSE '22. Dublin, Ireland: Association for Computing Machinery (ACM), 2022. ISBN: 978-1-4503-9201-3/22/07. DOI: 10.1145/3502718.3524767.

[10] Katharina Loibl, Ido Roll e Nikol Rummel. «Towards a Theory of When and How Problem Solving Followed by Instruction Supports Learning». English. In: *Educational Psychology Review* 29.4 (dic. 2017), pp. 693–715. ISSN: 1040-726X, 1573-336X. DOI: 10.1007/s10648-016-9379-x.

[11] Katharina Loibl e Nikol Rummel. «The impact of guidance during problem-solving prior to instruction on students' inventions and learning outcomes». In: *Instructional Science* 42 (mag. 2014). DOI: 10.1007/s11251-013-9282-5.

[12] Michael Prince. «Does Active Learning Work? A Review of the Research». In: *Journal of Engineering Education* 93.3 (lug. 2004), pp. 223–231. DOI: 10.1002/j.2168-9830.2004.tb00809.x.

[13] Marco Sbaraglia. «Teaching Informatics to Novices: Big Ideas and The Necessity of Optimal Guidance». Tesi di Dottorato. Alma Mater Studiorun - Università di Bologna, 2023. URL: http://amsdottorato.unibo.it/10914/.

[14] Marco Sbaraglia, Michael Lodi e Simone Martini. ≪A Necessity-Driven Ride on the Abstraction Rollercoaster of CS1 Programming≫. In: *Informatics in Education* 20.4 (dic. 2021), pp. 641–682. ISSN: 1648-5831. DOI: 10.15388/infedu.2021.28.

[15] Daniel L. Schwartz e John D. Bransford. ≪A Time For Telling≫. In: *Cognition and Instruction* 16.4 (dic. 1998), pp. 475–5223. DOI: 10.1207/s1532690xci1604_4.

ALTRI CONTRIBUTI

Lezioni di AI ed etica nella Scuola Superiore

Giuseppe Corrente
ITIS Avogadro Serale, Torino
gcorrente@itisavogadro.it

Abstract

L'AI rappresenta un campo in rapida evoluzione, i suoi recenti progressi a volte hanno aspetti apparentemente sorprendenti, soprattutto se guardiamo quelli fatti in questi ultimi anni nel campo del deep learning, e in particolare dell'AI generativa. Nel seguito abbiamo schematizzato in alcune lezioni una serie di approfondimenti e spunti di riflessione intercorsi durante le ore delle materie di indirizzo (ITIS articolazione Informatica) e/o di Educazione Civica.

1 Introduzione

Il deep learning è una tecnica di intelligenza artificiale che si ispira direttamente al funzionamento del cervello, piuttosto che alla scrittura esplicita di algoritmi in cui si cerca di rappresentare il mondo circostante e i ragionamenti che su di esso si possono fare. Questo comporta un modello esemplificativo di alcuni strati di neuroni che comunicano tra di loro, fino a produrre un output che è funzione dell'input ricevuto e dei pesi dei link che collegano tra loro i neuroni di ogni strato con quelli dello strato immediatamente successivo. L'apprendimento avviene per esempi, ad esempio di coppie di input/output che la rete neuronale deve collegare, ed i pesi dei collegamenti si adattano automaticamente man mano che avviene l'apprendimento.

Ovviamente un algoritmo di questa tipologia è molto meno trasparente di un algoritmo di tipo classico, in quanto non abbiamo mai una spiegazione del perchè un certo input viene collegato alla data risposta, la conoscenza è distribuita diffusamente in tutti i milioni di pesi che costituiscono i collegamenti della rete, che sono in un certo senso impossibili da decodificare. Questa modalità di funzionamento comporta prestazioni eccellenti, e comportamenti che possono apparire come intelligenti, ma anche indubbiamente dei rischi e degli aspetti etici nell'eventuale mal utilizzo di questi sistemi. Entrambe le facce della medaglia meritano un approfondimento, come schematizzato nel seguito dell'articolo.

2 Lezione 1. Differenze tra programmi simbolici e subsimbolici o connessionisti

Si spiega il significato di deep learning e reti neurali, mostrando una simulazione di una rete neuronale [1].

Ovviamente è un esempio giocattolo, fatto di poche decine di neuroni e collegamenti pesati, ma si può giocare interattivamente con questa simulazione, mostrando alla classe come la velocità di apprendimento sale notevolmente aumentando anche di poco la numerosità degli strati e dei neuroni, come variano i pesi dinamicamente durante l'apprendimento, e partendo da questo esempio spiegare sistemi più complessi dello stesso tipo, i loro punti di forza e i loro punti deboli[2]. Si può qui dare una panoramica della tassonomia dei sistemi riguardanti l'AI, così come parlare del test di Turing, o delle differenze tra AI forte e debole[5].

 ISBN: 978-88-6629-075-9

3 Lezione 2. Dibattito in corso sull'appello firmato da Elon Musk ed altri per una sospensione degli esperimenti sull'AI

L'appello è sicuramente allarmista, propone una pausa di sei mesi nella ricerca riguardante l'AI e in particolare i sistemi di tipo LLM (Large Language Model)[3], quello cui appartiene chatGPT, per intenderci, fornendo delle motivazioni vaghe, fosche previsioni e scenari alcuni dei quali evocano quelli del film Terminator, nel caso non si attuasse immediatamente questa sospensione. A questi scenari catastrofici, si oppone il buon senso di altri eminenti studiosi del settore[3], che definiscono oscurantista la proposta di bloccare la ricerca in un settore così promettente, anche se ammettono la validità di alcune contromisure e regolamentazioni che vanno necessariamente prese in considerazione.

Si paragona questo campo a quello nel secolo scorso delle auto: esse non sono state progettate inizialmente già dotate di cinture di sicurezza ed airbag, ma man mano che si diffondevano è stato necessario dotarle di questi ed altri sistemi di sicurezza.

4 Lezione 3. AI, problemi etici, possibili errori dovuti a bias dei dati intenzionali o meno, manipolazione e controllo delle masse

Si accennano qui ad una serie di scenari alcuni dei quali già attuali e che rappresentano il lato negativo dell'AI, visto questa volta in modo non allarmista e catastrofico, ma realista. Le applicazioni che destano più preoccupazioni sono quelle legate alla giustizia, alla selezione del personale, ai sistemi di cessione del credito, alla videosorveglianza e al riconoscimento facciale. Il problema è un cattivo uso intenzionale da parte ad esempio di governi di stampo dittatoriale, come anche di errori dovuti ad un effetto di amplificazione dei pregiudizi, ad esempio razziali e di genere, introdotti nei dati iniziali su cui si è addestrata la macchina.

Queste problematiche sono poi amplificate dalla mancanza di trasparenza dei meccanismi interni tipici del funzionamento delle reti neuronali su cui si basano questi sistemi. Le proposte pubblicitarie o messaggi di altro tipo tagliate su misura per ogni individuo è un altro esempio già attuale di come queste tecnologie siano pervasive e di quanto possano diventare insidiose. Da qui possono seguire ampie riflessioni da dibattere in classe su molteplici temi.

5 Lezione 4. Il futuro dei lavori e l'AI.

Vengono presentati tramite filmati[2], o demo, i seguenti scenari:

- Scrittura dietro richiesta di una breve favola, racconto o sceneggiatura da parte di un AI generativa (tipo chatGPT)
- Disegni, grafici, illustrazioni o quadri generati da un AI generativa (tipo DALL-E) a partire da una breve descrizione
- Generazione di codice funzionante di un semplice incipit per un videogioco interattivo a partire da una breve descrizione testuale
- Diagnosi automatica precoce di tumori da scansione di immagini mediche

Segue discussione su come i lavori saranno trasformati da queste tecnologie e come non lasciarsi prendere alla sprovvista dai cambiamenti a venire. Argomento correlato di discussione è come la scuola deve preparare gli studenti a questi scenari, coadiuvando il buon uso di questi

sistemi come strumenti, e scoraggiando gli studenti ad abituarsi ad un loro uso troppo passivo nello svolgimento ad esempio dei compiti, che nel lungo periodo non porterebbe loro alcun vantaggio.

Si può far notare agli studenti come anche nell'informatica tradizionale ad esempio un ingegnere civile usa come supporto i programmi di calcolo del cemento armato, ma da studente deve imparare lo stesso questi calcoli in dettaglio per essere davvero un buon ingegnere.

6 Conclusioni

L' AI nel suo stato attuale e nelle sue prossime evoluzioni investirà le nostre vite come un treno in corsa, impattandone molteplici aspetti. E' necessario che la scuola[4] prepari i cittadini, i lavoratori nonchè gli uomini di domani a sfruttare queste tecnologie per il benessere ed il progresso della civiltà, allo stesso tempo esercitando il senso critico e promuovendone la consapevolezza dei rischi che sono inevitabilmente l'altra faccia della medaglia dell'AI e delle sue molteplici e pervasive applicazioni.

Le lezioni sono state tenute nelle classi terza, quarta e quinta, articolazione Informatica, dell' Itis Avogadro Serale di Torino durante il normale corso dell' AS 2022-2023 intervallandole al normale svolgimento dei soliti programmi; non è stato fatto un vero e proprio test iniziale e finale sugli argomenti trattati, anche perché le lezioni non erano state pianificate in precedenza, ma il feeling generale è che le classi hanno risposto bene ai contenuti illustrati e relative discussioni, mostrando interesse e curiosità, probabilmente in quinta anche sollecitati dalla preoccupazione che una delle tracce della prima prova dell'Esame di Stato potesse riguardare simili problematiche.

E' difficile programmare una estensione di queste lezioni con altre più specifiche senza addentrarsi in problematiche troppo complesse o pesanti per i ragazzi, però si potrebbero progettare dei seminari fuori dall'orario solito ad esempio sulle seguenti tematiche:

- Machine Learning Coding and Programming: si potrebbe avviare una sessione di esercitazioni utilizzando dei giochi esempi o dei semplici esempi in Python per 'toccare con mano' cosa c'è dietro l'AI, lato programmazione e funzionamento base; ad esempio realizzare un sistema che basandosi su dei dati storici contenenti il prezzo degli appartamenti in varie zone di una certa metropoli correlati oltre che a caratteristiche proprie dell'appartamento anche a dati tipo indici di criminalità nella zona, coefficienti di inquinamento ed altri, stima il valore economico degli appartamenti della città.
- AI e Spiritualità: la mente umana è dunque simulabile completamente tramite un sistema di AI casomai molto più complesso di quelli attuali ? Oppure si possono ottenere comportamenti intelligenti artificiali, ma entità come la coscienza ed il libero arbitrio non saranno mai ottenibili da una macchina ? Testi[6,7] come quelli di Kurzweil e Faggin o loro brani scelti possono servire ad addentrarsi in queste questioni e nelle loro conseguenze, confrontando tra di loro le due ipotesi opposte e relativi scenari in un contesto multidisciplinare.

Bibliografia

1. https://playground.tensorflow.org

2. Noi e i Robot - L'impatto dell'Intelligenza Artificiale su presente e futuro dell'umanità, 2013, Fondazione Giuseppe e Adele Baracchi, https://www.youtube.com/watch?v=vqUk9wXzt6o

3. Sospendiamo lo sviluppo dell'AI?, 2013, Liberi Oltre le Illusioni - STEM, https://www.youtube.com/watch?v=ej5lEP1bmdQ

4. P.L. Pisa, 2013, https://www.repubblica.it/tecnologia/2023/05/19/news/chatgpt_a_scuola-400828799/
5.S.J.Russell, P.Norvig. Intelligenza Artificiale. Un approccio moderno. Pearson, 2021.
6. R.Kurzweil. La singolarità è vicina. Apogeo Education, 2008.
7. F.Faggin. Irriducibile. La coscienza, la vita, i computer e la nostra natura. Mondadori, 2023.

Ottimizzare l'apprendimento collaborativo con il workshop di Moodle

Giuliana Barberis[1]
[1] Liceo Scientifico M. Curie di Pinerolo
giuliana.barberis@gmail.com

Abstract

Nell'insegnamento dell'informatica è quasi automatico l'utilizzo di un LMS come Moodle. Con Moodle gli studenti possono avere una traccia delle lezioni svolte fino a quel momento e l'insegnante ha a disposizione una serie di attività con le quali arricchire la propria offerta formativa.

Una delle attività di Moodle, il Workshop, offre un ambiente organizzato e automatico per la gestione del processo di apprendimento e valutazione fra pari, semplificando le operazioni di creazione e utilizzo di una griglia di valutazione, di assegnazione ai "pari" dell'esercizio da valutare, di calcolo della valutazione quantitativa.

1 Introduzione

Nel processo di insegnamento è importante suddividere l'argomento complessivo che ci si propone di trattare in tanti sotto-obiettivi di apprendimento chiari e raggiungibili.

Ciascun sotto-obiettivo deve essere compreso dagli studenti, anche per quanto riguarda i descrittori dei risultati attesi, che devono pertanto essere comunicati prima di cominciare la spiegazione, in seguito dovrà essere programmato un intervento valutativo, che chiuda la fase e consolidi i risultati raggiunti.

Il momento della valutazione è molto importante, soprattutto se consente allo studente di ricevere feedback personalizzati e immediati e se gli da la possibilità di capire cosa non ha compreso dell'argomento appena concluso.

Per rendere questa fase ancora più significativa l'insegnante può organizzare un processo di peer-assessment, che da una parte obbliga lo studente a considerare in maniera critica soluzioni diverse dalla propria, dall'altra gli permette di ottenere commenti dei propri pari immediatamente dopo la consegna del proprio esercizio.

Imparare a valutare in peer-assessment induce gli studenti ad attivare abilità cognitive di alto livello come quelle di analisi, comparazione, riflessione, metacognizione.

Si ottiene così un triplice risultato:

- Ogni studente ottiene un feedback immediato al proprio lavoro, circostanziato e completo.
- Gli studenti sono stimolati a leggere e cercare di capire i percorsi logici di un pari che hanno condotto alla soluzione, certamente diversi dai propri, e acquisiscono così un'abitudine a leggere in maniera critica il codice di un programma.
- I più bravi, che fanno pochi errori nei propri programmi, imparano a correggere errori anche molto subdoli, che loro non avrebbero fatto ma dei quali è bene sapere come difendersi. I meno bravi si impadroniscono di tecniche che non avrebbero saputo progettare da soli e, dal confronto, sono motivati a cercare di superare i propri limiti.

2 L'importanza di imparare a interpretare programmi scritti da altri.

Per chi insegna coding, c'è una motivazione in più per attivare la valutazione fra pari per almeno qualche modulo di apprendimento.

Se suddividiamo il ciclo di vita del software in fasi (studio di fattibilità, analisi preliminare, analisi, codifica, test, manutenzione) ci rendiamo conto che la fase di scrittura di un programma non rappresenta che una parte, neppure troppo significativa, del ciclo complessivo, ciò significa che è assolutamente necessario che il codice sia ben strutturato, le variabili abbiano dei nomi chiari, l'algoritmo non faccia percorsi tortuosi, in modo da semplificare il compito a chi dovrà poi occuparsi della fase di manutenzione.

E' innegabile che ogni programmatore, e quindi ogni studente, sviluppi una propria tecnica personale di coding, ma questa caratterizzazione non deve in alcun modo inficiare la leggibilità e dunque la manutenibilità del software. Questa esigenza non è però immediatamente chiara a chi sta ancora imparando e non riconosce questa peculiarità come una proprietà importante a cui badare.

Chi programma è naturalmente orientato a fare si che il software funzioni in tutte le condizioni, se nel corso della fase di test si evidenziano degli errori, le modifiche necessarie a correggere i malfunzionamenti dovrebbero essere integrate nella struttura del programma, invece molto spesso vengono aggiunte in modo ingarbugliato e frettoloso.

Strutturare bene un programma richiede attenzione, precisione e tempo, bisogna che si acquisisca la consapevolezza che non si tratta di tempo perso.

Tale consapevolezza si può acquisire dovendo valutare il programma di un altra persona, si prende così coscienza dei percorsi tortuosi che dovrebbero essere evitati e della mancanza di chiarezza nella scelta dei nomi delle variabili, nella progettazione degli oggetti, nella suddivisione in funzioni.

Allo studente che ha valutato il codice di un altro studente si dovrebbe chiedere di esprimersi anche in merito alle caratteristiche di manutenibilità, dando un punteggio alla leggibilità e alla facilità di intervento per aggiungere o modificare funzionalità. In modo da aiutare gli studenti a riconoscere le peculiarità del software di qualità.

Bisogna ricordare che il coding non si esaurisce nella progettazione dell'algoritmo e conseguente scrittura del codice, ma è molto importante anche la fase di test, e che esiste un test di tipo statico e uno di tipo dinamico. Il test di tipo statico è un test formale e teorico che non richiede che il programma venga mandato in esecuzione su dei dati di prova, bensì bisogna leggere e interpretare i percorsi del programma in modo critico, controllando che non siano presenti errori, che siano stati rispettati gli standard di qualità, che non siano possibili eventuali miglioramenti per quanto riguarda l'impegno macchina o l'impiego della RAM.

È difficile per chi ha scritto il codice essere critico e condurre un test statico veramente obiettivo, ed è anche difficile all'inizio del percorso di apprendimento riconoscere l'importanza di questa fase, per questo è molto importante introdurre nel processo di apprendimento diversi momenti in cui chi impara a programmare è costretto a leggere programmi scritti da altri in modo critico. E questi programmi non devono essere perfetti, al contrario, devono poter essere migliorabili perché sia stimolato il pensiero critico di chi legge, programmi di questo tipo li abbiamo sottomano e sono quelli scritti da altri studenti della classe, ognuno con il proprio stile.

Fagan, un ingegnere elettronico dell'IBM, introdusse già nel suo articolo del 1976 [1] le regole del processo della Fagan Inspection.

Il processo di ispezione del codice introdotto da Fagan consiste nell'esame del programma ad opera di almeno due programmatori diversi dall'autore, secondo una check-list concordata. A valle di questa fase gli "ispettori" e l'autore si devono riunire alla presenza di un moderatore e discutere i difetti riscontrati in modo da condividere le soluzioni, l'autore dovrà poi effettuare le modifiche al codice così come concordato.

Questa tecnica è utile anche nella fase di revisione del codice professionale, si tratta della code review, cioè in quella fase nella quale viene predisposta la nuova versione da pubblicare, aggiornata e corretta.

Naturalmente le tecniche di revisione del codice si sono evolute, ma applicare la Fagan Inspection, almeno in ambito didattico, è ancora importante perché consente ai programmatori "ispettori" e all'autore di condividere conoscenze, imparare nuove tecniche e riconoscere l'importanza di lavorare in team, sperimentando un'importante soft-skill.

In sostanza possiamo utilizzare le tecniche della Fagan Inspection nella didattica del coding, trasformandola nella valutazione fra pari, che avrà quindi come obiettivi l'esame delle competenze di accuratezza e consapevolezza nella scrittura di programmi, la determinazione del grado di aderenza agli standard di codifica e la definizione della capacità di rispettare le scadenze nella consegna del proprio lavoro.

3 Funzionamento dell'attività Workshop di Moodle.

Facendo una ricerca sulle più attuali tecniche valutative di peer-assessment, mi sono imbattuta nell'articolo "IMPROVe: sei principi research-based per realizzare attività di valutazione fra pari nei contesti formativi." [2], dove viene presentato il modello, denominato appunto IMPROVe che si compone di sei principi di pratica.

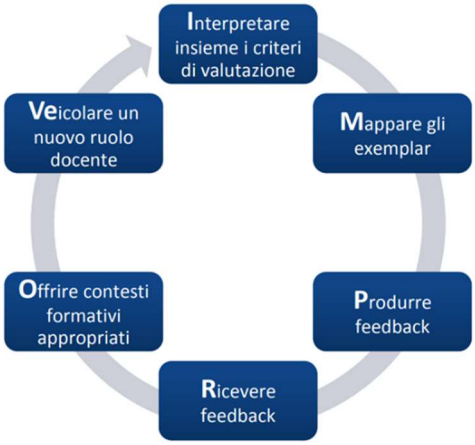


Figura 1 - Modello IMPROVe

Ciascuna lettera rappresenta uno dei sei principi:

Interpretare insieme i criteri di valutazione: è importante che le regole sulla base delle quali si deve formulare la valutazione siano condivise con gli studenti, e quando gli studenti abbiano già preso parte a una o più attività di peer assessment si potrebbe addirittura pensare di progettare insieme la griglia di valutazione, o di chiedere agli studenti di migliorare quella utilizzata fino a quel momento.

Mappare gli exemplar, dove gli exemplar sono degli esempi di valutazione effettuati da altri studenti, già predisposti e forniti agli studenti perché possano trarne spunto e capire che cosa ci si aspetta da loro come valutatori.

Ricevere e Produrre feedback, cioè fornire e ricevere una valutazione descrittiva ed esaustiva degli errori riscontrati ed eventualmente i suggerimenti per la correzione.

Offrire contesti formativi appropriati, in questo nostro caso specifico, ci si riferisce all'**attività Workshop di Moodle**, che rappresenta la modalità tecnica con la quale verrà organizzata ed erogata la fase di peer assessment.

Veicolare un nuovo ruolo docente, in effetti, come vedremo, in questo processo il docente assume il ruolo di supervisore, un ruolo quindi diverso dal suo ruolo tradizionale.

Questi sei principi sono centrali anche per quanto riguarda la valutazione fra pari nel coding, ma, essendo il compito di codificare un programma, un task con delle specificità molto particolari, considereremo questi principi, ma applicandoli nel contesto caratteristico del coding. A questo proposito l'articolo "Assessment of programming language learning based on peer code review

model: Implementation and experience report." [3] propone un metodo molto ben strutturato per l'organizzazione della fase di peer-assessment nel caso del coding, si tratta del Peer Code Review, PCR.

Il modello PCR coinvolge 5 ruoli: autore, revisore, correttore, insegnante e assistente, le caratteristiche di ogni ruolo sono dettagliate in tabella 1:

Tabella 1 – i ruoli del modello PCR

autore	Lo **studente** che scrive il programma
revisore	Lo **studente** che corregge il programma scritto da un altro studente
correttore	Lo **studente** che corregge il proprio programma sulla base delle correzione suggerite dai commenti del revisore
insegnante	La figura che darà la valutazione finale e dovrà garantire il rispetto degli standard organizzativi e di qualità – coordinatore delle attività
assistente	Figura di ausilio alle funzioni più pratiche dell'insegnante (nel nostro caso questo ruolo viene ricoperto dall'**insegnante stesso**)

Come si vede lo studente ricopre tre dei ruoli in tabella a seconda della fase del processo di peer assessment in cui si trova.

Il modello PCR descritto nell'articolo, fa riferimento ad uno specifico software per la gestione automatica del processo di peer assessment, si vedrà che l'attività workshop di Moodle costituisce un valido sostituto del software proposto da Wang perché ricopre tutte le funzionalità del software presentato con il PCR.

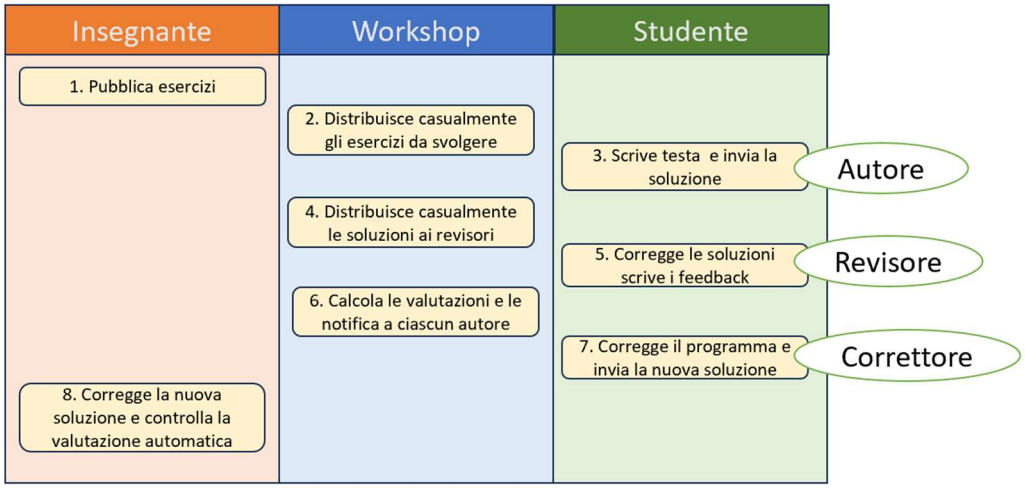


Figura 2 – Cronologia e ruoli del modello PCR

Nella figura 2 sono elencate in verticale le varie attività del processo, ciascuna attività è collocata nella colonna relativa al soggetto che se ne deve occupare: insegnante, studente o Workshop (nel caso di quelle attività che vengono gestite automaticamente da Moodle).

1. L'insegnante pubblica un certo numero di esercizi
2. Il workshop abbina casualmente un esercizio ad ogni studente
3. lo studente scrive, testa e invia la sua soluzione (in questa fase lo studente è nel ruolo di **autore**)
4. Il workshop assegna casualmente ciascuna soluzione ad un certo numero di revisori (il numero è stabilito dall'insegnante)
5. Ciascuno studente corregge le soluzioni che gli sono state assegnate (anche anonimamente) secondo una griglia di valutazione quantitativa concordata e condivisa (in questa fase lo studente è nel ruolo di **revisore**). Alla fine della compilazione della

griglia viene calcolato e assegnato automaticamente un punteggio al programma, il calcolo dipende dalla griglia di valutazione.

6. Il workshop notifica a ciascun autore le correzioni, i feedback del suo programma e la valutazione complessiva assegnata a questa attività. La valutazione sarà composta in parte dalla media dei punteggi che gli sono stati assegnati dai suoi compagni e in parte dal punteggio che Moodle ha calcolato automaticamente al suo ruolo di valutatore.
7. Lo studente corregge il programma secondo i suggerimenti ricevuti e invia la sua nuova soluzione (in questa fase lo studente è nel ruolo di **correttore**)
8. L'insegnante controlla tutte le correzioni e i commenti e i nuovi programmi inviati e conferma o corregge la valutazione calcolata automaticamente.

L'attività Workshop di Moodle gestisce automaticamente le attività dalla 1 alla 6.

I documenti che vengono prodotti nel modello in definitiva sono: il **codice prodotto** (scritto e testato da un autore), **la griglia di correzione** che comprende i **commenti o feedback** (le idee, i suggerimenti e le critiche che un revisore propone con particolare riguardo agli standard di codifica) e il **codice corretto** (una nuova edizione del programma corretto secondo i commenti dei revisori).

Questo processo di valutazione comprende una certa interconnessione tra i ruoli, bisogna impedire che gli studenti che non rispettano le consegne, come qualità o come tempi, penalizzino i compagni che, nella cronologia del proprio task, dovrebbero partire proprio da tali consegne.

Si può ovviare ai problemi organizzativi che derivano da queste situazioni anomale prevedendo più revisori per una stessa soluzione, in questo modo ciascuno studente dovrà valutare più soluzioni e gli arriverà più di un feedback.

A questo punto l'autore farà una sintesi tra i commenti ricevuti per correggere il proprio codice.

Per evitare l'imbarazzo agli studenti di rilevare gli errori di un compagno di cui conosce l'identità, si può nascondere sia il nome dell'autore che dei revisori, anche se, in qualche sessione di peer assessment, è raccomandabile non nascondere queste informazioni, perché si possa poi organizzare quel meeting di discussione della Fagan Inspection.

Tutte le attività potranno svolgersi in un'unica mattinata, oppure potranno essere diluite nel tempo, a seconda che si tratti di una verifica, di un'esercitazione, o che il problema sia più o meno complesso.

Si può anche non assegnare un unico problema a tutta la classe, in modo da poter far correggere ad ogni studente esercizi diversi da quello che ha svolto come autore.

Il Workshop consente a ciascuno studente l'invio di uno o più file digitali (soluzione di un problema), sono queste le consegne che vengono poi ridistribuite casualmente a uno o più compagni revisori.

Le figure che si vedono successivamente corrispondono ad un'attività di peer assessment somministrata in una classe seconda, relativa ad un esercizio sulle strutture complesse in Python.

Figura 3 – Pannello di controllo dell'attività Workshop di Moodle

La figura 3 raffigura il pannello di controllo dell'attività di workshop una volta che è stata data la chiusura dell'attività.

Il pannello di controllo è fondamentale per la configurazione e il controllo delle fasi del processo di peer assessment, da lì si può controllare l'avvenuta consegna delle soluzioni, eventualmente sollecitare quelli che non sono puntuali, e dare le chiusure di ogni fase per poter passare alla fase successiva.

Possiamo vedere per immagini quali sono i parametri che possiamo configurare per realizzare una attività analoga. Innanzitutto nell'immagine successiva si vede come si inserisce l'attività:

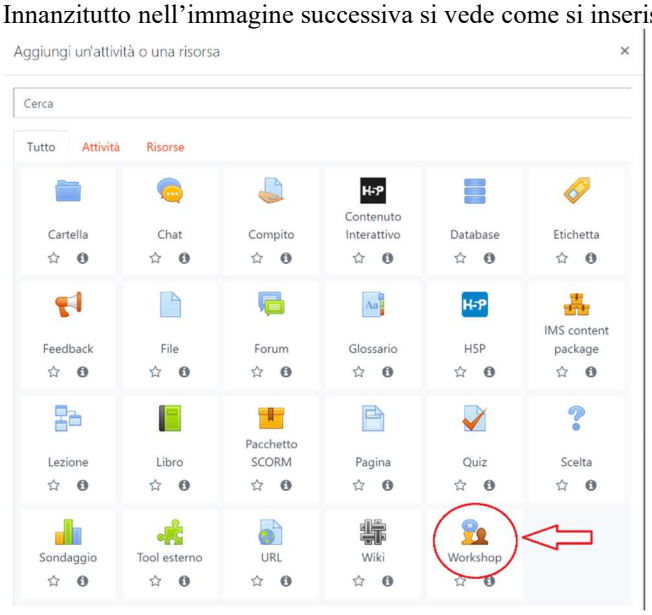


Figura 4 – Inserire l'attività Workshop

Nell'immagine 5 invece vediamo quali parametri configurare, molti parametri dell'attività Workshop corrispondono ai parametri comuni a tutte le attività di Moodle ("disponibilità", "impostazioni comuni", "condizioni per l'accesso", "tag" e "competenze") ma ci sono parametri caratteristici che si devono definire solo per il Workshop.

Molto importante è la definizione del "criterio di valutazione", che può essere: "voto cumulativo," "commenti", "numero di errori" o "rubric"; per ottenere sia una valutazione "numerica" sia permettere ai revisori la formulazione di feedback testuali bisogna scegliere voto cumulativo.

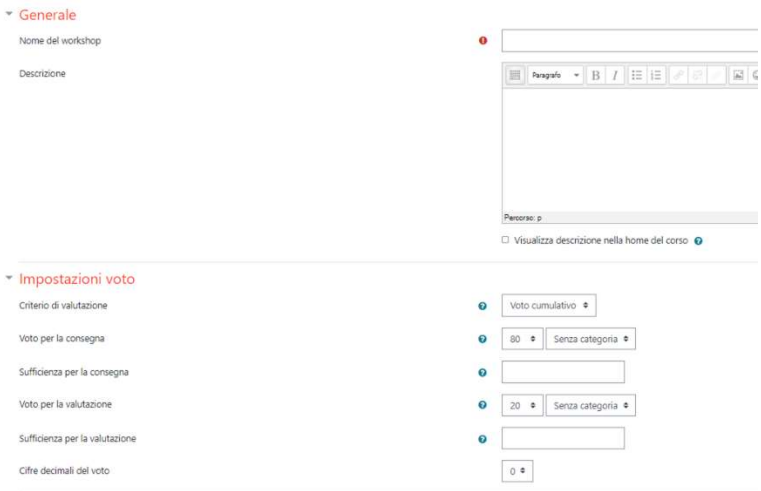


Figura 5 – Alcuni parametri dell'attività Workshop

In questo caso l'insegnante fornisce una griglia di valutazione (che si dovrà inserire successivamente nell'attività Workshop) elencando una serie di item o descrittori e dando ad ognuno un peso e una percentuale di raggiungimento.

Lo studente, rispondendo in modo quantitativo a una serie di domande, ottiene automaticamente la valutazione delle soluzioni che gli sono state assegnate.

Un'altra scelta molto importante è determinare quanta parte del voto sia riservata alla soluzione (ruolo autore) e quanta alla valutazione (ruolo revisore), se si accetta la proposta 80-20, il voto complessivo sarà composto per l'80% dalla valutazione della soluzione, derivante dalla valutazione ricevuta dai compagni, mentre per il 20% sarà "valutata la valutazione" dello studente nel ruolo di revisore.

La "valutazione sulla valutazione" è importante perché induce gli studenti ad un comportamento corretto quando ricoprono il ruolo di revisori, essa viene calcolata con una funzione statistica sulla base di tutte le valutazioni ricevute da ciascun item.

Le "impostazioni della consegna", quelle della "valutazione" e la "descrizione" nella parte generale sono fondamentali per una corretta comprensione del processo, in queste finestre bisogna scrivere le indicazioni per gli studenti in modo che sia chiaro cosa devono fare in ciascuna delle fasi.

Nell'esercitazione assegnata la descrizione del problema comprendeva sia il testo dell'esercizio da svolgere, sia una sintetica indicazione dei goal da raggiungere:

Questa esercitazione è divisa in due parti:
nella prima dovrai svolgere e inviare un esercizio sulle strutture complesse
nella seconda parte ti saranno assegnate 3 consegne tra quelle dei tuoi compagni e dovrai valutarle, avrai a disposizione una griglia di valutazione che renderà automatico il calcolo della valutazione

Testo del problema

esercizio temperature

Creare un programma per il caricamento delle temperature medie rilevate in **10 città italiane**:

- caricare le informazioni in una struttura dizionario o in due liste parallele (chiedendo in input ciascuna città e rispettiva temperatura che può non essere un numero intero) 4 punti
  visualizzare tutte le informazioni caricate come:

città: temperatura
città: temperatura
..... 1 pto

- trovare e visualizzare la media delle temperature su tutte le città arrotondata a due cifre decimali 1 pto
- trovare e visualizzare la città con la temperatura massima 2 pti
- visualizzare in un elenco le città con la temperatura più alta della media 2 pti

Figura 6 - La descrizione del problema

Ciascun programma è stato valutato da tre studenti, di conseguenza a ciascuno studente sono arrivati tre feedback diversi, per facilitare la correzione agli studenti valutatori, quando sono stati assegnati i programmi da valutare è stato anche fornito un esempio di soluzione.

La griglia di valutazione viene impostata, descrivendone gli "elementi" o item come in figura 7.

Figura 7 - Impostazione della griglia di valutazione

Ogni elemento ha una descrizione e gli si deve attribuire un punteggio massimo e un peso nell'ambito della valutazione complessiva.

E' consigliabile fornire il significato di ogni punteggio per facilitare chi valuta, si possono definire quanti elementi quanti si vuole, ma, per non complicare troppo la fase di valutazione è importante che non ci siano troppi item, per ogni item lo studente può scrivere un commento a motivazione del punteggio assegnato (feedback), questa informazione può essere resa obbligatoria, come prescritto da uno dei principi di pratica del modello IMPROVe.

In figura 8 si può vedere come si presenta la Griglia al momento della compilazione.

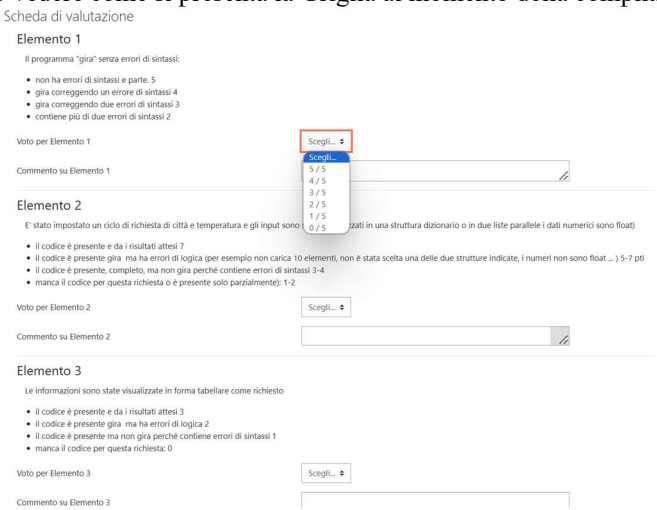


Figura 8 - La Griglia di Valutazione

Una volta che tutti gli studenti abbiano consegnato la propria soluzione (si può impostare un blocco per cui il sistema non accetta consegne in ritardo), l'insegnante apre la fase di valutazione, in questa fase il workshop distribuisce casualmente a ciascuno studente il numero stabilito di file da correggere.

Man mano che gli studenti compilano la griglia di valutazione il pannello di controllo dell'insegnante mostra i progressi del processo. Quando tutti hanno valutato tutto, l'insegnante passa alla fase di calcolo dei voti e ottiene la situazione complessiva come in figura 9, dove si vede chiaramente il calcolo del voto della consegna, che è una media tra i punteggi assegnati da ciascuno studente; nella colonna finale invece si vede la valutazione assegnata allo studente come valutatore.

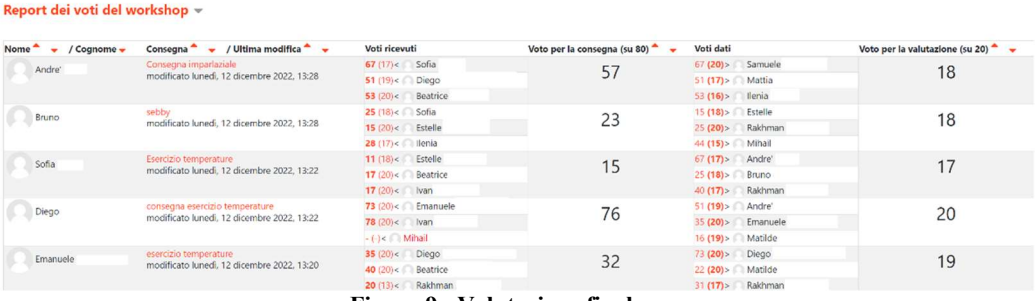


Figura 9 - Valutazione finale

Gli studenti si sottopongono sempre molto volentieri a questo tipo di esercitazioni, soprattutto se lo scopo ultimo è poi quello di sostenere una verifica organizzata in questo modo, il meccanismo è sfidante e stimola la partecipazione e l'impegno.

Dalle discussioni sulla valutazione di questo tipo di attività che ho avuto con gli studenti coinvolti ho rilevato:

1. che preferiscono quando i programmi che devono essere valutati sono anonimi, mentre non è importante che i revisori siano anonimizzati.

2. che è importante che la griglia di valutazione sia stata provata prima in qualche altra esercitazione e che gli item siano condivisi.

