Manuali

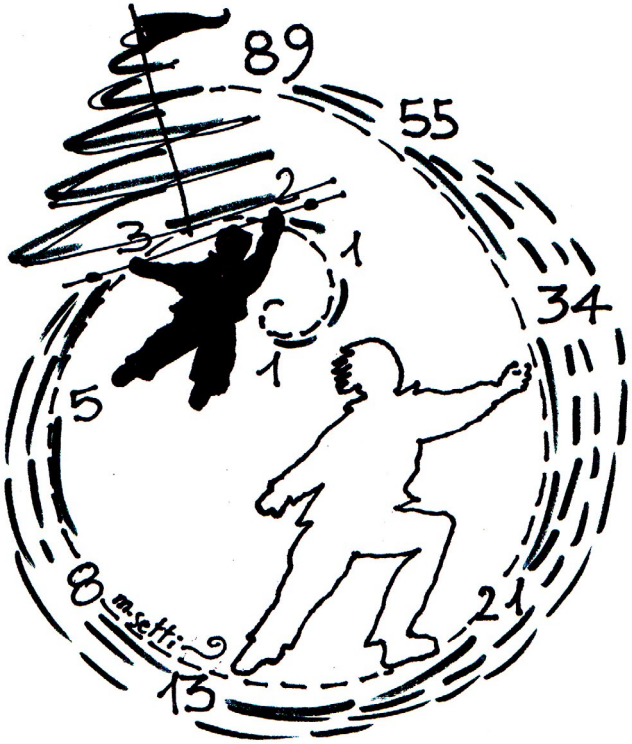
Scienze e Tecnologie – Formazione

E questo tutti chiamano Informatica

L'esperienza dei TFA nelle discipline informatiche

a cura di

Anna Labella



Sapienza Università Editrice

Collana Manuali 14

Scienze e Tecnologie
Serie Formazione

E questo tutti chiamano Informatica

L'esperienza dei TFA nelle discipline informatiche

a cura di
Anna Labella

SAPIENZA
UNIVERSITÀ EDITRICE
2015

Sapienza Università Editrice
Piazzale Aldo Moro 5 – 00185 Roma
www.editricesapienza.it
editrice.sapienza@uniroma1.it
Iscrizione Registro Operatori Comunicazione n. 11420
ISBN 978-88-98533-63-3
DOI 10.13133/ 978-88-98533-63-3

Distribuita su piattaforma digitale da:

Centro interdipartimentale di ricerca e servizi
Settore Publishing Digitale

In copertina: Marco Setti, *Respiro del mondo* (2015).

A tutti i nostri allievi
e ai loro allievi

Indice

Presentazione

Poca favilla gran fiamma seconda.
(Paradiso I, 34)

Questo volume affronta il problema di trasmettere il sapere guidando la nuova generazione e facendo sì che possa prendere in mano il testimone di qui a pochi anni.

Dalle scelte di cultura che facciamo oggi dipendono, infatti, il nostro futuro, il ruolo attivo o passivo che giocherà l'Italia, con la conseguente ricaduta su occupazione e benessere, ma anche il riconoscimento dei valori nei quali crediamo.

Saremo soltanto consumatori o produttori di cultura? Un paese che rincorre l'innovazione o uno che la genera e la promuove? Per promuoverla abbiamo bisogno di mettere in valore tutte le capacità innovative dei singoli, non in una concezione elitaria, ma in uno sforzo di tutti favorito da una corretta legislazione che non deprima l'immaginazione.

Data la velocità dell'evoluzione delle conoscenze e delle possibilità tecnologiche, la collaborazione tra scuola e università, tra didattica e ricerca diventa uno snodo importante. Ambedue queste istituzioni hanno bisogno di rinnovarsi per continuare a svolgere il loro ruolo, ma devono farlo in maniera armonica.

Alcune delle modifiche migliorative apportate dalla Commissione Cultura della Camera al disegno di legge di riforma del sistema nazionale di istruzione e formazione e delega per il riordino delle disposizioni legislative vigenti, c.d. per "La Buona Scuola", vanno in questa direzione grazie ad una delega al governo con precisi principi e criteri direttivi che ne delineano e chiariscono gli obiettivi: non può esserci

buona scuola senza buoni insegnanti. E per avere buoni insegnanti occorre formarli bene, reclutarli in base al merito e sostenerli affinché aggiornino competenze e metodologie didattiche.

Il provvedimento quindi accorpa la fase della formazione iniziale con quella dell'accesso alla professione.

Il nuovo sistema di accesso nei ruoli di docente nella scuola secondaria parte da un concorso nazionale riservato a chi possieda un diploma di laurea magistrale o, per le discipline artistiche e musicali, un diploma accademico di secondo livello, coerente con la classe disciplinare di concorso.

In possesso di questi titoli di studio gli interessati affronteranno direttamente le prove – basate sia sulle competenze disciplinari, sia su quelle psico-pedagogiche e trasversali – del concorso nazionale di assunzione con contratto triennale retribuito di formazione e apprendistato professionale. I vincitori saranno assegnati ad una scuola o ad una rete di scuole perché i posti messi a bando corrisponderanno al fabbisogno futuro delle scuole.

Nel triennio di contratto completeranno la loro formazione pedagogica e didattico-disciplinare con un tirocinio professionale retribuito, specializzandosi progressivamente e assumendo responsabilità di gestione delle classi, fino al definitivo ingresso in ruolo.

La parte formativa consisterà in un corso di specializzazione universitaria di durata annuale, mentre i restanti due anni del contratto saranno appunto destinati ai tirocini e alla graduale assunzione della funzione docente, anche in sostituzione di insegnanti temporaneamente assenti.

Al termine del triennio, sulla base del conseguimento del diploma di specializzazione e di una valutazione positiva del lavoro svolto, il contratto sarà trasformato nella immissione in ruolo.

Naturalmente, la formazione in servizio sarà obbligatoria e definita dalle scuole sulla base delle priorità indicate nel Piano di miglioramento dell'istituzione scolastica e nel Piano triennale nazionale di formazione.

Soffermarsi nel procedimento previsto è utile perché spiega il senso profondo del recente provvedimento sulla scuola, perché una tale procedura può risultare efficace non solo per formare un buon insegnante a tutto tondo ma anche per attrarre alla professione docente persone giovani, brillanti, motivate, appassionate, cioè persone di cui la scuola ha urgente bisogno per crescere continuamente e divenire luogo di fermento culturale e di crescita civile per le nuove generazioni.

Il modello non sarebbe però completo senza un forte impegno nella formazione continua e nell'aggiornamento professionale degli insegnanti in servizio, che è infatti presente nel testo in modo collegato con il resto.