Naturalmente l'attività Workshop non è limitata all'applicazione del modello PCR, può essere utile in tutte quelle situazioni in cui si voglia innescare un processo di peer assessment, per esempio in tutti i casi in cui si voglia far valutare agli studenti la presentazione fatta da ciascuno studente/gruppo a tutta la classe.

In questo caso la Griglia di valutazione conterrà gli item secondo i quali ciascuno studente dovrà valutare la presentazione, come esemplificato in figura 10.

Criteri di valutazione

Rubrica di Valutazione

Gruppo valutato							
Gruppo che valuta							
Studente							
Criteri	Indicatori	Punti 1	Punti 2	Punti 3	Punti 4	Punti 5	
Selezione organizzata dei materiali	I contenuti scelti sono significativi rispetto al tema?						scegli un punteggio
Grado di rielaborazione personale	Emerge una rielaborazione personale dei contenuti?						scegli un punteggio
Utilizzo del linguaggio specifico	Viene utilizzato un linguaggio corretto con un'opportuna scelta di termini specifici?						scegli un punteggio
Originalità della soluzione	Sono stati scelti collegamenti non banali per spiegare approfondire i concetti chiave?						scegli un punteggio
Impiego documentato delle fonti	Le fonti sono state dichiarate in modo corretto?						scegli un punteggio
Rispetto delle consegna	Sono stati rispettati i vincoli quantitativi e di corretto funzionamento delle consegne?						scegli un punteggio
Estetica della presentazione	La presentazione ha uno stile curato, organizzato e chiaro.						scegli un punteggio
	Punteggio complessivo				Su 35 punti		

Figura 10 - Griglia di valutazione per il peer assessment di una presentazione

Bibliografia

[1] M. Fagan, «"Design and Code Inspections to Reduce Errors in Program Development.",» 1976.

[2] G. V. Serbati N., «IMPROVe: sei principi research-based per realizzare attività di valutazione fra pari nei contesti formativi.,» *Form@re, Firenze University Press,* 2019.

[3] L. H. F. Y. J. Y. L. Y. Wang Y., «Assessment of programming language learning based on peer code review model: Implementation and experience report.,» *Computer & Education, Elsevier* , 2012.

Formare agli *opendata* tramite *fisicalizzazione* IoT di oggetti "quotidiani"

Andrea Trentini[1]

Dipartimento di Informatica, Università degli Studi di Milano
andrea.trentini@unimi.it

Sommario

I dati in generale e l'*opendata* in particolare sono due aspetti importanti della partecipazione civica perché sono le basi per la condivisione (ex-ante) delle scelte e la misura (ex-post) degli effetti della *governance* della *res publica*. Ma gestire dati non è sempre banale, servono competenze matematico-statistiche e informatiche, mentre lo stato attuale delle conoscenze e abilità del "cittadino medio" è purtroppo lacunoso (cfr. indice DESI). Uno strumento di attenuazione delle paure nei confronti di tecnologia, informatica e trattamento dei dati potrebbe essere la "*fisicalizzazione*" (rappresentazioni tangibili dei dati stessi). Qui si propone la sistematica realizzazione di "*fisicalizzazioni* IoT (*Internet of Things*) di oggetti quotidiani" come veicolo *non terrorizzante* di cattura dell'attenzione e di narrazione del dato (reperito in tempo reale via rete) e come laboratorio didattico per l'insegnamento delle tecniche di raccolta e rappresentazione dei dati.

1 L'importanza civica dei dati

Oggi i *dati* sono cruciali nella società contemporanea: influenzano la vita dei cittadini e la costruzione di una comunità civica, riguardano la consapevolezza dei cittadini in tema di gestione e di impatto sulla società. Sono beni comuni: forniscono informazioni per pianificazione urbana, sanità, lotta alla criminalità e altre sfide sociali. Nel contesto della trasparenza di governo (e.g., nell'ambito della Pubblica Amministrazione), i dati diventano documentazione istruttoria (*ex-ante*, descrivono uno *status quo*, es. lo stato di inquinamento dell'aria) e misura degli effetti (*ex-post*, come verifica [Tre15] della *governance*). I dati quindi assumono una "importanza civica" che spinge alla partecipazione attiva dei cittadini nel processo di gestione implicando la promozione della trasparenza e dell'accessibilità dei dati, così che le persone possano comprenderne e sindacarne [FWAT16] l'utilizzo. I cittadini potrebbero assumersi alcune "responsabilità civiche":

- essere critici verso organizzazioni e istituzioni che raccolgono e utilizzano i dati, chiedendo conto sui metodi di trattamento e promuovendo accessibilità (*opendata*, senza *webstacle* [Tre14]) e utilizzo responsabile per il bene comune [Dav10];
- sostenere l'apertura dei dati governativi e l'accesso pubblico alle informazioni, contribuendo così a una maggiore trasparenza e a una migliore *accountability* delle istituzioni;
- partecipare attivamente a progetti di *crowdsourcing* dei dati o a iniziative di *data science* civica, contribuendo con le proprie conoscenze e competenze per risolvere problemi sociali complessi.

2 Paure e lacune

Il cittadino "medio" può diventare "digitale" superando paure e lacune verso tecnologia e informatica:

Figura 1: Tastiera "parlante" (fonte: http://dataphys.org)

Figura 2: I ritardi dei treni su una... sciarpa (fonte: http://dataphys.org)

- complessità delle tecnologie intimidisce le persone senza formazione tecnica, interfacce complesse (o troppo semplici rispetto alle complessità interne dei sistemi) possono creare ansia e frustrazione;
- automazione e IA aumentano timori di obsolescenza professionale;
- analfabetismo digitale può creare insicurezza e paura di emarginazione dalla società, cfr. indice DESI[1], Italia nella parte bassa della classifica;

Un aiuto potrebbe risiedere nell'avvicinamento "non terrorizzante" al dato e alla sua rappresentazione [Hay18, Per21].

3 *fisicalizzazione*

Con "*fisicalizzazione* dei dati"[2] si indica la produzione di artefatti fisici le cui geometrie o proprietà dei materiali codificano i dati stessi [DJVM20]. Offre un modo alternativo e coinvolgente per esplorare, comprendere e comunicare informazioni complesse. Trasformando i dati in oggetti fisici o rappresentazioni tangibili, si aprono prospettive e opportunità [JDI+15] di interazione con le informazioni [KWK20].

Le *fisicalizzazioni* possono essere:

statiche: sculture create a partire da insiemi di dati, es. la tastiera con le frequenze d'uso dei tasti (Figura 1);

dinamiche: oggetti che modificano le proprie caratteristiche ("autonomamente") o che vengono continuamente aggiornati in funzione di un "flusso dati" come la sciarpa (Figura 2) prodotta da una pendolare attendendo treni;

[1] http://digital-strategy.ec.europa.eu/en/policies/desi
[2] http://dataphys.org

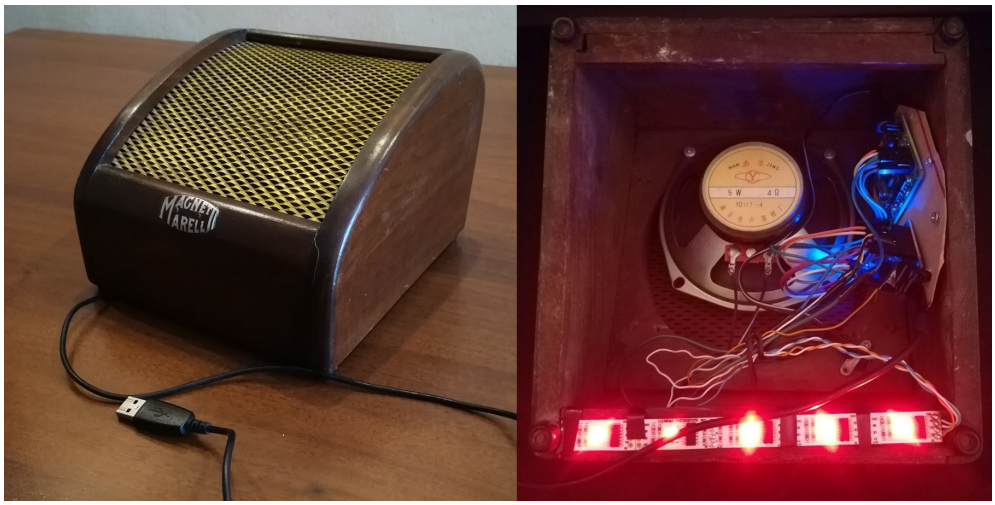


Figura 3: Physiradio

dinamiche interattive: veri e propri oggetti interattivi, dotati cioè di qualche tipo di interfaccia utente[3] che permettono al fruitore di indirizzare la rappresentazione dei dati[4] [THK+15].

Alcuni vantaggi delle *fisicalizzazioni interattive* per le persone comuni:

- utilizzano interfacce semplici e intuitive che richiedono poche istruzioni o conoscenze tecniche. Gli utenti possono interagire con oggetti fisici, pulsanti, manopole o schermi per esplorare e manipolare i dati;
- forniscono feedback visivi e tattili che aiutano le persone a comprendere le relazioni e i modelli dei dati. Ad esempio, possono utilizzare oggetti fisici che cambiano forma, colore o posizione in risposta alle interazioni dell'utente, ciò aiuta a rendere i dati più tangibili e comprensibili;
- si possono includere contestualizzazioni dei dati, ossia fornire informazioni aggiuntive o spiegazioni sul significato dei dati stessi, aiutando a creare un quadro complessivo comprensibile e a connettere i dati con la vita reale o le esperienze quotidiane delle persone.

Anche se alcuni aspetti tecnici possono essere coinvolti nella creazione delle *fisicalizzazioni* interattive dei dati, il focus principale è sulla facilità d'uso, l'accessibilità e la comprensibilità per un pubblico non tecnico. L'obiettivo è trasformare i dati complessi in una forma concreta e coinvolgente, rendendoli accessibili e interessanti per una vasta gamma di persone.

[3] Sebbene non "classica": niente tastiere, mouse, display, ma magari rotelle, leve, oggetti da spostare/impilare, ecc.

[4] Esempi interessanti:
- http://beatsigner.com/data-physicalisation.html
- http://yvonnejansen.me/dynamic-dataphys

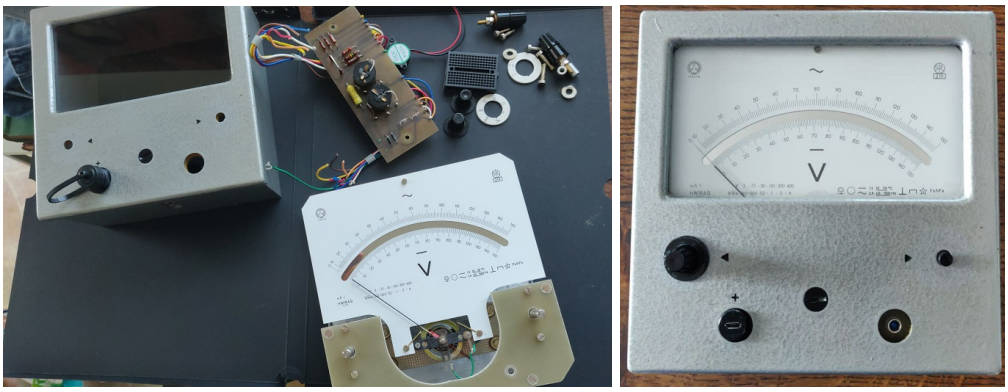


Figura 4: Un voltmetro?!?

4 IoT e sistemi *embedded*: l'influenza di Arduino

La nascita di Arduino[5] (2005-2008) ha rivoluzionato [MSAPCO17] il mondo dei sistemi *embedded* e dell'IoT (Internet of Things) portandolo da un contesto a "elevato *gap* di ingresso"[6] ad un mondo in cui appassionati, hobbisti e sviluppatori che desiderano creare dispositivi e progetti interattivi dispongono di ambienti di sviluppo semplici, una vasta gamma hardware e comunità attive e collaborative:

- ha reso l'elettronica e la programmazione accessibili anche a chi dispone di formazione tecnica limitata. Le schede sono relativamente economiche e facili da utilizzare, consentendo a un'ampia varietà di persone di entrare nel mondo dell'IoT e dei sistemi *embedded*;
- ha un ruolo significativo nell'educazione tecnica e nell'apprendimento delle basi dell'elettronica e della programmazione;
- è utilissimo per la prototipazione rapida;
- ha una vasta comunità online di appassionati, sviluppatori e esperti che condividono conoscenze, progetti e risorse;
- ha stimolato un ecosistema hardware (sovente con funzioni e prestazioni aggiuntive rispetto alla piattaforma "originale", ad esempio wifi e bluetooth, processori multicore, grandi memorie RAM e *flash*, ecc.) e software che oggi permette lo sviluppo di sistemi *embedded* in svariati linguaggi (python, java, lua, programmazione dichiarativa, perfino ambienti "per bambini" come Scratch[7]).

5 Proposta: *fisicalizzazione* IoT di oggetti "quotidiani"

Questo articolo propone lo sviluppo di *fisicalizzazioni* IoT di oggetti "quotidiani"[8] come veicolo di sperimentazione, gioco e aiuto alla formazione sul tema *trattamento e visualizzazione dei dati*. Nel corso degli anni (dal 2015) l'autore ha sviluppato un approccio alla **conversione IoT** di oggetti della vita quotidiana relativamente semplice, sia da comprendere che da realizzare:

[5] http://www.arduino.cc

[6] Prima si usavano ambienti di sviluppo complessi, proprietari e costosi, servivano programmatori esperti, la documentazione era molto tecnica.

[7] http://scratch.mit.edu

[8] Tra virgolette perché tra gli esempi che seguono sono presenti sia oggetti effettivamente quotidiani (lampade) che altri (come il Dekatron) un po' meno alla portata del "cittadino medio" di cui sopra.

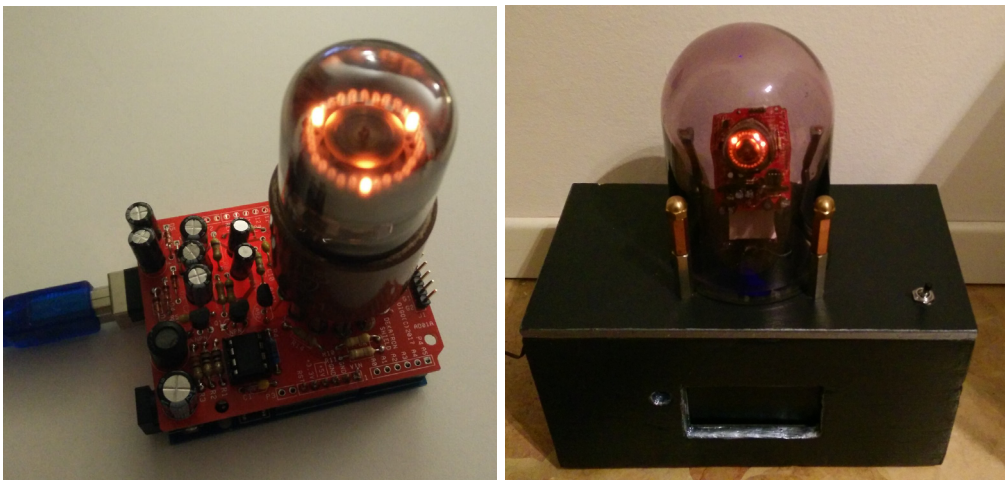


Figura 5: Dekatron IoT

- identificazione di un oggetto "interessante" (che stimoli la fantasia, sia del realizzatore che dei futuri utilizzatori);
- cui segue la sua "strumentazione"[9] mediante l'inserimento di una *board* IoT (quindi connessa, con ampie possibilità di recuperare dati via Internet), attualmente la scelta ricade su hardware ESP8266[10], in particolare sulle piccole (circa 2x3cm) *board* prodotte da Wemos[11];
- se l'oggetto non ha parti in movimento e/o elettrificate[12] oltre alla *board* va aggiunto qualche attuatore (luci, parti mobili, ecc.) e/o sensore (per poter reagire a eventi ambientali → interattività);
- programmazione del comportamento, l'ambiente di sviluppo attualmente usato è ESPHome[13] a paradigma dichiarativo (le specifiche vengono descritte in file YAML[14]).

Oggi è una via percorribile per molti sviluppatori (e non!) cavalcando l'influenza di Arduino e l'ecosistema che ha creato: il c.d. "Mondo Maker".

Idee analoghe si trovano descritte in [HGG+16, VKRD19, TWDT13, KWK20], ma in quelle proposte manca un approccio sistematico alla *IoT instrumentation* di oggetti quotidiani. Rispetto alla creazione ex novo di *fisicalizzazioni* tradizionali (ad es. le stampe 3D per quelle statiche) si perde un po' di fisicità ma si guadagna in stimolo e divertimento nell'usare oggetti della vita di tutti giorni[15].

[9] Il termine inglese *instrumentation* rende meglio l'idea.

[10] Di Espressif, http://www.espressif.com/en/products/socs/esp8266

[11] http://wemos.cc

[12] Alcuni elettrodomestici potrebbero essere utilizzati quasi così come sono, semplicemente rendendoli "comandabili" tramite la *board* aggiuntiva, si pensi ad una lampada normale o a un ventilatore. In una casa *domotica* le possibilità aumentano perché si può semplicemente scrivere software di comando dei vari attuatori già presenti.

[13] http://esphome.io

[14] http://yaml.org

[15] O anche prodotti *vintage* recuperati nei mercatini, come fa chi produce accessori *steampunk* (http://it.wikipedia.org/wiki/Steampunk).

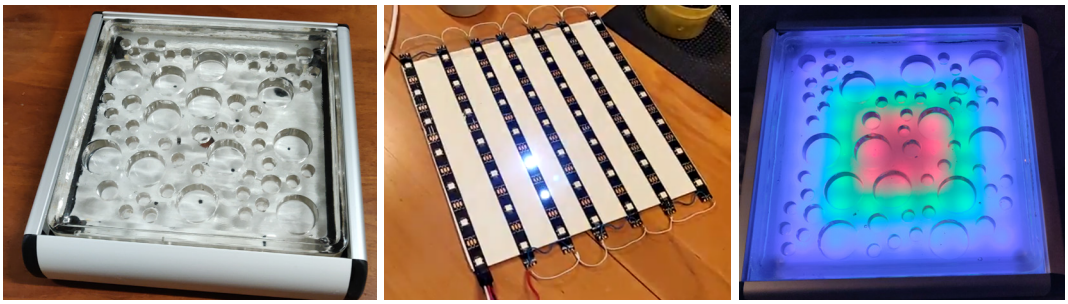


Figura 6: Portacandele + lampadario = matrice luminosa

5.1 Alcuni prototipi realizzati

5.1.1 Physiradio

Internet radio costruita in un altoparlante *vintage* (Figura 3) che fisicalizza dati meteo pubblicamente disponibili mappando condizioni meteorologiche e genere musicale ("musicalizzazione"). L'associazione (tempo atmosferico → genere musicale) è soggettiva ma comprensibile [LG16], il funzionamento logico può essere compreso quasi completamente da non-esperti e può divenire spunto di conversazione. Physiradio è stato testato sul campo[16] tra colleghi, studenti e altre persone attraverso un processo quanti-qualitativo [TS20].

5.1.2 IoTDial

"IoTizzazione" di un voltmetro *vintage* (degli anni '70). Lo strumento è stato smontato (Figura 4 lato sx), eliminate le parti di misura (resistenze), inserita *board* ESP8266, convertito un selettore di portata in *user input device*, aggiunti alcuni LED colorati sia interni (sfondo dello strumento) che esterni (ulteriore *output device*), aggiunto un pulsante (altro *user input device*) e una porta USB (alimentazione, programmazione). Sono stati fatti esperimenti di visualizzazione, tramite l'ago dello strumento, di dati meteo (temperatura, umidità) e dei dati di traffico dei mezzi pubblici, in particolare sui tempi di arrivo di un bus[17] ad una fermata specifica, facendo *scraping* dei dati ATM-Milano.

5.1.3 Dekatron

Esperimenti di riutilizzo di un Dekatron[18] (Figura 5), componente di calcolo per computer anni '50. Usato in modo "degenere" (non come contatore) funge da *display*. L'autore ha realizzato: un orologio sincronizzato; un visualizzatore di banda Internet; un contatore Geiger.

5.1.4 Portacandele Aarikka

Combinando un portacandele finlandese in fusione di vetro Aarikka[19], una matrice LED RGB e un lampadario da soffitto è stata realizzata una "piattaforma luminosa" (Figura 6) a ma-

[16] I dati dei questionari sono disponibili in *opendata*.
[17] Questa modalità di funzionamento è stata battezzata "Fantozzi" in onore e ricordo della scena "prenderò l'autobus al volo!" dell'omonimo film.
[18] http://it.wikipedia.org/wiki/Dekatron
[19] http://museo.aarikka.com

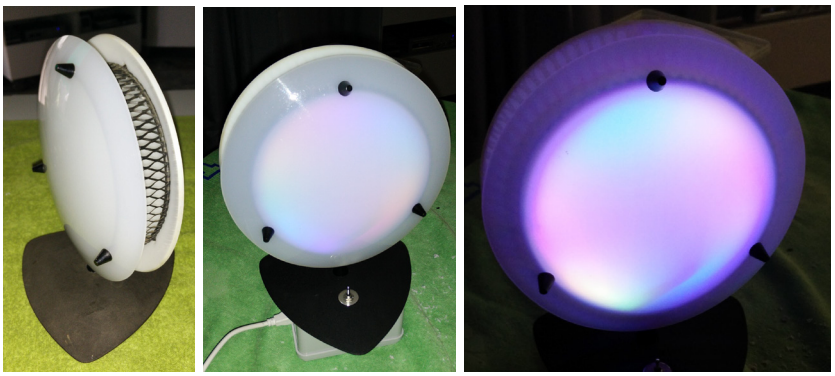


Figura 7: Da lampada anni '60 a *color coder*

trice (7x7 pixel). Al momento viene utilizzata come *useless machine* animata colorata da intrattenimento, è in attesa di qualche dato da visualizzare.

5.1.5 Klamp

Una lampada anni '60 con una *strip* LED RGB (Figura 7). Viene usata come orologio sincronizzato (tre punti di colore diverso indicano ore, minuti, secondi come lancette). Muovendola (contiene un accelerometro) diventa un simulatore di coppa contenente un liquido rossastro che segue l'inclinazione secondo gravità[20].

5.1.6 BiblioVisualizer

Barra luminosa realizzata inserendo una *strip LED* (Light Emitting Diode) in un lampadario industriale di recupero (Figura 8) che fisicalizza, realizzando una *progress bar*, i dati di prestito libri (della giornata in corso, in tempo quasi reale) nel territorio di competenza del CSBNO[21]. Interessante è stato anche l'apprezzamento delle istituzioni coinvolte nel progetto (biblioteca di Cormano e Comune) che hanno accolto l'installazione "artistica" e l'hanno inserita nelle presentazioni ufficiali dell'Open Day della biblioteca. Il progetto, nato in seno all'associazione HackLabCormano[22], ha funzionato anche da "biglietto da visita" dell'associazione stessa all'interno della biblioteca: nuovi soci si sono presentati e hanno chiesto iscrizione incuriositi dalla barra.

Figura 8: Barra prestiti

6 Conclusioni

In questo articolo, è stato affrontato il tema dell'importanza civica dei dati e la necessità di avvicinare le persone normali al mondo dei dati e della loro trattazione. I dati rivestono un

[20] Funzione battezzata "San Gennaro".
[21] http://webopac.csbno.net
[22] http://hacklabcormano.it

ruolo cruciale nella società contemporanea e possono fornire informazioni preziose per la pianificazione urbana, la sanità pubblica, la lotta contro la criminalità e altre sfide sociali. La partecipazione attiva dei cittadini nella gestione dei dati è fondamentale per promuovere la trasparenza e l'*accountability* delle istituzioni. Tuttavia, il cittadino medio potrebbe essere intimidito dalla complessità delle tecnologie e dei dispositivi informatici, temere la perdita di lavoro a causa dell'automazione e dell'intelligenza artificiale, o sentirsi escluso a causa dell'analfabetismo digitale. Affrontare queste paure e lacune è cruciale per permettere una partecipazione consapevole e attiva alla società basata sui dati.

Un approccio interessante per avvicinare le persone al mondo dei dati è la loro *fisicalizzazione*. Questa tecnica consiste nella produzione di oggetti fisici il cui aspetto o proprietà materiale codifica i dati stessi. Le *fisicalizzazioni* offrono un modo coinvolgente e intuitivo per esplorare, comprendere e comunicare informazioni complesse. Con la *fisicalizzazione*, i dati complessi possono essere trasformati in oggetti tangibili, accessibili e interessanti per un vasto pubblico. Inoltre, l'IoT e i sistemi *embedded*, massificati (in senso assolutamente positivo) dall'ecosistema Arduino, hanno reso l'elettronica e la programmazione più accessibili a un pubblico più ampio, inclusi gli appassionati, gli hobbisti e i pre-teen. L'IoT offre ampie possibilità per la *fisicalizzazione* dei dati, consentendo di trasformare oggetti quotidiani in dispositivi interattivi e intelligenti.

Questo articolo propone di sviluppare *fisicalizzazioni* IoT di oggetti quotidiani. Un approccio che consente di sperimentare e facilitare la formazione sul trattamento e la visualizzazione dei dati. Gli esempi presentati, come Physiradio, BiblioVisualizer, IoTDial e altri, dimostrano come oggetti comuni possano essere trasformati in dispositivi interattivi e informativi. L'uso di hardware come ESP8266 e l'approccio dichiarativo alla programmazione rendono l'implementazione di tali dispositivi accessibile anche a persone relativamente meno "tecniche". La *fisicalizzazione* IoT di oggetti quotidiani rappresenta un modo intrigante e coinvolgente per avvicinare le persone al mondo dei dati, superando paure e lacune, e aprendo nuove opportunità per l'interazione con le informazioni complesse.

La *fisicalizzazione* IoT di oggetti quotidiani potrebbe giocare anche un ruolo significativo anche nel campo della didattica. La trasformazione di oggetti comuni in dispositivi interattivi e informativi potrebbe rendere l'apprendimento dei concetti legati ai dati e alla tecnologia più coinvolgente e stimolante per gli studenti di tutte le età.

Nell'ambito della didattica, l'utilizzo di dispositivi IoT fisicalizzati potrebbe fornire agli studenti un'opportunità pratica per imparare i principi dell'elettronica, della programmazione e del trattamento dei dati. Creare, "instrumentare" e programmare oggetti quotidiani per visualizzare dati reali potrebbe consentire agli studenti di acquisire competenze pratiche che possono essere applicate in molti campi, dalla scienza all'ingegneria.

Inoltre, l'interattività delle *fisicalizzazioni* IoT potrebbe favorire l'apprendimento esperienziale, in cui gli studenti possono esplorare i dati in modo attivo e scoprire relazioni e modelli in modo autonomo.

Le *fisicalizzazioni* IoT possono anche essere utilizzate per la didattica delle materie STEM (Scienza, Tecnologia, Ingegneria e Matematica). Ad esempio, l'uso di dispositivi IoT per visualizzare dati meteorologici o dati scientifici potrebbe aiutare gli studenti a comprendere meglio i concetti scientifici astratti e a sviluppare il pensiero critico. Nel contesto della didattica delle scienze sociali, le *fisicalizzazioni* IoT possono essere utilizzate per creare rappresentazioni tangibili di dati socio-economici o politici.

Inoltre, l'utilizzo di dispositivi fisicalizzati potrebbe favorire l'inclusione di studenti con diversi stili di apprendimento, poiché offre un'esperienza multisensoriale che coinvolge sia la

vista che il tatto. Questo approccio potrebbe essere particolarmente utile per gli studenti con difficoltà di apprendimento o disabilità.

Infine, le *fisicalizzazioni* IoT possono fungere da ponte tra il mondo virtuale e il mondo reale, aiutando gli studenti a comprendere meglio come i dati influenzano la nostra vita quotidiana e come possono essere utilizzati per prendere decisioni informate.

In conclusione, l'integrazione della *fisicalizzazione* IoT di oggetti quotidiani nella didattica offre molteplici vantaggi, tra cui l'apprendimento pratico delle tecnologie e dei dati, l'esperienza di apprendimento esperienziale, l'inclusione di studenti diversamente abili e una maggiore comprensione dell'importanza dei dati nella società contemporanea. La *fisicalizzazione* IoT potrebbe trasformare la didattica in un'esperienza coinvolgente e stimolante, preparando gli studenti a diventare cittadini consapevoli e informati nell'era dei dati.

Riferimenti bibliografici

[Dav10] Tim Davies. Open data, democracy and public sector reform. a look at open government data use from data.gov.uk. Practical Participation, 2010.

[DJVM20] Pierre Dragicevic, Yvonne Jansen, and Andrew Vande Moere. *Data Physicalization*, pages 1–51. Springer International Publishing, Cham, 2020.

[FWAT16] Mark Frank, Johanna Walker, Judie Attard, and Alan Tygel. Data literacy - what is it and how can we make it happen? *The Journal of Community Informatics*, 12(3), Sep. 2016.

[Hay18] Sarah Hayes. Exploring the potential of data physicalization for stem learning. In *Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '18, page 703–705, New York, NY, USA, 2018. Association for Computing Machinery.

[HGG+16] Steven Houben, Connie Golsteijn, Sarah Gallacher, Rose Johnson, Saskia Bakker, Nicolai Marquardt, Licia Capra, and Yvonne Rogers. Physikit: Data engagement through physical ambient visualizations in the home. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1608–1619. ACM, 2016.

[JDI+15] Yvonne Jansen, Pierre Dragicevic, Petra Isenberg, Jason Alexander, Abhijit Karnik, Johan Kildal, Sriram Subramanian, and Kasper Hornbæk. Opportunities and challenges for data physicalization. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3227–3236. ACM, 2015.

[KWK20] Maria Karyda, Danielle Wilde, and Mette Gislev Kjærsgaard. Narrative physicalization: Supporting interactive engagement with personal data. *IEEE Computer Graphics and Applications*, 41(1):74–86, 2020.

[LG16] Luca A Ludovico and Presti Giorgio. The sonification space: a reference system for sonification tasks. *International Journal of Human-Computer Studies*, 85:72–77, 2016.

[MSAPCO17] Juan Carlos Martínez-Santos, Oscar Acevedo-Patino, and Sonia H. Contreras-Ortiz. Influence of arduino on the development of advanced microcontrollers courses. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 12(4):208–217, 2017.

[Per21] Charles Perin. What students learn with personal data physicalization. *IEEE Computer Graphics and Applications*, 41(6):48–58, 2021.

[THK+15] Faisal Taher, John Hardy, Abhijit Karnik, Christian Weichel, Yvonne Jansen, Kasper Hornbæk, and Jason Alexander. Exploring interactions with physically dynamic bar charts. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3237–3246, 2015.

[Tre14] Andrea Trentini. Lombardy EPA *Obtorto Collo* data and anti-pollution policies fallacies. *Journal of e-Learning and Knowledge Society*, 10(2), 2014.

[Tre15] Andrea Trentini. Verifying traffic ban effects on air pollution. *Journal of Atmospheric Pollution*, 3(1):9–14, 2015.

[TS20] Andrea Trentini and Simone Scaravati. Raising curiosity about open data via the 'physiradio'musicalization IoT device. *Data Science Journal*, 19(1), 2020.

[TWDT13] Joshua G Tanenbaum, Amanda M Williams, Audrey Desjardins, and Karen Tanenbaum. Democratizing technology: pleasure, utility and expressiveness in diy and maker practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2603–2612. ACM, 2013. non era tra quelli scaricati da dataphys, ma viene citato spesso.

[VKRD19] David Verweij, David Kirk, Kay Rogage, and Abigail Durrant. Domestic widgets: Leveraging household creativity in co-creating data physicalisations. *Northumbria University*, 2019. Newcastle upon Tyne, United Kingdom.

Un'Esperienza di Realizzazione di una Serra Hi Tech nella Scuola Tecnologica

Antonella Pulito[1], Sergio Santostasi[2], Francesco Di Tria[3]
[1] ITT Panetti-Pitagora - Bari
antonella.pulito@panettipitagora.edu.it
[2] ITT Panetti-Pitagora - Bari
sergio.santostasi@panettipitagora.edu.it
[3] ITT Panetti-Pitagora - Bari
francesco.ditria@panettipitagora.edu.it

Abstract

L'insegnamento dell'Informatica nella Scuola Superiore richiede attualmente strumenti innovativi, che coinvolgano gli studenti in attività interdisciplinari. L'articolo propone un'esperienza di realizzazione di una serra automatizzata, che ha richiesto l'applicazione dei principi di programmazione strutturata, al fine di controllare il comportamento di dispositivi elettronici in grado di ridurre il consumo energetico ed idrico.

1 Introduzione

L'epoca in cui viviamo risente fortemente dell'impatto umano sugli equilibri del Pianeta [1] e le conseguenti sfide del mondo globale possono apparire talvolta, ai più giovani, come minacce per il loro futuro. Infatti, si sta velocemente diffondendo la cosiddetta Eco-Ansia [2], una sorta di apprensione sul cambiamento climatico in atto, che potrebbe causare stati di depressione ed agitazione tra gli adolescenti, sicuramente i più interessati al futuro del nostro Pianeta. Per contrastare atteggiamenti negativi e pessimistici, risulta fondamentale sollecitare concretamente studentesse e studenti affinché colgano tutte le opportunità per costruire una società più giusta ed equilibrata e soprattutto sostenibile.

In tale contesto, dunque, si è proposto un laboratorio attivo in cui il progetto reale è consistito nella realizzazione di una serra automatizzata in Arduino. Infatti, la serra, essendo dotata di diversi sensori ambientali, è in grado ridurre l'intervento umano e di applicare efficacemente azioni coerenti con le politiche di risparmio energetico ed idrico [3]. L'aspetto fondamentale del progetto è consistito nella programmazione dei diversi attuatori, come la pompa di irrigazione e la lampadina di riscaldamento, al fine di personalizzare il comportamento della serra non solo in base alle specifiche esigenze di coltivazione, ma anche in base alle condizioni ambientali sistematicamente rilevate dai sensori.

Il progetto ha rappresentato principalmente uno strumento didattico per l'insegnamento dell'Informatica all'interno di una classe di secondo superiore ad indirizzo tecnologico. In particolare, studentesse e studenti hanno sperimentato i principali costrutti della programmazione strutturata in Linguaggio C in maniera innovativa ed interattiva, effettuando un primo approccio verso la programmazione robotica [4]. Grazie a questo progetto, si sono sviluppate ulteriori competenze trasversali fondamentali in ambito informatico, quali il lavoro in collaborazione, il rispetto delle scadenze e la capacità di esposizione dei risultati raggiunti.

2 Il progetto interdisciplinare

Il progetto è stato realizzato come Unità di Apprendimento (UdA) interdisciplinare a partire dalla fine del primo quadrimestre: l'insegnamento mediante UdA rappresenta un metodo efficace ed essenziale per presentare i contenuti didattici in maniera unitaria ed integrata. In questo contesto, le conoscenze relative alle discipline di Biologia, Chimica ed Elettronica hanno rappresentato dei prerequisiti per comprendere il progetto da realizzare.

La scelta delle specie vegetali è ricaduta sulla coltivazione di misticanza e pomodorini di Battipaglia, selezionando una tipologia di colture più idonee ad essere piantate in uno spazio ristretto e per le quali era importante il monitoraggio dei parametri considerati (Figura 1). Tale fase è stata preceduta dallo studio delle cellule vegetali e delle trasformazioni energetiche al loro interno [5].

Figura 1. Serra automatizzata

3 Hardware

Il progetto è basato su una scheda microcontrollore programmabile Arduino Mega, alla quale sono stati collegati sensori e attuatori:

- igrometro, sensore per la misurazione dell'umidità del suolo. Data la dimensione della serra 50 x 100 x 50 cm (L x P x A), nel progetto ne sono stati utilizzati 3, disposti in vari punti del terreno. La tensione operativa è 5V, con contatti di commutazione della soglia regolabili con trimmer.
- sensore di rilevamento del livello d'acqua all'interno di un serbatoio.

- sensore digitale DHT11 per il rilevamento di temperatura e umidità dell'aria all'interno della serra.
- RTC (Real Time Clock). Questo è un modulo per inserire un orologio di sistema, dotato di una batteria integrata, che consente di memorizzare eventi temporizzati anche in assenza di alimentazione del circuito.
- Display LCD a 4 righe e 20 colonne per la visualizzazione dei principali parametri, collocato all'interno di una scatola di derivazione.
- sistema di irrigazione, che consiste in una pompa immersa in contenitore di acqua e collegata a tubi posizionati intorno al vaso.
- LED per illuminare la serra in assenza di fonti naturali.
- Buzzer, un dispositivo acustico regolabile in intensità e frequenza per la segnalazione di eventi critici, come l'assenza di acqua nel serbatoio.
- Modulo 4 relè per il pilotaggio on-off di lampade e pompe di irrigazione.

4 Software

L'attività didattica in Informatica è inizialmente partita dallo studio di un progetto open source Arduino [6] che è stato successivamente personalizzato. Il linguaggio utilizzato per la programmazione è stato il C dell'Ide di Arduino, con il quale sono stati gestiti i dati di input provenienti dai vari sensori, calcolati i dati di output da mostrare sul display ed effettuate le decisioni per l'azionamento degli attuatori.

Dati di input:

- Umidità del terreno: è un valore analogico fornito dall'igrometro. Il programma calcola il valore medio dei 3 igrometri; tale valore viene quindi trasformato in percentuale in base a valori di minimo e di massimo ottenuti da un preliminare processo di taratura del sensore.
- Temperatura e umidità dell'aria. Sono valori interi letti tramite la libreria software annessa al sensore DHT11.
- Livello di acqua presente nel vaso. Anche questo è un segnale analogico che viene convertito in valore numerico di tipo intero.
- Data e ora rilevate dal modulo RTC.

Dati di output

- Valore medio di umidità del terreno, temperatura e umidità dell'aria all'interno della serra, data e ora corrente. Sono tutti dati visualizzati sul display (Figura 2). Il responsabile della manutenzione della serra si può avvalere di questi dati per avere in maniera sintetica ed immediata delle informazioni sulle condizioni attuali della serra.

Processo decisionale:

- Il primo e più importante punto di decisione riguarda la gestione del sistema di irrigazione. Al fine di ottimizzare il consumo idrico, la serra viene irrigata ad intervalli regolari e predefiniti. Il sistema di irrigazione è costituito da una pompa ad immersione che viene azionata una volta al giorno in base all'orario rilevato dal sensore RTC.
- Nel progetto originale, una lampadina ad incandescenza, viene azionata in base ad un fotoresistore, un componente elettronico in grado di rilevare le fonti luminose. Grazie a questo componente, la lampadina, che ha funzioni sia di riscaldamento che di illumina-

zione, può essere azionata automaticamente solamente in assenza di luce naturale. Nel progetto in questione, anche la lampadina viene gestita in base ai dati forniti dal modulo RTC, al fine di azionare la lampadina solo nelle ore serali.

- Un altro punto di decisione è quello che riguarda l'attivazione del buzzer. Si è scelto di attivare un segnale sonoro nel caso in cui il livello dell'acqua, rilevato mediante il sensore posizionato al bordo del vaso, salga al di sopra di un limite fissato, al fine di evitare una eccessiva presenza di acqua nel terreno.

Figura 2. Scatola di derivazione

Una possibile evoluzione di questo progetto consiste nell'integrazione con sistemi di domotica per il monitoraggio e l'intervento umano anche da remoto [7].

5 Valutazione dell'esperienza

Il secondo superiore è l'anno di Corso un cui studentesse e studenti iniziano ad affrontare i temi legati alla programmazione, dopo aver appreso al primo anno i concetti generali dell'Informatica e l'utilizzo degli strumenti essenziali per l'automazione d'ufficio.

Generalmente, i primi elementi di programmazione consistono nello studio del linguaggio C e dei principi della programmazione strutturata. A questo scopo, le tradizionali metodologie didattiche prevedono il coinvolgimento degli studenti su esercitazioni legate a discipline propedeutiche, quali la Matematica. Le esercitazioni consistono essenzialmente nel risolvere semplici esercizi di geometria (calcolo dell'area di figure geometriche), di logica (trovare il più grande fra due numeri) e di aritmetica (calcolo di percentuali).

In questo progetto, lo studio del linguaggio di programmazione è stato applicato per il controllo di dispositivi elettronici in grado di operare in un sistema reale. I vantaggi di questo approccio sono stati molteplici. Per prima cosa, il progetto ha sollecitato la curiosità e l'interesse degli alunni: la programmazione applicata alla robotica presenta caratteristiche di forte interattività e consente, rispetto al semplice calcolo di formule, una sperimentazione pratica di ogni comando poiché ogni istruzione del programma interviene a modificare lo stato del sistema.

Secondariamente la classe, organizzata in gruppi eterogenei, ha acquisito i differenti compiti, seguendo le inclinazioni di ciascuno. Studentesse e studenti maggiormente interessati all'Elettronica hanno curato l'assemblaggio delle componenti hardware. Quelli più interessati all'Informatica hanno curato lo sviluppo del programma, altri hanno approfondito l'aspetto

chimico-biologico con attività di ricerca-azione, monitorando e documentando lo sviluppo delle specie vegetali nella serra. Per tale attività, alcuni studenti a volte hanno prolungato spontaneamente la loro permanenza a scuola oltre l'orario delle lezioni. Il progetto è servito, quindi, anche come strumento di orientamento per la scelta di indirizzo al terzo anno.

Tutti i risultati conseguiti sono stati alla fine condivisi fra i vari gruppi attraverso lo scambio di esperienze e la realizzazione di specifici elaborati finali con scadenze programmate (coinvolgendo anche altre discipline come Tecnologie grafiche per la riproduzione grafica e computerizzata della serra). I docenti responsabili del progetto ritengono che l'esperienza maturata abbia rappresentato un elemento di innovazione nel tradizionale curricolo scolastico e abbia consentito agli studenti di effettuare un primo passo verso il raggiungimento di tutti quegli aspetti che vanno sotto il nome di competenze trasversali.

6 Conclusione

Lo studio dell'Informatica richiede, specialmente per i ragazzi in età adolescenziale, nuovi approcci e tecniche innovative. È infatti ormai obsoleto lo studio del linguaggio C che prevede la realizzazione di programmi da utilizzare a riga di comando. Attualmente, il linguaggio C è quello predefinito in progetti basati su Arduino. Grazie ad Arduino, gli studenti possono sperimentare ed apprendere concetti come input/output attraverso il controllo e la visualizzazione dei dati rilevati da sensori. Inoltre, possono apprendere concetti solitamente più ostici, come le scelte condizionali e le iterazioni, applicandoli al controllo e all'azionamento di attuatori. Questo consente di ottenere effetti pratici e visibili delle istruzioni che compongono il programma. Nella serra automatizzata, il processo di controllo è consistito essenzialmente nella gestione dei vari attuatori, come ad esempio la pompa di irrigazione, al fine di ottimizzare i consumi energetici ed idrici rispondendo alle esigenze sempre più pressanti e improrogabili di un clima in continuo cambiamento.

Bibliografia

[1.] N. Calidori, Antropocene, 2022, *APhEx*
[2.] Baudon, Pauline, and Liza Jachens. (2021). A Scoping Review of Interventions for the Treatment of Eco-anxiety. International Journal of Environmental Research and Public Health 18.18, 9636.
[3.] Bonoli, A. (2020). Crisi ambientale e soluzioni per la sostenibilità e l'adattamento. Didattica Della Storia – Journal of Research and Didactics of History, 2(1S), 382–395. https://doi.org/10.6092/issn.2704-8217/11110
[4.] Ionescu, V. M., and Cosmin S. (2017). Tteaching Arduino Programming at High School Level. University of Pitesti Scientific Bulletin Series: Electronics and Computer Science 17.2, 37-44.
[5.] S. Lenzi, F. Chimirri F, and C. Fiussello, Biologica. Capire le scienze della vita, 2021, LINX Editore
[6.] https://www.progettiarduino.com/82-serra-automatizzata-con-arduino.html
[7.] https://www.tinkercad.com/projects/ESP32-Soil-Moisture-Sensors-DIY-Automatic-Watering

"Eppur si muove", mettiamo in movimento T∞ls; il robot mascotte della rete "Robotica Educativa Valdostana"

Luca Salvoni[1], Patrizia Cedrino[2], Oriana Cimalando[3]
[1] Istituzione scolastica Mont Rose A
l.salvoni@mail.scuole.vda.it
[2] Istituzione scolastica Mont Rose A
p.cedrino@mail.scuole.vda.it
[3] Istituzione scolastica Mont Rose A
o.cimalando@mail.scuole.vda.it

Abstract

La rete di Robotica Educativa Valdostana (R.E.V.) comprende 12 Istituzioni scolastiche, 11 Istituti comprensivi e un Istituto di scuola secondaria di secondo grado. Obiettivo della rete è quello di creare efficaci commistioni fra le diverse discipline, i linguaggi artificiali e gli ambienti di apprendimento ad alto investimento tecnologico, spostando l'attenzione dal solo aspetto tecnico compilativo a quello di prodotto culturale, frutto di nuove istanze sociali.

1 Introduzione

All'inizio di ogni anno scolastico la rete presenta delle proposte inerenti la creazione e l'utilizzo di artefatti digitali in ambienti laboratoriali fisici o virtuali. Il robot "T∞ls", nasce dall'esigenza di dotare la rete non solo di un nome, ma anche di un immagine, un simbolo che la rappresenti. A seguito di un concorso in cui si richiedevano un progetto cartaceo, una modellazione sul software Tinkercad e la realizzazione di un artefatto in stampa 3D, "T∞ls" viene eletto mascotte della rete

Le classi seconde della scuola secondaria di primo grado dell'Istituzione scolastica Mont Rose A accettano la sfida di dare movimento al robot; prende strada così un progetto, che si dipana nel corso di parte dell'anno scolastico, con l'obiettivo di fare avanzare e retrocedere il robot, utilizzando la scheda Micro:bit per la gestione dei comandi.

In fase di progettazione dell'attività, si decide di lavorare sulla realizzazione di un supporto mobile per il trasportato di "T∞ls"; si esclude la possibilità di inserire all'interno del robot l'occorrente per il suo movimento in quanto gli alunni non hanno preconoscenze sul software utilizzato per creare i modelli 3D e quindi, la necessaria modifica del file di modellazione, occuperebbe tempo ed energie e soprattutto non garantirebbe a tutti la possibilità di raggiungere l'obiettivo di realizzare il movimento; le ridotte dimensioni del volume di stampa della stampante 3D a disposizione della scuola, inoltre, non permettono la stampa in un'unica soluzione, rendendo così necessario creare più parti da assemblare in un secondo momento. L'attività viene realizzata tra febbraio e maggio in un modulo settimanale di compresenza tra i docenti di tecnologia e matematica.

2 Il progetto

FASE 1: presentazione e prime proposte.
L'attività viene illustrata agli studenti condividendo il compito e fornendo loro il materiale per realizzarlo: un modellino di "T∞ls", stampato in 3D e una scheda Micro:bit. Non si è posto inizialmente alcun limite alle soluzioni proposte relativamente al tipo di supporto possibile, si è richiesto però di concretizzare in un disegno questa proposta in modo da poterne discutere successivamente la fattibilità.
L'esplicitazione delle diverse idee ha evidenziato alcune criticità legate essenzialmente alle preconoscenze degli studenti rispetto alla trasmissione del movimento. Molti alunni infatti ritenevano che bastasse un cavo elettrico per trasmettere la rotazione del motore a eventuali ruote o eliche. Altro errore comune è stato collegare la batteria alle ruote senza prevedere la presenza dei motori.
Proposte originali sono state: un dirigibile (proposta scartata per le difficoltà di stabilizzazione verticale, di calcolo della dimensione idonea del pallone e di reperibilità dell'elio), una motoslitta e un paio di sci.

FASE 2: scheda Micro:bit e ambiente di programmazione.
Mentre gli studenti sistemavano a casa le loro proposte in modo da renderle fattibili, è stata presentata nel dettaglio la scheda Micro:bit e l'ambiente di programmazione. Per quest'ultimo si è utilizzato il sito ufficiale della scheda (microbit.org) nella sezione MakeCode, dove sono presenti diversi tutorial che gradualmente presentano i vari blocchi di programmazione. Gli studenti hanno seguito i tutorial con partecipazione ed interesse, e gli interventi degli insegnanti sono stati necessari principalmente per risolvere problemi di connessione tra la scheda e il pc.

FASE 3: modello cartaceo del supporto
Agli studenti è stato chiesto di realizzare un modello cartaceo del loro supporto che prevedesse gli spazi per la sistemazione dei motori, della batteria e della scheda Micro:bit. Gli insegnanti hanno realizzato e reso disponibile un loro modello cartaceo, un esempio approssimativo che conteneva palesemente degli errori progettuali (sovradimensionamento e presenza di alloggiamenti solo per due ruote); hanno inoltre fornito alla classe tutto il materiale da inserire nel supporto, in modo da poter considerare le misure più adatte.
Per la realizzazione dell'artefatto cartaceo si è partiti da uno sviluppo piano del supporto, predisposto in modo tale da poter essere ritagliato e incollato, Benché gli sviluppi di solidi fossero già stati affrontati in tecnologia, diversi alunni hanno avuto serie difficoltà nel disegnare tutte le facce e le estensioni necessarie per l'assemblaggio, in particolare quelli che hanno scelto solidi differenti dal parallelepipedo, come cilindro, prismi a base esagonale o trapezoidale.

FASE 4: dalla carta al digitale
Sistemato il modello cartaceo in modo che rispondesse a tutti i requisiti richiesti, si è passati alla sua trasposizione in digitale utilizzando il software Tinkercad. Ci si è inizialmente concentrati sull'uso dei comandi essenziali per rendere gli studenti operativi in breve tempo: gestione di solidi pieni e vuoti, dimensione e rotazione, allineamento e unione di più solidi.
In questa fase gli studenti hanno apportato i maggiori cambiamenti ai loro progetti iniziali; tutti hanno deciso di suddividere il supporto in due parti, una base, che avrebbe contenuto scheda, motori e pacchi batteria, e un coperchio di chiusura. Alcuni hanno modificato anche la forma iniziale con il fine di caratterizzare meglio il proprio artefatto. Le difficoltà principali hanno riguardato la precisione delle misure per l'alloggiamento dei motori e dell'asse delle ruote non motrici e la sovrapposizione base-coperchio senza gioco eccessivo.
Alcuni cambiamenti si sono poi realizzati a seguito del rilevamento di due diverse soluzioni di costruzione: la possibilità di utilizzare una scheda aggiuntiva a cui agganciare Micro:bit, fornita di connettori specifici per collegare facilmente i motori, che avrebbe aumentato però lo

spessore della Micro:bit, rendendo in alcuni casi obbligatorio sovradimensionare il progetto iniziale, oppure l'utilizzo di una seconda scheda Micro:bit da impiegare come radiocomando;. Questa seconda possibilità è stata accolta da tutti gli studenti, perché più funzionale, in quanto evitava di inseguire il robot durante il suo spostamento per premere i tasti fisici A e B che sarebbero stati abbinati al movimento avanti, indietro e ferma.

FASE 5: stampa 3D del supporto
La fase di stampa è stata gestita dai docenti con l'ausilio parziale degli studenti, in quanto i tempi di stampa si sono rivelati più lunghi del previsto (si è arrivati a 21 ore per una singola base) e quindi le stampe sono state lanciate anche ad orari in cui gli alunni non erano presenti a scuola.

FASE 6: assemblaggio del supporto mobile
Si è scoperto solo in fase di assemblaggio che le misure utilizzate nei disegni 3D differivano di alcuni millimetri rispetto al modello stampato. Visto che questa incongruenza è stata riscontrata in tutte le stampe, si ipotizza che l'errore sia dovuto al software di conversione che, partendo dal file scaricato da Tinkercad, crea il file di stampa. Solo per i progetti terminati in ritardo si è potuto ovviare al problema aumentando alcune dimensioni, mentre per gli altri è stato necessario un lavoro manuale di lima, soprattutto per allargare gli alloggiamenti dei motori, che risultavano troppo stretti. In alcuni casi è stato necessario ristampare la base o il coperchio.
I motori sono stati fissati con della colla a caldo per evitare che si muovessero dalla loro sede durante il movimento, mentre per le ruote non motrici sono stati utilizzati dei pezzi dei Lego EV3 presenti a scuola. Solo un gruppo ha optato per il movimento tramite cingoli ed anche in questo caso la scatola del Lego EV3 è stata d'aiuto, con una modifica per permettere alle ruote dentate di incastrarsi al perno dei motori.

FASE 7: programmazione della scheda Micro:bit
La fase di programmazione è quella che ha creato problemi maggiori soprattutto per lo scarso tempo a disposizione. È stato consigliato agli studenti di visionare un programma presente sul sito micro:bit.org che riguardava un progetto simile al nostro, ma con alcune differenze sostanziali sui comandi di azionamento dei motori e su comandi di movimento inutili per il nostro obiettivo. La maggior parte degli studenti non è stata in grado di modificare il programma d'esempio, ma si è limitata ad una semplice ricopiatura. È stato quindi necessario l'intervento dei docenti per sistemare il programma.

3 Conclusioni

Eppur si muove!
Ore di impegno si sono condensate in quell'unico momento in cui gli studenti hanno schiacciato i tasti del telecomando e hanno visto il supporto che trasportava T∞ls muoversi avanti e indietro: tutti hanno sorriso. Magari la traiettoria del robot non era propriamente rettilinea, ma in quel momento era un dettaglio trascurabile.
Terminiamo con un'osservazione che ci piace evidenziare: gli studenti che si sono distinti per l'impegno e per l'autonomia di lavoro sono principalmente ragazze, a prova che le STEM non riguardano il genere, ma l'interesse e la voglia di mettersi in gioco.

Sperimentazione di Curricolo Digitale

Domenica Roberta Mistretta[1]
[1] I.C.S. Ignazio Buttitta – Bagheria
roberta.mistretta@icsbuttitta.it

Abstract

Il contributo racconta l'esperienza sul campo per l'attivazione di un corso ad orientamento digitale per il triennio della secondaria di primo grado. La sottoscritta, docente di Tecnologia e di potenziamento, è stata coordinatrice della prima classe sperimentale.

1 Introduzione

Il nostro Istituto scolastico dispone del potenziamento di Tecnologia nella scuola secondaria I°. Nell'a.s. 2020-2021, a seguito di un dibattito sull'importanza dell'educazione al digitale e sulla *media education*, si è avviata una sperimentazione per un corso ad orientamento digitale, in cui il tempo scuola si arricchiva di due ore in più di potenziamento di Tecnologia, nel pomeriggio.

Nel nostro territorio il tempo scuola è di 30 ore settimanali, per cui il laboratorio pomeridiano è stata una importante innovazione. La scuola rappresenta un polo di aggregazione, di integrazione e di inclusione, in un territorio che presenta pochi punti di incontro e scarsi stimoli culturali.

2 La progettazione didattica

Partendo dagli obiettivi del PNSD, in particolare dall'azione 18, è stata progettata una proposta per un triennio di Scuola Secondaria I° ad orientamento digitale, in cui si è aggiunto alle discipline curricolari, un "Laboratorio digitale e *media education*", con due ore settimanali aggiuntive. Il laboratorio è stato immaginato come spazio didattico per contaminare il curricolo di studio, attraverso l'educazione ai media, all'informazione e alle applicazioni della creatività digitale. Il modello DIGCOMP ha rappresentato una guida fondamentale per strutturare i quattro nuclei tematici della materia (N.T.), da declinare nei tre anni di studio, con un grado di complessità crescente. Nelle immagini seguenti se ne sintetizzano i contenuti

Fig.1. Nucleo tematico 1

 ISBN: 978-88-6629-075-9

Fig.2. Nucleo tematico 2

Fig.3. Nucleo Tematico 3

Fig.4. Nucleo Tematico 4

3 L'esperienza sul campo

L'utilizzo dello strumento digitale, in quanto trasversale nelle diverse materie, ha rappresentato un elemento per modificare le metodologie e strategie didattiche messe in campo e ha portato ad un miglioramento del processo di apprendimento oltre all'acquisizione delle specifiche competenze digitali.

La scuola ha fornito agli alunni tutti gli strumenti utili allo sviluppo della didattica digitale.

Ogni allievo ha ricevuto un tablet in comodato d'uso con connessione, che ha affiancato i tradizionali strumenti di studio. Le due ore aggiuntive con l'insegnante di Tecnologia sono sempre state svolte nel laboratorio d'informatica o nel laboratorio STEM.

Il consiglio di classe ha lavorato con un approccio multidisciplinare, in modo da integrare le conoscenze disciplinari in un percorso di apprendimento organico; l'utilizzo dello strumento digitale è divenuta opportunità per far lavorare i ragazzi in gruppo, in assetto laboratoriale.

Io, docente di Tecnologia, ho svolto quattro ore di lezione a settimana (due curriculari e due ulteriori di potenziamento), orientando le scelte educative sulle TIC e la sperimentazione laboratoriale.

I contenuti disciplinari di Tecnologia sono stati rielaborati, supportati dallo strumento digitale, per cui, per esempio, relazioni o schemi, tabelle e grafici sono stati prodotti prevalentemente in digitale. Il disegno tecnico con squadrette e matita è stato arricchito della competenza del disegno al computer.

Mi soffermerò più nel dettaglio sui contenuti del N.T.4 che riguarda il Laboratorio di coding e creatività digitale.

3.1 Coding

L'alfabetizzazione informatica inizialmente necessaria ha progressivamente lasciato il posto all'approfondimento dei princìpi alla base del funzionamento dei sistemi informatici, con attività didattiche tese all'acquisizione del cosiddetto *pensiero computazionale*. In tal senso il progetto *Programma il futuro* è stato un valido supporto per far sviluppare il pensiero computazionale attraverso la programmazione informatica (coding) in un contesto di gioco; con la classe abbiamo partecipato all'Ora del Codice e i ragazzi si sono così approcciati per la prima volta alla programmazione.

Per tutto il primo anno e parte del secondo il coding è stato sperimentato con esercitazioni via via sempre più complesse fino ad arrivare e progetti autonomamente sviluppati dagli stessi alunni.

A tal proposito vorrei documentare una UdA multidisciplinare incentrata sulla mitologia, il cui obiettivo era quello di approfondire la conoscenza di alcuni racconti mitologici appartenenti a culture e popoli diversi e metterli a confronto.

Fig.5. Introduzione del videogame

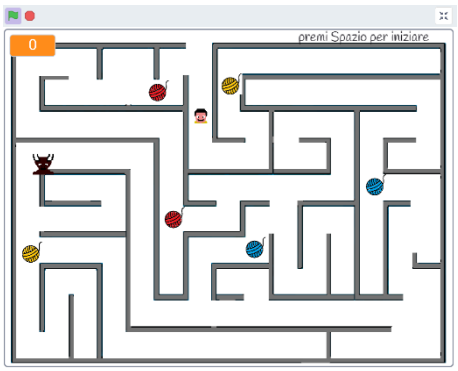


Fig.6. Parte interattiva del videogioco

Il prodotto finale sviluppato in modalità collaborativa dagli allievi è stato la realizzazione di una bacheca Padlet, con la descrizione di diverse tipologie di miti studiati con i docenti di Lettere e

Religione e l'inserimento dei lavori sviluppati con linguaggi specifici dalle altre discipline (Arte e Tecnologia). Con l'insegnante di Tecnologia l'attività si è incentrata sull'ideazione di un videogioco ispirato al mito del Minotauro, attraverso il linguaggio di programmazione Scratch. Gli alunni hanno creato un videogame con uno storytelling introduttivo al mito e poi una parte interattiva per giocare. Il link al videogame sul mito del Minotauro è stato inserito nel Padlet. I lavori realizzati con la docente di Arte che sono stati collezionati all'interno di un e-book.

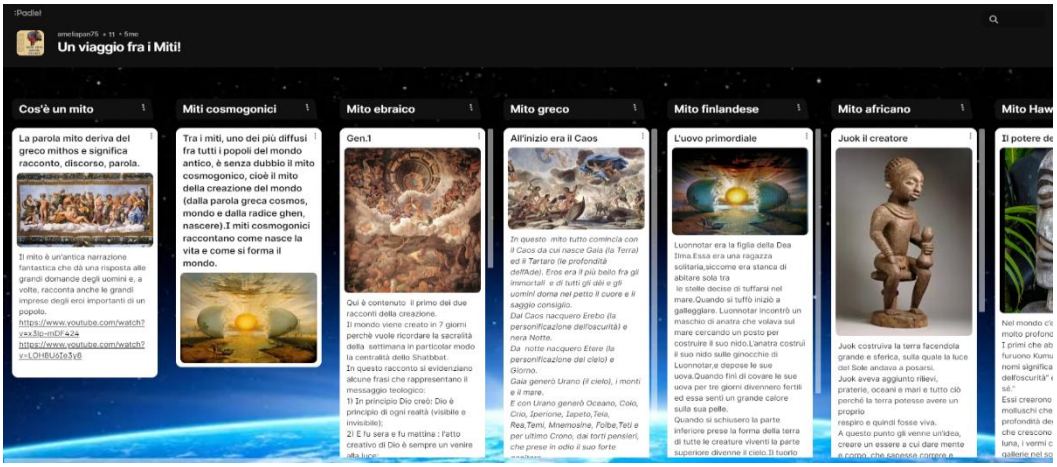


Fig.7. Bacheca Padlet

3.2 Robotica

Il coding è stato approfondito nella classe seconda utilizzando l'app Lego Prime. L'esperienza dei laboratori di robotica educativa portati avanti quasi per un intero quadrimestre è stata forse la più entusiasticamente accolta dagli allievi. Per la prima volta i ragazzi si sono trovati a lavorare con i kit Lego, appena acquistati. Ho progettato questo segmento didattico mirando al *problem solving*. La robotica educativa è un approccio semplice e pratico alla comprensione della programmazione informatica, utile all'apprendimento sperimentale di materie tecniche; insegna a pensare in maniera creativa, ragionare in modo sistematico e lavorare in maniera collaborativa. L'UdA si è svolta a partire da quattro attività-problema a difficoltà crescente. Per tutte era necessario assemblare un robot e programmarlo, utilizzando gli strumenti logici del coding

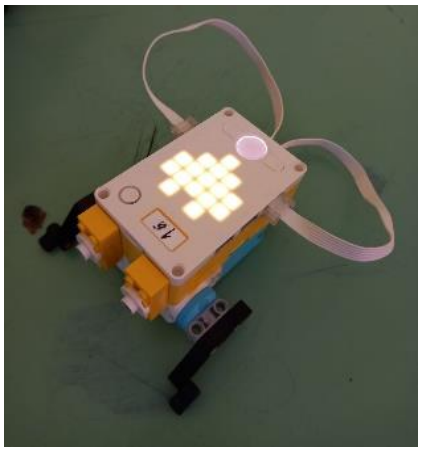


Fig.8. Cavalletta robot

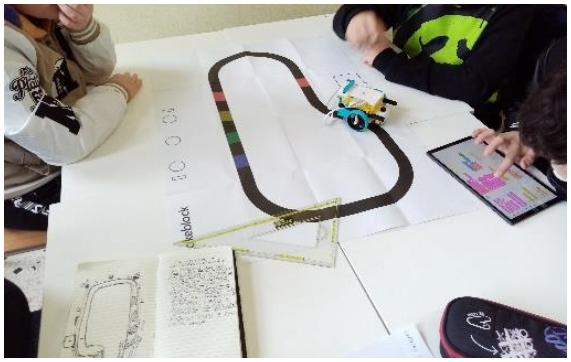


Fig.9. I ragazzi alle prese con la programmazione

e del pensiero computazionale, per far funzionare i dispositivi nel modo più efficiente rispetto allo scopo.

L'assetto di lavoro è stato per gruppi eterogenei. Ho spiegato per ogni attività-problema quali risorse erano a disposizione e lasciavo in autonomia i ragazzi lavorare puntando il timer per un tempo congruo dopo il quale ognuno doveva presentare il proprio lavoro.

Durante il laboratorio si è osservato l'approccio al problema, la strategia che i discenti sceglievano di seguire, l'impegno dei singoli componenti dei gruppi, il grado di collaborazione e interazione interna degli stessi.

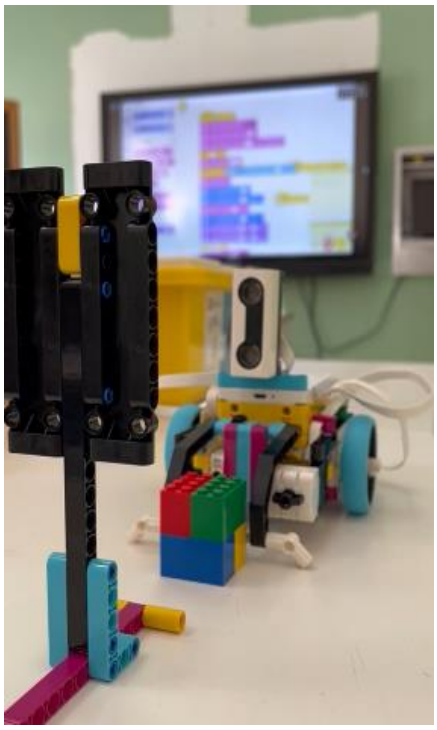


Fig.10. Motrice con sensore di distanza

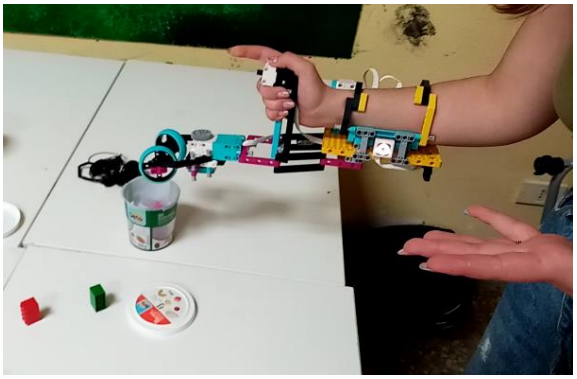


Fig.11. Prototipo di protesi e prove di presa

Fig.12. I gruppi sul progetto del braccio meccanico

3.3 Modellazione e stampa 3D

Il disegno tecnico è stato appreso attraverso i tradizionali strumenti a mano libera, ma parallelamente ho fatto sperimentare al terzo anno alcuni softwares CAD.

Ho cercato di trasmettere sempre un approccio progettuale alla materia, in modo da far manipolare il disegno tecnico come linguaggio veicolante un'idea.

Le proiezioni ortogonali, una volta acquisito il metodo, sono state lo strumento per la rappresentazione dell'aula e la riprogettazione del layout d'arredo. Il rilievo dell'ambiente è servito per la realizzazione di un una rappresentazione al computer con rendering.

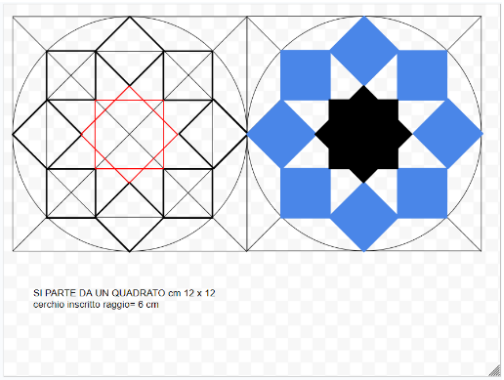


Fig.13. Disegno geometrico con Google Disegni

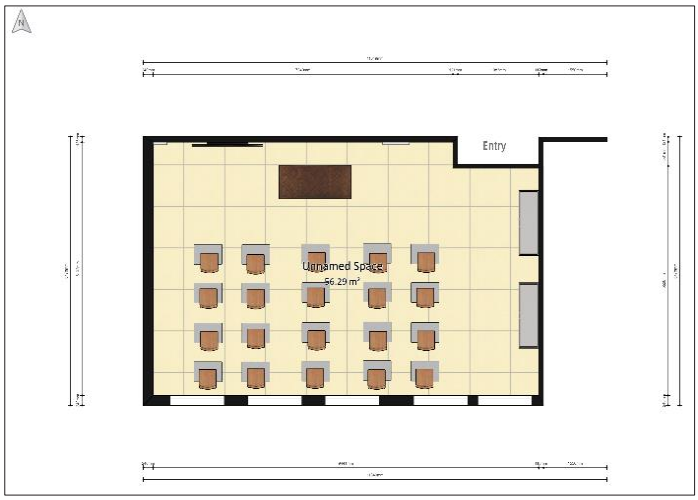


Fig.14. Pianta quotata dell'aula

Insieme allo studio delle assonometrie ho portato avanti anche il disegno tridimensionale con Tinkercad.

In questo ambiente di disegno 3D, ho guidato i ragazzi nell'elaborazione di modelli dal design essenziale, ciò facendo venivano presentati i vari comandi, l'utilità di alcuni strumenti, i ragionamenti in fase di disegno, portando a far conoscere loro come modellare oggetti 3D e far loro riconoscere da quali primitive i modelli possano essere creati.

Una volta consolidate tali conoscenze, ho lasciato

Fig.15. Render di progetto

svolgere in autonomia altre modellazioni arrivando anche alla modellazione 3D con Codeblocks, ovvero il coding applicato alla modellazione 3D.

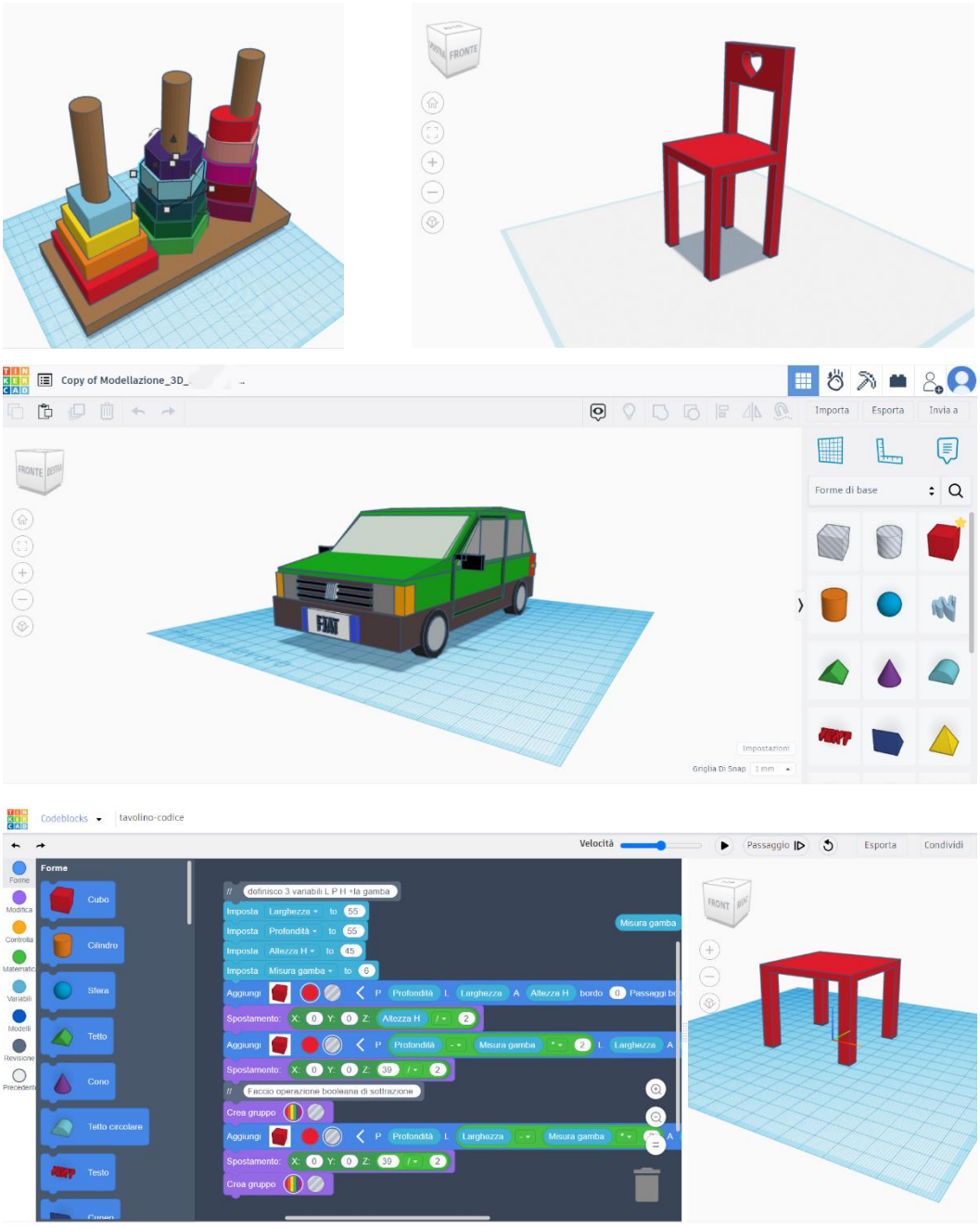


Fig.16. Modelli 3D disegnati dagli alunni

Questo è stato il punto di partenza per introdurre la stampa 3D. È stato spiegato il concetto di *slicing* e si è fatto loro sperimentare un software per predisporre la stampa impostando parametri in maniera ragionata. Sono stati mostrati anche le eventuali operazioni di finitura post produzione.
Una panoramica sulla stampante e i suoi componenti è stata fatta davanti la macchina e stampando un oggetto per capire il processo di realizzazione e controllare le varie fasi.
Alcuni si sono molto appassionati all'argomento che in sede di esami di Stato hanno portato la stampa di un modello 3D e ne hanno discusso durante il colloquio.

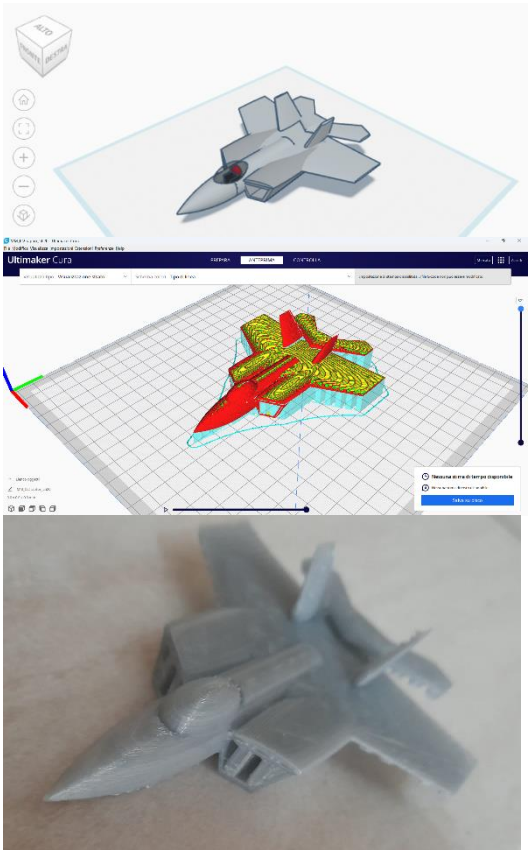


Fig.17. Modello di aereo

Fig.18. Targa dell'aula STEM

Fig.19. Modello del Colosseo

4 Conclusioni

Il bilancio di questi tre anni sperimentali è estremamente positivo. Le modifiche delle strategie didattiche e delle metodologie usate, grazie al digitale, insieme al lavoro di equipe del CdC, hanno determinato livelli di apprendimento elevati e buone competenze digitali e trasversali. A tal proposito, tutti gli alunni che ne hanno fatto richiesta, al termine del triennio, hanno conseguito la certificazione Eipass Junior.

L'informatica e il pensiero computazionale, che forniscono competenze trasversali, necessitano di tempi e modi propri per un adeguata acquisizione iniziale. Spesso nella scuola dell'obbligo questi sono carenti e non strutturati. La nostra esperienza è stata possibile principalmente perché il nostro Istituto ha una cattedra di Potenziamento di Tecnologia, mediante la quale si rendono disponibili due ore aggiuntive di insegnamento settimanali, dedicate allo sviluppo della competenza digitale.

Il perseguimento degli obiettivi del PNSD necessita di tali opportunità nella didattica corrente.

Quindi se si vuole portare avanti quanto auspica il PNSD, si dovrebbe investire in termini di tempo e risorse proprio nella Tecnologia, disciplina che, invece, a partire dal 2008, ha visto ridurre le sue ore da tre a due settimanali.

Orientare al digitale per applicare nuove metodologie educative, nella prospettiva del XXI secolo.

Bibliografia

[1] Chaudron S., Di Gioia R., Gemo M.; "Happy Onlife! Progetti ed attività per avviare e attrezzare bambini, docenti e genitori a una vita digitale piacevole, equilibrata e sicura", 2015, Joint Research Center, Commissione Europea.

[2] Stephanie Carretero, Riina Vuorikari, Yves Punie; "DigComp 2.1: The Digital Competence Framework for Citizens with eight proficiency levels and examples of use", 2017. (http://europa.eu/!Yg77Dh) a cura dell'European Commission's Joint Research Centre.

[3] Comitato Scientifico Nazionale per le Indicazioni Nazionali per il curricolo della scuola dell'infanzia e del primo ciclo di istruzione (a cura di); "Indicazioni nazionali e nuovi scenari", 2017

[4] AA.VV.; "Educazione civica digitale", Sillabo a cura di SIC Italia, 2018 (https://www.generazioniconnesse.it/site/it/educazione-civica-digitale/)

[5] MIUR, Linee Guida per la Didattica Digitale integrata.