L'idea di affidare in parte alle università la preparazione dei nuovi docenti è sicuramente un passo in questa direzione: gli atenei possono così riversare sulla scuola quanto viene prodotto attraverso la ricerca, impedendo la fossilizzazione dei contenuti. Questo processo richiede, però, un adattamento alle esigenze della scuola che deve educare tutti e non soltanto gli specialisti. Abbiamo così uno stimolo anche per valutare correttamente la ricerca, non confrontandola soltanto con il mondo economico, troppo interessato a usufruirne immediatamente.

La scuola quindi si pone come momento di riflessione sulla ricerca per valutarne gli aspetti che contribuiscono veramente alla crescita individuale e collettiva delle persone. Non abbiamo dubbi che ciò avrà anche una ricaduta economica come sta avvenendo in altre parti del mondo.

In questo volumetto ritroviamo il significato profondo dell'intervento legislativo per "La Buona Scuola": lo Stato investe di nuovo nei suoi insegnanti come parte cruciale della ripresa degli investimenti nella scuola, vedendoli come uomini di cultura prima di ogni altra cosa e lo testimonia l'aver deciso di mettere in rete, per ognuno di loro, non solo punteggio di laurea e di abilitazione ma tutto il percorso culturale che ha segnato la loro vita professionale.

Vediamo in azione il ruolo della scuola nella messa a fuoco dell'Informatica (con la lettera maiuscola) nella cultura contemporanea come metodo e, quindi, come elemento di formazione. L'occasione di formare i futuri formatori, grande sfida di ogni generazione, ha costretto gli Autori a pensare, discutere, sperimentare e questo ci sembra un buon segno per un buon inizio.

Flavia Piccoli Nardelli
Presidente Commissione Cultura,
Camera dei Deputati

Prefazione

The people who are crazy enough to think they
can change the world are the ones who do.
(S. Jobs)

Sempre di più l'informatica si presenta, nella vita professionale e in quella personale, come un sapere generale, necessario alla cultura di ognuno così come lo sono i fondamenti della conoscenza della lingua o dell'aritmetica. Non è dunque possibile usare criticamente gli strumenti dell'informatica prescindendo dalla conoscenza della teoria che essa comporta. Non basta il saper fare o il saper usare, ma occorre il sapere in sé, sapere cosa sia l'Informatica e come operi sulle informazioni che comunichiamo e manipoliamo.

L'insegnamento dell'informatica e la formazione degli insegnanti di informatica costituiscono un binomio indivisibile su cui agire per affermare la concezione dell'informatica nella scuola quale disciplina scientifica autonoma, caratterizzata da obiettivi, contenuti, metodi e strumenti di apprendimento propri. A differenza di altri ambiti disciplinari, per l'informatica questo non è scontato, ma va affermato e sostenuto con forza perché le indicazioni ministeriali e i programmi di insegnamento soffrono ancora di una impostazione essenzialmente legata alle abilità informatiche e al saper fare.

La recente istituzione di percorsi universitari abilitanti all'insegnamento dell'informatica nella scuola secondaria di secondo grado ha rappresentato una preziosa occasione per la comunità accademica italiana per contribuire alla definizione delle competenze dei futuri docenti di informatica secondo la concezione sopra menzionata.

Il Gruppo di Lavoro Informatica e Scuola del GRIN (l'Associazione Italiana del Docenti universitari di Informatica – http://www.grin-informatica.it) si è fatto promotore di varie iniziative di coordinamento nazionale nell'ambito dell'insegnamento dell'informatica e della formazione degli insegnanti di informatica. Questo volume, a cura di alcuni componenti del Gruppo di Lavoro GRIN Informatica e Scuola, vuole testimoniare l'attività svolta dalla comunità accademica italiana e raccoglie i contributi relativi alle esperienze di formazione degli insegnanti attuate nelle diverse sedi universitarie che hanno ospitato un Tirocinio Formativo Attivo di Informatica sul territorio nazionale.

Siamo consapevoli che c'è ancora molto da fare, ma le esperienze raccolte in questo ciclo di formazione sono un primo passo nella direzione giusta, quella di riconoscere ore dedicate all'insegnamento dell'informatica come disciplina formativa autonoma da parte di insegnanti laureati in Informatica "ben formati".

Silvana Castano
Università degli Studi di Milano,
Past-President GRIN

Introduzione

La preparazione dei futuri insegnanti di informatica ha una problematica abbastanza diversa da quella degli insegnanti di altre materie. Ciò è dovuto all'ambiguità ancora presente nel concetto di informatica e all'assenza di programmi ben definiti per i vari corsi di studio così come alla carenza di libri di testo. Cominciamo con l'ambiguità del concetto di informatica.

Occorre fare una precisa distinzione tra abilità informatiche e informatica propriamente detta. Per la maggior parte dell'opinione pubblica l'informatica coincide con le abilità informatiche, come se saper leggere e scrivere coincidesse con l'essere poeta. Tuttavia non ci si accorge di questa aberrazione che riscontriamo non soltanto presso coloro che si suppone siano poco informati, ma anche nella scuola e negli organi di comunicazione. Si attribuiscono quindi all'informatica caratteristiche proprie delle abilità informatiche causando un travisamento che danneggia soprattutto i giovani che stanno effettuando la scelta della loro professione futura. Quando andiamo a presentare i nostri corsi di laurea ci rendiamo immediatamente conto di dover affrontare subito questo fraintendimento, anche se non è facile definire cosa sia l'informatica, come del resto non lo è qualunque altra scienza. Per le altre però abbiamo una così antica tradizione, che in genere la definizione non è necessaria, ma per l'informatica il problema si pone, soprattutto per chi non è un "addetto ai lavori". Perfino la normativa sembra non aver chiara la distinzione tra abilità informatiche e informatica.

Per questo motivo abbiamo intitolato il volume parafrasando la famosa frase di Tommaso d'Aquino, che, nell'impossibilità di definire di Dio, ci arriva da diverse prospettive, dandone una descrizione complessa e coscientemente insufficiente.

I testi che qui presentiamo sono il frutto di una riflessione avvenuta in diverse sedi italiane che hanno ospitato un TFA di tipo informatico (classe A042 e A033): Bologna, Bolzano, L'Aquila, Milano, Roma, Torino, Udine, Verona. La riflessione è stata a più riprese discussa in workshop che l'hanno avuta come tema a Torino, Pisa e Roma. Questo volume è in un certo senso il punto di arrivo dell'ultimo incontro tenutosi il 21-22 febbraio 2014 presso il dipartimento di Informatica della Sapienza, anche se nell'ultimo anno tale esperienza è ulteriormente maturata attraverso la sperimentazione sul campo.

Nel ribadire che non è possibile "definire" l'informatica, anche perché essa è in un continuo divenire, al momento possiamo dire ciò che essa non è e sottolineare alcuni aspetti dei quali siamo a questo punto abbastanza sicuri. Vedremo dunque in questo volume interventi di carattere generale, ma anche interventi fattivi di alcuni degli Autori che hanno lavorato per sperimentare determinate tecniche di insegnamento e del modo di affrontare specifici problemi.