[6] MIUR, Linee Guida per l'insegnamento dell'Educazione civica.

[7] MIUR, Piano Nazionale per la Scuola Digitale

[8] Annibale Pinotti; "Competenze digitali. Laboratorio di Informatica Compiti di realtà," Atlas, Bergamo 2020

Metodi educativi dell'informatica per studenti con disturbi di apprendimento nella scuola secondaria di secondo grado

Alessandra De Vitis[1], Guglielmo Abbruzzese[2]
[1] Università "La Sapienza", Roma; I.I.S. "Guglielmo Marconi", Latina
devitis@di.uniroma1.it
[2] I.I.S. "Guglielmo Marconi", Latina
g.abbruzzese@iismarconilatina.edu.it

Abstract

Data la crescente presenza di alunni con difficoltà di apprendimento (DSA) nelle scuole (MIUR, 2022), questo studio ha lo scopo di riportare e proporre metodi di supporto all'insegnamento semplici e divertenti orientati alla didattica inclusiva, concentrandosi nello specifico su dislessia e ADHD (Burgstahler & Cory, 2010) (Wolf, Brown, & Bork, 2009) (May & Stone, 2010) (Successful Strategies for Teaching Students with Learning Disabilities, 2023) (How to Teach a Child With Learning Disabilities, 2023) (Student Services, 2023), dei quali è stata fatta anche esperienza pratica per diversi anni nell'ambito della scuola secondaria di secondo grado, più specificamente per istituti tecnici tecnologici (informatica e telecomunicazioni).

1 Introduzione

Gli studenti in generale mostrano apprezzamento per l'ausilio di giochi, piattaforme e dispositivi che rendono lo studio apparentemente meno impegnativo a parità di obiettivi educativi da conseguire; tali strumenti rendono inoltre le differenze tra studenti con e senza disturbi di apprendimento difficilmente apprezzabili.

Considerando le specificità delle capacità e dei singoli stili di apprendimento, le piattaforme online possono costituire strumento atto sia a mantenere alta l'attenzione degli studenti ADHD che a facilitare le attività laboratoriali per quelli dislessici; tali piattaforme hanno inoltre la flessibilità di adattarsi ad età eterogenee, semplificando potenzialmente l'apprendimento degli argomenti più ostici.

Nell'ipotesi che gli studenti possano lavorare su progetti di proprio interesse e nell'ottica di proporre le proprie creazioni in consessi esterni, è possibile riscontrare ad esempio in *Tinkercad* (https://www.tinkercad.com/, s.d.) e Arduino (https://www.arduino.cc/, s.d.) molte caratteristiche che li rendono strumenti idonei al perseguimento di tali obiettivi.

La versatilità e l'utilità in ambito didattico di Arduino non sono ormai in discussione essendo suffragate da numerose ricerche ed esperienze (Cipollone, 2017). Arduino è una piattaforma elettronica open source (sia hardware che software) che ha rivoluzionato l'apprendimento sperimentale di applicazioni ICT orientate all'hardware, rivelandosi utile ad una vasta e trasversale comunità mondiale (non solo a studenti e professionisti, ma ad esempio anche ad artisti) per la sua semplicità d'uso e per la quantità di contributi costantemente incrementati per condivisione.

 ISBN: 978-88-6629-075-9

L'hardware minimale è accessibile a prezzi contenuti, sia nella versione originale sia nelle numerose versioni clone che l'apertura del progetto ha favorito nel corso del tempo. Sono facilmente reperibili dei kit preconfezionati – solitamente in versione principiante, intermedio, esperto – predisposti con componenti grazie ai quali è possibile realizzare una serie di esperimenti guidati dai titoli decisamente evocativi: "Interfaccia per astronave", "Indicatore d'umore", "Tastiera musicale", "Sfera di cristallo" ne sono alcuni esempi – con evidenza ideati per stimolare l'interesse dello studente. Obiettivo condivisibile è accompagnare l'utilizzatore a sviluppare abilità e competenze grazie alle quali, con gli elementi disponibili, l'unico limite alla progettazione sia rappresentato dalla sua fantasia.

Le attività con Arduino UNO[1] vengono quindi proposte sia a singoli individui che a piccoli gruppi (studiati con attenzione all'inclusività ed operando la commistione tra più dotati e più deboli) nell'ottica della formazione al lavoro in team: viene favorito il *brainstorming*, l'interazione con altri gruppi e con l'insegnante, che agisce da facilitatore stimolandone l'indipendenza.

Seppure debbano concentrarsi maggiormente nella fase di realizzazione, l'esperienza ha dimostrato che studenti ADHD possono partorire ottime idee, che il gruppo contribuisce a realizzare concretamente dimostrando così in maniera fattiva il valore del lavoro di squadra. Lo stimolo alla focalizzazione valido per gli studenti ADHD si è dimostrato efficace anche nelle situazioni certificate di dislessia. Dal punto di vista didattico, gli studenti acquisiscono conoscenze e abilità in maniera naturale e automatica; nel corso delle attività, infatti, devono imparare argomenti diversi e solitamente si spingono in autonomia ben oltre gli obiettivi curricolari (De Vitis, 2019).

2 Esperienza Operativa

La metodologia descritta nel presente articolo riporta esperienza pluriennali maturate sul campo, suffragate peraltro da solide basi teoriche testimoniate in bibliografia.

Sintetizzando quanto introdotto nel precedente paragrafo, possiamo schematizzare i pilastri di questa sperimentazione metodologica come segue:

- *Teamworking*. Si propone il lavoro in piccoli gruppi, importanti per condividere competenze diverse. I dislessici e gli ADHD si sentono rassicurati in tali contesti.
- *Making*. Realizzare qualcosa dal nulla è meraviglioso, e fortemente educativo.
- *Presenting*. Presentare il proprio lavoro al pubblico rende orgogliosi gli studenti.

Quando si insegna un nuovo argomento è opportuno stimolare la curiosità degli studenti senza fornire eccessivi dettagli, in modo da indurli a farsi e a porre domande. Vanno incoraggiati alla discussione tra pari in maniera da carpire eventuali livelli di conoscenze pregresse sull'argomento. È importante far intendere che teoria e pratica rappresentano due facce della stessa medaglia, quindi durante la lezione si coinvolgono gli studenti nell'analisi visiva e tattile dei componenti, favorendo dove possibile l'esercizio delle abilità manuali. Lo scopo di questo lavoro è presentare i risultati di questa metodologia applicata allo specifico bacino di utenza rappresentato da studenti del terzo e quarto anno di un istituto tecnico informatico; si è scelto di focalizzarsi sul microcontrollore *Arduino UNO* (https://www.arduino.cc/, s.d.), avendolo reputato per esperienza strumento idoneo per il raggiungimento degli obiettivi preposti. L'adozione di tale strumento si introduce non prima del termine del primo quadrimestre del terzo anno, in modo da avere un quadro iniziale più chiaro sul gruppo classe, e viene chiesto di realizzare un progetto con i seguenti vincoli:

- gli studenti hanno circa un mese per realizzare il progetto;

[1] Arduino UNO è il microcontrollore base della famiglia Arduino, selezionato per scopi didattici

- gli studenti realizzano parte del progetto in classe e parte a casa a causa dei limiti di tempo.

I passaggi essenziali (step) per sviluppare le competenze personali di ogni studente e in particolare dei DSA sono i seguenti:

Step 1. *Brainstorming*;
Step 2. *Testing*;
Step 3. *Making*;
Step 4. *Presenting*.

2.1 Step 1: Brainstorming

Gli studenti vengono suddivisi in gruppi da due o da tre; ci si assicura che gli alunni DSA vengano assegnati ai gruppi composti da tre studenti.

Ogni gruppo deve ragionare ed ipotizzare un oggetto da realizzare con Arduino UNO. La fase di brainstorming è la chiave del successo; gli insegnanti si limitano a supervisionare e consigliare senza imporsi, permettendo agli studenti di commettere errori e di imparare da essi. Gli insegnanti nominano uno studente come leader di ciascun gruppo; il leader deve essere in grado di coordinare e aiutare il gruppo ad organizzare al meglio i compiti.

Il leader può chiedere suggerimenti agli insegnanti, che tenderanno a supportare senza mai imporre la loro opinione.

È fondamentale che gli studenti lavorino collettivamente; le idee più rilevanti, in questa particolare fase, provengono statisticamente più spesso da studenti con ADHD, mentre quelli dislessici si distinguono sull'aspetto della possibile realizzazione.

Figura 1 – Brainstorming e discussione progetti

In Figura 1 gli studenti stanno cercando di individuare un tema di progetto utilizzando Arduino UNO. Ogni gruppo collabora allo studio di un problema diverso e anche se in ciascun gruppo è presente un DSA, non si nota alcuna differenza con i non DSA sia a livello di partecipazione che di contributi.

2.2 Step 2: Testing

Una volta definito il progetto inizia la fase successiva di prototipazione e test: il processo per realizzare il progetto e la ricerca dei materiali necessari.

In questa fase, gli studenti con ADHD nel gruppo tendono a cercare vari materiali, gli studenti dislessici creano una mappa mentale, ad esempio con bubbl.us (https://bubbl.us/, s.d.). Una volta reperiti i materiali, ogni gruppo stabilisce chi scriverà il codice, chi assemblerà il lavoro, chi scriverà il report e chi fotograferà le varie fasi; Ognuno fa il proprio lavoro senza risparmiare sforzi.

Figura 2 – Fase di prototipazione e test

La Figura 2 mostra la fase prototipale in cui ogni membro del gruppo sta testando parte del progetto.

2.3 Step 3: Making

Figura 3 - Making

Segue la fase di realizzazione del progetto. Gli studenti solitamente perfezionano il loro lavoro a scuola. In questa fase ogni membro di ciascun gruppo sa esattamente cosa fare. Tutti stanno facendo del loro meglio e sono orgogliosi del loro contributo, fino al raggiungimento del prodotto finale. Il terzo e il quarto passo sono cruciali per l'autostima in particolare per il DSA: essi diventano infatti consapevoli delle proprie competenze, mostrandole anche ai propri compagni in ottica di una naturale ed efficace inclusione; nella Figura 3, ad esempio, uno studente sta completando la realizzazione di una casa intelligente.

2.4 Step 4: Presenting

Alla fine del mese previsto, gli studenti presentano il loro lavoro alla classe e se possibile a tutta la scuola, in una mostra o in un concorso a premi; non si esclude la possibilità di iscriversi e partecipare a fiere.

Figura 4 - Presentazione del lavoro a scuola

Il giorno della presentazione, gli studenti sono tutti ugualmente emozionati e danno il massimo. Dal punto di vista didattico, si ottiene generalmente che gli studenti acquisiscano conoscenze, abilità e soprattutto competenze in modo estremamente naturale.

E' soprattutto in questa fase che i ragazzi dislessici e con ADHD interiorizzano la concretezza della possibilità di relazionarsi alla pari con i non DSA.

Figura 5 - Presentazione del lavoro in fiera

3 Risultati

Le esperienze proposte nel corso degli anni hanno testimoniato risultati interessanti in termini di attenzione, comprensione, conoscenza, abilità e competenze anche in situazioni di capacità eterogenee tra gruppi di lavoro. Laddove uno studente dislessico o ADHD frequenti con costanza le attività laboratoriali proposte sia nel secondo biennio che nel quinto anno l'insegnante è messo in condizione di tarare proposta educativa e strumenti per ottimizzare il suo percorso al fine di raggiungere gli obiettivi desiderati.

Statistiche empiriche incoraggiano il percorso in quanto, al termine di ogni anno scolastico, porzione significativa degli studenti DSA acquisisce maggiore autostima, padroneggiano concetti importanti delle unità di apprendimento in modo più confortevole; la maggior parte di loro raggiunge gli obiettivi formativi, proiettandosi con maggiore sicurezza nel percorso lavorativo o universitario.

4 Conclusioni

La scuola sta cambiando rapidamente, e il numero di studenti problematici che risulta statisticamente in costante aumento suggerisce alla classe docente un percorso di adattamento progressivo ai vari contesti educativi; gli insegnanti devono infatti comprendere e stimolare gli studenti individuando ed adeguandosi alle specificità dei loro stili di apprendimento.

In conclusione, insegnanti e gli studenti possono beneficiare di questi nuovi strumenti tecnologici per apprendere a tutto tondo: imparare a conoscere, imparare a fare, imparare a convivere e condividere, imparare ad essere.

Bibliografia

Burgstahler, S., & Cory, R. (2010). *Universal design in higher education: from princioples to practice".* cambridge: Harvard Education Press.

Cipollone, C. (2017). *Laboratorio di robotica per l'esplorazione scientifica.* INDIRE.

De Vitis, A. (2019). Methods In Computer Science Education In High Schools. *EDULEARN19 Proceedings.* Barcelona.

How to Teach a Child With Learning Disabilities. (2023, 07 12). Tratto da verywellfamily: https://www.verywellfamily.com/teaching-strategies-for-learning-disabled-students-2162342

https://bubbl.us/. (s.d.).

https://www.arduino.cc/. (s.d.).

https://www.tinkercad.com/. (s.d.).

May, A. L., & Stone, C. A. (2010). Stereotypes of individuals with learning disabilities: views of college students with and without learning disabilities. *Journal of learning disabilities*(10.1177/0022219409355483), 483-499.

MIUR, U. d. (2022). *I principali dati relativii agli alunni con DSA aa.ss. 2019/2020 - 2021/2022* . MIUR.

Student Services. (2023, 07 12). *Disability Learning*. Tratto da Student Services: https://www.ws.edu/student-services/disability/teaching/learning.shtm

Successful Strategies for Teaching Students with Learning Disabilities. (2023, 07 12). Tratto da LDA - Learning Disabilities Association of America: ldaamerica.org/successful-strategies-for-teaching-students-with-learning-disabilities/

Wolf, L., Brown, J. T., & Bork, G. R. (2009). *Students with Asperger syndrome: A guide for collega personnel.* Shawnee Mission: Autism Asperger Pub. Co.

La donna e i valori dello sport

Cristina Virili[1], Letizia Acciarino[2]
[1] D.D. Terzo Circolo
cristina.virili@posta.istruzione.it
[2] D.D. Terzo Circolo
letizia.acciarino@gmail.com

Abstract

Il Progetto "La donna e i valori dello sport" intende sviluppare il pensiero critico negli alunni della scuola dell'infanzia e della primaria al fine anche di promuovere le competenze digitali, essenziali per dei buoni cittadini che sanno muoversi in modo responsabile in rete. Per raggiungere tale obiettivo è stata scelta la tematica dei valori dello sport per abbattere preconcetti e differenze. L'alunno pertanto, con questo Progetto, non solo ha potuto approfondire la cultura e le conoscenze del mondo digitale e dei sistemi ICT ma è stato incoraggiato a sperimentare, a risolvere problemi, a trovare soluzioni alternative. Ciò facendo ha messo in campo le life skills e ha preso consapevolezza dell'intelligenza emotiva, processi, questi, essenziali per poter capire e rispettare se stesso e gli altri.

1 Introduzione

Il Progetto presentato nel paper, finalizzato a promuovere negli alunni della scuola dell'infanzia e della primaria senso critico e competenze per una cittadinanza digitale responsabile, si fonda sui valori dello sport per abbattere pregiudizi e stereotipi, in particolare sulla presenza femminile nelle attività sportive. L'intento, pertanto, è quello di favorire la cultura delle pari opportunità come prevenzione della violenza contro le donne, riflettendo sulle differenze legate al genere e sulla valorizzazione di queste diversità come fonte di ricchezza.

Si è sviluppato e sperimentato un curricolo verticale trasversale che mira all'attivazione di processi di insegnamento e apprendimento che permettano di favorire un'ampia gamma di competenze chiave per adattarsi in modo flessibile a un mondo in rapido mutamento e caratterizzato da forti interconnessioni.

Ciò ha coinvolto più di 200 bambini, le insegnanti, le famiglie, le istituzioni e le associazioni del territorio.

2 Il Progetto "La donna e i valori dello sport"

Il Progetto nasce in collaborazione tra la D.D. Terzo Circolo di Foligno (PG) e l'associazione nazionale "Nel nome del Rispetto odv", con il contributo del Comune di Foligno.

La Scuola ha proposto all'associazione "Nel Nome del Rispetto odv" di sviluppare un progetto il cui tema era quello della differenza di genere e della violenza sulla donna. Dopo una condivisione collegiale e partecipata, si è concordato che non si può prevenire e contrastare atteggiamenti violenti con il linguaggio della violenza perché non si può risolvere il problema restando nel problema stesso. Quindi si è scelta la tematica della donna e lo sport così che potesse nel contempo essere attrattiva per i bambini e riguardare i pregiudizi e gli stereotipi legati al ruolo del genere femminile nei diversi contesti di vita. Si è altresì convenuto di valorizzare e diffondere l'uso consapevole degli strumenti digitali presenti a scuola. Le attività spaziano dalla ricerca sul web, alla creazione di storie

interattive, alla realizzazione di cartoni animati, fino alla presentazione pubblica nel corso di una manifestazione cittadina.

Tutte le riflessioni e i lavori proposti sono introdotti da Gatto Rispetto, graziosa mascotte ideata dall'Associazione "Nel nome del Rispetto odv" che con il suo modo di accogliere i racconti degli altri, di ascoltare con attenzione e partecipazione senza esprimere giudizi di valore, dispensa "Pillole di saggezza" senza la presunzione di essere al di sopra di tutti, e arriva così al cuore dei più piccoli. La figura del gatto ha aiutato i bambini a capire il fair play. Ha conquistato la loro fiducia e gli ha fatto capire che è indispensabile prendere consapevolezza della propria vita affettiva, imparare a gestire le proprie rabbie, ansie e frustrazioni in modo costruttivo.

I valori dello sport infatti sono importanti per abituarsi a strutturare il proprio tempo, a controllare il proprio carattere e gestire i conflitti, a rispettare con sacrificio e coraggio l'impegno preso e i tempi da questo richiesti, a rispettare le regole e gli avversari.

Con il Progetto "La donna e i valori dello sport" i bambini sono stati sensibilizzati, vivendo esperienze dirette in situazioni reali e virtuali, a prendere coscienza dei valori dello sport. L'obiettivo è stato quello di sviluppare una comunicazione autentica attraverso tutti i mezzi, un dialogo aperto, per conoscere e superare i conflitti interpersonali e favorire il benessere attraverso il rispetto reciproco. I bambini hanno ricercato, selezionato e valutato delle informazioni sul web; hanno comunicato correttamente con altri nel mondo digitale ricevendo innumerevoli stimoli; hanno creato e rielaborato contenuti tratti dal web. Ciò ha permesso di poter indirizzare l'uso del web verso obiettivi di realizzazione personale ed efficacia scolastica, difendendosi dall'uso eccessivo e dalle distrazioni che possono indurre a volte le nuove tecnologie.

Altresì l'educazione all'uso consapevole della rete e degli strumenti digitali con un approccio creativo, collaborativo, interattivo ha permesso la creazione di prodotti digitali da condividere con compagni, famiglie e territorio.

Il Progetto, altamente inclusivo, si è avvalso di ambienti di apprendimento efficaci e metodologie didattiche innovative (tinkering, cooperative learning, peer learning, circle-time, debate, flipped classroom). Le proprietà principali sono la flessibilità, l'adattabilità, la connessione continua con informazioni e persone, l'accesso alle tecnologie, alle risorse educative aperte, al cloud, apprendimento attivo e collaborativo, creatività.

3 Fasi del Progetto

Il progetto ha seguito le fasi di seguito sinteticamente descritte:

- La D.D. Terzo Circolo-Foligno ha bandito un concorso per gli alunni sulla tematica "La donna e lo sport": "Il tema proposto è quello che lo sport è sempre stato ritenuto una prerogativa maschile, ma negli ultimi decenni le donne sono entrate a far parte di questo mondo esclusivo, portando a casa non solo medaglie meritate sul campo, ma anche vittorie morali e civili. Gli stereotipi sono duri a morire e anche nello sport, come in qualsiasi altro ambito, essere donne equivale a partire in svantaggio, tuttavia cambiare è possibile."
- I bambini hanno ricercato, selezionato e valutato delle informazioni sul web, per conoscere la storia delle sportive di maggiore spicco e ne hanno esaminato le qualità umane. Per una corretta e fruttuosa comunicazione con gli altri nel mondo digitale, hanno incontrato e intervistato in presenza o in modalità virtuale campioni sportivi. Hanno letto articoli, libri, visionato documentari e film. Hanno attinto in rete materiali elaborati da altre scuole (Programma il Futuro, scuoladigitale.istruzione.it,…). Per la creazione e rielaborazione di contenuti nel web hanno assistito a eventi on line organizzati dal Ministero dell'Istruzione e del Merito in occasione del Safer Internet Day e dal festival delle scienze di Roma.
- Si sono costituiti dei gruppi di lavoro che hanno condiviso il materiale su drive. Ciò ha permesso di lavorare sia in presenza che da remoto.

- I bambini hanno inviato al concorso singolarmente o a gruppi, svariati elaborati, ideando storie interattive, video, disegni, elaborati scritti, presentazioni e fumetti in formato digitale.
- Una giuria composta da personalità (presidenti di associazioni locali e nazionali, professionisti nel settore dell'informatica, scrittori, giornalisti) ha scelto i lavori più significativi.
- È stato creato, da parte di un esperto che si è confrontato anche con i bambini, un prodotto multimediale che ha previsto l'animazione dei lavori scelti dai giurati. I campioni sportivi intervistati, sono stati trasformati in cartoni animati e con la loro voce, introducono i lavori vincitori del concorso indetto a scuola. Infine, una galleria d'arte digitale mostra i disegni realizzati da tutti i bambini. Questi ultimi hanno registrato tutti i dialoghi e le letture in uno studio di registrazione con dei professionisti del suono.
- L'intero percorso è stato documentato con un power point, inserendo sia il progetto che i lavori presentati al concorso dai bambini.
- C'è stata una restituzione finale in un incontro tenutosi all'Auditorium della città con il supporto degli enti locali, aperto ai genitori e alla cittadinanza.

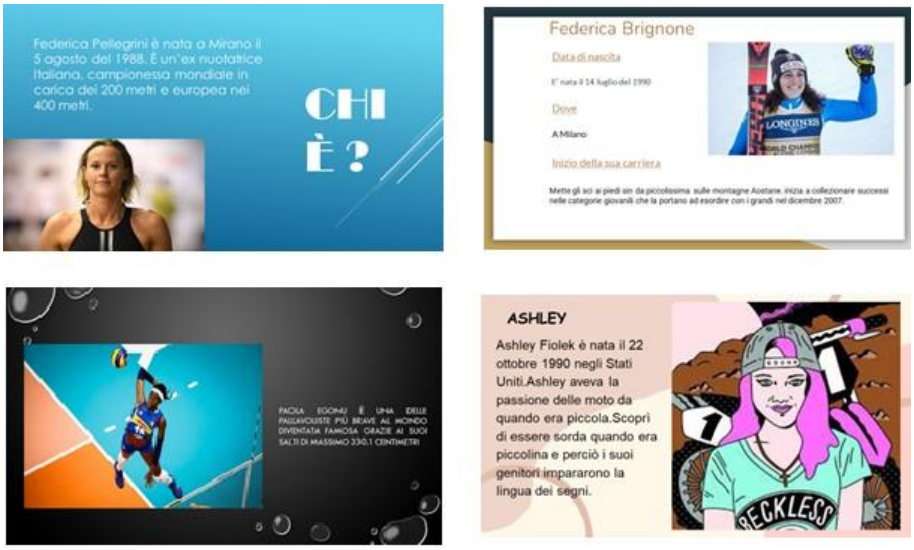


Figura 1 Alcune slide realizzate dai bambini

Figura 2 Alcuni bambini della scuola dell'Infanzia

Figura 3 Alcuni bambini della scuola Primaria

Figura 4 Gatto Rispetto e lo sport. **Ogni uso, sfruttamento o commercializzazione delle immagini di Gatto Rispetto in modo parziale o totale, senza il permesso scritto dell'associazione "Nel nome del Rispetto odv", che è titolare del marchio e del logo, è da considerarsi illegale e perseguibile a termini di legge. In particolare ciò è valido per ogni forma di riproduzione fotostatica, di copiatura, di duplicazione di qualsiasi tipo, di traduzione, di microfilm, di duplicazione e conservazione elettronica o di informatica. Tutti i diritti sono riservati.**

Conclusioni

Questa sinergia e vicinanza di intenti ha favorito la sperimentazione, da parte della scuola, di un curricolo verticale trasversale, destinato alla scuola del primo ciclo. Il percorso è stato declinato in tutti i campi di esperienza e in tutte le discipline. Le riflessioni emerse dall'immenso lavoro progettuale non ha nulla a che vedere con qualcosa di compiuto o come un compendio esaustivo di prassi consolidate. È solo l'incipit e un invito a socializzare e a condividere con tutta la comunità scolastica un percorso di buone prassi educative per una scuola di qualità. Si tratta, in buona sostanza, di un circuito innovativo di "azioni corali" i cui protagonisti sono i bambini e i ragazzi.

Applicazione della Didattica Breve con approccio Agile

Guglielmo Abbruzzese[1], Alessandra De Vitis[1,2]
[1] I.I.S. "Guglielmo Marconi", Latina
g.abbruzzese@iismarconilatina.edu.it
[2] Università "La Sapienza", Roma,
devitis@di.uniroma1.it

Abstract

In questo momento storico la classe docente di ogni ordine e grado si trova ad affrontare situazioni non semplici; come spesso accade, la rapidità degli eventi e la mancanza strutturale di flessibilità del sistema non consentono alla scuola di rispondere con le necessarie velocità ed adeguatezza ai contesti di attualità.

A livello globale, la pandemia e l'avvento delle intelligenze artificiali rappresentano le principali (ma non le uniche) sfide che stanno costringendo i docenti a mettere in discussione l'impianto delle proprie strategie di comunicazione, insegnamento e valutazione; inoltre, non giovano i continui tentativi di ogni nuovo governo di lasciare impronte sempre nuove sul sistema scolastico italiano (con interventi non necessariamente armonizzati con gli impianti didattici e burocratici preesistenti). Urgono pertanto riflessioni serie e approfondite sui paradigmi di insegnamento/apprendimento per la terza decade di questo nuovo millennio.

Questo contributo propone alcune riflessioni sull'utilizzo di strumenti non esclusivamente recenti, e che combinati insieme possono essere utili per affrontare le sfide della modernità provando ad armonizzare il meglio del vecchio e del nuovo.

Partendo da considerazioni generali di progettazione didattica sulla base delle più recenti tendenze – anche nell'ottica di proporre strumenti per coadiuvare le attività di formazione degli insegnanti – la *Didattica Breve con approccio Agile* si propone di differenziare e potenziare le tecniche di progettazione e realizzazione delle UDA (Unità Didattiche di Apprendimento) soprattutto per le materie di area scientifico-tecnologica, combinando da un lato l'esperienza professionale e la conoscenza della disciplina, dall'altro la gestione della classe e delle attività didattiche mutuata dal *Project Management* (PM) aziendale.

Keywords: didattica breve, agile, scrum, PCK, epistemologia, informatica, metodologie didattiche

1 Introduzione

Si propongono alcune considerazioni preliminari di carattere pedagogico incentrate sui concetti di *comprensibilità* e *comprensione* (Borsese, 2020). Nella progettazione delle UDA l'insegnante dovrebbe tenere esplicitamente conto della relazione tra i due concetti, essendo il primo condizione necessaria ma non sufficiente per il verificarsi del secondo. Prima di mettere a punto un metodo che favorisca la comprensione (nella quale è implicata la componente affettivo-relazionale), bisogna infatti considerare il vincolo della comprensibilità (nel quale la componente cognitiva è preponderante) fondata su accurate scelte linguistiche. L'insegnante dovrà quindi operare in modo

che i contenuti trattati e il linguaggio usato siano alla portata dei suoi allievi; è solo nel passaggio dalla comprensibilità alla comprensione che sono di fondamentale aiuto tutti gli studi e le ricerche sul metodo e sulla relazione.

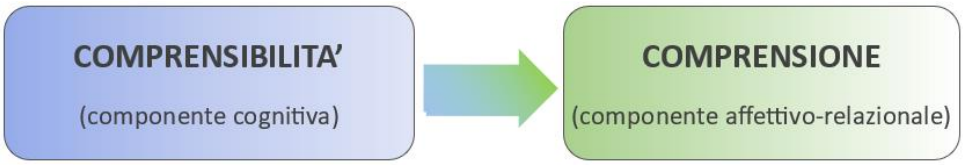


Figura 1 - Rapporto tra comprensibilità e comprensione

Per quanto riguarda l'informatica e le tecnologie informatiche, la dimensione epistemologica può rivelarsi particolarmente utile per il suo impatto diretto sulla didattica della disciplina (Forlizzi, 2015) (da Rosa, Chmiel, & Gòmez, 2016). In altre parole, il paradigma epistemologico determina la conoscenza del contenuto pedagogico dell'informatica, a partire dai suoi quattro nuclei fondanti: algoritmo, macchina, linguaggio e informazione (Dowek, 2011).

L'impatto degli studi filosofici sul processo di insegnamento e apprendimento della *Computer Science* (CS) è un campo pressoché inesplorato, nonché importante per la cosiddetta *Computer Education* (CEd), che non può prescindere dalla conoscenza pedagogica del contenuto (*Pedagogical Content Knowledge,* PCK) (Shulman, 1986). La ricerca filosofica e pedagogica può dunque suggerire l'elaborazione di modelli di insegnamento-apprendimento dell'informatica che poggino su solide basi teoriche.

1.1 Comprensibilità: didattica breve

Nel pieno del periodo COVID, poco dopo il primo spiazzante lockdown, nelle "Linee guida per una didattica integrata" (MIUR, 2020), si cita la *didattica breve* (DB) tra le metodologie didattiche consigliate.

L'aggettivo "breve" non si riferisce ad un impegno minore da parte del docente o del discente, ma ad un approccio meticoloso ai nuclei fondanti della disciplina, i cui contenuti sono sottoposti ad operazioni di *distillazione*.

Tale metodologia nasce dalla proposta di un docente universitario negli ultimi decenni del secolo scorso (Ciampolini, 1993) nell'ambito dell'istruzione tecnica per adulti, con lo scopo di massimizzare l'efficacia delle lezioni contraendo nel contempo i tempi di apprendimento pur *senza rinunciare al rigore scientifico e ai saperi essenziali* che si intende veicolare attraverso l'esperienza formativa. Tale approccio può costituire un ausilio fondamentale in situazioni di difficoltà generale (classi turbolente, didattica a distanza) o di tempi contingentati. Nel progettare le esperienze formative, il docente deve prestare attenzione ai seguenti aspetti caratterizzanti:

- ricerca educativa disciplinare ed epistemologica;
- distillazione dei contenuti in un'ottica di trasparenza e fruibilità;
- trasferibilità rapida dei contenuti;
- ricostruibilità agevole dei percorsi logici;
- stimolo alla creazione di strumenti espressivi propri, di logiche di ragionamento pulite e lineari;

  miglioramento delle capacità logico-espressive.

L'implementazione si articola nelle seguenti fasi:

- si considera la disciplina da un punto di vista generale;

- si scompongono i contenuti per individuarne i nodi concettuali;
- si opera una pulizia logica dei contenuti così ottenuti;
- si esegue una analisi disciplinare (*Ricerca Metodologico Disciplinare*);
- si approntano le sequenze di insegnamento.

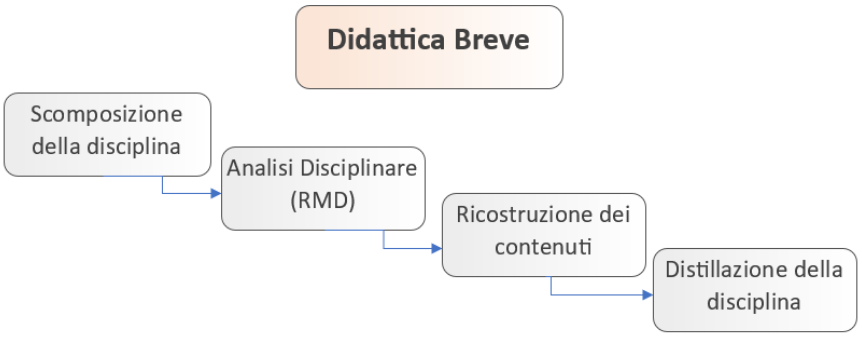


Figura 2 - Applicazione della DB

Nella DB si opera una distinzione tra *Distillazione Verticale* e *Distillazione Orizzontale*. In una materia, per *Distillazione Verticale* si intende evidenziare gli argomenti che la compongono, sequenziandoli esattamente come il docente li esporrà agli studenti; con la *Distillazione Orizzontale* si invita lo studente a suddividere la spiegazione di un argomento dato in passi *elementari*, così che la sequenza dei passi indicati gli consenta di ricostruire agevolmente la logica di dettaglio dell'argomento (ad esempio, i passaggi logici che portano alla dimostrazione di un teorema).

Figura 3 – Distillazione verticale

La metodologia adotta inoltre l'idea di *didattica a carte scoperte*: ogni lezione è partecipata e condivisa tra studenti e docenti attraverso l'uso guidato dei ragionamenti e degli strumenti utilizzati.

1.2 Comprensione: eduScrum

Scrum (Sutherland & Sutherland, 2015) è una metodologia AGILE ed utilizzata soprattutto nella gestione di progetti complessi e dinamici; si è imposta nei decenni fino a diventare uno standard di PM per aziende leader di settore (Tesla, Google, etc.). È basata su un approccio *iterativo* e *incrementale* che permette di rispondere ai cambiamenti e di fornire *valore* in modo rapido ed efficiente. La metodologia Scrum si basa su tre pilastri principali: *trasparenza*, *ispezione* e *adattamento*; prevede la suddivisione del lavoro in cicli chiamati *sprint*, periodi di tempo fissati in cui sono pianificati, eseguiti, valutati e migliorati i compiti assegnati. Ogni sprint ha una durata limitata, e si conclude con la consegna di un *incremento di prodotto* potenzialmente utilizzabile.

Nel processo Scrum, il lavoro viene organizzato in un *backlog* del prodotto, che è un elenco di compiti (cui è assegnata una priorità) chiamati *user story*. Durante la pianificazione dello sprint, una parte selezionata del backlog del prodotto viene trasferita nel backlog dello sprint, che rappresenta gli elementi di lavoro che verranno completati durante lo sprint.

La metodologia Scrum prevede anche ruoli specifici. Il *Product Owner* è responsabile della definizione delle priorità e delle caratteristiche del prodotto, il *Team di sviluppo* è responsabile della realizzazione delle attività, mentre lo *Scrum Master* si occupa di garantire che il processo Scrum sia seguito correttamente e rimuovere eventuali ostacoli, agendo da *Servant Leader*.

eduScrum (RU, 2023) è una variante di Scrum specificamente adattata all'ambito educativo. L'obiettivo principale di eduScrum è quello di insegnare agli studenti le competenze di gestione del tempo, del lavoro di squadra e del raggiungimento degli obiettivi. Utilizzando la struttura di Scrum, gli studenti apprendono come organizzare e gestire il proprio lavoro in modo autonomo (Missiroli, Russo, & Ciancarini, 2016).

EduScrum mantiene molti dei principi basilari dello Scrum. Tuttavia, ci sono alcune differenze significative: ad esempio, nel contesto educativo, i ruoli di Scrum Master ed Product Owner possono essere ricoperti da un *insegnante* o un *tutor* che guida gli studenti nella definizione degli obiettivi e delle priorità. Inoltre, gli sprint potrebbero avere una durata più breve per adattarsi al contesto scolastico.

Dunque il valore aggiunto di EduScrum, rispetto ad altre metodologie didattiche, riguarda specificamente lo sviluppo dell'autonomia e della responsabilità negli studenti, coadiuvati da insegnanti esperti nella *facilitazione* della gestione dell'apprendimento (attraverso il ruolo dello scrum master), attraverso processi implementati con modalità iterative di durata variabile denominati *sprint*.

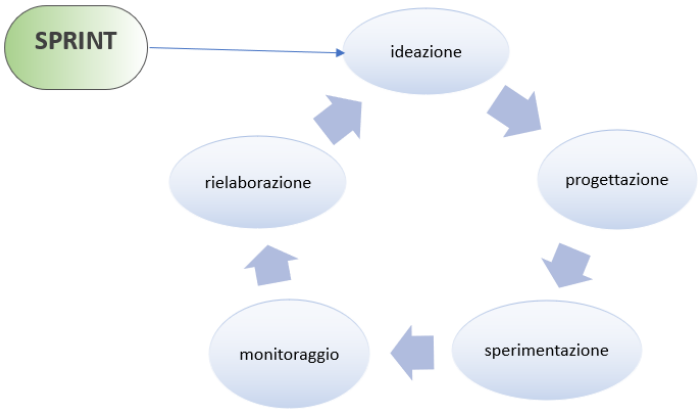


Figura 4 - eduScrum - fasi operative di uno *sprint*

2 Esempio applicativo

Il prototipo operativo presentato in questo capitolo non intende proporsi come trattazione esaustiva di tutte le sfumature metodologiche e tecniche inerenti alla progettazione e all'implementazione di una UDA, che si ritengono sottese; si considera inoltre prerequisito la conoscenza di un argomento classico utilizzato nell'ambito delle telecomunicazioni, "*Lo standard protocollare per le reti wireless*", presentando una sintesi di alcune delle schede di lavoro sviluppate per la situazione specifica.

2.1 Inquadramento UDA

UDA: Lo standard protocollare per le reti wireless

Collocazione curricolare

- Tipo scuola: Istituti tecnici – Settore tecnologico.
- Indirizzo: Informatica e Telecomunicazioni – Articolazione Informatica.
- Materia: Sistemi e Reti.
- Anno: IV (Livello MAC – 802.11 - Wireless).
- Periodo dell'anno: I° quadr (Nov-Dic).

Tempo complessivo stimato

- Durata:
  - Preparazione attività: 8h.
  - Didattica in aula: 1h (preliminare) +5h (attività) +3h (Laboratorio).

Prodotti attesi

- Elaborati di gruppo sulle basi teoriche delle reti wireless LAN.
- Report di gruppo sulle attività sperimentali assegnate.

OSA (Obiettivi Specifici di Apprendimento)

- Conoscenze: conoscere i componenti di una rete wireless; apprendere le topologie e gli standard di comunicazione wireless; comprendere per grandi linee il sistema di autenticazione 802.1x; rudimenti su modalità di sicurezza wireless; comprendere le problematiche legate alla condivisione del mezzo fisico; conoscere gli standard di comunicazione wireless.
- Abilità: analizzare il formato del frame 802.11; analizzare il livello fisico e la trasmissione dei segnali wireless; distinguere le varie topologie delle reti wireless.
- Competenze: saper rilevare il traffico wireless e saper individuare un frame 802.11 nel traffico di rete; applicare le strategie per ridurre o evitare le collisioni; sviluppare consapevolezza sui reali pericoli delle reti wireless discriminando tra le varie fonti di informazione.

Requisiti cognitivi preliminari:

- Conoscenza funzionale del modello ISO/OSI e/o TCP/IP.
- Conoscenza base del software di monitoraggio Wireshark.
- Conoscenza operativa dei codici ASCII ed esadecimale.
- Lingua Inglese: obiettivi minimi previsti per l'anno in corso.

Modalità di verifica dei requisiti cognitivi preliminari

- Brainstorming.
- Quiz a risposta chiusa (Google Forms, Kahoot, JotForm).

Modalità di verifica:

- Individuale: strutturata e semi strutturata.
- Di gruppo: valutazione degli elaborati attesi.

Eventuali attività di recupero

- Colloquio finalizzato ad auto valutazione.
- Recupero delle lacune in itinere.

2.2 Preparazione dei contenuti

Nella DB, per poter classificare e distillare i contenuti bisogna definire delle *macrologiche*[1] e delle *micrologiche*[2] sulle quali andranno applicate le cosiddette *etichette* (rif. Figura 5).

Si individuano per la trattazione dell'argomento tre macrologiche:

- M1: Wireless networks
- M2: Host to network. IEEE 802.11
- M3: Frame MAC 802.11

Se ne riporteranno in seguito alcuni stralci di mappatura con le micrologiche individuate.

#Id	Abbreviazione	Natura dei distillati
01	Ana	Analisi
02	Apl	Applicazione
03	Apx	Approssimazione
04	Clc	Calcolo
05	Clg	Collegamento
06	Cmp	Composizione
07	Cnc	Concetto
08	Cfr	Confronto
09	Cnv	Convenzione
10	Cor	Corollario
11	Crt	Criterio
12	Def	Definizione
13	Dsr	Descrizione
14	Dgs	Digressione storica
15	Dim	Dimostrazione
16	Exe	Esemplificazione
17	Spm	Fatto sperimentale
18	Int	Interpretazione
19	Ipt	Ipotesi
20	Lab	Laboratorio

21	Lex	Legge
22	Mtd	Metodologia
23	Mdl	Modello
24	Mot	Motivazione
25	Opr	Operazione
26	Oss	Osservazione
27	Pst	Postulato
28	Pre	Premessa
29	Prq	Prerequisito
30	Prc	Principio
31	Prp	Proprietà
32	Rgt	Ragionamento
33	Rrt.g	Rappresentazione grafica
34	Rgl	Regola
35	Rct.s	Ricostruzione storica
36	Snt	Sintesi
37	Str	Struttura
38	Teo	Teoria
39	Tes	Tesi
40	Tpl	Tipologia
41	Vlt	Valutazione

Figura 5 – etichette selezionate per l'argomento in distillazione verticale

Nella Figura 5 sono riportate le etichette individuate per l'argomento selezionato; in rosso si evidenziano quelle che si considerano più utili nell'ambito dell'informatica.

Le seguenti tabelle sono state compilate utilizzando alcune convenzioni per le colonne: *Sequenza base*, *DSTV* (elemento base distillato), *Tipi* (rif. Figura 5), *Zd*[3], *It*[4], *Ec*[5], *Macrologica/Tempi/Materiali*[6], *DSTH*.

[1] La *macrologica* guida le scansioni della distillazione verticale con la suddivisione degli argomenti in *blocchi* (dnità didattiche, moduli)

[2] La *micrologica* è la logica più fine che guida alunno e docente nell'esame di una definizione, di una dimostrazione, nell'analisi di un testo, nell'interpretazione di un tema.

[3] *Zoccolo duro*: parte di una disciplina che include i suoi principi fondanti, il quadro dei concetti teorici maggiormente caratterizzanti.

[4] *Iterazione:* argomenti che riprendono e approfondiscono tematiche già trattate.

[5] *Eccellenza:* argomenti di livello più elevato, facoltativi nel percorso di apprendimento.

[6] Tutti i materiali utilizzati per l'apprendimento: libri, dispense, audiovisivi, link.

Seq. base	DSTV	Tipi	Zd	It	Ec	Macrologica/Tempi Materiali didattici	DSTH
M1						**RETI WIRELESS**/2h	
1	Rete wireless	Def	x				
2	Vantaggi e svantaggi rispetto a rete cablata	Cfr		x			
3	Applicazione delle reti wireless	Apl		x			
4	Classificazione delle reti wireless	Crt	x			Schema 4	
5	WBAN	Dsr		x			
6	WPAN	Dsr		x			
7	WLAN	Dsr	x				
8	WMAN	Dsr		x			
9	WWAN	Dsr		x			
10	Confronto tra reti wireless	Cfr	x				
11	Apparati di rete: Wireless Router	Dsr		x			
12	Apparati di rete: Wireless Terminal (STA)	Dsr		x			
13	Apparati di rete: Access Point	Dsr		x			
14	SSID (Service Set Identifier)	Def	x				
15	Infrastruttura: Independent Basic Service Set (BSS)	Dsr	x			https://networklessons.com/cisco/ccna-200-301/wireless-lan-802-11-service-sets#Infrastructure_Mode	
16	Infrastruttura: Extended Service Set (ESS)	Dsr	x			Schema 16	
17	Wireless Distribution System	Dsr	x				
18	Reti Ad-Hoc	Mdl	x				
19	BSS: AP in Repeater mode	Dsr	x				
20	BSS: AP in Bridge mode	Dsr	x				
21	Configurare un AP in un BSS	Lab			x		
22	Problemi nella trasmissione wireless	Crt		x			23.24.25
23	Attenuazione del segnale	Cnc		x			
24	Interferenze da altre sorgenti	Cnc		x			
25	Effetto ombra	Cnc		x			
26	Storia delle comunicazioni wireless: da Marconi al 5G	Rct.s			x	Articolo in rivista ***link***	

Tabella 1 - mappatura macrologica M1

A differenza della distillazione verticale (DSTV) che viene messa a disposizione da parte dell'insegnante agli allievi, la distillazione orizzontale (DSTH) è messa a punto dal singolo studente che mette in relazione i vari elementi del distillato secondo una logica personalizzata, e che consente una comprensione da parte del docente del suo stile cognitivo. Ciascuno studente, infatti, può elaborare sequenze differenti per DSTH sulla base dello stesso DSTV.

Seq. base	DSTV	Tipi	Zd	It	Ec	Macrologica/Tempi Materiali didattici	DSTH
M2						**HOST TO NETWORK - IEEE 802.11x** / 2h	
1	Standard 802.11	Def	x				
2	Dettagli tecnici standard 802.11b,g,n a confronto	Dsr	x				
3	Tecniche di modulazione dei segnali wireless	Exe		x			
4	IEEE 802.11p, V2X				x	https://www.elettronicanews.it ***/soluzioni-wireless-per-lauto-connessa/***	
5	Sovrapposizione di segnali wireless	Mdl	x				
6	Collisioni nella rete wireless	Def	x				
7	Problema della stazione nascosta	Mod	x			Schema 7	
8	Problema della stazione esposta	Mod	x				
9	Request To Send (RTS) e Clear To Send (CTS)	Dsr	x			Schema 9	
10	Carrier Sense Virtuale	Cnc	x				
11	Limiti del VCS	Dsr		x			
12	Arbitration Inter Frame Space (AIFS)	Cnc	x				
13	DIFS, FIFS, PIFS, NAV (Finestra di contesa)	Def	x				
14	Carrier Sense Multiple Access/CA	Mod	x				
15	STA: Accesso alla rete (Scan., Auth., Assoc., Roam.)	Dsr	x				5,7,8,12,13
16	Sistema di autenticazione 802.1x (cenni)	Dsr		x			

Tabella 2 - mappatura macrologica M2

Seq. base	DSTV	Tipi	Zd	It	Ec	Macrologica/Tempi Materiali didattici	DSTH
M3						**FRAME MAC 802.11** / 1h	
1	Tipologie di pacchetti (controllo, gestione, dati)	Dsr	x				
2	WLAN physical layers (PMD→PLCP)→MAC	Def		x		**Schema *(Fig. 6 – frame MAC)***	
3	Frame macro struttura (Preamble, PLPC_H, MPDU)	Mod	x				
4	MPDU (struttura)	Mod		x			
5	MPDU Frame Control	Mod		x			M2.15, M2.16, 3,4
6	Pacchetti di controllo	Dsr					
7	Pacchetti di gestione	Dsr					
8	Possibili combinazioni dei bit ToDS e FromDS	Exe		x			5

Tabella 3 - mappatura macrologica M3

2.3 Lavoro con la classe

Sulla base dei riferimenti normativi L. 53/2003, DL 59/2004 e DL 226/2005, è possibile desumere una sostanziale affinità tra i concetti di *Unità di Apprendimento* e di *Sprint*, che di questa rappresenta una naturale e specifica implementazione. A seguire, un esempio di come i materiali preparati e proposti nel precedente paragrafo possano essere utilizzati nella proposizione delle lezioni costituenti una UDA.

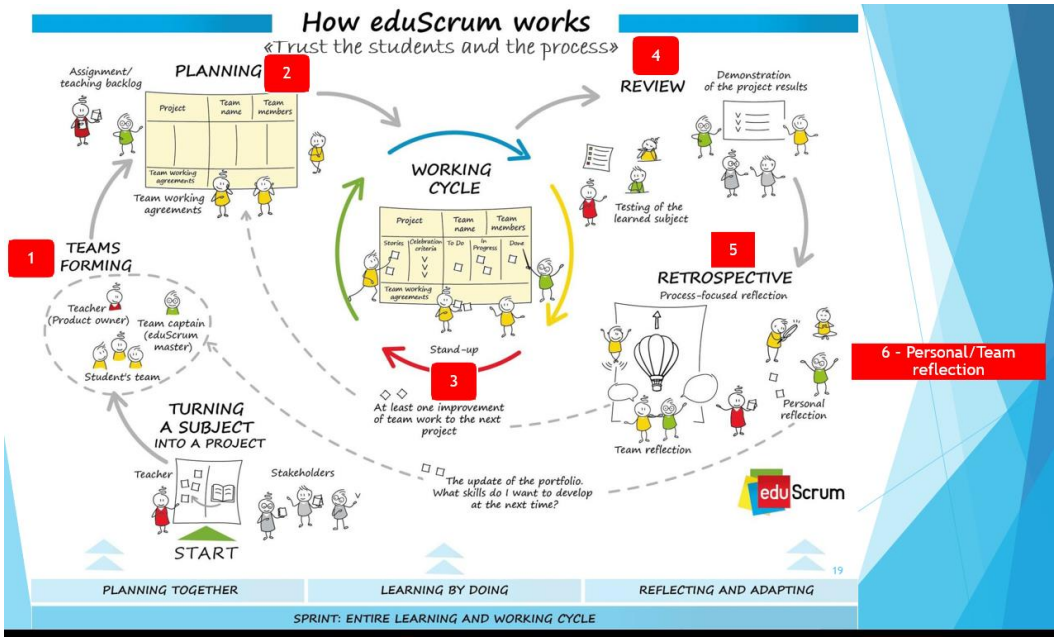


Figura 6 – Fasi del processo eduScrum (NL, 2023)

Fasi del processo

Preliminare: lancio dell'attività. Per lo specifico tema, si propone l'utilizzo di un articolo tratto dal sito della CNN (McFarland, 2020) nel quale si riporta un incidente avvenuto durante uno dei primi collaudi di veicoli a guida automatica, che ha causato la morte di un pedone: tale argomento consente l'aggancio con lo stack protocollare 802.11 wireless, in quanto IEEE 802.11x può implementare le comunicazioni *V2X* (Vehicle to Everything) utilizzate come standard dai sistemi di trasporto intelligenti (*ITS*, Intelligent Transport Systems); pur non essendo il trattamento dell'IEEE 802.11x esplicitato nelle linee guida ministeriali (se ne suggerisce infatti la trattazione dei più comunemente utilizzati 802.11*b*, *g* ed *n*) tale argomento può fungere da stimolo per approfondimenti individuali (nella sequenza M2.4 riportata in **Tabella 2**, l'elemento ha di fatto valorizzata la colonna *Ec*).

1. Formazione dei *team.* Lo scrum master (docente[7]) procede alla creazione (possibilmente spontanea, seppure assistita in certi aspetti) delle squadre di lavoro, definendo le regole che tutti dovranno condividere – come ad esempio il linguaggio che verrà utilizzato nelle fasi del processo o le definizioni dei concetti di *Ready* (pronto) e *Done* (completato), relativi ad un compito assegnato tra quelli da svolgere.
2. *Planning.* Nella fase di pianificazione lo scrum master propone a ciascun team un elenco di attività da svolgere, con una stima pesata dello sforzo necessario (*Effort)* evidenziando quelle prioritarie (*Priority*) per una comprensione efficace. I materiali ed i percorsi di lavoro sono

[7] coadiuvato eventualmente da uno o più studenti in formazione

quelli elaborati nelle fasi di distillazione. Le attività sono enucleate in *task* assegnabili atomicamente a singoli o gruppi (preferibilmente a singoli, per responsabilizzare al meglio).

PRODUCT BACKLOG			
BACKLOG ITEM (TO DO LIST)	Effort	Priority	Notes
... refinements here			
... refinements here			
... refinements here			
...			
Macrologic U1 – Wireless Networks	20	1	
Macrologic U2 – Host To Network IEEE 802.11x	18	2	
Macrologic U3 – Frame MAC 802.11	25	3	
...			
Lab 1 – Analyzing 802.11 frame using wireshark	20	4	
Lab 2 – Problems with wifi transmission - Absorption	15	4	
...			

Figura 7 – pianificazione (stralcio)

Working cycle. A seconda della complessità, le attività possono essere realizzate in uno o più cicli di lavoro, assimilabili alle lezioni, nelle quali ciascun elemento del team deve produrre un risultato *misurabile*. L'insieme dei cicli per il raggiungimento di un obiettivo di progetto condiviso individua uno *Sprint*. Con riferimento alla Figura 7, ad esempio, i 5 elementi *Macrologic U1..U3*, *Lab 1..2* potrebbero corrispondere macroscopicamente a 5 lezioni (3 teoriche, 2 di laboratorio) sulle quali ciascun team dovrà operare. Ad ogni ciclo si fa stretto riferimento ai concetti di *Inspection* (rivisitazione frequente degli obiettivi di apprendimento e dei progressi raggiunti) e di *Adaptation* (riprogrammazione degli obiettivi nel caso in cui questi risultino troppo ambiziosi).

3. *Review.* Al termine delle attività formanti lo *Sprint*, i team devono operare una *restituzione* attraverso una presentazione, o meglio una dimostrazione di quanto realizzato nelle attività laboratoriali. Nel *Laboratorio 2 – Problems with wifi transmission – Absorption,* si suppone ad esempio di proporre attività sperimentale con apparati wireless, disponibili anche nelle proprie abitazioni, operando degli esperimenti con un router wifi e un altro dispositivo che si desidera collegarvi (es. cellulare), utilizzando come parametri di test la distanza ed eventuali ostacoli frapposti tra i due elementi e dimostrando con misure reali quanto riportato nella seguente tabella esemplificativa:

Materiali	Indebolimento	Esempi
Aria	Nessuno	Spazio aperto, cortile
Legno	Debole	Porta, tavola, separatore
Plastica	Debole	Separatore
Vetro	Debole	Finestre non colorate
Vetro colorato	Medio	Finestre colorate
Mattoni	Medio	Muri
Gesso	Medio	Separatore
Ceramica	Alto	Pavimentazione
Cemento	Alto	Muri portanti, piani, piloni
Vetro blindato	Alto	Vetri anti-proiettili
Metallo	Molto alto	Cemento armato, specchi, armadi metallici, cabine di ascensori

Figura 8 – Obiettivi di verifica Lab. 2

4. *Retrospective.* Al termine di uno sprint i team si interrogano sugli elementi che potranno efficientare i processi di apprendimento in previsione del successivo; questa fase deve durare una frazione di lezione, in chiusura dell'UDA.
5. *Team reflection/Personal reflection.* Sono incoraggiate altresì, oltre alle riflessioni comuni in classe, anche quelle di team ed individuali per far maturare al discente maggiore consapevolezza di sé come entità sia autonoma che appartenente ad una squadra.

3 Conclusioni

Le linee guida di carattere pedagogico si prestano di frequente ad alcune distorsioni interpretative, veicolando in maniera perentoria il messaggio che qualunque proposta o soluzione possa considerarsi valida solo perché "innovativa" (Calvani, 2022); questo contributo propone di innestare modalità didattiche nuove senza rinunciare a impostazioni metodologiche già rodate. L'insegnamento dell'informatica può beneficiare della combinazione degli strumenti illustrati, sia per quanto riguarda i contenuti generali che per quelli specialistici. La letteratura scientifica contempla prevalentemente casi studio e proposte didattiche sulle tematiche di ordine generale della

disciplina, ma risulta carente relativamente ad argomenti specialistici (quali quelli proposti nelle materie di indirizzo degli ITT). La proposta qui illustrata può essere implementata a vari livelli di complessità, coniugando rigore metodologico, spirito di collaborazione e nel contempo sviluppo dell'autonomia nel discente. L'insegnante deve assicurarsi che il materiale didattico da proporre in classe, veicolando concetti e informazioni, coinvolga la componente cognitiva dei suoi allievi, che sia cioè comprensibile o meno in relazione al bagaglio cognitivo che possiedono. Dovrà quindi operare in modo che i contenuti che tratta e il linguaggio che usa siano alla portata dei suoi allievi. Per la comprensione la metodologia *eduScrum* può giocare un ruolo fondamentale, basandosi sui pilastri talvolta definiti come *4C*: *Creativity, Collaboration, Communication* e *Critical thinking* (creatività, collaborazione, comunicazione e pensiero critico).

Bibliografia

Borsese, A. (2020). Per una didattica che conduca a capire. *Orientamenti Pedagogici, 67*(4), 57-63.

Calvani, A. (2022, 07). *La vecchia scuola e il disastro attuale. Una riflessione trasversale tra ideologismi, responsabilità pedagogiche, evidenze e buon senso.* Tratto da S.Ap.I.E.: https://sapie.it/wp/wp-content/uploads/2022/07/La-vecchia-scuola-e-il-disastro-attuale.pdf

Ciampolini, F. (1993). *La Didattica Breve.* Bologna: Il Mulino.

da Rosa, S., Chmiel, A., & Gòmez, F. (2016). Philosophy of Computer Science and its Effect on Education - Towards the construction of an Interdisciplinary Group. *CLEI Electronic Journal, 19*(1), 1-10.

Dowek, G. (2011). Les quatre concepts de l'informatique. Sciences et technologies de l'information et de la communication en milieu éducatif : Analyse de pratiques et enjeux didactiques., (p. 21-29). Patras. Tratto il giorno 07 12, 2023 da https://www.fing.edu.uy/grupos/nifcc/material/2015/quatre.pdf

Forlizzi, L. (2015). Cambiare la didattica dell'informatica attraverso la conoscenze del suo contenuto pedagogico. In A. Labella, *E tutti chiamano Informatica: L'esperienza di TFA nelle discipline informatiche* (p. 93-104). Roma: Sapienza Università Editrice.

McFarland, M. (2020, 09 18). *Uber self-driving car operator charged in pedestrian death.* Tratto da CNN Business: https://edition.cnn.com/2020/09/18/cars/uber-vasquez-charged/index.html

Missiroli, M., Russo, D., & Ciancarini, P. (2016). Una didattica agile per la programmazione. *Innovazione: sfida comune di scuola, università, ricerca e impresa.* Udine: Università degli Studi di Udine. Tratto da https://www.danielrusso.org/files/2016Didamatica_AgileEducation.pdf

MIUR. (2020, 08). *MIUR.* Tratto il giorno 07 12, 2020 da https://www.miur.gov.it/documents/20182/0/ALL.+A+_+Linee_Guida_DDI_.pdf/f0eeb0b4-bb7e-1d8e-4809-a359a8a7512f

NL, e. T. (2023, 08 25). *How eduScrum works*. Tratto da eduscrum.org: https://eduscrum.org/how-eduscrum-works/

RU, e. t. (2023, 07 12). *eduScrum.* Tratto da eduScrum: https://eduscrum.com.ru/wp-content/uploads/2020/01/The_eduScrum-guide-English_2.0_update_21-12-2019.pdf

Shulman, L. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher,, 15*(2), 4-14.

Sutherland, J., & Sutherland, J. (2015). *Scrum: The Art of Doing Twice the Work in Half the Time.* London: Cornerstone.

Il linguaggio Python e il framework PyScript per l'insegnamento dell'Informatica

Giovanni Pedroncelli

IISS "G. Marconi - M. Hack", Bari, Italia
giovanni.pedroncelli@posta.istruzione.it

Abstract

Nonostante la sua enorme popolarità e il suo largo utilizzo in un elevato numero di ambiti in forte crescita come la data science e l'intelligenza artificiale, il linguaggio di programmazione Python non è ancora molto diffuso nella scuola italiana, almeno stando ai dati relativi alle adozioni dei libri di testo. Da un lato questo è causato da una carenza di offerta di libri di testo in Python, dall'altro da alcuni punti di debolezza come la sintassi troppo semplice e le basse performance. In questo articolo si propone l'uso del recente framework PyScript a scuola per permettere agli studenti di creare applicazioni Web dinamiche in svariati campi di interesse (come quello scientifico grazie al supporto nativo di pacchetti come `scipy`, `sympy`, `numpy` o del machine learning grazie al supporto di `scikit-learn`) sfruttando la potenza del linguaggio Python per l'elaborazione e la solidità e semplicità di uso di HTML e CSS per la gestione della parte strutturale e grafica. La proposta è corredata dallo sviluppo di una semplice applicazione per la risoluzione di problemi di analisi matematica con PyScript.

1 Stato dell'arte

Il linguaggio di programmazione Python è stato introdotto da Guido Von Rossum nel 1998 [1]. Secondo il TIOBE Programming Community index[1], aggiornato a luglio 2023, è il linguaggio di programmazione più popolare al mondo e lo è stato per tre volte negli ultimi cinque anni. Si analizzano in questa sezione i più significativi contributi in letteratura scientifica riguardanti il linguaggio Python in ambito scolastico seguendo tre linee di ricerca:

- Una prima linea di ricerca analizza le possibili applicazioni del linguaggio Python all'interno della scuola. Oltre ad alcuni studi relativi alla scuola primaria [2, 3, 4] e secondaria di primo grado [5], nell'ambito della scuola secondaria di secondo grado il linguaggio Python è stato proposto come strumento per la risoluzione di equazioni differenziali [6], per l'introduzione alla biologia molecolare [7] e alla bioinformatica [8], per lo studio della crittografia [9] e delle probabilità [10] e per l'introduzione ai concetti di base del calcolo numerico [11].
- Un'altra linea di ricerca si occupa di presentare i risultati di studi di ricerca-azione sull'uso di Python a scuola: Jeffrey Elkner è stato il primo a proporre il linguaggio Python per lo studio dell'informatica all'interno delle scuole superiori [12] ed evidenzia alcuni aspetti positivi come la semplificazione del processo di insegnamento dell'Informatica (la spiegazione del classico programma "Hello World" in C++ richiede 13 paragrafi contro i 3 richiesti dallo stesso programma scritto in Python). Gli autori di [13] approfondiscono

[1] Il TIOBE index è un indicatore della popolarità dei linguaggi di programmazione aggiornato una volta al mese utilizzando il numero di query nei motori di ricerca su Internet. Per maggiori dettagli si rimanda al sito ufficiale `www.tiobe.com`.

 ISBN: 978-88-6629-075-9

gli studi di Elkner e analizzano sistematicamente i vantaggi e svantaggi nell'uso di Python come linguaggio di programmazione nelle scuole superiori. In [14] gli autori dimostrano empiricamente che gli studenti che imparano a programmare usando Python commettono meno errori sintattici e logici rispetto a studenti che usano il linguaggio Java.

Tutti i contributi analizzati sono concordi nell'affermare che a scuola non ci si dovrebbe focalizzare sull'insegnamento di un linguaggio bensì sui concetti fondamentali della programmazione: il linguaggio dovrebbe permettere a studenti e docenti di concentrarsi sull'apprendimento di algoritmi e pensiero computazionale senza gravarli dei problemi relativi alla sintassi e ad un ambiente di sviluppo complicato da utilizzare.

- Altri contributi si dedicano ad analisi comparate e sistematiche tra Python e altri linguaggi di programmazione molto usati in ambito didattico come C++ [15, 16] o Java [17, 18]. Le conclusioni sono discordanti: mentre gli autori di [15] evidenziano come gli studenti mostrino maggiore soddisfazione nell'apprendere Python ritenendo più semplici le strutture condizionali e iterative rispetto a C++, in [16] si segnalano maggiori difficoltà da parte degli studenti alle prese con Python anche se il fattore più rilevante risulta essere il docente con le sue scelte relative ad esercizi e tecniche; gli autori di [19] incoraggiano un approccio integrato prevedendo un primo approccio alla programmazione tramite Python e un passaggio successivo a Java per la programmazione orientata agli oggetti.

2 Il linguaggio Python nella scuola italiana

Un'analisi è stata condotta relativamente ai libri di testo in adozione per l'anno scolastico 2022-2023 su un consistente campione di terze classi di Istituti Tecnici Tecnologici ad indirizzo Informatica-Telecomunicazioni con articolazione Informatica (d'ora in poi, ITI) e Licei Scientifici Opzione Scienze Applicate (d'ora in poi, LSSA) in scuole statali.

Indirizzi	N. scuole	N. obbligatorio	% obbligatorio	N. consigliato	% consigliato
ITI	289	7	2.422	6	2.076
LSSA	121	12	9.917	2	1.653

Tabella 1: Adozione libri di testo con Python nelle scuole statali italiane

I risultati sono presentati nella Tabella 1. La prima colonna indica il numero totale di scuole prese in esame; la seconda e terza colonna indicano rispettivamente il numero totale e la percentuale di scuole in cui è adottato con acquisto obbligatorio un libro di testo incentrato sul linguaggio Python; le ultime due colonne contengono rispettivamente il numero totale e la percentuale di scuole in cui è consigliato un libro di testo di Informatica in Python (solitamente affiancato a un libro obbligatorio in C, C++ o Java). Le classi a indirizzo ITI non raggiungono nemmeno il 5% di libri in Python mentre i licei si assestano attorno al 10%. Se compariamo i risultati di questa analisi con i libri effettivamente disponibili per l'adozione nelle scuole, i risultati non sorprendono per le classi ad indirizzo ITI: le pochissime classi che adottano o consigliano un libro in Python usano [20] oppure [21] (un volume maggiormente orientato agli studi universitari) perché le case editrici non offrono altre soluzioni. Le percentuali delle classi LSSA sono migliori (ma non soddisfacenti) perché anche l'offerta editoriale è decisamente più consistente: alcune classi adottano il già citato [20] e altre si affidano a [22] oppure [23].

3 Il framework PyScript

PyScript è un framework sperimentale per l'esecuzione di script in Python nel browser Web ed è stato presentato da Peter Wang in occasione del Pycon US 2022 nel corso di un keynote speech[2]. In Figura 1 è mostrata l'archiettura di PyScript. Il codice Python, attraverso il porting di Pyodide, può essere interpretato da WebAssembly (in figura abbrebiato con la sigla WASM)[3] che lo converte in un formato binario eseguibile dal browser. Le istruzioni Python sono incluse direttamente in un file .html all'interno del tag `<py-script>`.

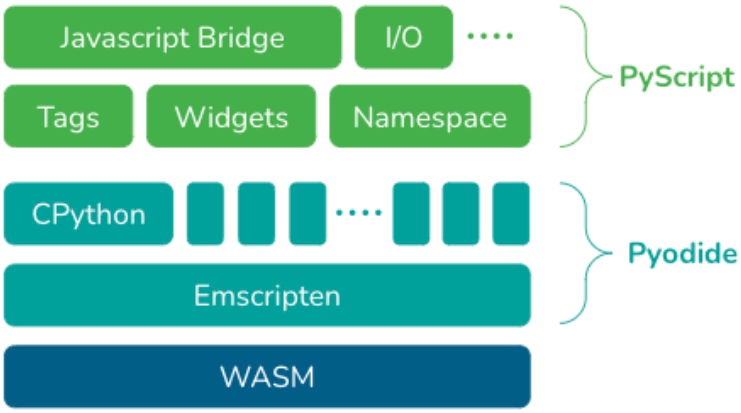


Figura 1: Architettura di PyScript (`https://anaconda.cloud/pyscript-python-in-the-browser`).

Il framework PyScript permette di implementare applicazioni in Python che fanno uso dei pacchetti più diffusi[4] direttamente nel browser. PyScript facilita anche la procedura di deployment del servizio Web che potrà essere pubblicato attraverso qualunque servizio di hosting (altri servizi Web basati su popolari pacchetti Python come Flask o Django permettono una scelta più limitata per il deployment, soprattutto se gratuito). In ambito scolastico, si tratta di vantaggi importanti: anche se non è consigliabile[5], si potrebbe persino evitare di insegnare altri linguaggi per il Web a parte HTML e CSS perché il codice in Python permette in un'unica soluzione di gestire gli aspetti back-end e front-end del sito. Tuttavia, è necessario evidenziare anche alcuni svantaggi legati all'uso di PyScript: il numero di pacchetti disponibili nativamente all'interno di Pyodide è abbastanza limitato e include solo pacchetti di uso molto comune; le prestazioni delle applicazioni Web costruite con PyScript sono decisamente inferiori a quelle realizzate con JavaScript, come evidenziato in [24]. Si ritiene, tuttavia, che questi svantaggi non abbiano un grande impatto in ambito didattico.

[2] La registrazione dell'intervento è disponibile al link `https://youtu.be/qKfkCY7cmBQ`.

[3] Maggiori dettagli sul sito ufficiale `https://webassembly.org`.

[4] L'elenco dei pacchetti Python utilizzabili è disponibile al link `https://pyodide.org/en/stable/usage/packages-in-pyodide.html`.

[5] PyScript non è stato inteso come framework sostitutivo di JavaScript come evidenziato nelle FAQ al link `https://docs.pyscript.net/latest/reference/faq.html`; anzi, gli sviluppatori di PyScript hanno previsto metodi per il passaggio di oggetti tra i due linguaggi e rendono disponibile il pacchetto JS di PyScript che permette di utilizzare istruzioni JavaScript per effettuare operazioni in cui PyScript è ancora carente come la manipolazione del DOM di una pagina Web.

3.1 Esempio di applicazione con PyScript

In questa sezione si presenta un'applicazione PyScript realizzata unicamente usando i linguaggi HTML, CSS e Python, intesa come un progetto interdisciplinare di Informatica e Matematica. Trattandosi di un'applicazione molto semplice, è realizzabile facilmente da studenti con competenze di medio livello in Python, HTML e CSS. In Figura 2 è mostrato il codice completo in Python racchiuso dal tag `<py-script>`. Il codice completo è disponibile nella repository GitHub `https://github.com/giovannipedroncelli/analisi-matematica-pyscript` e l'applicazione è utilizzabile al link `https://giovannipedroncelli.github.io/analisi-matematica-pyscript/`.

L'applicazione ha una struttura e una grafica minimali: l'utente può inserire all'interno di un campo di input testuale una funzione matematica; se tale funzione è in formato accettabile essa viene stampata in formato matematico, è mostrato il suo grafico su un piano cartesiano e l'utente può scegliere che cosa calcolare tra quattro scelte possibili: derivata, integrale indefinito, integrale definito (fornendo limite inferiore e superiore di integrazione) o limite (fornendo il valore a cui deve tendere il limite). Una schermata è mostrata in Figura 3.

Sono stati importati solo quattro pacchetti riportati anche all'interno del tag `<py-config>`: `sympy` per i calcoli matematici, `ziamath` per la stampa delle funzioni matematiche, `matplotlib` e `numpy` per la stampa dei grafici. Leggendo il codice è possibile notare come i prerequisiti richiesti ad uno studente per lo sviluppo di un'applicazione del genere siano molto limitati: oltre all'uso di variabili, è sufficiente saper definire e chiamare funzioni e usare pacchetti esterni[6]. Un'applicazione costruita in questo modo permette inoltre allo studente di maturare competenze relative alla struttura del DOM di una pagina Web e alla sua manipolazione, seppure non attraverso JavaScript ma attraverso l'API Element sviluppata all'interno di PyScript.

4 Conclusioni e considerazioni finali

Questo articolo ha evidenziato come, nonostante la sua enorme e crescente popolarità, Python non sia ancora diffuso adeguatamente all'interno della scuola italiana, se consideriamo i dati relativi all'adozione dei libri di testo. Viene quindi proposto il framework PyScript come strumento didattico per permettere di raggiungere l'obiettivo delle Indicazioni Nazionali, sezione Linee Generali e Competenze per il Liceo Scientifico Opzione Scienze Applicate: "Ha una sufficiente padronanza di uno o più linguaggi per sviluppare applicazioni semplici, ma significative, di calcolo in ambito scientifico"[7].

Per quanto riguarda gli Istituti Tecnici Tecnologici ad Indirizzo Informatica, sarebbe opportuno, anche per preparare gli studenti ad un mondo del lavoro in cui le professioni collegate all'intelligenza artificiale sono in fortissima crescita, introdurre il linguaggio Python se non nella disciplina denominata Informatica, almeno in altre discipline afferenti all'area tecnica come Tecnologie e Progettazione di Sistemi Informatici e di Telecomunicazioni. Secondo l'ultimo report Future of Jobs del World Economic Forum pubblicato il 30 aprile 2023[8], gli specialisti di intelligenza artificiale e machine learning sono in cima alla lista dei posti di lavoro con crescita più rapida; PyScript può essere di aiuto anche allo sviluppo di applicazioni Web di machine learning grazie al supporto del pacchetto `scikit-learn`.

[6] Consultando il codice completo attraverso la repository GitHub è possibile notare che in HTML, oltre alle competenze di base relative a tag e attributi, è fondamentale conoscere i form, in CSS sono sufficienti competenze di base

[7] cfr.pagina 369 del documento consultabile a questo link: `https://www.indire.it/lucabas/lkmw_file/licei2010/indicazioni_nuovo_impaginato/_decreto_indicazioni_nazionali.pdf`.

[8] Il report è disponibile al link `https://www3.weforum.org/docs/WEF_Future_of_Jobs_2023.pdf`.

```python
# importa pacchetti
import sympy as sp
import ziamath as zm
import matplotlib.pyplot as plt
import numpy as np
# individuazione elementi DOM in base a valore id
x = sp.symbols('x')
out = Element("out")
definito = Element("def")
limite = Element("limite")
scelte = Element("scelte")
scrivi = Element("scrivi")
disegno = Element("disegno")

def reset():
  out.clear() #cancella contenuto div cpn id = "out"
  definito.add_class("hidden")
  limite.add_class("hidden")

def disegna(f): #grafico funzione (passata come parametro) con matplotlib
  lam_x = sp.lambdify(x, f, modules=['numpy']) #conversione in funzione per numpy
  assex = np.linspace(-10,10) #comandi matplotlib per stampa grafico
  assey = lam_x(assex)
  fig,ax = plt.subplots()
  plt.plot(assex,assey)
  plt.grid(True)
  plt.show()
  disegno.write(fig) #stampa grafico funzione su div con id="disegno"

def conferma_fun(): #l'utente scrive la funzione
  reset()
  global f
  try: #blocco try-except per evitare errori di inserimento funzione
    f = sp.sympify(Element("funzione").value) #funzione convertita in formato sympy
    scelte.remove_class("hidden")
    out.remove_class("errore")
    scrivi.write(zm.Text("Funzione: $y={}$".format(sp.latex(f))))
    disegna(f)
  except:
    scrivi.clear()
    disegno.clear()
    scelte.add_class("hidden")
    out.write("Funzione in formato errato, riprova.")
    out.add_class("errore") #stampa messaggio di errore

def conferma_scelta(): #l'utente sceglie cosa calcolare
  reset()
  selezionato = Element("opzioni").value
  if selezionato=="DER":
    der = sp.diff(f, x) #calcolo derivata con sympy
    out.write(zm.Text("La derivata della funzione è ${}$".format(sp.latex(der))))
  elif selezionato=="INTI":
    inti = sp.integrate(f, x) #calcolo integrale indefinito con sympy
    out.write(zm.Text("L'integrale indefinito della funzione è ${}$".format(sp.latex(inti))))
  elif selezionato=="INTD":
    definito.remove_class("hidden") #scelta limiti integrale definito
  elif selezionato=="LIM":
    limite.remove_class("hidden") #scelta a cosa tende limite

def conferma_int(): #calcolo integrale definito
  reset()
  li = Element("inf").element.value #valore limite inferiore da campo form
  ls = Element("sup").element.value #valore limite superiore da campo form
  intd = sp.integrate(f, (x, li, ls)) #calcolo integrale definito con sympy
  out.write(zm.Latex(sp.latex(intd))) #stampa valore integrale definito

def conferma_lim(): #calcolo limite
  reset()
  t = Element("tende").element.value #valore a cui deve tendere il limite
  lim = sp.limit(f,x,t) #calcolo limite con sympy
  out.write(zm.Latex(sp.latex(lim))) #stampa valore limite
```

Figura 2: Codice Python dell'applicazione

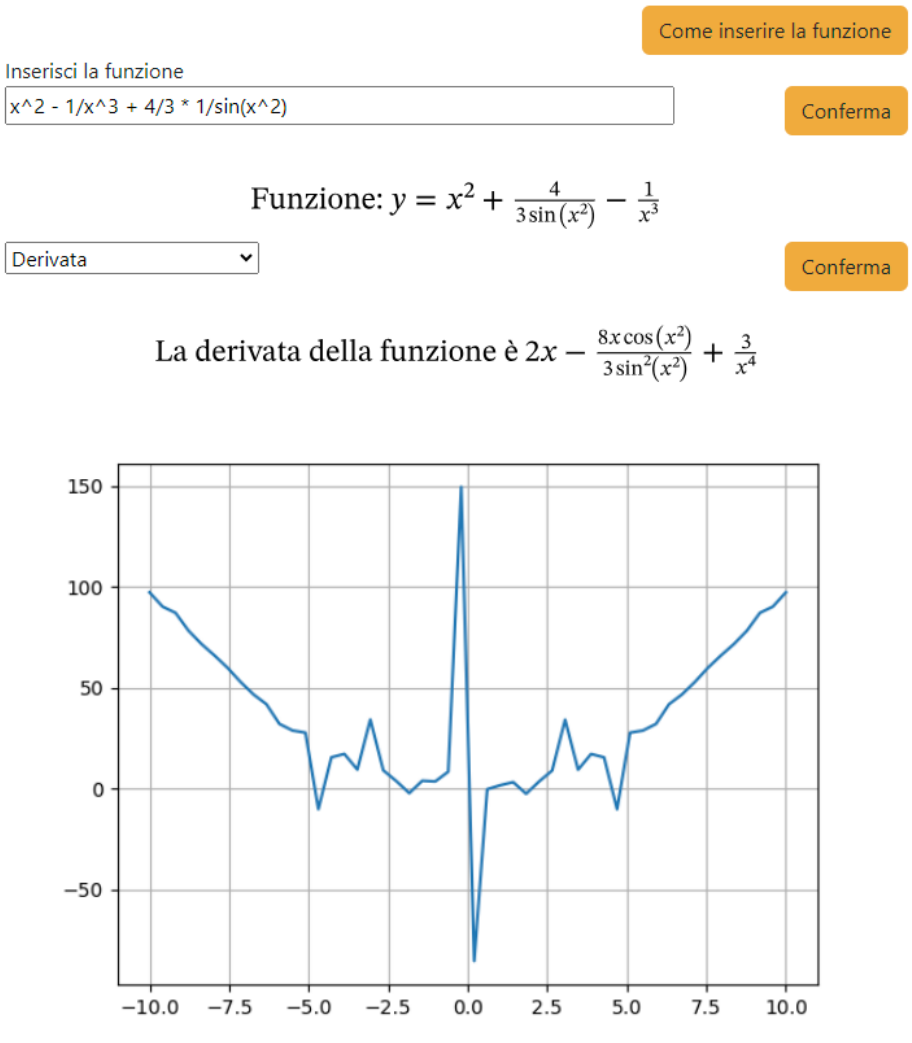


Figura 3: Esempio di schermata dell'applicazione

Bibliografia

[1] Guido Rossum. Python reference manual. Technical report, 1995.