Il campo è sicuramente ampio ed aperto a innumerevoli proposte e soluzioni, con l'obiettivo di formare tramite l'insegnamento persone libere, mature e appassionate alla scienza, ad ogni scienza e non succubi di un monopolio di mercato.

Non siamo certamente in grado di affrontare il problema dell'insufficienza di normativa per quanto riguarda i programmi di studio, perché non sta a noi agire in questo senso, e nemmeno della carenza di testi adatti che ne consegue. Infatti già la concezione di "libro di testo" risulta problematica in un'epoca nella quale l'apprendimento, proprio a causa degli strumenti informatici, si è trasformato. Ci vorrà ancora tempo, riflessione e sperimentazione da parte di tutti per arrivare a dare un supporto a coloro che con entusiasmo e sacrificio si preparano all'insegnamento dell'informatica.

1. La didattica dell'informatica libera nel Secondo Rinascimento

Renzo Davoli[1]

La rivoluzione digitale ha portato a cambiamenti mai visti in passato. Si può affermare che si è aperta una nuova era che alcuni chiamano già il "Secondo Rinascimento". E' cambiato completamente il rapporto con la conoscenza e quindi con l'apprendimento. Questo articolo vuole iniziare ad esaminare come questi cambiamenti hanno influenzato la didattica ed in particolare la didattica dell'informatica. In questa trasformazione l'informatica gioca un ruolo fondamentale perché è al tempo stesso una scienza giovane ed è la chiave di volta che ha reso possibile la rivoluzione digitale: è oggetto ed attore del cambiamento.

La conoscenza nel secondo Medioevo (prima del Secondo Rinascimento) era scarsa, doveva essere faticosamente distribuita in libri, dischi, ora, libera finalmente dal supporto hardware, può essere replicata a costo praticamente nullo.

L'ambiente culturale nel quale oggi apprendono i nostri studenti è completamente diverso da quello nel quale hanno studiato le precedenti generazioni. L'alibi di riempire le aule di gadget tecnologici per adattarsi alla modernità è una prospettiva miope e dannosa per il futuro dei nostri giovani concittadini.

La trasformazione è molto più ampia e richiede profondi cambiamenti nel metodo di insegnamento e nuovi strumenti. È l'educazione alla creatività la frontiera dell'insegnamento e il software libero l'unica scelta possibile per il futuro.

[1] Dipartimento di Informatica – Scienza e Ingegneria, Alma Mater Studiorum, Università di Bologna.

1.1. Alcune idee sull'insegnamento

- L'insegnamento è un fenomeno di risonanza. Ogni studente ha proprie capacità e propri talenti innati, ha frequenze proprie di risonanza. Deve essere esposto a quanti più domini del sapere e metodologie sia possibile. Quando incontrerà una vibrazione simile ad una delle proprie frequenze inizierà a risuonare, e sarà l'inizio di una magnifica avventura.
- Fornire nozioni ad uno studente è come spostarlo nello spazio della conoscenza dal punto A al punto B. Il punto B può essere interessante, ma le nozioni da sole non consentono allo studente di spostarsi dal punto B. Fornire un metodo allo studente è come dotarlo di una derivata prima nello spazio della conoscenza, una velocità. Potrà proseguire a partire dal punto B e raggiungere altri punti. Ma non è abbastanza. Solo con una velocità non si può cambiare direzione. Le derivate di ordine superiore nello spazio della conoscenza sono la passione e l'entusiasmo per la materia trattata. Se riuscite a trasmettere le derivate superiori, i vostri studenti saranno veramente liberi di muoversi nella conoscenza.

1.2. Benvenuti nel III millennio

Hardware/Software è solamente la nuova formulazione della dualità spirito/materia.
L'Hardware (materia):

- è un oggetto fisico. Non si crea, non si distrugge, si trasforma; è disponibile in quantità limitata
- è un aggregato di atomi
- se si cede ad altri un elemento hardware se ne perde il possesso.

Il Software (spirito):

- è conoscenza/informazione
- si crea, si propaga, si duplica senza costo
- più è disponibile, più se ne genera di nuovo!
- se si cede conoscenza/informazione ad altri non si perde la propria.

Oggi tutto è spirito (πάντα πνεύμα): (definizione tratta da un seminario di Angelo Raffaele Meo).

Se ieri c'era un legame nella cultura comune fra il software/la conoscenza e il supporto hardware che la conteneva, oggi questo collegamento non esiste più. Ieri si poteva confondere l'idea di libro con l'idea romanzo, un disco con la musica. Oggi la rivoluzione digitale ha liberato il software, un romanzo può essere stampato su carta, ma anche trasportato su una chiavetta USB o semplicemente scaricato dalla rete. Oggi è chiaro che una ricetta di cucina, una teoria matematica, un brano musicale, un programma per elaboratore, una poesia, una barzelletta ... hanno la stessa natura: software.

Questa è la nuova rivoluzione.

L'invenzione della stampa a caratteri mobili è stata fra le cause del (primo) Rinascimento. La stampa ha solamente portato ad una riduzione dei costi e tempi per la riproduzione del software e tanto è bastato per traghettare il mondo fuori dal medioevo. Oggi il software si riproduce a costo di fatto nullo e la rivoluzione che sta causando questo cambiamento è molto superiore a quella della stampa.

L'avvento della digitalizzazione e della rete ha cambiato e sta cambiando radicalmente le strutture sociali, economiche e culturali. È chiaro che questo mutamento ha forti riflessi anche nell'insegnamento di tutte le materie, ma in modo particolare dell'Informatica che è in grado di fornire la chiave di lettura autentica delle trasformazioni in atto.

A volte mi piace pensare come saranno i libri di storia dei miei propropronipoti e di cosa rimarrà traccia per questa epoca che stiamo vivendo. Senza dubbio l'avvento della digitalizzazione e della rete Internet avranno un capitolo dedicato, anche se non so ora cosa verrà detto negli ultimi paragrafi.

Non è un caso se si inizia a parlare di Secondo Rinascimento.

La trasformazione in atto è epocale.

Se ieri la conoscenza era difficile da ottenere, se era limitata la quantità di libri e documenti ai quali ogni essere umano, anche il più fortunato, potesse accedere, oggi al contrario abbiamo libero accesso a una quantità indescrivibile di materiale. Se ieri era difficile ottenere la conoscenza, oggi è difficile proteggersi dalla conoscenza, orientarsi in un mare di informazione così ampio.

In ogni rivoluzione c'è un ancien régime. Ovviamente le lobby politiche ed economiche dominanti prima dell'avvento della rivoluzione digitale non vogliono cedere il passo.