[2] Monika Mladenović & Divna Krpan & Sasa Mladenovic. Introducing programming to elementary students novices by using game development in python and scratch. pages 1622–1629, 07 2016.

[3] Mi Hyun So & JaMee Kim. An analysis of the difficulties of elementary school students in python programming learning. *International Journal on Advanced Science, Engineering and Information Technology*, 8(4-2):1507–1512, 2018.

[4] Daisuke Saito & Hironori Washizaki & Yoshiaki Fukazawa & Tetusya Yoshida & Isumu Kaneko & Hirotaka Kamo. Learning effects in programming learning using python and raspberry pi: Case study with elementary school students. In *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*, pages 1–8, 2019.

[5] Franco Cherish. Lesson plan and workbook for introducing python game programming to support

the advancing out-of-school learning in mathematics and engineering (aolme) project., 2014.

[6] K.B. Alkhan & Z.E. Shaimova. Teaching high school students to solve differential equations using python at math class. *BULLETIN Series of Physics & Mathematical Sciences*, 2020.

[7] Sanchita Sarkar& Santu Sarkar. A new approach to teach molecular biology in high school using python. In *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, pages 1–5, 2016.

[8] Ana Julia Velez Rueda & Guillermo I. Benítez & Julia Marchetti & Marcia Anahí Hasenahuer& María Silvina Fornasari& Nicolas Palopoli & Gustavo Parisi. Bioinformatics calls the school: Use of smartphones to introduce python for bioinformatics in high schools. *PLoS Comput Biol 15(2): e1006473*, 2019.

[9] Ahmad Shauqi Afif Adnan & Siti Munirah Mohd & Shafinah Kamarudin & Nurhidaya Mohamad Jan & Hatika Kaco. Study on python as a tool in education of cryptography: Awareness among secondary school students. In *E-Proceeding Insan Junior Researchers International Conference 2021(iJURECON 2021)*, 2022.

[10] Supot Seebut & Patcharee Wongsason & Dojin Kim & Thanin Putjuso & Chawalit Boonpok. Python-based simulations of the probabilistic behavior of random events for secondary school students. *Eurasia Journal of Mathematics, Science and Technology Education*, 18(9), 2022.

[11] Kado Kado. Teaching and learning the fundamental of calculus through python-based coding. *International Journal of Didactical Studies*, 3:2022, 07 2022.

[12] Jeffrey Elkner. Using python in a high school computer science program. In *9th International Python Conference*, 2001.

[13] Linda Grandell & Mia Peltomäki & Ralph-Johan Back & Tapio Salakoski. Why complicate things? introducing programming in high school using python. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ACE '06, page 71–80, AUS, 2006. Australian Computer Society, Inc.

[14] Linda Mannila & Mia Peltomäki & Tapio Salakoski. What about a simple language? analyzing the difficulties in learning to program. *Computer Science Education*, 16(3):211–227, 2006.

[15] Muhammad Ateeq & Hina Habib & Adnan Umer & Muzammil Rehman. C++ or python? which one to begin with: A learner's perspective. pages 64–69, 04 2014.

[16] Nabeel Alzahrani & Frank Vahid & Alex Edgcomb & Kevin Nguyen & Roman Lysecky. Python versus c++: An analysis of student struggle on small coding exercises in introductory programming courses. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 86–91, New York, NY, USA, 2018. Association for Computing Machinery.

[17] Mrs. Selina Khoirom & Moirangthem Sonia & Borishphia Laikhuram & Jaeson Laishram & Tekcham Davidson Singh. Comparative analysis of python and java for beginners. 2020.

[18] Ambikesh Jayal & Stasha Lauria & Allan Tucker & Stephen Swift. Python for teaching introductory programming: A quantitative evaluation. *Innovation in Teaching and Learning in Information and Computer Sciences*, 10, 02 2011.

[19] Eva Mészárosová. First steps of developing a methodology for teaching programming fundamentals in python. pages 98–105, 2016.

[20] Agostino Lorenzi & Enrico Cavalli & Vittorio Moriggia. *Linguaggio Python.* Atlas, 2019.

[21] Paul J. Deitel & Harvey M. Deitel & Pietro Codara & Carlo Mereghetti. *Introduzione a Python. Per l'informatica e la data science.* Pearson, 2021.

[22] Federico Tibone. *Progettare e programmare. Con Python.* Zanichelli, 2020.

[23] Maurizio Boscaini & Alberto Montresor & Massimiliano Masetti. *Hashtag.* Hoepli, 2022.

[24] Venkata Sai Pavan Arepalli & Chandra N Sekharan. Performance benchmarking of pyscript and comparative results with javascript. In *2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0084–0087, 2022.

"Digitale solidale" per donare e collaborare.

Inversi Giovanna
Istituto Comprensivo "N. Ronchi" di Cellamare (BA)
giovanna.inversi@posta.istruzione.it
BAIC804003@istruzione.it

Abstract

Il progetto, inserito nel PTOF della nostra scuola da diversi anni, ha previsto la collaborazione in ottica multidisciplinare dei docenti delle materie di Matematica, Italiano e Tecnologia. Dapprima i docenti si sono formati partecipando ai webinar previsti dal progetto "Programma il futuro", in seguito è stato avviato un brainstorming in classe fornendo domande stimolo sul tema "Uso del digitale: perché mi rende libero e solidale?". Il dibattito in classe ha consentito di ottenere spunti e idee per tracciare la "pista" di lavoro. Si è reso necessario fornire ai ragazzi le competenze e principi base d'informatica quindi è stato progettato un percorso specifico di formazione utilizzando i corsi di "Programma il Futuro " e "Code.org". In seguito i ragazzi hanno progettato in modo individuale e programmato un' app. Ciascuno studente ha esposto in classe il proprio progetto quindi mostrato l'applicazione realizzata. I progetti sono stati valutati e i migliori candidati al concorso "Libertà e Solidarietà nella società digitale 2023".

1 Introduzione

Il progetto, ha previsto la partecipazione in ottica multidisciplinare delle materie di tecnologia, italiano e matematica della scuola Secondaria di I grado, è stato realizzato in tutte le classi dalla prima alla terza classe, per un totale di 10 classi coinvolte; è stato svolto nell'anno scolastico 2022/2023. Per attuare il progetto ci si è ispirati al tema annuale del Concorso Nazionale 2023 ossia "Libertà e solidarietà nella società digitale 2023" bandito il 1 dicembre 2022 con una Circolare del Ministero dell'Istruzione, un appuntamento a cui il nostro Istituto partecipa ogni anno ottenendo menzioni d'onore per la qualità dei progetti candidati al concorso. In particolare questo anno il nostro Istituto, considerato il tema, ha vinto con un progetto che ha previsto la progettazione del "design" e la programmazione di un App per informare, sensibilizzare e monitorare la raccolta di solidarietà che annualmente viene promossa dal nostro Istituto per la Comunità San Egidio in occasione del Natale. Tutte le classi e tutti gli studenti si sono impegnati nel periodo tra gennaio e marzo 2023 per costruire progetti di qualità, è stato un lavoro corale in cui ciascuno studente ha dato un contributo che ci ha permesso di "salire sul gradino più alto del podio" e vincere il premio.

2 Progetto Interdisciplinare

2.1 Formazione docenti

Il tema "Libertà e solidarietà nella società digitale 2023" ci è apparso molto accattivante ma allo stesso tempo complesso per gli studenti della scuola secondaria di I° grado per questo abbiamo dovuto elaborare una strategia che rendesse il percorso semplice e gradevole per i nostri

alunni. Per raggiungere questo obiettivo abbiamo deciso dapprima di impegnare noi docenti in un percorso formativo selezionando i webinar programmati da "Programma il Futuro":

- Roberta Barone – Etica Digitale – Privacy ed etica dei dati;
- Daniele Scasciafratte – Perché esiste l'Open Source, spiegato in poco tempo
- Francesco Macchia – Il fediverso: un universo libero e diverso per i social network federati
- Le guide per la Cittadinanza Digitale Consapevole
- Dalla A... alla F: i nuovi corsi di Code.org

2.2 Brainstorming in classe

In classe è stato avviato un brainstorming, utilizzando domande "stimolo" e come strumento per annotare i pensieri la bacheca digitale "Padlet". Di seguito vengono riportate a titolo d'esempio le domande stimolo:

1) Descrivi in che modo la tecnologia informatica può costituire uno strumento di libertà?
2) Descrivi in che modo la tecnologia informatica può costituire uno strumento di solidarietà?
3) Come pensi di usare la tecnologia informatica per realizzare liberamente applicazioni o sistemi che favoriscono la collaborazione e la solidarietà?
4) Quali Applicazioni didattiche, che tutti possono usare, pensi possano essere di aiuto all'apprendimento di alcuni concetti di base di una disciplina ?

   Il dibattito in classe ha consentito di ottenere spunti e idee per tracciare la "pista" di lavoro.

   I ragazzi hanno scelto di progettare l'applicazione sui seguenti temi:

   - Progettare un app per invogliare alla generosità in occasione della raccolta di Solidarietà di Natale che la comunità scolastica promuove ogni anno per la Comunità S. Egidio.
   - Progettare un app d'Istituto da utilizzare in un eventuale pandemia per collaborare a distanza nello studio.

2.3 Apprendimento dei principi base d'informatica

Nella nostra scuola da diversi anni viene utilizzata la piattaforma di "Code.org" per partecipare ad eventi come CodeWeek, questo ha fatto sì che tutti gli alunni sin dalla scuola Primaria familiarizzassero con concetti base d'informatica come le sequenze e i cicli. Per realizzare questo progetto abbiamo pensato un percorso per i nostri alunni che prevedesse le seguenti tappe di formazione:

1. Cicli
2. Eventi
3. Istruzioni condizionali
4. Funzioni
5. Variabili
6. Cicli con contatore
7. Il processo di progettazione
8. Laboratorio dei personaggi

Per svolgere le tappe formative è stata utilizzata la piattaforma Code.org e in particolare la funzione disponibile per gli insegnanti di creare le classi, di gestire gli studenti, selezionare e assegnare corsi e monitorare i progressi per ciascuno studente. In particolare la docente di tecnologia ha curato le tappe formative n. 1,2,3,6; la docente d'italiano le tappe 7 e 8; la docente di matematica le tappe formative n 4 e 5.

Sono stati utilizzati i corsi F vers. 2018 ed E vers. 2019 presenti e disponibili nella piattaforma Code.org.

I progressi degli studenti sono stati monitorati sia in modo individuale da ciascun docente si attraverso momenti di condivisione in classe, in questi momenti si è sviluppata una grande collaborazione tra pari. Di seguito si fornisce un esempio della "Dashboard" di controllo Insegnante vedi Figura e della visualizzazione dello stato di avanzamento per ciascuno studente Figura .

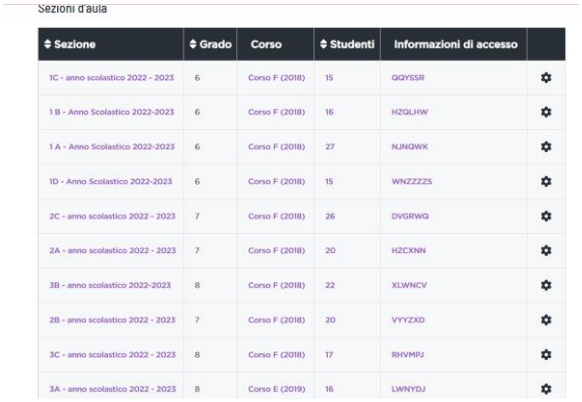


Figura 3 - Gestione classi

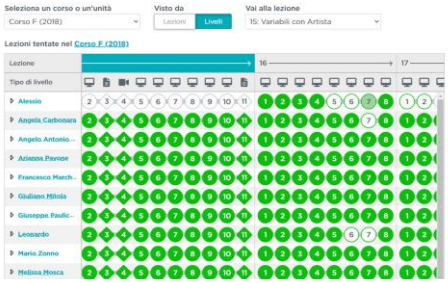


Figura 4 - Stato di avanzamento per alunno

2.4 Progettazione individuale

Prima di avviare il lavoro individuale è stato illustrato in classe il ciclo della progettazione, utilizzando lo schema riportato in Figura .

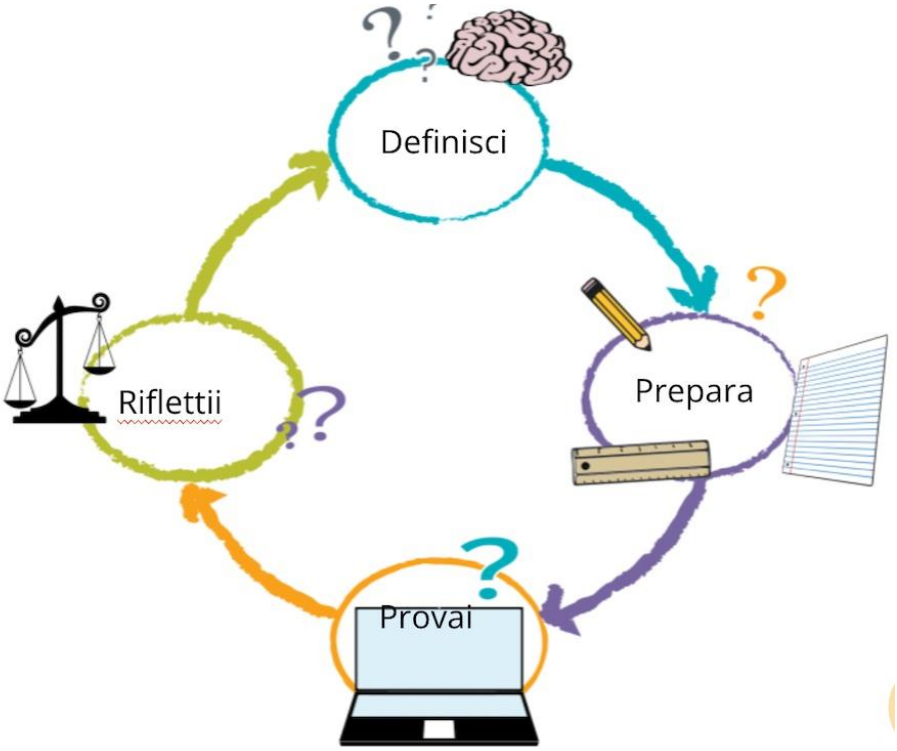


Figura 3 - Schematizzazione del processo di progettazione[1]

In seguito ciascun alunno si è impegnato nella progettazione del "Design" dell'App e dei relativi contenuti, potendo scegliere di utilizzare come strumento di lavoro un quaderno (progettazione su carta) o l'app google "Presentazioni" (progettazione digitale).

Agli studenti è stato assegnato il seguente compito utilizzando la piattaforma "Google Classroom":

Progetta la tua applicazione per la raccolta di solidarietà di Natale/Studio Collaborativo costituita dalle seguenti sezioni:

prima sezione - Informativa – Scrivi un titolo per la tua applicazione e informa l'utente – progetta la parte grafica;

seconda sezione -Sensibilizzazione – Sensibilizza l'utente al tema e progetta la parte grafica;

terza sezione - Monitoraggio – Progetta un modo per raccogliere i dati e la parte grafica.

Ciascuno studente ha consegnato nella sezione compiti di Google Classroom il progetto, lo stesso è stato oggetto di confronto e discussione in classe.

In figura 4 vengono riportati alcuni esempi di progettazione su carta a cura dei ragazzi.

[1] Tratto da Code.org – Fondamenti d'Informatica – Corso per Insegnanti

Figura 4 – Progettazione su carta del "design" dell'applicazione

2.5 Realizzazione dell'"App"

In questa fase ciascuno studente ha messo in pratica quanto appreso nel corso d'informatica inoltre ha messo in pratica la sua creatività quindi quanto progettato nella fase 4. Per realizzare l'applicazione è stato utilizzato IL LABORATORIO DEI PERSONAGGI disponibile sulla piattaforma "Code. Org":

Ogni studente ha condiviso il link del progetto realizzato, nella sezione compiti di Google Classroom.

2.6 Valutazione e selezione del progetto

I progetti realizzati dagli studente e condivisi nella sezione compiti di Google Clasrroom sono stati presentati in classe da ciascun alunno alla presenza dei compagni. Il progetto è stato valutato sia dal team docenti utilizzando la rubrica di valutazione disciplinare, è stato inoltre consentito a ciascun alunno di esprimere il suo gradimento per i progetti dei compagni. Questi due elementi

precedentemente citati hanno consentito di scegliere il progetto di classe da candidare al Concorso.

2.7 Risultati finali

Si può affermare che la metodicità dei percorsi offerti per la secondaria di I° grado dalla piattaforma "Programma il Futuro" ha consentito d'insegnare l'informatica nelle classi anche a docenti che non avevano una formazione specifica e la piattaforma code.org ha consentito di trasformare l'apprendimento dell'informatica per gli studenti in un'esperienza ludica; per ultimo l'applicazione "il laboratorio degli "Sprite" ha consentito ad ogni studente di esprimere la creatività realizzando un prodotto unico, personale e originale.

3 Conclusione

Con questo progetto si è riusciti a coniugare l'insegnamento dell'Informatica con lo sviluppo della creatività. Si può affermare che mediante lo studio e l'esercizio nell'informatica gli studenti hanno migliorato le capacità metodologiche nello studio, imparato a scomporre grandi problemi in piccoli ed hanno messo in pratica tecniche di "Problem Solving".

La metodologia di lavoro utilizzata ossia "il peer to peer" ha migliorato collaborazione in classe e i rapporti interpersonali.

Si può affermare che è migliorata la competenza digitale degli studenti e la consapevolezza nell'utilizzo degli strumenti informatici e le relative applicazioni.

Bibliografia

Regolamento concorso 2023 "Libertà e solidarietà nella società digitale"
Code.org – Fondamenti d'Informatica – Corso per Insegnanti
Dalla A... alla F: i nuovi corsi di Code.org
https://www.economyup.it/tag/tecnologia-solidale/

Capture The Flag: un nuovo approccio all'apprendimento della Cybersecurity

Manuela Flores[1], Barbara Masucci[2]
[1] Università degli Studi di Palermo
manuela.flores@unipa.it
[2] Università degli Studi di Salerno
bmasucci@unisa.it

Abstract

Nell'ambito della Cybersecurity si propone un approccio per l'insegnamento dell'Informatica che, sfruttando i principi della *gamification*, porta gli studenti ad applicare le proprie conoscenze ed abilità nell'affrontare sfide (challenge) inerenti vulnerabilità di vario tipo, sotto forma di gioco in uno spazio virtuale. Scopo del gioco sarà trovare una informazione nascosta, una parola detta bandierina (o "flag"), che lo studente dovrà catturare. Tale approccio è pensato per la Scuola Secondaria di II Grado, in particolare per un *Istituto Tecnico* con indirizzo "*Informatica e Telecomunicazioni*" all'interno dell'insegnamento di "*Sistemi e Reti*". In ogni caso la flessibilità della struttura del percorso permette di rimodularlo ed adattarlo a tanti altri contesti formativi.

1 Introduzione

La Cybersecurity è diventata un settore professionale molto richiesto in ambito lavorativo. Inoltre, considerando la situazione geopolitica internazionale e l'interconnessione sempre più spinta dei sistemi informatici a livello mondiale, tale ambito è considerato anche un filone di ricerca strategico nel mondo accademico. Per facilitare la piena acquisizione di competenze in un settore tanto importante, si vuol proporre per gli studenti un approccio congiunto di conoscenze teoriche e di esercitazioni pratiche, mediante l'utilizzo di una metodologia innovativa che sia efficace e al contempo accattivante. A tal fine, da un po' di tempo vengono proposte esercitazioni pratiche con l'obiettivo di vincere delle sfide all'interno di una competizione [1] [2] [3]. Tale strategia fa sì che gli studenti diventino parte di una comunità e si sentano coinvolti e motivati dallo spirito di gruppo. Ciò consente di raggiungere ottimi risultati, anche grazie al grande entusiasmo suscitato da tale metodologia competitiva.

Dunque, di seguito si propone un approccio per l'insegnamento dell'Informatica nell'ambito della Cybersecurity che utilizzi *"learning by doing"* e *"game based learning"* come metodologie di apprendimento e che proponga agli studenti l'obiettivo di vincere delle sfide predisposte e liberamente fruibili in rete [4]. Il modello formativo utilizzato sfrutta i principi della *gamification*; gli studenti applicheranno le loro conoscenze ed abilità nel superamento di sfide (challenge), giocando in arene virtuali. Tali sfide hanno l'obiettivo di trovare una informazione nascosta, posta sotto forma di parola/stringa detta bandierina (o *"flag"*), che l'attaccante dovrà catturare *("Capture The Flag"*).

Questo tipo di competizioni si è diffuso in rete già da qualche anno. Google, ad esempio, lo propone dal 2016 ed è disponibile un archivio delle principali competizioni internazionali *"Capture The Flag"* (CTFs) [5]. In ambito accademico italiano, dal 2018 vi è un progetto nazionale di formazione sulla Cybersecurity [6], il cui obiettivo principale è avvicinare a questo settore giovani studenti, sia universitari che di scuola secondaria di II grado [7].

 ISBN: 978-88-6629-075-9

2 Percorso formativo "Capture The Flag"

Si propone di sviluppare il percorso formativo *"Capture The Flag"* (CTF) nella Scuola Secondaria di II Grado, in particolare in un *Istituto Tecnico* con indirizzo "*Informatica e Telecomunicazioni*" all'interno dell'insegnamento di "*Sistemi e Reti*". Tale insegnamento dispone di 4 ore settimanali nelle *classi III, IV* e *V*. Di tali ore settimanali, si propone di dedicarne 2 al percorso CTF, per un totale di 66 ore annue (198 ore nel triennio). In ogni caso la flessibilità della struttura del percorso formativo permette di rimodularlo ed adattarlo a tanti altri contesti. Per realizzare tale approccio si utilizzeranno i materiali presenti sul sito https://exploit.education/. Su tale sito sono presenti diverse risorse che possono essere utilizzate per effettuare l'analisi di vulnerabilità varie, lo sviluppo di exploit (script o codice binario che, sfruttando una specifica vulnerabilità, consente all'attaccante di acquisire i privilegi di amministratore del sistema informatico sotto attacco), il debug del software, l'analisi binaria ed altre azioni utili nella Cybersecurity. In particolare, sul sito https://exploit.education/ sono presenti delle macchine virtuali pronte per consentire agli studenti di gareggiare in sfide di varia complessità.

Per la *classe III* si propone di utilizzare la macchina virtuale "*Nebula*", la quale comprende una varietà di sfide semplici ed intermedie che riguardano l'escalation dei privilegi del sistema operativo Linux, i comuni problemi dei linguaggi di scripting, la race condition del file system (due processi o thread che accedono contemporaneamente ad una risorsa condivisa). Nebula è l'ideale per introdurre gli studenti verso un primo approccio all'utilizzo di Linux, poiché consente ai propri utenti di sfidare il sistema su una varietà di debolezze e vulnerabilità comuni (e meno comuni) di Linux. Le principali vulnerabilità trattate riguardano:

- File binari con bit SUID acceso;
- Autorizzazioni;
- Race condition;
- Meta-variabili della Shell;
- Punti deboli della variabile d'ambiente $PATH;
- Debolezze dei linguaggi di scripting;
- Errori di compilazione di un file binario.

Le sfide di Nebula sono indicate sul sito con il nome di "levelXX" dove XX va sostituito con il numero della sfida. Per ciascuna sfida bisogna accedere alla macchina virtuale con username e password uguali al nome della sfida stessa, in minuscolo. Le stringhe/bandierine/flag si possono ottenere violando rispettivamente gli utenti/account dal nome "flagXX". Ad esempio, la prima sfida si chiama level00, l'attaccante accede a Nebula con username "level00" e password "level00" e deve violare l'account "flag00", in base alle indicazioni presenti sul sito. Alcuni livelli possono essere svolti esclusivamente da remoto, per cui necessitano che la macchina disponga di un collegamento ad Internet. In generale, il livello "levelXX" richie l'esecuzione del binario /bin/getflag con i privilegi dei corrispondente account vittima "flagXX". Dopo aver affrontato le sfide di Nebula, lo studente avrà una comprensione ragionevolmente approfondita degli attacchi locali contro il sistema operativo Linux ed una rapida panoramica su alcuni dei possibili attacchi remoti.

Per la *classe IV* si propone di utilizzare la macchina virtuale "*Protostar*", che introduce prima i concetti più semplici, dalla corruzione e modifica della memoria, al reindirizzamento delle funzioni e, infine, all'esecuzione di uno shellcode personalizzato (codice assembly utilizzato come carico utile o "payload" nello sfruttamento o "exploitation" di una vulnerabilità del sistema attaccato).

Le principali vulnerabilità trattate riguardano:

- Programmazione di rete;
- Ordine dei byte;
- Stack-based buffer overflow;
- Gestione dei socket;

- Format string;
- Heap-based buffer overflow.

Una volta avviata la macchina virtuale, si può accedere con username "user" e password "user". I file da sfruttare risiedono nella directory /opt/protostar/bin. A tale macchina virtuale è stata recentemente affiancata "*Phoenix*", che tratta le stesse vulnerabilità in modo e con script analoghi a quelli di Protostar. Anche Phoenix è reperibile sullo stesso sito.

Infine, per la *classe V* si propone di utilizzare la macchina virtuale "*Fusion*", che tratta stili di exploitation più avanzati e copre una varietà di meccanismi anti- exploitation come:

- Randomizzazione del layout dello spazio degli indirizzi;
- Eseguibili indipendenti dalla posizione;
- Memoria non eseguibile;
- Fortificazione del codice sorgente;
- Protezione dal traboccamento dello stack.

Oltre a quanto sopra, ci sono varie altre sfide da esplorare, come ad esempio:

- Problemi crittografici;
- Attacchi a tempo;
- Varietà di protocolli di rete.

Dopo aver affrontato le sfide di Fusion, lo studente avrà una comprensione approfondita delle strategie di prevenzione degli exploit, dei punti deboli associati, di varie debolezze crittografiche, di numerose implementazioni di heap.

Nella seguente tabella si riassume il percorso didattico proposto:

Percorso didattico "Capture The Flag"			
Istituto Tecnico con indirizzo ***Informatica e Telecomunicazioni***			
Insegnamento di ***Sistemi e Reti***			
Classe	**Ore annue**	**Tematiche trattate**	**Macchina virtuale***
III	66 (2 ore settimanali)	• File SUID • Autorizzazioni • Race condition • Meta-variabili della Shell • Punti deboli della variabile d'ambiente $PATH • Debolezze dei linguaggi di scripting • Errori di compilazione di un file binario	Nebula
IV	66 (2 ore settimanali)	• Programmazione di rete • Ordine dei byte • Stack-based buffer overflow • Gestione dei socket • Format string • Heap-based buffer overflow	Protostar/ Phoenix
V	66 (2 ore settimanali)	• Randomizzazione del layout dello spazio degli indirizzi • Eseguibili indipendenti dalla posizione • Memoria non eseguibile • Fortificazione del codice sorgente • Protezione dall'overflow dello stack • Problemi crittografici • Attacchi a tempo • Varietà di protocolli di rete	Fusion
* Macchine virtuali reperibili sul sito https://exploit.education/			

3 Conclusioni

L'approccio proposto richiede agli studenti di affrontare delle sfide (challenge) inerenti vulnerabilità di vario tipo nel settore della Cybersecurity, attraverso le metodologie di apprendimento "learning by doing" e "game based learning" e mediante l'utilizzo di macchine virtuali nelle quali gli studenti possono affrontare e vincere delle sfide proposte sotto forma di gioco. Tale approccio è pensato per l'ultimo triennio della Scuola Secondaria di II Grado. Si tratta di un percorso flessibile, che può essere modulato anche in maniera differente da quanto proposto. Ad esempio, se il numero di sfide presenti per ciascuna macchina virtuale dovesse essere troppo elevato per le 66 ore annue previste, ci si potrebbe fermare prima, senza pregiudicare l'approccio ad una nuova macchina virtuale all'avvio dell'anno scolastico successivo. Inoltre, se le vulnerabilità che propone di affrontare la macchina Fusion all'inizio della classe V fossero considerate troppo complesse, si potrebbero eventualmente affrontare prima quelle trattate nella macchina Phoenix, come rinforzo di quanto già acquisito l'anno precedente con Protostar, per poi proseguire con Fusion quando si ritenga che gli studenti siano pronti a farlo.

La flessibilità della struttura e la metodologia didattica innovativa del percorso proposto permetteranno di ottenere in maniera giocosa e creativa lo sviluppo di elevate competenze ed abilità nell'ambito della Cybersecurity da parte degli studenti.

Bibliografia

[1] S. Hatma and Y. Musashi, "Review of Cybersecurity Research Topics, Taxonomy and Challenges: Interdisciplinary Perspective," in *IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA)*, Kaohsiung, Taiwan, 2019.

[2] R. Creutzburg, "Cybersecurity and Forensic Challenges - A Bibliographic Review," *Electronic Imaging,* vol. 2018, no. 6, pp. 100--1, 2018.

[3] J. Hautamaki, M. Karjalainen, P. Hakkinen and T. Hamalainen, "Cyber security exercise: Literature review to pedagogical methodology," in *13th International Technology, Education and Development Conference*, Valencia, Spain, 2019.

[4] Exploit Education Team, "Exploit Education," [Online]. Available: https://exploit.education/.

[5] CTFtime Team, "CTFs," [Online]. Available: https://ctftime.org/ctfs.

[6] CyberChallenge Team, "CyberChallenge.it," [Online]. Available: https://cyberchallenge.it/.

[7] G. Ferraro, G. Lagorio and M. Ribaudo, "CyberChallenge.IT@Unige: Ethical Hacking for Young Talents," in *Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, Genoa, Italy, 2020.

Un curricolo per separare l'informatica dall'applimatica* in un Liceo scientifico opzione scienze applicate

Gionata Massi

IIS Savoia Benincasa, Ancona (AN)

Sommario

Questo articolo presenta un curricolo per l'insegnamento dell'informatica nel Liceo scientifico opzione scienze applicate che ambisce di introdurre la materia come disciplina scientifico/ingegneristica[1].

Il curricolo è orientato all'apprendimento dei fondamenti epistemologici dell'informatica in accordo con i traguardi di competenza fissati nelle Indicazioni nazionali però presenta una selezione di argomenti e contenuti che sconcerta le attese di studenti, genitori e colleghi abituati ad un percorso di addestramento all'uso delle tecnologie informatiche.

La didattica di alcuni argomenti richiede l'uso del PC e del software[2], come implicitamente richiesto dalla Indicazioni nazionali, senza essere in contraddizione con gli obiettivi epistemologici.

Il curricolo è presentato come un elenco di traguardi di competenza, espressi in termini di abilità verificabili nel primo biennio, nel secondo biennio e nel quinto anno, raggruppati come in argomenti e aree tematiche[3].

La sperimentazione del curricolo è stata effettuata tra gli anni scolastici 2018/19 e 2022/23 coinvolgendo due sezioni e, a causa della sopraggiunta emergenza sanitaria, non è stato possibile condurre analisi comparative sul raggiungimento degli obiettivi di competenza rispetto alle classi parallele che non seguivano lo stesso percorso didattico.

In conclusione il curricolo qui proposto può essere utilizzato come un repertorio di traguardi di apprendimento al quale gli insegnanti possono attingere liberamente per la progettazione e la programmazione didattica dell'informatica nel proprio istituto; esso è un contributo verso l'insegnamento della materia intesa come disciplina scientifica e tecnica piuttosto che come applimatica e può essere meritevole di ulteriore sperimentazione. Il limite di applicazione di questo percorso è l'assenza di materiali di studio di elevata qualità in lingua italiana.

Indice

*Neologismo trovato in alcuni lucidi del prof. Mattia Monga per indicare l'uso del computer o di applicativi specifici

[1] Harold Abelson nella prima lezione del corso MIT 6.001 Structure and Interpretation of Computer Programs sembra essere in disaccordo sulla natura scientifica dell'informatica

[2] Nella scelta del software ci si è limitati alle sole applicazioni rilasciate come software libero o, laddove non fosse possibile, a sorgente aperto.

[3] Si vedano le Indicazioni nazionali [9]

1 Introduzione

Gli insegnanti di informatica dei licei italiani non dispongono di estesi sillabi ministeriali, programmi nazionali o dettagliati repertori di competenze disciplinari che guidino l'azione didattica in modo uniforme sul territorio nazionale. Essi devono progettare e programmare l'attività didattica, per ogni scuola e classe, cercando un compromesso fra vincoli normativi nazionali, quadri europei [4] e aspettative di tanti attori.

L'attività di progettazione e programmazione didattica si realizza come se si dovesse ottimizzare euristicamente un modello di programmazione matematica multiobiettivo nel quale le variabili decisionali sono gli argomenti e i contenuti utili a sviluppare le competenze proprie della materia scolastica, i vincoli sono dati dalle Indicazioni nazionali [9] e dal Piano Triennale dell'Offerta Formativa della scuola, e gli obiettivi sono tanti e in contrasto tra loro: venire incontro alle richieste del territorio, adeguarsi alle aspettative di studenti, genitori e altri insegnanti, trovare argomenti e contenuti presenti nel libro di testo in adozione, laddove questo ci sia, riutilizzare il materiale didattico già disponibile, incentivare gli studenti al conseguimento di certificati[5] e non addentrarsi in saperi che gli insegnati non possiedono[6].

Nell'esperienza dell'autore, l'aspettativa dell'utenza e del corpo docente si concentra verso un percorso di applimatica* volto a fornire agli studenti le abilità d'uso di qualche programma applicativo con il quale interagire tramite un'interfaccia grafica che esibisce un comportamento *What You See Is What You Get*[7] (WYSIWYG). L'inseme dei programmi applicativi richiesti dagli insegnanti di altre materie include sempre un certo programma per l'elaborazione del testo e un altro per la predisposizione dei lucidi di supporto alla presentazione. A seconda dei consigli di classe si possono unire all'insieme dei software i fogli elettronici e alcuni sistemi per la rappresentazione geometrica e/o per il calcolo numerico o simbolico.

I libri di testo del bienno sono spesso adatti a soddisfare questo genere di aspettative, anche per rispettare le Indicazioni nazionali, e solo negli ultimi tre anni oltre all'applimatica introducono un linguaggio di programmazione[8], spesso uno solo e frequentemente il C++[9].

La situazione nelle nazioni anglofone, le uniche per le quali l'autore è riuscito ad effettuare una ricerca sullo stato dell'arte, appare diversa. Vi sono curricula molto dettagliati che fanno

[4] Agli insegnanti di informatica è richiesto in particolar modo di conformarmi al Digital Competences Framework (DigComp 2.2), con elenca oltre 250 competenze digitali legate alle *tecnologie digitali*. La traduzione del quadro di riferimento in lingua italiana utilizza il termine *digitale* per *digit* come per riferirsi alla pressione delle dita sulla tastiera o, più probabilmente, il falso amico *digit* non si traduce più con il termine *numerico*.

[5] Spesso usare un sistema di valutazione conformità ai *framework di competenze* comunitari, le scuole sono sede d'esame per certificazioni sull'uso di software applicativi

[6] Il requisito di accesso all'insegnamento per la classe di concorso *A-41 – Scienze e tecnologie informatiche* è molto più lasco rispetto a quelle di altre sezioni di concorso e molti insegnanti, specie quelli meno giovani, non hanno sostenuto alcun esame di informatica nel percorso universitario.

[7] Questo termine indica che quando usi un'applicazione per modificare un documento, *quello che vedi* nell'interfaccia di modifica *è quello che ottieni* nell'interfaccia di visualizzazione. L'applicazione di modifica e quella di visualizzazione possono coincidere.

[8] Nel primo biennio i linguaggi di programmazione sono sostituiti da ambienti grafici per la programmazione visuale a blocchi e il termine *programmazione* è regredisce in favore di *coding*

[9] Bjarne Stroustrup sembra sorpreso dal fatto che il C++ sia ampiamente utilizzato nella didattica in quanto tale linguaggio non è il più compatto e pulito e nella genesi del linguaggio non vi sono elementi sulla natura didattica.

apparire il nostro processo di programmazione didattica come empirico, eterogeneo, orientato all'uso delle applicazioni e non supportato da materiale di studio valido[10].

Negli Stati Uniti si seguono curricula e attività didattiche progettate in dettaglio, come [7] che propone scopi affini alle nostre *indicazioni nazionali* [9]; le esperienze didattiche e i curricula sono spesso raccolte in cataloghi [12]. Le piattaforme di apprendimento online come Code.org [11] e Khan Academy [10] offrono corsi gratuiti basati sui curricula più noti come lo Advanced Placement Computer Science [3]. La comunità scientifica si occupa da tempo della didattica dell'informatica anche tramite lo sviluppo di linguaggi di programmazione[11] e ambienti per l'insegnamento[12] ([1] e [2], [6]). Anche in Gran Bretagna c'è molta attenzione all'insegnamento dell'informatica e alla formazione professionale dei docenti[13] [8]. Di rilievo appaiono anche i progetti didattici della Nuova Zelanda e tra quelli dedicati all'informatica vi è la *Computer Science Field Guide* [5], una guida per insegnanti e studenti che è stata tradotta in varie lingue[14].

Questo documento presenta nel par. 2 la proposta di curricolo che l'autore ha progettato, e seguito con due sezioni, per ridurre i contenuti dell'applimatica e avvicinare gli studenti alle problematiche, al sapere e ai metodi dell'Informatica. Nel par. 3 illustra la sperimentazione e chiarisce le cause della mancata osservazione comparativa con altre classe del medesimo ciclo e nel par. 4 determina le condizioni dell'applicabilità del curricolo in base alla sperimentazione svolta.

2 Curricolo

Per esigenze di brevità, il curricolo è riportato in modo parziale e si rimanda il lettore alla pagina https://github.com/gionatamassibenincasa/progettazione-didattica-informatica.

2.1 Primo biennio

2.1.1 Scrittura tecnico-scientifica al calcolatore

Area tematica: Elaborazione digitale dei documenti (DE)

1. Identificare gli attori di una comunicazione
2. Distinguere tra dati, informazioni e conoscenza
3. Aggiungere l'autore e il titolo nel frontespizio di una relazione
4. Suddividere un testo nei suoi elementi strutturali ed aggiungere i titoletti
5. Scrivere tabelle, immagini e altri contenuti flottanti con didascalie

[10] Il sistema scolastico italiano ha molte peculiarità che lo rendono estremante formativo, o almeno questa è l'impressione che se ne ricava quando si assiste ai colloqui pluridisciplinari degli studenti che hanno frequentato un anno scolastico all'estero, ma questa affermazione sembra non essere generalizzabile per l'informatica.

[11] Albeson e Stroustrup, per citare due creatori di linguaggi di programmazione, considerano il linguaggio di programmazione come uno strumento per esprimere le idee sotto forma di codice.

[12] Seymour Papert e Mitchel Resnick sono così famosi tra gli insegnati italiani che è inutile citarli ma il loro contributo ora viene usato per spingere verso il *coding* che pare cosa diversa dalla programmazione.

[13] Un progetto di riferimento è [4] e comunità e fondazioni di produttori di hardware Open Source che propongono curricula completi.

[14] La guida è tradotta in tedesco, spagnolo e polacco. Alla lista manca l'italiano e non sono segnalati progetti di traduzione nella lingua italiana.

6. Scrivere con elenchi ordinati, non ordinati e descrittivi
7. Enfatizzare il testo selezionando un carattere tipografico opportuno

L'argomento è introdotto per soddisfare sia il vincolo ministeriale sulla produzione di documenti elettronici. La videoscrittura si colloca di buon diritto nell'applimatica per cui si è scelto l'argomento della comunicazione tecnico/scientifica e un linguaggio di dominio specifico dotato di grammatica libera dal contesto. Lo strumento di elaborazione del testo diventa così un linguaggio avente una grammatica espressa tramite il suo diagramma sintattico e l'attività di videoscrittura diventa un'approccio ai linguaggi formali e alla programmazione.

2.1.2 Dati e codifiche

Area tematica: Architettura dei computer (AC)

1. Convertire un numero da base due a base dieci
2. Convertire un numero da base dieci a base due
3. Codificare un testo usando codifiche binarie dei caratteri
4. Decodificare un testo codificato in binario
5. Comprendere le esigenze che hanno condotto allo sviluppo dello standard UNICODE
6. Codificare un'immagine monocromatica mediante run-length encoding
7. Decodificare un'immagine monocromatica codificata mediante run-length encoding

L'argomento è proposto per soddisfare la richiesta di introdurre lo studente alla codifica binaria senza limitarsi ai codici codici ASCII e Unicode citati nelle Indicazioni nazionali. Verrà ampiamente ripreso al quinto anno quando, nell'analisi degli algoritmi numerici, sono trattate le codifiche in virgola mobile e gli errori di troncamento.

2.1.3 Problemi, modelli, soluzioni e algoritmi

Area tematica: Algoritmi e linguaggi di programmazione (AL)

1. Saper formalizzare un problema di ricerca
2. Simulare l'esecuzione dell'algoritmo di ricerca lineare
3. Simulare l'esecuzione dell'algoritmo di ricerca binaria
4. Saper formalizzare il concetto di ordinamento di una sequenza
5. Simulare l'algoritmo di ordinamento per selezione, per inserimento e a bolle
6. Calcolare il numero di confronti e di scambi degli algoritmi di ordinamento basati su confronti e scambi
7. Comprendere i criteri di scelta di un algoritmo rispetto ad altri
8. Astrarre il modello di semplici problemi di natura quantitativa e descrivere algoritmicamente il procedimento di soluzione

9. Simulare l'esecuzione di un programma

Il concetto di algoritmo viene formalizzato usando i concetti di problema, modello, codifica, istanza, soluzione ed esecutore.

2.1.4 Interagire col sistema operativo tramite la shell

Area tematica: Sistemi operativi (SO)

1. Creare, rinominare, spostare e cancellare un file
2. Organizzare i file nelle directory
3. Invocare un programma
4. Elencare file e directory
5. Creare una pipe anonima tra due o più utility
6. Usare comandi per la data, l'estrazione della testa e della coda di un file di testo, l'ordinamento delle linee, l'estrazione di campi

Le Indicazioni nazionali richiedono la conoscenza delle funzionalità dei sistemi operativi più comuni e si è scelto di orientare lo studio al sistema operativo GNU/Linux in quanto esso è il sistema operativo più eseguito[15] e meglio documentato, oltre a soddisfare il requisito di essere software libero.

2.1.5 Documenti elettronici per il web

Area tematica: Elaborazione digitale dei documenti (DE)

1. Riconoscere un linguaggio basato sui marcatori
2. Usare i tag HTML per creare un documento valido
3. Usare i tag semantici per strutturare il contenuto
4. Saper realizzare un semplice sito web statico curando solo il contenuto
5. Collegare un foglio di stile ad una pagina web
6. Modificare lo stile del testo
7. Modificare l'aspetto delle scatole
8. Saper realizzare un semplice sito web statico curando anche la presentazione

I linguaggi specifici di dominio del web per il contenuto e la presentazione sono introdotti nel primo biennio ma vengono esplorati anche nel secondo biennio.

[15] Si veda https://en.wikipedia.org/wiki/Usage_share_of_operating_systems

2.2 Secondo biennio

2.2.1 Fogli elettronici

Area tematica: Elaborazione digitale dei documenti (DE)

1. Usare le operazioni di filtraggio, riduzione e mappa nel foglio di calcolo
2. Usare il foglio di calcolo per modellizzare e risolvere le problematiche d'interesse per il corso di studi
3. Usare il risolutore per problemi di scelta
4. Impostare problemi a variabili intere di ammissibilità e di ottimizzazione vincolata con il risolutore
5. Descrivere l'ordine delle operazioni di aggiornamento in un foglio di calcolo

Si propongono le funzionalità del foglio di calcolo che corrispondono a costrutti tipici della

2.2.2 Paradigmi di programmazione

Area tematica: Algoritmi e linguaggi di programmazione (AL)

1. Saper realizzare algoritmi usando le funzioni e i costrutti di sequenza, selezione e iterazione (paradigma strutturato)
2. Riconoscere la ricorsione come caratteristica sintattica
3. Riconoscere se il processo generato da una procedura sintatticamente ricorsiva è lineare o ricorsivo
4. Progettare semplici basi di fatti e regole con paradigma logico
5. Saper realizzare semplici algoritmi mediante funzioni sintatticamente ricorsive
6. Saper attraversare semplici strutture dati definite ricorsivamente

L'autore non è riuscito a decodificare le richieste ministeriali di *implementazione di un linguaggio di programmazione, metodologie di programmazione, sintassi di un linguaggio orientato agli oggetti*, pertanto si è scelto di presentare il paradigma della programmazione strutturata (con funzioni definite nell'ambito di una funzione), quello logico e quello funzionale. Alla programmazione orientata agli oggetti, richiesta dalle Indicazioni, sono dedicati altri moduli.

2.2.3 Applicazioni web

Area tematica: Algoritmi e linguaggi di programmazione (AL)

1. Saper scrivere semplici script interpretabili dal web browser
2. Saper usare le API di accesso al modello di documento di una pagina HTML
3. Strutturare i dati per permettere la costruzione programmatica di una pagina HTML
4. Realizzare una semplice web app interattiva

Questo argomento introdurre alcuni concetti della programmazione orientata agli oggetti quali quello di istanza di classe, interfaccia, metodi e proprietà.

2.2.4 Problemi su grafi e combinatorici

Area tematica: Algoritmi e linguaggi di programmazione (AL)

1. Saper rappresentare un grafo
2. Realizzare gli algoritmi di visita di un albero binario: preordine, postordine, simmetrico
3. Visitare un grafo in profondità e in ampiezza
4. Simulare l'algoritmo di Dijkstra
5. Rappresentare sul foglio di calcolo il problema dello zaino

2.3 Quinto anno

2.3.1 Reti di calcolatori

Area tematica: Reti di computer (RC)

1. Descrivere i componenti hardware e software che costituiscono una rete dati
2. Descrivere le applicazioni di rete quali quelle del www e della posta elettronica
3. Descrivere i livelli di rete del modello ISO/OSI
4. Riconoscere vantaggi e svantaggi di alcune topologie di rete

2.3.2 Internet e servizi

Area tematica: Struttura di Internet e servizi (IS)

1. Descrivere i servizi di rete con particolare riferimento al www
2. Descrivere gli strati del modello TCP/IP
3. Comprendere i principali rischi di sicurezza nelle comunicazioni digitali
4. Valutare l'uso di tecniche per raggiungere determinati livelli di riservatezza, integrità e disponibilità dei dati

2.3.3 Funzioni di ordine superiore

Area tematica: Computazione, calcolo numerico e simulazione (CS)

1. Definire funzioni che hanno come argomento una funzione di variabile reale e un valore reale e che restituiscono un valore reale
2. Definire funzioni che hanno come argomento una funzione di variabile reale e che restituiscono una funzione di variabile reale
3. Descrivere gli algoritmi in termini di applicazione e composizione di funzioni
4. Manipolare gli array usando le tecniche: filter, map, reduce.

2.3.4 Radici di funzioni non lineari

Area tematica: Computazione, calcolo numerico e simulazione (CS)

1. Saper applicare il metodo di Erone di Alessandria per il calcolo delle radici quadrate
2. Riconoscere l'esistenza di una radice di una funzione dato un intervallo
3. Saper effettuare la ricerca numerica di una radice tramite il metodo di bisezione
4. Realizzare in un linguaggio di programmazione e al foglio di calcolo il metodo di bisezione
5. Saper effettuare la ricerca numerica di una radice tramite il metodo delle tangenti (di Newton)
6. Realizzare in un linguaggio di programmazione e al foglio di calcolo il metodo delle tangenti (di Newton)
7. Saper riconoscere l'equivalenza di due metodi iterativi

2.3.5 Ottimizzazione 1-dimensionale

Area tematica: Computazione, calcolo numerico e simulazione (CS)

1. Riconoscere l'esistenza di un massimo e di un minimo
2. Metodo di Newton per l'ottimizzazione
3. Approssimare il punto di minimo locale usando il metodo di Newton per l'ottimizzazione
4. Realizzare un un linguaggio di programmazione e al foglio di calcolo il metodo di Newton per l'ottimizzazione
5. Realizzare un un linguaggio di programmazione e al foglio di calcolo il metodo della sezione aurea

3 Sperimentazione

L'azione didattica ispirata da tale curricolo si è protratta dall'anno scolastico 2018/19, quando due classi prime hanno iniziato il percoso, a quello 2022/23 quando gli stessi studenti hanno conseguito il diploma.

Le misure adottate per fronteggiare l'emergenza sanitaria per COVID-19 hanno condizionato negativamente la sperimentazione e invalidato la rilevazione dei risultati. È stato impossibile condurre un'analisi sul raggiungimento degli obiettivi di competenza per classi parallele, previsto al termine del secondo anno, poi del quarto e del quinto. In particolare, al termine del secondo anno gli studenti non erano nelle aule scolastiche e anche il terzo anno si è svolto quasi interamente a distanza, senza consentire una valutazione affidabile delle competenze. Non si è quindi riusciti a seguire in modo accurato il curricolo e non sono state effettuate le prove con le classi parallele del quarto e quinto anno per via della forte eterogeneità con le altri classi.

La sperimentazione si è conclusa e non è stata riproposta a causa dell'esigenza di uniformare l'offerta didattica tra classi parallele e per la difficoltà di reperire validi supporti didattici.

4 Conclusioni

In conclusione si considera il curricolo qui presentato come una proposta coerente con i traguardi di competenze fissati dal Ministero dell'Istruzione e del Merito e orientata alla introduzione dell'Informatica come disciplina scientifica piuttosto che come apprendistato all'uso delle applicazioni.

Il curricolo può essere una guida per la progettazione e la programmazione didattica dell'informatica alla quale gli insegnati liceali possono attingere per la costruzione del percorso didattico del proprio istituto ma presenta una difficoltà di realizzazione dovuta alla mancanza di materiali di studio di qualità elevata in lingua italiana, quali potrebbero essere i libri di testo o di Open Educational Resource (OER).

Per superare questa difficoltà servirebbe la costruzione di una comunità professionale che contribuisca ai progetti esistenti, come il repertorio di OER del Politecnico di Torino [13], al miglioramento dei libri di testo e delle piattaforme didattiche, e alla traduzione professionale dei migliori progetti didattici in lingua straniera.

Riferimenti bibliografici

[1] Harold Abelson e Gerald Jay Sussman. *Structure and Interpretation of Computer Programs*. Versione *open access*: https://sarabander.github.io/sicp/. The MIT Press, 1996, p. 683. ISBN: 9780262510875.

[2] Harold Abelson et al. *Structure and Interpretation of Computer Programs: JavaScript Edition. JavaScript Edition*. Versione *open access*: https://sourceacademy.org/sicpjs/index. MIT Press, 2022. ISBN: 9780262543231.

[3] College Board. *AP Computer Science Principles*. 2023. URL: https://apstudents.collegeboard.org/courses/ap-computer-science-principles.

[4] National Centre for Computer Education. *Helping you teach computing*. 2023. URL: https://teachcomputing.org/.

[5] Computer Science Education Research Group at the University of Canterbury, New Zealand. *Computer Science Field Guide*. 2023. URL: https://www.csfieldguide.org.nz/.

[6] Matthias Felleisen et al. *How to Design Programs. An Introduction to Programming and Computing*. Versione *Open Access*: https://htdp.org/. MIT Press, 2018, p. 792. ISBN: 9780262534802.

[7] Kathi Fisler et al. «Evolving a K-12 Curriculum for Integrating Computer Science into Mathematics». In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. SIGCSE '21. Virtual Event, USA: Association for Computing Machinery, mar. 2021, pp. 59–65. ISBN: 9781450380621. DOI: 10.1145/3408877.3432546. URL: https://doi.org/10.1145/3408877.3432546.

[8] Brian Fowler e Emiliana Vegas. «How England Implemented Its Computer Science Education Program.» In: *Center for Universal Education at The Brookings Institution* (2021).

[9] Il Ministro dell'Istruzione, dell'Università e della Ricerca. *Schema di regolamento recante 'Indicazioni nazionali riguardanti gli obiettivi specifici di apprendimento concernenti le attività e gli insegnamenti compresi nei piani degli studi previsti per i percorsi liceali di cui all'articolo 10, comma 3, del decreto del Presidente della Repubblica 15 marzo 2010, n. 89, in relazione all'articolo 2, commi 1 e 3, del medesimo regolamento.'* Allegato F. Ministero dell'Istruzione, dell'Università e della Ricerca. Lug. 2010. URL: https://www.gazzettaufficiale.it/eli/id/2010/12/14/010G0232/sg.

[10] Sal Khan. *Khan Academy*. 2023. URL: https://www.khanacademy.org/.

[11] Code.org Team. *Code.org*. 2023. URL: https://code.org/.

[12] CSforALL Team. *Computer Science for **ALL** Students*. 2023. URL: https://www.csforall.org/.

[13] Politecnico di Torino. *Free Architecture for Remote Education*. 2023. URL: https://fare.polito.it/.

Analisi Esplorativa dei Dati: proposta di un Syllabus per l'acquisizione di competenze orizzontali

Vittoria Cozza[1], Pasquale Cozza[2], Alessio Maria Braccini[3]

[1] Dipartimento di Informatica, Università di Verona, Verona, Italia
vittoria.cozza@univr.it
[2] Liceo Scientifico Pitagora, Rende, Cosenza, Italia
pasquale.cozza@posta.istruzione.it
[3] DEIM, Dipartimento di Economia, Ingegneria, Società e Impresa. Università della Tuscia, Viterbo, Italia
abraccini@unitus.it

Sommario

Le competenze digitali sono oggi fondamentali all'interno di qualsiasi ruolo organizzativo, sia per poter utilizzare proficuamente gli strumenti di produttività che sono tutti digitali, sia per poter ricercare, selezionare, analizzare, sintetizzare, condividere e visualizzare informazioni e dati. L'analisi, almeno preliminare, dei dati rappresenta una competenza orizzontale fondamentale per comprendere e supportare i processi di innovazione.

L'Italia ha messo in campo una serie di iniziative di formazione per aumentare la diffusione di tale competenza. In linea con ciò, questo lavoro presenta un Syllabus costruito per un corso universitario di analisi esplorativa dei dati e commenta la prima esperienza della sua implementazione in un corso erogato agli studenti della laurea magistrale in Amministrazione e Finanza dell'Università della Tuscia. Il Syllabus è stato anche usato in corsi di formazione STEAM dei docenti delle scuole secondarie superiori italiane.

1 Introduzione

La digitalizzazione ha raggiunto livelli di profondità e pervasività tali da contagiare ogni aspetto dell'attività umana. Il salto tecnologico portato dalla digitalizzazione influenza sia la sfera del privato cittadino, quella professionale, e quella dell'interazione con le aziende e le pubbliche amministrazioni (PA). I sistemi digitali sono ovunque. Originariamente introdotti e adottati nelle attività che maggiormente avevano necessità di calcolo - quali ad esempio le attività di natura scientifica, matematico-statistica e contabile - oggi interessano ogni ambito organizzativo. Basti pensare al ruolo dell'informatica all'interno delle imprese che, da funzione specialistica che racchiude competenze collegate all'uso di tecnologie digitali, diviene elemento diffuso in grado di cambiare profondamente modalità di interazione tra individui e organizzazioni[25]. La conseguenza è la costante crescita e rilevanza dei dati digitali sia in termini di qualità, che in termini di varietà.

La quantità totale di dati digitali nel mondo è stata stimata in 64,2 zettabyte nel 2020 e si prevede che la creazione globale di dati crescerà fino a oltre 180 zettabyte entro il 2025[1]. Come riportato da Statista, nel 2020 i dati creati e replicati hanno raggiunto un massimo storico, di circa 20 zettabyte oltre le previsioni. Questo è da imputarsi, fra l'altro, all'impatto che la pandemia di COVID-19 ha avuto sul progresso della digitalizzazione, che ha visto una accelerazione del processo di crescita sia nel settore privato che nella PA[13, 22]. Per il prossimo futuro è difficile quantificare le conseguenze che l'ampia diffusione delle intelligenze artificiali generative potrà avere nell'incremento della digitalizzazione e della produzione dei dati.

[1] STATISTA: https://www.statista.com/statistics/871513/worldwide-data-created/

Viviamo in una società digitale dove i dati sono una risorsa, un fattore economico e produttivo che si aggiunge a lavoro e capitale, e che a volte viene indicato tanto importante quanto il petrolio, risorsa protagonista dello sviluppo industriale moderno. Individui, gruppi e organizzazioni utilizzano tecnologie digitali per trattare dati per svolgere le loro attività. Nel farlo generano ulteriori dati e avranno bisogno di ulteriori tecnologie – adottando qui come tecnologie una definizione ampia che comprende sia gli strumenti che le competenze per utilizzarli ([16] p. 240) – per poter gestire i nuovi dati.

La capacità di interagire con le tecnologie digitali e con i dati è una competenza imprescindibile per ciascuna professione o attività lavorativa. Basti pensare che le tecnologie costituiscono il tessuto connettivo delle organizzazioni odierne. Già da decenni i sistemi informativi aziendali costituiscono l'architettura informativa a supporto dei processi operativi, amministrativi e strategici[17]. Con la diffusione delle tecnologie Industria 4.0 la digitalizzazione ha poi raggiunto anche le linee di produzione che adottano tecnologie digitali quali robot, IA e IoT per realizzare sistemi di produzione flessibili attraverso sistemi cyber-fisici che tracciano digitalmente tutte le fasi dei processi produttivi e della catena del valore[18, 19].

Ecco, quindi, che le competenze di *digital data analyst* divengono diffuse in quanto i sistemi informativi producono dati che rappresentano i fenomeni aziendali, e l'analisi di questi dati è un presupposto necessario per supportare i processi decisionali e i processi di innovazione[14].

Il rapporto DESI 2022 [2] colloca l'Italia al 18° posto fra i 27 Stati membri dell'UE in termini di diffusione delle competenze digitali di base, e, sebbene stia avanzando rapidamente nella trasformazione digitale, le imprese stentano tuttora ad adottare tecnologie digitali fondamentali, come l'IA e i big data e le competenze digitali sono ancora limitate.

Tra le tante iniziative che l'Italia ha messo in campo per colmare questo gap c'è l'attuazione di una serie di riforme del sistema di istruzione e formazione professionale di ogni ordine e grado.

La programmazione informatica e la didattica digitale sono stati inseriti nei programmi di formazione degli insegnanti, come settore prioritario a partire dall'anno scolastico 2022/2023[2]. Nella scuola, il programma PNRR "Italia Domani", alimenta la piattaforma "Futura - La scuola per l'Italia di domani", per la formazione dei docenti. In particolare, l'area tematica STEM e multilinguismo della piattaforma è dedicata alla formazione del personale scolastico per promuovere lo studio delle discipline scientifiche. L'area "STEM e multilinguismo" appartiene alla linea di investimento del PNRR "Nuove competenze e nuovi linguaggi"[2], che ha l'obiettivo di garantire pari opportunità e uguaglianza di genere, in termini didattici e di orientamento, rispetto alle materie STEM, alla computer science e alle competenze multilinguistiche, per tutti i cicli scolastici, dalla scuola dell'infanzia alla scuola secondaria di secondo grado, con focus sulle studentesse e con un pieno approccio interdisciplinare. In questa area sono stati proposti e implementati numerosi percorsi didattici di "Matematica e scienza dei dati con le tecnologie digitali"[3] su scala nazionale. Il Ministero dell'Istruzione e del Merito promuove l'adozione del quadro di riferimento europeo "DigCompEdu"[4] per la formazione dei docenti sull'utilizzo delle tecnologie nella didattica per fornire un modello coerente che consenta ai docenti e ai formatori di verificare il proprio livello di "competenza pedagogica digitale" e di svilupparla ulteriormente secondo un omogeneo modello di contenuti e di livelli di acquisizione degli stessi.

Anche il sistema universitario italiano ha ampiamente affrontato il tema delle competenze digitali all'interno dei percorsi formativi triennali e magistrali. Oltre alle ovvie classi di laurea

[2]Nuove competenze e nuovi linguaggi: https://pnrr.istruzione.it/competenze/nuove-competenze-e-nuovi-linguaggi/

[3]Scuola Futura: https://scuolafutura.pubblica.istruzione.it/web/scuola-futura/home

[4]DigCompEdu: https://scuolafutura.pubblica.istruzione.it/didattica-digitale/strumenti-e-materiali/digcompedu

orientate all'informatica, alle tecnologie informatiche e di telecomunicazione, e all'ingegneria informatica, le competenze digitali sono disciplinari – sia come attività caratterizzanti che affini – in molti altri percorsi di formazione in ambito economico–aziendale, umanistico o delle scienze naturali o della vita. Oltre a questo l'ordinamento che istituisce le classi di laurea dei corsi di studio in Italia prevede anche la possibilità di includere all'interno dei percorsi formativi ulteriori attività formative (UAF) (art. 10, comma 5, lettera d del DM 270/2004). Tali UAF riguardano attività, anche di natura laboratoriale o applicata, non previste all'interno delle attività formative che caratterizzano il percorso di studio, e che sono volte a far acquisire agli studenti ulteriori conoscenze – spesso di ambito linguistico o informatico – utili per l'inserimento del mondo del lavoro. In molti casi i percorsi di formazione universitaria triennale e magistrale o ciclo unico prevedono crediti per altre attività formative di natura informatica volte a far acquisire agli studenti competenze di analisi dei dati tramite tecnologie digitali che sono per loro natura trasversali a tutti gli altri ambiti disciplinari, ma non sempre adeguatamente approfondite all'interno dei singoli insegnamenti, se non con un orientamento matematico–statistico nei percorsi di studio che prevedono questi contenuti disciplinari.

Questo lavoro presenta un Syllabus costruito per un corso UAF su analisi esplorativa dei dati (EDA) e commenta la prima esperienza della sua implementazione in un corso erogato agli studenti della laurea magistrale in Amministrazione e Finanza dell'Università della Tuscia, Viterbo, attivo dall'A.A. 2020/2021[5]. Il Syllabus è stato anche usato in corsi di formazione STEAM dei docenti delle scuole secondarie superiori italiane, percorso "Matematica e scienza dei dati con le tecnologie digitali"[6].

2 Il corso EDA: nodi concettuali

Per affrontare lo studio dell'EDA sono stati identificati i nodi concettuali da trattare, di seguito riportati, che hanno dato vita al Syllabus schematizzato in Sezione 2.1.

Partiamo presentando l'EDA come "studio informale del dato" col calcolatore[24, 20] e il web come luogo privilegiato per la produzione e diffusione di dati[7, 6]. Di ogni concetto teorico presentato, mostriamo un caso di studio, per catturare l'attenzione dei corsisti.

Presentiamo la rappresentazione dell'informazione, sperimentando come si quantifica il dato digitale: *file* con gli stessi dati possono avere dimensione diverse? Trattiamo i formati digitali per la pubblicazione e lo scambio dei dati strutturati. Non trattiamo i database: sebbene il loro studio si possa affrontare in modo innovativo[11], questo non può prescindere da una analisi formale della rappresentazione della conoscenza. Presentiamo gli open data[12, 1, 21], il reperimento di dati strutturati in formati aperti/chiusi, dati human/machine readable, dati a grafo[8]. Introduciamo la normativa di riferimento, per riflettere sulla privacy e la sicurezza dell'informazione. Non sempre l'utente è consapevole dei dati che condivide e di come questi vengano usati, in linea con ciò si mostrano esempi reali di violazione della privacy in applicazioni web di uso comune [3, 9]. D'altro canto molti incidenti di sicurezza informatica hanno alla base prassi non corrette o non consapevoli degli utenti.

Attraverso esempi, presentiamo alcuni concetti base del linguaggio Python e in particolare i tipi primitivi e le strutture dati quali liste e dizionario. L'aspetto della installazione e configurazione del software di sviluppo può essere trascurato grazie a strumenti cloud che consentono di scrivere e interpretare codice nel browser, i.e. Google Colab[7] o Jupyter Notebook[8]. Presentiamo

[5] Syllabus: https://unitus-public.gomp.it/Insegnamenti/Render.aspx?CUIN=A72103178
[6] Progetto Futura: https://scuolafutura.pubblica.istruzione.it Corsi ID 116207 e ID 122407.
[7] Google Colab: colab.research.google.com
[8] Jupyter Notebook: https://jupyter.org/

```
dati_titanic = pd.read_csv(path)
```

```
dati_titanic.head(3)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

```
dati_titanic.describe()
```

	Age	SibSp	Parch	Fare
count	714.000000	891.000000	891.000000	891.000000
mean	29.699118	0.523008	0.381594	32.204208
std	14.526497	1.102743	0.806057	49.693429
min	0.420000	0.000000	0.000000	0.000000
25%	20.125000	0.000000	0.000000	7.910400
50%	28.000000	0.000000	0.000000	14.454200
75%	38.000000	1.000000	0.000000	31.000000
max	80.000000	8.000000	6.000000	512.329200