Gli editori rappresentano un caso emblematico. L'economia è la scienza che studia la gestione delle risorse scarse. Ieri gli editori avevano il ruolo nell'economia di gestire l'accesso ai supporti che contenevano la conoscenza. Si preoccupavano di stampare, trasportare e mettere a disposizione per la vendita i supporti (libri, dischi/CD ecc.). Nel terzo millennio la risorsa scarsa sulla quale costruire un sistema economico è la capacità di selezionare, catalogare, rendere fruibile l'informazione cercata. Invece, assurdamente, continuano a fare lobby per rendere scarsa la risorsa che oggi scarsa non è più. Sarebbe come avvelenare l'aria per poter vendere bombolette di ossigeno.

Per i nostri studenti un brano musicale sul loro cellulare è come una barzelletta divertente (sono giustamente entrambi elementi di software). E' naturale condividere la barzelletta per ridere insieme come condividere la musica, proprio non riescono a concepire perché la barzelletta si può "copiare" e la musica (se proprietaria) no. Le leggi non vanno violate, ma quando sono anacronistiche vanno modificate. Il diritto di autore, in particolare il diritto di paternità intellettuale, è sacrosanto. Ogni autore deve poter decidere quale tutela dare alla propria opera e chi viola la volontà dell'autore deve pagare i danni arrecati.

Ma il software è un servizio, non un bene! Quindi copiare abusivamente di software è come non pagare la parcella al commercialista. Non si capisce per quale arcano meccanismo legislativo la copia abusiva sia reato penale e il non pagare il commercialista dia luogo a una causa civile (o meglio si capisce bene quale potente lobby abbia condizionato il legislatore).

La soluzione nella scuola è chiara e oggi più che mai obbligatoria: il software libero, inteso come conoscenza libera di tutti i tipi. Oggi è diventato anacronistico farsi pagare la licenza di copia, farsi cioè pagare per ciò che non costa più nulla.

Mi preme ora mettere a fuoco la situazione degli studenti del III millennio. Molti esperimenti hanno mostrato che la sovraesposizione alla conoscenza ha trasformato i loro processi cognitivi. La BBC recentemente nel programma "The Virtual Revolution 4: Homo Interneticus?" ha presentato uno studio per vedere come la rete abbia modificato il processo di ricerca di informazioni e di apprendimento. Si è visto che gli utenti più giovani navigano nella conoscenza sempre verso nuove direzioni, raramente tornano nello stesso sito o visitano molte pagine mentre gli utenti con qualche anno in più sembrano muoversi seguendo un modello, approfondiscono la ricerca cercando

sempre di confermare o confutare le proprie tesi. Prendendo spunto da un proverbio di Archiloco, che più o meno suona così "Una volpe sa mille trucchi, un istrice ne conosce uno solo ma funziona!", I più giovani sono stati chiamati "volpi" mentre i secondi "istrici". (In [12] la tassonomia di "animali" è più ampia).

Voglio qui semplificare un po' la situazione. Nelle mie esperienze didattiche ho visto alcune difficoltà specifiche delle "Volpi", i ragazzi della google generation.

1. Non avvertono la necessità di formalizzare, modellizzare la conoscenza. Tutto è "one click away". Perché memorizzare, perché creare modelli?
2. Si sentono spettatori del mondo. Sono sperduti nel mare di conoscenza. Tutto è già stato detto e fatto, il "muro di accesso" per poter essere protagonisti appare troppo alto da superare.
3. È tutto molto bello e appariscente. Non si può competere con ciò che è già disponibile. La ricchezza non dà la felicità, e neanche ricchezza di conoscenza dà la felicità nell'apprendere.

Prima di conoscere il lavoro della UCL io usavo diverse terminologie, ma avevo empiricamente avvertito simili differenze. Io parlavo di "Commodore Generation" vs "Nintendo Generation" (quella di oggi potrebbe anche essere chiamata di "I-*" generation)

Io sono della Commodore Generation (o forse ancora di un'era geologica precedente). Avevamo computer semplici, occorreva capire cosa succedeva "dietro le quinte" per poterli usare. Ma non era solo un problema di computer, la domenica mattina in cortile si smontava il carburatore del motorino. La vita era una gara di hacking, per conquistare un chilometro orario o per riuscire a far girare un programma in più.

Oggi non si costruiscono più aquiloni. Sugli scaffali dei supermercati si possono trovare aquiloni colorati e bellissimi. Ma insieme a quelle due canne con la stoffa recuperata chissà dove, volava la nostra fantasia.

Il bene perduto si chiama "ricompensa dell'artigiano", nel nostro caso la "ricompensa dell'artigiano intellettuale". È il carburante della gioia di apprendere.

Quando un artigiano vede una propria opera realizzata, come il ragazzo che vede l'aquilone volare o l'informatico che osserva il suo programma funzionare, viene ricompensato della fatica fatta e prende consapevolezza delle proprie capacità, di essere protagonista e non spettatore.

1.3. La crisi (e la resurrezione) dell'informatica

L'insegnamento scolastico ed universitario dell'informatica ha subito alterne vicende (non solo in Italia).

È un fenomeno complesso e ovviamente si sommano cause internazionali e cause locali, cause culturali ed economiche. Proverò a darne una spiegazione basata su anni di partecipazione al gruppo SIGCSE dell'ACM che studia la didattica dell'informatica, ma che ovviamente non pretende di essere né la spiegazione *autentica*, né *esaustiva* (tanto meno avere valenza scientifica).

L'informatica (la Computer Science e non la Information Technology) era l'icona del futuro negli anni '60 e '70. Nell'iconografia comune i computer erano quelle improbabili macchine piene di lampadine e bottoni che comparivano nei film di fantascienza, macchine capaci di risolvere ogni problema.

Sull'onda di queste immagini le scuole e le università avevano un boom di iscrizioni nei corsi di Informatica. Sono gli anni in cui nascono le icone dei nerd e dei geek brufolosi e solitari che sono sempre attaccati ad una tastiera.