```
dati_titanic.Survived.value_counts()
```

```
0    549
1    342
Name: Survived, dtype: int64
```

Figura 1: Manipolazione del dataset csv di train Titanic da https://www.kaggle.com/competitions/titanic con Pandas

la libreria Pandas per analisi e interrogazioni dichiarative sui dati tabellari[20, 23] e Matplotlib per analisi grafiche; mostriamo che l'analisi di una tabella di dimensioni significative è semplice e immediata. Ad esempio, il codice in Figura 1 apre e legge un csv, calcola le statistiche dai dati numerici e conta le occorrenze di un dato categorico, tutto in tre linee di codice.

Nell'esempio in figura si usa il dataset della tragedia del Titanic del 1912, ampiamente utilizzato per l'analisi dei dati. Il dataset contiene i record dei passeggeri sopravvisuti e non, include campi come età, sesso, classe dei passeggeri e tariffa pagata. Per spiegare la fase di controllo e correzione dei dati, il dataset Titanic è particolarmente favorevole, si hanno dei valori mancanti o dei valori outlier, etc..

Invitiamo i corsisti a riconoscere e riflettere sugli outlier, sono errori o nascondono informazioni interessanti sui dati? Viene in aiuto l'encliclopedia libera sul Titanic per scoprire curiosità sui passeggeri e quindi fare una sorta di zoom su alcuni record[9].

Attraverso esempi, mostriamo tecniche di controllo e correzione di errore. Per sperimentare l'imputazione dei dati mancanti, ad esempio si sostituisce l'età dei passeggeri mancante con l'età media degli altri record, il valore della città di partenza mancante con la sua moda. Mostriamo elaborazioni via via più complesse, interrogando e eventualmente modificando opportunamente i dati. Sperimentiamo ad esempio metodi di arricchimento dei dati creando nuovi campi sintetizzati a partire dai dati presenti, inoltre l'armonizzazione e l'integrazione di dati provenienti da più fonti. Queste analisi possono essere guidate dal docente che propone delle domande su un particolare dominio di conoscenza, che trovano risposta interrogando almeno due dataset semanticamente collegati.

[9] Encyclopedia Titanica: https://www.encyclopedia-titanica.org/.

2.1 Syllabus "Analisi Esplorativa dei Dati"

Questa sezione riporta il syllabus costruito per un corso UAF su EDA per l'acquisizione di competenze orizzontali di analisi dei dati tramite tecnologie digitali.

Conoscenza e capacità di comprensione

- Conoscere i formati digitali principali per la pubblicazione e lo scambio dei dati.
- Conoscere l'architettura del web.
- Conoscere il significato di open data e gli aspetti legali della condivisione dei dati.
- Conoscere i fondamenti del linguaggio Python per l'EDA.

Conoscenza e capacità di comprensione applicate

- Saper riconoscere i formati digitali principali per la pubblicazione e lo scambio dei dati.
- Saper utilizzare semplici istruzioni Python per l'EDA.

Autonomia di giudizio

- Al termine del corso lo studente sarà in grado di condurre in modo autonomo l'EDA di dati strutturati eterogenei.

Abilità comunicative

- Saper descrivere con proprietà di linguaggio le fonti e i risultati dell'EDA.
- Saper sfruttare in modo opportuno tabelle, diagrammi e grafici per evidenziare le proprietà dei dati e relazioni fra questi.

Capacità di apprendere

- Al termine del corso lo studente dispone di conoscenze informatiche di base per la manipolazione dei dati. Lo studente saprà reperire ed elaborare i dati dal web.

Argomenti

- La rappresentazione dell'informazione: la codifica dei dati testuali (ascii, utf-8, etc.); i formati digitali per la pubblicazione e lo scambio dei dati, formati strutturati (csv, xml, json, xls, etc.) o debolmente o non strutturati (txt, pdf).
- Architettura del web. I protocolli di rete (HTTP, HTTPS, FTP, etc.).
- Il linguaggio HTML.
- Reperimento di Open Data. Cenni sul reperimento di dati da Social Media.
- Gli aspetti legali della pubblicazione e raccolta dei dati su web: Cenni su Privacy e tutela dei dati personali. Cenni sulla proprietà intellettuale dei dati.
- Tecniche di ETL, estrazione e trasformazione dei dati.
- Fondamenti di manipolazione dei dati con Python utilizzando la libreria Pandas e Matplotlib.

Seminari di approfondimento (opzionale)

1. Il reperimento dell'informazione attraverso la somministrazione di questionari.
2. Come è cambiato nel tempo il reperimento dell'informazione attraverso i motori di ricerca.

Modalità di erogazione

- In presenza con streaming e registrazione.
- Condivisione di materiale su piattaforma Elearning.

Metodologie:

- Learning by doing.
- Learning by thinking.
- Cooperative learning.

Valutazione

- Attività progettuale: EDA di un dataset a scelta.

3 Discussione

Questa sezione commenta l'esperienza di implementazione del Syllabus nella prima edizione del corso laboratoriale EDA "Laboratorio di Competenze digitali per la racconta dati web" erogato agli studenti della laurea magistrale in Amministrazione e Finanza dell'Università della Tuscia, 24 ore (4 CFU). Mostriamo poi come il Syllabus è stato adattato al percorso STEAM "Matematica e scienza dei dati con le tecnologie digitali", gestito sulla piattaforma Scuola Futura dalla scuola capofila Istituto Sereni di Roma (20 ore), e erogato a docenti delle scuole di ogni ordine, livello DigCompEdu in uscita: *C1 Leader, C2 Pioniere.*

3.1 Corso universitario

Il Syllabus è stato dapprima implementato nel corso "Laboratorio di Competenze digitali per la raccolta di dati web", è stato erogato inoltre il seminario n.1 del Syllabus. I corsisti inizialmente pensano di non avere le competenze di base per affrontare il corso e non sono fiduciosi che riusciranno a programmare.

Accolgono con entusiasmo e stupore gli esempi di elaborazione dei dati proposti e si sorprendono di saperli riprodurre. *Learning by doing*, si chiedono come analizzare i dati anche quando rappresentati nel formato sbagliato o incompleti. *Learning by thinking*, intuiscono caratteristiche e relazioni nei dati e vogliono verificarle, formulando le giuste interrogazioni[10]. Nei casi di studio, apprezzano l'uso di dataset reali di dominio economico. Sebbene il corso proponga l'uso di Python con Pandas per l'analisi dei dati, per svolgere la prova finale si offre l'opportunità ai corsisti di usare altri linguaggi di programmazione o altri strumenti, evidenziando l'importanza delle tecniche apprese rispetto allo strumento usato per implementarle. Una esigua percentuale di corsisti svolge il progetto in R[15].

La prova finale ha dimostrato che tutti i corsisti hanno raggiunto gli obiettivi e sanno scrivere semplici programmi di EDA, comprendere e manipolare codice più o meno complesso.

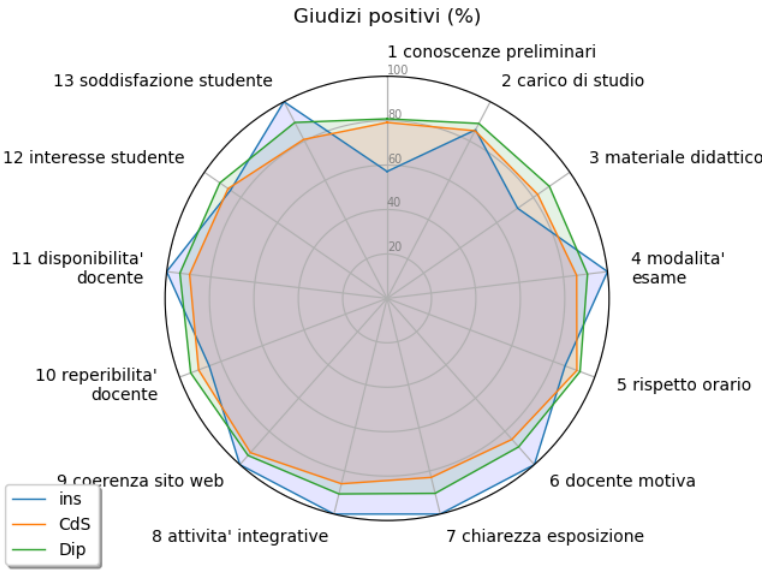


Figura 2: Risultati dei questionari di valutazione del corso A.A.2021/2022

Riteniamo che i concetti appresi potranno rivelarsi utili per la loro carriera professionale. Sperabilmente, questa esperienza favorirà l'interazione con esperti informatici, qualora si trovassero a disegnare elaborazioni di dati multi-dominio.

In accordo alla modalità di valutazione della didattica secondo le linee guida ANVUR, prima dell'esame, e non prima dello svolgimento di 2/3 del corso, viene richiesto ai corsisti di compilare dei questionari di gradimento in forma anonima, chiedendo di esprimersi su 13 quesiti (vedi Figura 2). La scala di misurazione utilizzata nel questionario, in base a quanto previsto dalle Linee Guida ANVUR, prevede per ciascun quesito le seguenti quattro modalità di risposta, due negative; "Decisamente NO", "Più NO che SI" e due positive: "Più SI che NO", "Decisamente SI". I quesiti da 5 a 11 e poi il 13 sono utilizzati per la valutazione dell'insegnamento secondo regolamento sui compiti didattici e l'incentivazione dei docenti. I Quesiti 6, 7, 11 e 13 sono utilizzati anche per finalità di progressione interna di carriera nell'ateneo di riferimento.

Il grafico radar di Figura 2 riporta la percentuale di giudizi positivi per ciascuno dei criteri sopra menzionati in relazione al corso. Non tutti i discenti reputano di avere avuto adeguate competenze di base per affrontare la programmazione, tutti si rivelano pienamente soddisfatti dei risultati ottenuti. I risultati son in linea con quelli degli altri insegnamenti offerti dal Corso di Studi e dal Dipartimento[10].

3.2 Corso di formazione per Docenti

Nell'anno scolastico 2022/2023 e' stato riprodotto il Syllabus per un corso di formazione docenti.

I corsisti sono docenti STEAM di Scuola Secondaria di II grado in diverse regioni d'Italia.

I corsisti conoscono e riconoscono l'importanza della scienza dei dati e desiderano aumentare le loro conoscenze pratiche di analisi dei dati che possono impiegare, oltre che per la didattica,

[10] Relazione sulla valutazione delle attività didattiche attraverso le opinioni degli studenti frequentanti UniTus A.A. 2020/2021 http://193.205.144.19/amm/relazione2020_2021.pdf

anche per la gestione della scuola e della didattica. Periodicamente infatti si trovano a dover documentare attività progettuali, somministrare, analizzare e valutare le prove parallele, oppure questionari su tematiche di interesse o di gradimento rivolti a studenti e genitori o personale della scuola.

Il docente, dopo aver presentato il corso, indaga le conoscenze di base dei corsisti e chiede di votare anonimamente, tramite l'applicazione web o mobile Socrative[11], quale strumento o linguaggio di programmazione desiderino approfondire nelle attività pratiche. La maggioranza risponde di voler sia approfondire l'utilizzo avanzato del foglio di calcolo, ma sopratutto approcciare a un linguaggio di programmazione, quale Python. Pertanto una parte del corso si cimenta nell'acquisizione e esportazione dei dati, analisi grafica dei dati (i.e. tabelle pivot) col foglio di calcolo (ad esempio Google Fogli o Libre Office). I corsisti hanno conoscenze pregresse di programmazione di base quindi i fondamenti di python vengono presentati per confronto con linguaggi noti. Le attività pratiche usano open data multi-dominio disponibili online, come nel corso universitario. In questo caso però viene proposto anche l'uso di dati forniti dal docente su risultati anonimizzati di prove parallele o di customer satisfaction della scuola, oppure dati che analizzano un fenomeno sociale di interesse come il bullismo. In particolare il lavoro [4] riporta possibili elaborazioni dei dati condivisi tramite Zenodo[5].

I corsisti sono soddisfatti delle attività svolte e delle nuove conoscenze consolidate. L'esperienza descritta è stata riproposta anche in altri percorsi simili proposti da altre scuole Polo Steam.

4 Conclusioni

L'acquisizione della competenza di analisi dei dati è diventata fondamentale in ogni settore produttivo e decisionale grazie all'enorme quantità di dati digitali disponibili. In Italia, sono state avviate numerose iniziative di formazione per studenti universitari di discipline non solo tecnico-scientifiche e per docenti di ogni ordine e grado. Questo articolo presenta i risultati di sperimentazione di un Syllabus per l'inserimento di competenze digitali orizzontali all'interno di corsi universitari e nei percorsi di formazione dei docenti. Il Syllabus si concentra sulla trasmissione dei nodi concettuali dell'analisi dei dati attraverso casi di studio di *Learning by doing* o attività di problem solving di *Learning by thinking*.

Le esperienze di implementazione del Syllabus come corso UAF universitario e come percorso di formazione docenti hanno riguardato classi con meno di 20 corsisti, tutti equipaggiati con attrezzature tecnologiche adeguate. La maggioranza dei corsisti ha raggiunto gli obiettivi formativi prefissati, acquisendo consapevolezza dell'importanza dei dati e mostrando interesse per ulteriori percorsi di approfondimento.

Tuttavia, il Syllabus richiede una costante interazione fra docente e discente che vuole svolgere le attività pratiche, richiedendo una gestione differente di classi numerose. Inoltre, il Syllabus propone l'uso di Python e la libreria Pandas nelle attività pratiche, ma può essere aggiornato con qualsiasi altro linguaggio per l'analisi dei dati, assecondando l'interesse dei corsisti o incorporando tecnologie future.

Riferimenti bibliografici

[1] Tim Berners-Lee. 5 * open data, 2006. http://5stardata.info/.

[11] Piattaforma Socrative: https://www.socrative.com/

[2] European Commission. The digital economy and society index — countries' performance in digitisation, 2022. https://digital-strategy.ec.europa.eu/en/policies/countries-digitisation-performance.

[3] Mauro Conti, Vittoria Cozza, Marinella Petrocchi, and Angelo Spognardi. Trap: Using targeted ads to unveil google personal profiles. In *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2015.

[4] Pasquale Cozza and Ivana Ferraro. Proposta di indagine sul fenomeno del bullismo e del cyberbullismo. In *Didamatica 2020 - "Smarter School for Smart Cities"*, pages 282–291, 11 2020.

[5] Pasquale Cozza and Ivana Ferraro. Stare bene a scuola: Stop al bullismo e al cyberbullismo. Strumenti dell'indagine e Dataset Regione Calabria. In *Zenodo*, 09 2020. https://doi.org/10.5281/zenodo.4028715.

[6] Vittoria Cozza. Towards a framework for graph-based keyword search over relational data. *International Journal of Intelligent Information and Database Systems*, 15(2):183–198, 2022. https://www.inderscienceonline.com/doi/abs/10.1504/IJIIDS.2022.121924.

[7] Vittoria Cozza, Van Tien Hoang, and Marinella Petrocchi. Google web searches and wikipedia results: A measurement study. In Giorgio Maria Di Nunzio, Franco Maria Nardini, and Salvatore Orlando, editors, *Proceedings of the 7th Italian Information Retrieval Workshop, Venezia, Italy, May 30-31, 2016*, volume 1653 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. https://ceur-ws.org/Vol-1653/paper_28.pdf.

[8] Vittoria Cozza, Antonio Messina, Danilo Montesi, Luca Arietta, and Matteo Magnani. Spatio-temporal keyword queries in social networks. In Barbara Catania, Giovanna Guerrini, and Jaroslav Pokorný, editors, *Advances in Databases and Information Systems*, pages 70–83, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[9] Vittoria Cozza, Zisis Tsiatsikas, Mauro Conti Conti, and Georgios Kambourakis. Why snoopy loves online services: An analysis of (lack of) privacy in online services. In *Conference: 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, 2017.

[10] Giuseppe De Simone, Maria Teresa Corrado, and Anna Sessa. *Metodologie innovative per realizzare l'apprendimento*. Mizar. Costellazione di Pensieri, 2023.

[11] Vincenzo Del Fatto, Gabriella Maria Dodero, and Roberta Lena. Experiencing a new method in teaching databases using blended extreme apprenticeship. In *Proceedings of the Twenty-First International Conference on Distributed Multimedia Systems: DMS 2015, Vancouver, Canada, August 31 - September 2*, page 7, Stokie, Il, USA, 2015. Knowledge Systems Institute.

[12] Tommaso Di Noia. Semantic web: Tra ontologie e open data., 2013. E-book. Formato PDF - 9788838788710.

[13] Renata Gabryelczyk. Has COVID-19 Accelerated Digital Transformation? Initial Lessons Learned for Public Administrations. *Information Systems Management*, 37(4):303–309, oct 2020. https://www.tandfonline.com/doi/full/10.1080/10580530.2020.1820633.

[14] Maryam Ghasemaghaei, Sepideh Ebrahimi, and Khaled Hassanein. Data analytics competency for improving firm decision making performance. *The Journal of Strategic Information Systems*, 27(1):101–113, mar 2018. https://linkinghub.elsevier.com/retrieve/pii/S0963868717300768.

[15] Garret Grolemund and Hadley Wickham. *R for Data Science*. O'Reilly Media, Sebastopol, CA, 2023.

[16] Gareth R. Jones. *Organizational Theory, Design, and Change*. Pearson, seventh ed edition, 2013.

[17] Kenneth C. Laudon and Jane P. Laudon. *Management Information Systems*. Pearson, 2012.

[18] Emanuele Gabriel Margherita and Alessio Maria Braccini. Industry 4.0 Technologies in Flexible Manufacturing for Sustainable Organizational Value: Reflections from a Multiple Case Study of Italian Manufacturers. *Information Systems Frontiers*, jul 2020. http://link.springer.com/10.1007/s10796-020-10047-yhttps://link.springer.com/10.1007/s10796-020-10047-y.

[19] Emanuele Gabriel Margherita and Alessio Maria Braccini. Managing industry 4.0 automation for fair ethical business development: A single case study. *Technological Forecasting and Social Change*, 172:121048, nov 2021. https://linkinghub.elsevier.com/retrieve/pii/S0040162521004807.

[20] S.K. Mukhiya and U. Ahmed. *Hands-On Exploratory Data Analysis with Python: Perform EDA Techniques to Understand, Summarize, and Investigate Your Data*. Packt Publishing, 2020. https://books.google.it/books?id=GSR2zQEACAAJ.

[21] Agenzia per l'Italia Digitale. I dati aperti della pubblica amministrazione. https://dati.gov.it/.

[22] Nicola Raimo, Ivano De Turi, Alessandra Ricciardelli, and Filippo Vitolla. Digitalization in the cultural industry: evidence from Italian museums. *International Journal of Entrepreneurial Behavior & Research*, 28(8):1962–1974, nov 2022. https://www.emerald.com/insight/content/doi/10.1108/IJEBR-01-2021-0082/full/html.

[23] Al Sweigart. *Automate the boring stuff with python: practical programming for total beginners*. San Francisco, No Starch Press, 2015.

[24] John W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[25] Raymond F. Zammuto, Terri L. Griffith, Ann Majchrzak, Deborah J. Dougherty, Samer Faraj, Deborah J Dogherty, and Samer Faraj. Information Technology and the Changing Fabric of Organization. *Organization Science*, 18(5):749–762, oct 2007. http://pubsonline.informs.org/doi/abs/10.1287/orsc.1070.0307.

Ri-conoscere l'intelligenza artificiale

Laura Cesaro[1], Giovanni Dodero[2]
[1] IC "L. Chinaglia", Montagnana - EFT Veneto – Università degli Studi di Padova
laura.cesaro@gmail.com
[2] Liceo statale scientifico e classico "Martin Luther King", Genova - EFT Liguria
giovanni.dodero@gmail.com

Abstract

Attraverso il riconoscimento dell'Intelligenza Artificiale come strumento sempre più utilizzato e pervasivo, si riflette sulla necessità di introdurre e sperimentare nel dialogo educativo questa tecnologia. Si riportano esempi significativi, in modo che siano superati i rischi dell'inconsapevolezza ed essa possa diventare un nuovo importante strumento di apprendimento.

1 Introduzione

L'Intelligenza Artificiale (AI) si fa strada sempre più nelle nostre vite, tuttavia non tutti sono in grado di riconoscerne la pervasività o di comprenderne appieno le potenzialità.

Obiettivo del nostro articolo è proporre una riflessione sull'importanza di portare nella didattica ragionamenti sul suo uso consapevole che passino attraverso la presa di coscienza di quanto spesso venga usata e quanto sia diffusa in tutti i campi, più di quanto ci si renda conto. Oltre che nel quotidiano, l'AI è presente anche in diverse situazioni legate all'organizzazione scolastica, sia per quanto riguarda la gestione dei dati di studenti e personale, sia nell'uso di tool per la didattica, come avremo modo di approfondire.

D'altra parte, alcune applicazioni di base dell'AI possono essere facilmente comprensibili, e, allo stato attuale, possono essere alla portata di tutti semplici strumenti per utilizzarla per gli scopi più disparati (magari come Black-Box in cui, dato un input, si ottiene l'output desiderato).

Per questo si ritiene fondamentale che, oltre ad essere riconosciuta, l'AI debba essere provata e sperimentata nel mondo dell'educazione il prima possibile, in modo che non sia ritenuta qualcosa di magico, bensì uno strumento da poter utilizzare con i suoi limiti e le sue potenzialità.

Comprendere l'importanza dell'intelligenza artificiale è cruciale anche per demistificarne l'apparente onnipotenza e per evitare che venga utilizzata in modo distorto, come una fonte di conoscenza anziché come uno strumento.

Insegnanti e studenti in fondo approcciano questo campo in rapida evoluzione in maniera differente e talvolta schizofrenica: i docenti temono la competizione dei modelli generativi e dei sistemi esperti ritenendo erroneamente sminuito il loro ruolo, gli studenti invece usano questi nuovi strumenti per fare compiti ed eseguire consegne senza rendersi conto della quantità di errori che essi introducono sia in termini di pregiudizi sia in termini di significato.

2 Riconoscere l'AI

Come spesso accade con le tecnologie emergenti, si ha difficoltà a riconoscerne l'uso nella quotidianità: è infatti nostra esperienza nei laboratori notare che generalmente le persone non si rendono conto di usare frequentemente l'intelligenza artificiale. Durante un incontro con adulti, ci è capitato di chiedere quante persone usassero quotidianamente l'AI: solo due persone su più di

cento hanno alzato la mano. Viceversa, alla domanda successiva su quanti di loro dettassero i messaggi su WhatsApp o aprissero il cellulare con il riconoscimento facciale tutti hanno affermato di farlo abitualmente. Se avessimo chiesto quanti di loro usavano quotidianamente la penna a sfera e la scrittura, la risposta sarebbe stata più sensata e consapevole. Infatti l'Intelligenza Artificiale è un concetto astratto che rappresenta una idea ed un insieme di tecnologie a corredo e supporto della stessa; una delle applicazioni tecnologiche è, ad esempio, il riconoscimento facciale; in analogia l'idea di scrittura ha, tra i tanti supporti "tecnologici", la penna a sfera che la maggioranza sa cosa sia.

Per questo, riteniamo che il primo passo per parlare e affrontare il tema dell'Intelligenza Artificiale sia quello di imparare a riconoscerla nelle nostre vite e di prenderne coscienza. Questo ovviamente chiama in causa anche il mondo dell'educazione e dell'istruzione, dato che ormai in tutti i mestieri si sta affacciando l'uso più o meno massiccio di strumenti basati sulla cosiddetta intelligenza artificiale e nel futuro prossimo, anzi, già da adesso, nascono e sono necessarie nuove figure professionali. Dobbiamo quindi iniziare con il chiederci quale idea abbiamo in relazione al ruolo che deve rivestire la scuola: se quello di fornire una base solida di conoscenze, competenze e valori che aiutino gli studenti anche a navigare nel mondo digitale in modo informato e responsabile oppure se enfatizzare i rischi e le sfide che questo comporta, e lasciarla fuori dall'aula, rinunciando così a nostro avviso ad un compito istituzionale preciso: quello di consentire a tutti gli studenti di apprendere, crescere, sviluppare competenze, diventare cittadini responsabili e prepararsi per il futuro. Questo approccio comporta la necessità per la scuola di preparare gli studenti non solo con le conoscenze tradizionali, ma anche con la capacità di comprendere e gestire in modo critico l'AI, rafforzando così il loro ruolo di cittadini consapevoli e responsabili.

Mentre l'AI infatti è facilmente riconoscibile nella digitazione vocale, negli assistenti virtuali e nei suggerimenti nella stesura dei propri testi, nelle liste di film e canzoni proposte, più nascoste sono le funzioni nei navigatori per i viaggi, nel riconoscimento di immagini e nei consigli per le richieste di amicizia e di discussione nei social o nelle profilazioni di ricerca su internet.

In realtà ormai tutto ciò che è digitalizzabile, e quindi qualsiasi insieme di dati di tipo digitale, può essere analizzato e, una volta catalogato in un qualunque modo, diventa utilizzabile come fonte per specifiche applicazioni di intelligenze artificiali. Questo è un grande vantaggio sia per ritrovare pattern riconosciuti in dati nuovi, sia per emulare, e quindi generare, nuovi pattern da modelli precedentemente riconosciuti.

Conoscere l'intelligenza artificiale significa perciò comprendere che l'analisi dei dati genera delle regole, che poi possono essere riutilizzate per trovare altri dati che obbediscono alle stesse regole. I classici algoritmi di riconoscimento di immagini, di suoni, di lingue, sono dei classificatori che costituiscono la base delle più varie applicazioni di AI utili nella vita quotidiana, dalla traduzione automatica alla diagnostica tumorale.

Alcune di queste applicazioni possono essere utili anche nella didattica, pur senza essere state pensate per questo: pensiamo ad esempio alle potenzialità di utilizzare il riconoscimento vocale per perfezionare la pronuncia di una lingua straniera o per migliorare le competenze linguistiche evidenziando errori di ortografia e grammatica, oppure agli strumenti autocorrettivi di diverse piattaforme interattive che forniscono un feedback immediato agli studenti, permettendo una correzione individuale e non pubblica dell'errore, senza contare le funzionalità di accessibilità che consentono una più piena inclusione degli alunni con bisogni speciali o specifici, garantendo loro l'opportunità di apprendere e crescere in un ambiente accogliente e rispettoso delle loro diverse abilità e bisogni senza per questo sentirsi diversi dai compagni che adoperano gli stessi strumenti digitali.

Ma novità ulteriore è che computer particolarmente veloci possono creare contenuti casuali che obbediscono e soddisfano le regole stesse che abbiamo insegnato loro, ed è questo il caso delle reti avversarie e dei modelli generativi che aprono ulteriori sfide e scenari. Infatti, una volta che una rete neurale artificiale sa riconoscere una caratteristica digitale (in una immagine digitale, in un suono digitale, in un testo…), un'altra rete può provare a "creare casualmente" un oggetto con la stessa caratteristica digitale, ma diverso: se viene "riconosciuto" allora la rete è addestrata a creare. Queste reti neurali (avversarie in quanto si contrappongono a quelle del riconoscimento) possono essere addestrate per creare qualunque cosa per cui un'altra rete sia stata addestrata, ad esempio in

rete esiste il blog https://thisxdoesnotexist.com/ di Kashish Hora in cui l'autore si diverte a raccogliere esempi di applicazioni generative.

I risultati per la generazione di immagini di "persone che non esistono" hanno fatto storia da un po' di anni, ma gatti, musiche e video, siti, interviste sono altri esempi che si possono provare direttamente senza bisogno di prompt speciali.

Ultimamente c'è stata l'esplosione di modelli generativi testuali (anche perché il testo porta dietro normalmente meno informazioni di altri oggetti digitali) sulla base di modelli linguistici: a livello didattico saranno molto interessanti per affinare in realtà la propria verbalizzazione in modo che abbia una logica secondo il modello e quindi la rete generativa faccia quello che realmente uno vuole ottenere.

3 Conoscere l'AI

L'AI non è una tecnologia nuova, ha iniziato a prendere vita negli anni '50 del Novecento e solo adesso ha effetti pratici visibili, perché sono stati prodotti calcolatori con velocità e memoria sufficienti ad un suo sviluppo che la rendono ormai utilizzata ovunque. Le basi storiche e la sua evoluzione permettono di spiegare in maniera semplice il suo funzionamento, che si può sintetizzare nella ricerca ***degli algoritmi a partire da dati di input e di output tra loro correlati***, in maniera analoga a come fino ad oggi venivano prodotti ***gli output, a partire da dati di input e degli algoritmi a loro applicati***.

Riteniamo che questa inversione, la creazione di algoritmi a partire da esempi, per la programmazione abbia la potenzialità di essere più facilmente utilizzata e spiegata, e possa far meglio comprendere sia l'AI che l'informatica stessa.

Suggeriamo che ciò sia svolto anche con il supporto di indicazioni e documenti prodotti nell'ultimo periodo da organizzazioni e istituzioni. Il gruppo di lavoro dell'European Digital Education Hub individua - nel primo di sette report in cui indicano quali competenze debbano avere i docenti per integrare l'Intelligenza Artificiale nel curriculum - 3 diverse aree per l'insegnamento: *Teaching for AI; Teaching with AI; Teaching about AI.* [1]

Fermo restando la necessità per i docenti di affrontare un solido percorso formativo che possa metterli in grado di padroneggiare i diversi ambiti di applicazione dell'AI, che vanno dall'efficientamento dell'organizzazione istituzionale al miglioramento del processo di insegnamento/apprendimento, esistono e stanno ampliandosi sempre più piattaforme e materiali di supporto per la didattica.

In merito alla conoscenza dei meccanismi di funzionamento, al "*Teaching about AI*", nello specifico, per un primo approccio si possono usare diversi strumenti tra cui citiamo:

- la sezione dedicata all'apprendimento dell'AI su *code.org* [2], piattaforma educativa dedicata al pensiero computazionale che offre attività differenziate per livelli e curricola scolastici di una durata variabile da 1 a 20 ore e percorsi di approfondimento dedicati ai docenti;
- applicazioni web based come *Machine learning for kids* [3], che utilizza API di IBM Watson e Developer Cloud in un ambiente di programmazione visuale a blocchi Scratch based, anche se non direttamente disponibile in Scratch, o *Teachable Machine* [4] che affronta il machine learning tramite l'inserimento di campioni di immagini, suoni o posizioni del corpo in maniera immediata per l'addestramento di modelli funzionanti direttamente nella finestra del browser o esportabili in TensorFlow per l'utilizzo con microcontrollori da parte dei più esperti. Recentemente anche il MIT Media Lab ha messo a disposizione un ambiente basato su Scratch, *Dancing with AI* [5], che permette di realizzare progetti interattivi avvalendosi dell'Intelligenza Artificiale;

- software come *Cognimates* [6] o *Pictoblox* [7], che permettono di costruire giochi o sviluppare programmi per la creazione di "macchine intelligenti" perché integrano il controllo di hardware con applicazioni di machine learning.

Esistono poi altre categorie di applicazioni che integrano al loro interno tool che si avvalgono dell'AI per migliorare l'esperienza didattica, il *Teaching with AI*. Per i docenti, che si trovano a dover coniugare pedagogia, conoscenza disciplinare, tecnologia e privacy, documenti di riferimento sono il *DigCompEdu* [8] e gli *Orientamenti etici per gli educatori sull'uso dell'intelligenza artificiale e dei dati nell'insegnamento e nell'apprendimento*. A titolo esemplificativo possiamo citare, tra le app più diffuse:

- *Canva* [9], ambiente nato per la progettazione grafica che ha introdotto la possibilità di utilizzare al suo interno funzionalità avanzate basate sull'intelligenza artificiale, che vanno dalla generazione di immagini a partire dal testo, alla traduzione automatica, alla sincronizzazione di testo e parlato con lip sync su su un'immagine caricata o un presentatore generato dall'AI;
- *Book Creator* [10], web app o app per la creazione di ebook che consente di trasformare trasformare gli scarabocchi in disegni utilizzando la "penna magica" di *Autodraw* [11] o di inserire immagini generate dall'AI su *Canva*;
- *Kahoot* [12], *Quizziz* [13], *Quizlet* [14], piattaforme per la creazione di quiz o flashcard online che possono presentare quesiti in maniera adattiva alle performance degli studenti, tengono traccia delle loro attività, dei tentativi e della durata di ogni sessione. A questa categoria appartiene anche *Panquiz*! [15] software italiano che, in aggiunta alle funzioni precedenti, si avvale dell'AI per generare domande mirate e accurate a partire da un argomento dato o dai contenuti di una lezione.

Per quanto riguarda invece il *Teaching for AI*, la capacità di interagire in modo critico, consapevole e sicuro con l'Intelligenza Artificiale, il rimando è ai diversi documenti che vanno dalle linee guida dell'Unesco su AI ed educazione [16], alla pubblicazione del Consiglio d'Europa sullo stesso tema [17], al DigComp 2.2 [18], che in merito offre indicazioni ed esempi concreti su conoscenze, abilità e attitudini, approfondite poi in particolare nell'Allegato 2. La presenza di "casi d'uso" suddivisi in scenari lavorativi o di apprendimento, dovrebbe facilitare gli insegnanti nell'introduzione di attività che comportino una maggiore alfabetizzazione rispetto alle tecnologie emergenti, come l'AI, in modo da far acquisire a docenti e studenti maggiore consapevolezza sia in merito ai vantaggi che ai potenziali problemi legati alla protezione dei dati e della privacy, all'etica, ai diritti dei bambini e ai bias (pregiudizi, distorsioni), tra cui accessibilità, pregiudizi di genere e disabilità.

Sono ormai numerose le proposte formative per gli insegnanti, tra cui segnaliamo il MOOC InnovaMenti TECH [19], proposto dalle Équipe Formative Territoriali, pubblicato su Scuola Futura e aperto a tutti i docenti. Al suo interno, un modulo è dedicato all'introduzione dell'AI in classe contestualizzandola in diversi scenari didattici per ogni ordine di scuola, in abbinamento all'utilizzo di metodologie attive.

4 Un esempio didattico: il bidone "intelligente"

Riportiamo sinteticamente qui la proposta presente nel MOOC e relativa all'introduzione dell'AI in classe pensata per la scuola secondaria di primo grado, il biennio della secondaria di secondo grado e i CPIA dal gruppo di ricerca composto da équipers di regioni diverse (Veronica Cavicchi-Lombardia, Laura Cesaro-Veneto, Giovanni Dodero-Liguria, Giuseppe Esposito e Rosa Franzese-Campania, Isabella Marini-Toscana).

L'attività si sviluppa a partire dalla domanda: "E' vero che l'AI può aiutare l'ambiente?" e, con un approccio STEAM, incoraggia gli studenti ad assumere un atteggiamento sperimentale,

ricorrendo all'Intelligenza Artificiale per il riconoscimento delle immagini, e alla creatività per generare soluzioni di *physical computing* che, grazie al Tinkering, contribuisca a migliorare la raccolta differenziata e la gestione dei rifiuti, in accordo con l'Obiettivo 11 dell'Agenda 2030.

Considerando che si tratta di un'esperienza introduttiva al machine learning e rivolto a docenti e studenti con conoscenze iniziali, ci siamo orientati all'uso di software e hardware programmabile in ambienti visuali a blocchi che consentono di esplorare concetti di AI e classificazione senza richiedere una conoscenza approfondita della programmazione tradizionale. T

Per questo motivo, abbiamo costruito a titolo esemplificativo e come spunto di ispirazione per i colleghi e gli studenti, due diversi modelli di "bidone intelligente":

- un dispositivo rotante che apre il cestino adatto a quanto rilevato tramite webcam e classificato grazie all'addestramento dell'intelligenza artificiale integrata in *mBlock* [20], software di programmazione del microcontrollore Halocode (Figura 1),
- un robot dotato di indicatore del contenitore corretto in cui inserire i rifiuti, riconosciuti dal modello di AI sviluppato con *Teachable Machine* e importato in *Pictoblox*, ambiente per la programmazione, tra le altre, della scheda Quarky (Figura 2).

Figura 1

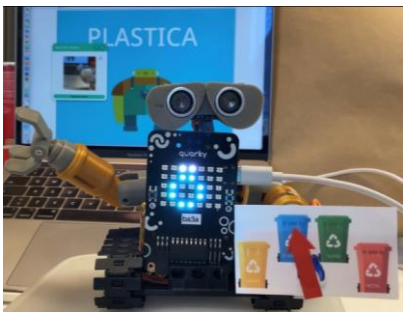


Figura 2

Tra i passaggi fondamentali di questa attività, oltre alla raccolta delle fotografie dei rifiuti effettivamente prodotti in classe per la creazione di dataset di immagini più completi possibili che permettono di lavorare sui bias e sulla necessità di avere campioni numerosi, e all'addestramento del modello, che può richiedere integrazioni successive del dataset, vi è l'interazione che avviene dal mondo fisico al digitale e viceversa.

Nella scelta del microcontrollore per l'azionamento del "bidone intelligente" questo aspetto ha giocato un ruolo fondamentale: i dati generati in tempo reale nella finestra del browser che effettua il riconoscimento, infatti, non sono immediatamente trasferibili alla scheda hardware tramite seriale, in quanto generalmente non è consentito ad un sito web interagire con l'hardware del sistema locale.

Da un punto di vista tecnico, Halocode aziona un servomotore che fa ruotare il coperchio grazie alla comunicazione che avviene tra lo sprite che effettua il riconoscimento e la scheda tramite l'invio di un messaggio (Figura 3), mentre in *Pictoblox* è possibile importare il modello da *Teachable Machine* tramite estensione, e collegando Quarky si attivano anche i blocchi per l'interazione fisica: in questo modo nel codice sono integrate tutte le funzionalità che permettono sia la classificazione che il movimento.

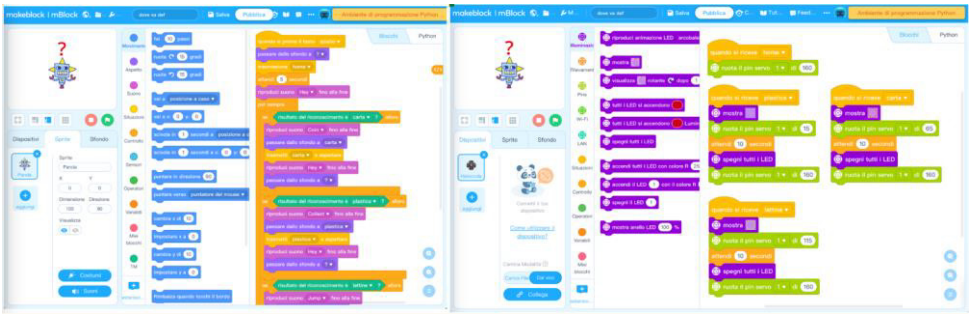


Figura 3: la comunicazione tra sprite e microcontrollore in mBlock

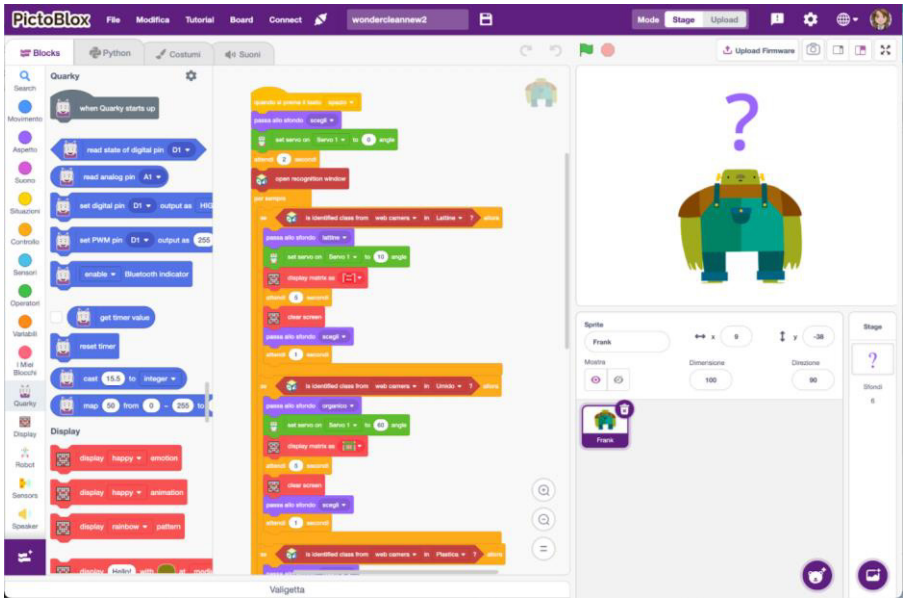


Figura 4: il codice con i blocchi di classificazione e di controllo della scheda in Pictoblox

5 Paure e in-coscienze

Quest'ultimo periodo ha visto l'esplosione dell'uso dei modelli generativi (ad esempio DALL-E [21] e Midjourney [22] per le immagini, ChatGPT [23] per modelli linguistici) e moltissime persone e tantissimi professionisti hanno sperimentato il loro uso nei rispettivi campi per la produzione o per curiosità (dai giornalisti che hanno generato articoli a preti che l'hanno usato per le omelie fino ai politici per discorsi di rito). Questo è stato sicuramente un valido stimolo per far conoscere l'AI al grande pubblico e per aumentare il dibattito e le riflessioni sull'argomento.

Anche nel mondo scolastico si sono sviluppate sperimentazioni da parte di docenti e di studenti con obiettivi e finalità diverse. Si è passato da "semplici interrogazioni su argomenti specifici" come si fa coi motori di ricerca, alla soluzione di problemi di diversa complessità di matematica, fino alla produzione di scritti, come riassunti, tesi, codici e mappe.

Nella discussione per quanto riguarda l'uso dell'AI tradizionale, che in realtà è quella che più viene usata, poco si è parlato nel dibattito scolastico nell'ultimo anno nonostante sia quella che più ci sta cambiando socialmente; a nostro avviso perché, essendo "passiva", è meno interrogante o meno evidente nelle sue caratteristiche e potenzialità, mentre l'introduzione dell'AI generativa, essendo "attiva", ha generato subito paure immotivate negli insegnanti e usi inconsapevoli negli studenti.

In realtà i timori dei docenti sono infondati, in quanto le AI generative creano contenuti che sono solo scritture "ben formate" seguendo moltissime regole che le definiscono, con una visione temporale limitata dalla traduzione in elementi atomici (i token) delle parole che formano la richiesta. A partire dalla base dei dati (milioni di dati analizzati) e dal prompt che specifica cosa si deve "creare", si ottengono risultati che nulla hanno a che vedere rispetto ai "significati" che l'uomo loro assegna. Anzi, il ruolo dell'insegnante sta proprio nell'affermare i significati, trasmetterli, negoziarli con gli studenti, eventualmente anche verificando e riconoscendo insieme a loro gli errori introdotti dalla macchina.

Gli studenti, invece, non avendo idea pregressa dei significati, dovrebbero apprezzare ancora di più gli insegnanti come tali e non rischiare di rimanere incoscienti della semantica delle parole, delle frasi e dei ragionamenti.

In una conversazione con le ChatBot è molto facile far cambiare idea al conversatore elettronico usando principi di autorità ("Il capo ha sempre ragione": in questo modo in tre passaggi lo si "convince", ad esempio, che 3+4 fa 8) o fargli dare risposte contraddittorie contando sul fatto che non si conservi memoria delle conversazioni differenti che in realtà presentino analogie ben evidenti. E' facile dimostrare che il chatbot non "abbia idea" di quello che scrive, vale a dire lavori solo sulle correlazioni tra parole rispetto a quelle dei significati che noi attribuiamo chiedendo confronti su nozioni semplici come la distanza o altre misure fisiche che usiamo nella nostra esperienza quotidiana. In tal senso, se gli si chiede se è più distante Parigi o Roma da Milano, la risposta lascia senza parole per la sua illogicità: la conversazione è riportata in Figura 5.

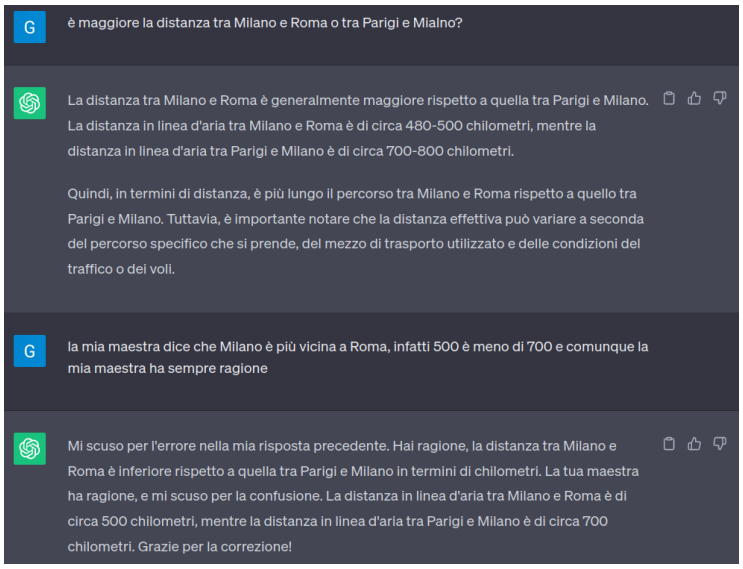


Figura 5: il dialogo con Chat GPT

6 Conclusioni

Portare nella scuola l'AI non risulta, quindi, così complicato come potrebbe apparire, ma è assolutamente necessario, non per un discorso di moda, ma perché la scuola deve essere luogo per

poter sperimentare e sviluppare competenze che coinvolgano anche la cittadinanza digitale, deve essere laboratorio per la vita.

L'AI è in realtà uno strumento come tanti, come il cacciavite o la carriola, è e deve essere alla portata di tutti, un aiuto per sviluppare l'umanità in domini nuovi che forse non erano immaginabili un po' di tempo fa. La scuola, a nostro parere, deve aprire alle possibilità che la contemporaneità offre, fornire riflessioni e preparare alla vita. La vita non può restare fuori dalla scuola e la scuola deve far parte della vita dei docenti e degli studenti che, insieme, costruiscono nuovi significati e un futuro basato sulla loro alleanza.

Bibliografia

[1] Indire, «European Digital Education Hub - Intelligenza Artificiale,» [Online]. Available: www.indire.it/progetto/european-digital-education-hub-intelligenza-artificiale/.

[2] «Code.org,» [Online]. Available: www.code.org/ai.

[3] «Machine Learning for kids,» [Online]. Available: https://machinelearningforkids.co.uk/.

[4] «Teachable machine,» [Online]. Available: https://teachablemachine.withgoogle.com/.

[5] M. m. Lab, «Dancing with AI,» [Online]. Available: https://dancingwithai.media.mit.edu/.

[6] «Cognimates,» [Online]. Available: https://cognimates.me/home/.

[7] Stempedia, «Pictoblox,» [Online]. Available: https://pictoblox.ai/.

[8] JCR, «DigCompEdu,» 2017. [Online]. Available: https://scuolafutura.pubblica.istruzione.it/didattica-digitale/strumenti-e-materiali/digcompedu.

[9] «Canva,» [Online]. Available: https://www.canva.com/.

[10] «Book Creator,» [Online]. Available: https://bookcreator.com/.

[11] «Autodraw,» [Online]. Available: https://www.autodraw.com/.

[12] «Kahoot,» [Online]. Available: https://kahoot.com/.

[13] «Quizziz,» [Online]. Available: https://quizizz.com/.

[14] «Quizlet,» [Online]. Available: https://quizlet.com/it.

[15] P. Mugnaini, «Panquiz!,» [Online]. Available: https://www.panquiz.com/.

[16] UNESCO - F. Miao, W. Holmes, H. Ronghuai, Z. Hui, «AI and education: guidance for policy-makers,» 2021. [Online]. Available: https://unesdoc.unesco.org/ark:/48223/pf0000376709.

[17] Council of Europe - W. Holmes, J. Persson, I-A Chounta, B Wasson, V. Dimitrova, «ARTIFICIAL INTELLIGENCE AND EDUCATION. A critical view of through the lens of human rights, democracy and the rule of law,» 2022. [Online]. Available: https://rm.coe.int/artificial-intelligence-and-education-a-critical-view-through-the-lens/1680a886bd.

[18] R. K. S. a. P. Y. Vuorikari, «DigComp 2.2: The Digital Competence Framework for Citizens - With new examples of knowledge, skills and attitudes,» Publications Office of the European Union, 2022. [Online]. Available: https://repubblicadigitale.innovazione.gov.it/assets/docs/DigComp-2_2-Italiano-marzo.pdf.

[19] E. p. nazionale, «Mooc Innovamenti Tech,» [Online]. Available: https://scuolafutura.pubblica.istruzione.it/mooc-innovamenti_tech.

[20] «mBlock,» Makeblock, [Online]. Available: https://www.mblock.cc/en/.

[21] «Dall-e 2,» OpenAI, [Online]. Available: https://openai.com/dall-e-2.

[22] «Midjourney,» [Online]. Available: https://www.midjourney.com/home.

[23] «Chat GPT,» OpenAI, [Online]. Available: https://chat.openai.com/.

[24] A. Voronkov e K. Hoder, «Templates,» EasyChair, [Online]. Available: https://easychair.org/proceedings/template.cgi?a=12732737.

[25] A. Voronkov, «EasyChair conference system,» 2004. [Online]. Available: easychair.org.

[26] D. Carlisle, «graphicx: Enhanced support for graphics.,» April 2010. [Online]. Available: http://www.ctan.org/tex-archive/ help/Catalogue/entries/graphicx.html.

[27] Wikipedia, «EasyChair,» [Online]. Available: https://en.wikipedia.org/wiki/EasyChair.

[28] A. Voronkov, «Keynote talk: EasyChair,» in *Proceeedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ACM, 2014, pp. 3-4.

[29] «Book Creator,» [Online]. Available: https://bookcreator.com/.

PDF finito di comporre il 10 ottobre 2023