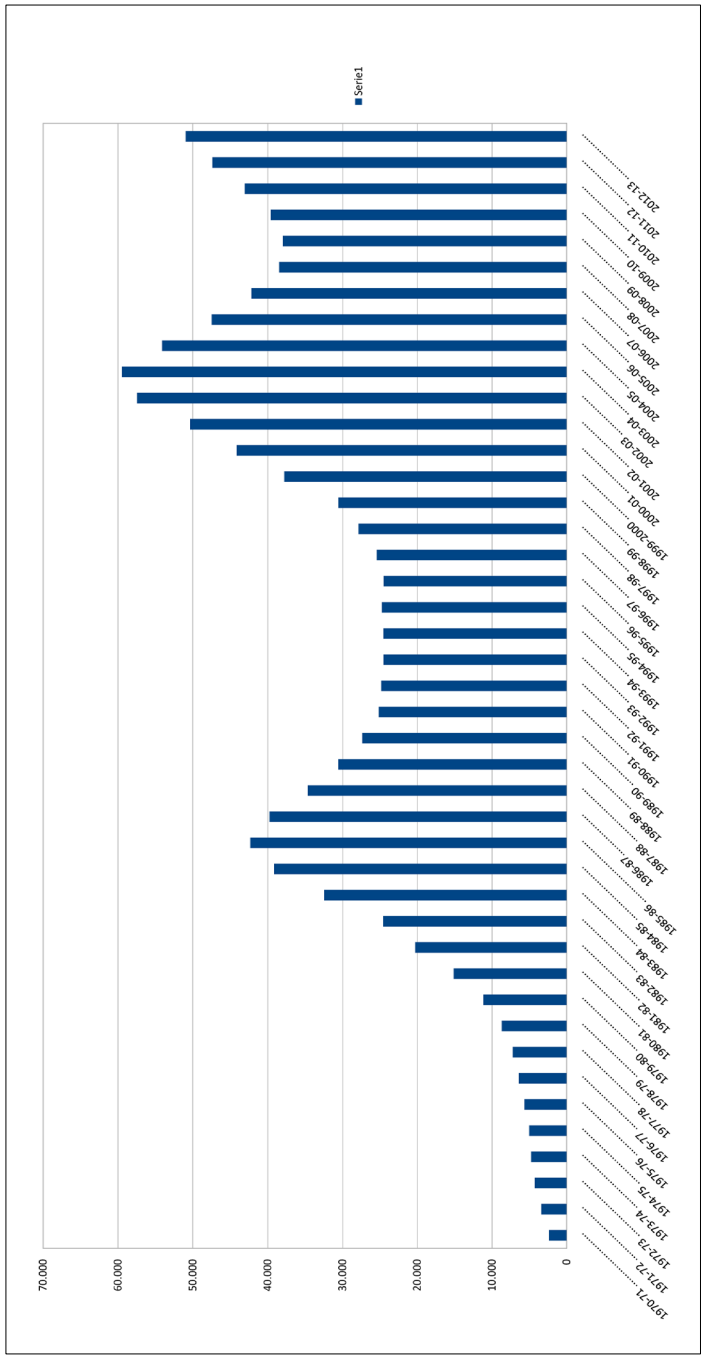
In questa fase c'era una certa armonia fra scuola/università e studenti, tutti pionieri di un mondo da esplorare (e forse con le aziende che non avevano ancora iniziato a *mettere i bastoni fra le ruote* per limitare il futuro e fare soldi!).

In questa fase era chiaro che il computer fosse solo strumentale allo studio dell'informatica.

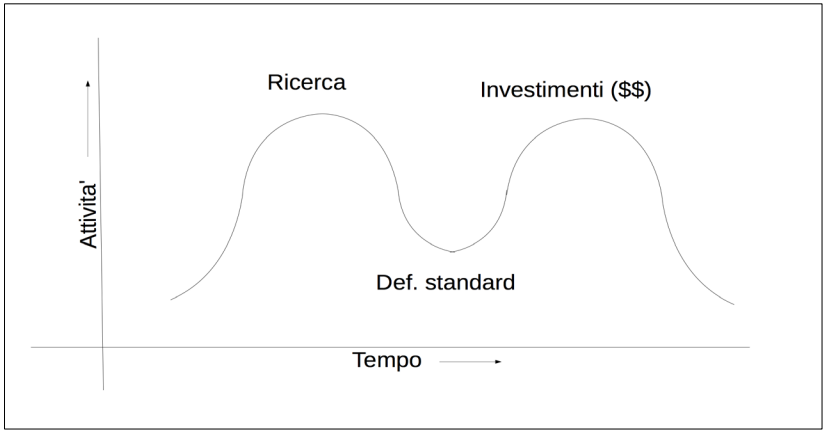
Negli anni '90 c'è stato un notevole decremento delle iscrizioni. L'informatica aveva perso il suo interesse. I grafici che trovate in rete (come quello di B. Webter riportato in Tabella 1.1, a sinistra) mostrano il netto calo. Il riferimento è ai dati americani, ma vi assicuro che lo stesso capitava anche in Europa.

A Bologna dal 1987 c'è un corso di laurea in Informatica (prima Scienze dell'Informazione a 4 anni, poi Informatica a 5 anni, Informatica 3+2, e infine 3 e 2). Nei primi anni era a numero chiuso (120 persone) e avevamo tantissime richieste, poi c'è stato un calo.

Il boom degli anni 80 era anche dovuto all'avvento dei personal computer e all'esplosione della domanda di programmatori.



Tab. 1.1. Andamento delle iscrizioni ai corsi di Informatica. Elaborazione dell'autore su dati del National Center for Education Statistics (NCES, http://nces.ed.gov), tabella 325.35, simili risultati si possono trovare nelle illustrazioni dell'articolo: http://brucefwebster.com/2008/03/05/the-decline-in-computer-science-students/.



Tab. 1.2. Rielaborazione dell'autore del concetto conosciuto col nome di "apocalisse dei due elefanti", descritto da David Clark dell'MIT, "those who write standards live in a somewhat boring valley between two exciting peaks: the first peak the moment of invention and the second peak the billion dollar return on investment" http://groups.csail.mit.edu/ana/People/DDC/Apocalypse.html).

Gli anni 1987-88 sono stati anni di crisi (lunedì nero: 19 ottobre 1987). Scrive Webster nella pagina sopra citata: "by 1990, the running joke around Silicon Valley was: “Do you know what the status symbol of the 90s is? A job.”".

Seguono anni difficili, e poi un nuovo boom legato alla "bolla" del dot com che porta ad un picco attorno all'anno 2000. Poi però la bolla dot com scoppia (come è normale per le bolle), e questo causa un nuovo forte decremento di iscrizioni. Ora l'interesse sta nuovamente crescendo, spero in modo stabile e definitivo.

Ciò che mi rende fiducioso è che sono sfumate le aspettative dei film di fantascienza degli anni ‘70 (che hanno causato il boom degli anni ‘80), sono sfumate le aspettative di mercato della bolla del dot com che hanno causato il boom del 2000. Ora nelle scuole e nelle università si torna al pensiero computazionale, agli algoritmi.

In altre parole, passata l'euforia iniziale per la nuova scienza che ha innescato fenomeni periodici (smorzati) di entusiasmo/disillusione, l'informatica si può ora misurare con le altre scienze più *datate*.

Mi ha sempre stupito la somiglianza del grafico delle immatricolazioni con la apocalisse dei due elefanti (Tabella 1.2). Questo grafico (teorico) mostra l'andamento degli investimenti in una tecnologia.

Il primo picco è relativo allo sviluppo, il secondo alla commercializzazione/diffusione. Nella prima fase si investe per l'innovazione, nella seconda fase si investe perché non ci sia innovazione. Per le aziende occorre sfruttare lo status quo.

Se questo è naturale nella vita aziendale, non possiamo fare in modo che succeda nell'ambito della scuola. In Italia siamo passati dalla splendida anarchia creativa della didattica degli anni '80/'90 con splendide esperienze e progetti, alla omologazione degli anni 2000 (complice anche ECDL). Ovviamente la riorganizzazione centralistica dei programmi di informatica ha consentito alle lobby delle software house di intervenire e le lobby hanno investito perché si realizzasse la seconda gobba del grafico (contro l'innovazione, per il mantenimento del proprio mercato).

Nel mondo ormai (spero presto anche in Italia) ci si è accorti che questa prospettiva è miope. E se ne sono accorte le stesse software house che non riescono più ad assumere personale qualificato [13].

1.4. Cosa/come insegnare informatica oggi

Prima di tutto occorre insegnare informatica. Ricordate sempre che (come sembra aver detto E. W. Dijkstra) "Computer Science is no more about computers than astronomy is about telescopes". In altre parole il computer sta all'informatico come il microscopio al biologo o il telescopio all'astronomo. Lo studio della tecnologia attuale è solo funzionale (solo i concetti sui quali la tecnologia si basa sono interessanti).

Nella didattica possiamo inserire diversi tipi di conoscenza:

- Conoscenza di base, a lungo termine. Esempi: algoritmi, complessità, computabilità...
- Conoscenza a medio termine. Esempi: linguaggi, formati, architetture di processori e bus.
- Conoscenza a breve termine. Esempi: uso di strumenti software.

Occorre concentrarsi sulla conoscenza fondamentale che quindi ha una valenza di più lungo termine. Tra l'altro la didattica sul mero uso di un computer, come quella dell'uso di un microscopio è per gli studenti noiosa, ripetitiva e per nulla entusiasmante.

Gli studenti avvertono tantissimo la *partecipazione* del docente all'insegnamento. Il corso che segue passo passo un libro di testo, uguale negli anni, senza novità non viene apprezzato. Oggi le aule scolastiche

sono sempre pubbliche: con i social network i ragazzi di diversi anni di corso e di diverse scuole si confrontano. Occorre conquistare l'interesse dei ragazzi.

Considerate che nell'insegnamento occorre combattere alcuni preconcetti che i ragazzi della Nintendo generation spesso hanno. Per esempio sono convinti che non sia necessario conoscere il funzionamento delle cose per usarle. In questo caso occorre mostrare loro che se non si conosce il funzionamento di uno strumento si è inermi quando lo strumento fornisce risultati inattesi. In altre parole si è schiavi dello strumento e non si è flessibili nei casi non comuni. Un altro preconcetto tipico è che tutto sia troppo complesso per poter essere compreso. Tutta l'ergonomia ha prodotto una forma di pigrizia culturale (che fa comodo a chi non vuole che il mondo evolva perché ci lucra sopra!). Io penso che un aspetto importante della didattica sia insegnare ai ragazzi a non aver paura della materia studiata. Un laureato della materia X, nel mio modo di vedere le cose, è colui che può dire: *se un essere umano esperto in X ha realizzato Y, io posso capire Y*. Per la scuola superiore magari basta ridurre il senso di spavento, dare fiducia...

Un concetto che un docente deve sempre fornire come effetto collaterale è la perseveranza, la pazienza che occorre nel cercare il risultato voluto. Il senso di velocità e di bombardamento informativo che i nostri giovani hanno fa in modo che non abbiano (o non pensino di avere) il tempo per costruire, fare esperimenti, lasciar sedimentare la conoscenza. La noia è il nostro nemico maggiore nell'aula.

L'insegnante è nell'aula un allenatore culturale. Se al posto di studenti in un'aula di informatica noi fossimo allenatori di atletica ed avessimo di fronte aspiranti atleti, studieremmo esercizi al limite delle loro capacità fisiche per allenarli. Ogni aspirante atleta arriva a casa stanco dopo un allenamento, è normale. Non esiste allenamento che non comporti fatica. E' una fatica mirata ad ottenere soddisfazioni nelle gare. L'allenatore che fa fare pochi esercizi per non stancare la squadra non è un buon allenatore. E' giusto e normale che gli studenti facciano fatica a scuola, che si stanchino perché sono sedute di allenamento culturale. L'importante è che sia chiaro l'obiettivo, quali gare della vita si possono vincere con un buon allenamento.

Per riassumere i concetti mi basta dire che i nostri ragazzi sono stati *viziati* ad avere soluzioni pronte, appariscenti e subito. Li hanno *stupiti*

con effetti speciali, come diceva un antico messaggio pubblicitario, continuava, *ma noi siamo scienza non fantascienza*. Noi dobbiamo fare in modo che si stupiscano di loro stessi.

Nel mondo sono state studiate diverse proposte per motivare allo studio dell'informatica. L'idea è di far leva sugli interessi e sugli ideali dei ragazzi. Ne elenco alcune:

- Computing for social good: lo studio degli algoritmi viene legato a problemi che sono correlati con il bene pubblico, per far comprendere l'utilità sociale dell'informatica. Algoritmi sui grafi possono essere correlati a problemi di logistica per la protezione civile che deve portare soccorso in caso di emergenza o problemi di pattern matching possono essere applicati a sequenze di DNA. Questo approccio mostra agli studenti che l'informatica non risolve solo problemi teorici legati a inesistenti commessi viaggiatori, all'ottimizzazione di spazio negli zaini o a filosofi che stanno sempre a cena ma fornisce soluzioni effettive a problemi reali. Se oggi il riconoscimento delle sequenze di DNA potrà consentire di guarire da malattie gravi lo si deve ai Biologi ma anche agli Informatici che creano algoritmi in grado di gestire enormi moli di dati [1,17].
- Storytelling: è una metodologia per studenti più giovani o con maggiori difficoltà: si tratta di trasformare il racconto di una storia in un programma che genera una animazione grafica. Alice e Scratch sono due progetti che consentono di creare programmi in maniera grafica, incastrando blocchi come se fossero costruzioni virtuali. Il risultato è una azione compiuta da uno o più personaggi [4,5].
- Didattica con robot: non vi è dubbio che i robot siano molto appealing nell'immaginario collettivo, e vedere i propri algoritmi muoversi è fonte di grande soddisfazione. Esistono numerose esperienze con prodotti hardware proprietari o con componenti autocostruiti e schede à la Arduino [3].
- Realizzare videogiochi: anche il mondo dei videogiochi è molto interessante: riuscire a poter giocare e far giocare gli amici ad un videogioco appena scritto è appassionante [8].
- Lo studio dei repository di Software Libero come musei per visitare i capolavori dell'arte della programmazione [2]. Come gli studenti dell'accademia visitano i musei così i nostri studenti possono visitare i repository di software. Saper apprezzare i capolavori fa parte della sensibilità dell'artista come dell'informatico. Entrambi devono essere in grado di riconoscere le opere di valore da quelle da evitare.

- Lo studio tramite Arduino/Raspberry PI per esperimenti con sensori/attuatori [6]. Questa modalità consente di unire alla ricompensa dell'artigiano intellettuale quella del reale artigiano. Lo studio è molto più interessante se si riescono a costruire circuiti che forniscono un feedback anche fisico dell'elaborazione e che possono diventare dispositivi utili. In Inghilterra ogni anno la BBC trasmette durante le vacanze di Natale alcune lezioni scientifiche per ragazzi organizzate da "The Royal Institution". E' una trasmissione molto seguita. Nel 2014 il tema è stato "How to hack your home". Per esempio in uno degli esperimenti un grattacielo di Londra è stato trasformato in uno schermo col quale giocare a tetris [18].
- Lo studio legato alla matematica, scrivere programmi per la soluzione di problemi matematici (ad es. con Sage, project Euler). [9,10]. Nel III millennio lo studio della matematica non può più essere scorrelato dalle metodologie di calcolo simbolico e numerico. Questo trait d'union fra le due scienze può essere utilizzato per creare un maggior interesse per entrambe.
- Per gli studenti della scuola elementare e media l'informatica deve essere insegnata senza alcuna enfasi su tecnicismi e formalismi. In questo campo c'è una ultradecennale esperienza neozelandese con il progetto "computer science unplugged" che ho tradotto in italiano con il collega Giovanni Michele Bianco dell'università di Verona. La didattica dell'informatica avviene tramite giochi in aula e in giardino categoricamente senza l'uso dei computer [14].

I linguaggi di programmazione meriterebbero un discorso a sé stante (e ci sarebbe materia per un ulteriore capitolo). Specialmente per chi è alle prime armi: occorre limitare al minimo la necessità di usare sintassi pesanti e contorte. Se nelle scuole elementari si possono del tutto evitare (csunplugged) e nelle scuole medie si possono usare anche linguaggi grafici (scratch); occorre poi avvicinarsi ai linguaggi di programmazione testuali.

Python è diventato il più popolare linguaggio per i corsi di introduzione alla programmazione delle più importanti università americane[19]. Fra gli altri pregi, ha una sintassi molto leggera, educa a scrivere in modo ordinato (l'indentazione fa parte del linguaggio), non richiede la definizione delle variabili che assumono a tempo di esecuzione il tipo del valore assegnato (quindi tutte le funzioni diventano

template), è portabile, funziona anche sui telefonini, ha tantissime librerie disponibili...

Molti dei testi citati negli esempi di metodi didattici della lista della pagina precedente usano Python come linguaggio di riferimento [3,7,8,10].

1.5. Non c'è possibilità di scelta: soltanto Software Libero nella scuola

La scuola è un servizio pubblico, ha la funzione di trasmettere conoscenza. La scuola "copia" software come suo dovere istituzionale. La scuola pubblica non è un'azienda, non deve minimizzare i costi, deve massimizzare i servizi evitando gli sprechi.

Anche il ruolo dell'Imprenditore è completamente diverso da quello di un Pubblico Funzionario, nel nostro caso un professore o un dirigente scolastico. L'imprenditore deve portare avanti gli interessi della propria azienda. Il pubblico funzionario rappresenta lo Stato e deve portare avanti gli interessi del cittadino e difenderlo dagli interessi delle aziende.

Pubblici funzionari e imprenditori sono parti contrapposte nell'esercizio delle rispettive funzioni.

C'è una differenza fondamentale: gli errori degli imprenditori vengono pagati dall'economia aziendale; gravi errori possono portare al fallimento; gli errori dei funzionari pubblici vengono pagati dalla comunità.

Per il software nelle scuole, non ha alcun senso considerare il costo delle licenze d'uso (che spesso favorisce il software libero) o il "costo" per cambiare il know how (che favorisce l'inerzia a mantenere il software proprietario). Il vero costo da considerare è quello della dipendenza culturale che si induce nei nostri studenti e quindi poi nelle imprese sul territorio. Con il software proprietario si condanna al costo delle licenze tutta la nostra economia. Non è ammissibile.

Nella Pubblica Amministrazione è stato giustamente deciso con una disposizione di Legge (art.68 Codice dell'Amministrazione Digitale) l'obbligo di uso del software libero. Come unica eccezione viene ammesso l'uso di software proprietario solo in caso di comprovata *impossibilità* di uso del software libero. Mi preme sottolineare che la Legge usa la parola impossibilità non la parola convenienza. Quindi il

Codice dell'Amministrazione Digitale vincola le P.A. all'uso del Software Libero anche quando questo costi di più del software proprietario.

A scuola il nostro dio è Galileo.

La scuola deve sempre partire dalla Scienza, è la conoscenza a lungo termine. La didattica della tecnologia da sola, è una conoscenza sterile se non basata sulla Scienza. Per esprimere chiaramente il concetto si può dire che un edificio con aule piene di Lavagne Interattive e dotato di ogni gadget tecnologico non è di per sé una scuola. Occorrono gli insegnanti per fare una scuola, e maggiore è la professionalità degli insegnanti, migliore è la scuola. L'investimento in conoscenza libera, sia essa software libero o formazione degli insegnanti, è una scelta lungimirante. Ogni gadget tecnologico continuerà a costare per manutenzione e in pochi anni diventerà obsoleto e dovrà essere buttato.

La scuola non può chiedere atti di fede, solo atti di scienza, quindi nessuna conoscenza deve essere negata alla curiosità degli studenti e degli studiosi, altrimenti si spegne la passione verso la scienza.

Il software proprietario nega questo diritto. È vietato per lo studente curioso conoscere i segreti di funzionamento di un software proprietario. La scuola non può permettersi questa delusione, è la negazione della propria missione.

La scuola deve essere *super partes*.

La scuola lavora al solo servizio della conoscenza. Una scuola che diventasse mezzo pubblicitario per aziende perderebbe ogni credibilità: si diffonderebbe il dubbio che la didattica e la ricerca vengano focalizzate per difendere solo interessi economici. Con il software proprietario, di fatto, la scuola fa pubblicità alle aziende editrici del Software La situazione diventa gravissima nei settori dove contribuisce a mantenere situazioni dominanti sul mercato.

Il Libero Software forma Liberi Cittadini.

La scuola non deve usare la didattica per generare dipendenze culturali. La Scienza crea libertà, la didattica della tecnologia fine a se stessa crea dipendenze. Non è un'attività didattica: è ammaestramento. Diffondere la dipendenza da software proprietario è come far contrarre un debito ai propri studenti, quindi alla società. È come aumentare il debito pubblico.

Il Software Libero nella didattica produce anche ulteriori effetti positivi:

- Educa alla collaborazione (per una vera e sana globalizzazione)
- Educa alla leale competizione sul merito e sulla conoscenza
- Elimina discriminazioni nell'accesso al software (Diritto allo Studio)
- Educa ad essere attivi nella società (cittadini, protagonisti) e non passivi (consumatori, spettatori)
- Potenzialmente crea minori costi (tutti i costi delle licenze d'uso vengono azzerati). Si spostano comunque da costi improduttivi (spesso importazioni immateriali) a costi di lavoro nel territorio, cioè posti di lavoro per i nostri laureati.

Nell'ambito della ricerca scientifica il conflitto con il software proprietario è palese. Il metodo scientifico prevede la che gli esperimenti possano essere ripetibili per poter verificare la validità delle teorie. Il software proprietario contiene conoscenza nascosta e quindi invalida il metodo scientifico.

1.6. I danni dell'ECDL

È mia opinione che l'ECDL abbia fatto grandi danni nella scuola italiana.

L'ECDL forma "spettatori": ECDL incarna il mito di "usare senza capire". Io non riuscirei a passare gli esami dell'ECDL. L'"ambiente di esame" è utile per ammaestrare le scimmiette, non è stato pensato per esseri umani. Il simulatore chiede di ripetere sequenze di click per fare le operazioni richieste. I miei concetti di cultura, di conoscenza, di didattica non riescono ad avere nessuna intersezione con l'ECDL.

L'ECDL è inutile. L'uso di un word processor è sicuramente più intuivo delle mille app che gli studenti usano quotidianamente sui loro cellulari. Eppure nessuno insegna loro ad usare i loro telefonini con i quali spediscono filmati in rete, si interfacciano a tanti social network diversi, fanno a gara fra chi riesce a configurare una funzionalità originale, utile o divertente. Scrivere una lettera con un word processor o realizzare una presentazione sono letteralmente "giochi da ragazzi" per loro, a patto che ne vedano l'utilità.

L'ECDL non è "europea": provate a cercare in giro per la rete chi "fornisce" l'ECDL in giro per l'Europa o il mondo. Sono in larga parte centri di formazione, scuole private. Io mi aspetto che siano aziende

come la nostra Cepu a dare titoli simili... non le scuole pubbliche o le università. Da anni faccio l'esperimento, quando sono all'estero per qualche convegno, di chiedere ai colleghi se conoscono l'ECDL. I risultati sono molto scarsi. Quei pochi che conoscono l'ECDL (per esempio in Inghilterra è un po' diffusa) la concepiscono come strumento di marketing di scuole o centri di formazione privati.

L'ECDL crea dipendenza: Questi corsi ed esami sono fatti per saper usare specifici applicativi. È una pubblicità per Microsoft fatta a spese dello stato e delle famiglie illuse di contribuire alla preparazione dei giovani. La presenza di ECDL con strumenti liberi è stata solo un alibi per continuare a propinare l'altra con strumenti proprietari: la quasi totalità dei corsi e test center usano Windows e Office, ma esistendo (su alpha centauri) quella con Ubuntu e OpenOffice appare tutto meno "di parte". È un po' come il discorso di poter avere computer senza un sistema operativo preinstallato, in teoria si può e qualche modello in vendita esiste... ma chi sa come fare? In poche parole l'ECDL ha limitato la flessibilità degli utilizzatori di programmi.

L'ECDL contribuisce a diminuire la competitività italiana ed aumenta il debito pubblico: Avendo "assuefatto" tanti ragazzi, pubblici dipendenti, all'uso (senza comprensione) di determinati strumenti ora si è contratto un debito a lungo termine relativo ai costi di licenza. È una dipendenza economica nazionale, quindi limita la competitività e aumenta il debito (visto che i costi di licenza varcano l'oceano, e generalmente non contribuiscono neanche alle entrate fiscali perché le licenze vengono vendute dalle filiali Irlandesi delle multinazionali).

L'ECDL è noiosa per gli studenti: Avete mai parlato con i vostri figli quando hanno subito una lezione di uso di word processor o di foglio elettronico? Io sto male a pensare alla nostra bellissima scienza, fatta di sfide per trovare l'algoritmo più elegante ed efficiente, recepita come l'obbligo di mettere in grassetto il testo o di fare la famigerata "stampa unione". Uno studente normodotato, se non ha problemi cognitivi, impara ad usare un word processor da solo come impara ad usare il proprio telefonino.

L'ECDL genera frustrazione fra gli insegnanti: Gli studenti riescono autonomamente a trovare trucchi e metodi per risolvere problemi. L'insegnante non può competere in gare di furbizia con i nativi digitali. Così spesso c'è un senso di frustrazione e si tende ad inibire la creatività dei ragazzi nel cercare soluzioni originali ai problemi. È proprio

l'antitesi della mia idea di scuola! L'insegnante ha esperienza, capacità di creare modelli risolutivi. Mentre lo studente può trovare da solo il menu per creare un "collegamento" non può per tentativi arrivare al quicksort. Per il quicksort serve l'esperienza del docente.

L'ECDL genera confusione sul ruolo dell'informatica: Non è un Biologo chi sa usare un microscopio, né chi conosce alla perfezione la rotellina della messa a fuoco o come disporre un vetrino per vedere bene l'immagine. Ora in tanti diciamo che l'ECDL non è informatica, ma è difficile sostenerlo, visto che è stato per anni portato avanti da un'associazione dove erano presenti in massa docenti universitari di informatica. Non stupiamoci poi se il ministero chiede di conoscere l'uso dell'F5 come precondizione di cultura digitale per ogni insegnante [15] o se pensa che insegnanti di "Trattamento testi" (ai miei tempi si chiamava dattilografia) possano insegnare informatica.

L'ECDL consuma inutilmente risorse (denaro e tempo) nelle scuole e nelle università: l'orario scolastico è limitato. Il tempo per imparare l'uso di word processor e fogli elettronici è per me rubato alla didattica degli algoritmi di ordinamento o di ricerca. Appare a tutti chiaro che il teorema di Pitagora sia fondamentale, anche se non incontriamo tutti i giorni triangoli rettangoli sul nostro cammino. Allo stesso modo occorre che vengano riconosciuti altrettanto fondamentali gli algoritmi di base. Si possono insegnare a partire dalle scuole elementari, senza computer, come ha dimostrato Tim Bell con il suo bellissimo progetto "computer science unplugged".

L'ECDL ha tolto corsi di informatica a scuola e nelle Università: la confusione generata ha fatto sì che in molte Facoltà (oggi Scuole) e corsi di Studio, insegnamenti di Informatica venissero assimilati all'ECDL come programma. In questo modo il ruolo dell'Informatica è stato denigrato. Vige l'idea che chiunque possa insegnare informatica nelle Università. Fondamenti di Informatica è il corsetto per dare quattro soldi al post-doc in radiologia o geologia. Tanto il word processor lo sanno utilizzare tutti e magari quel post-doc è un mago nell'uso del programma della TAC. Non parliamo dei colleghi fisici o matematici che si sentono dei della programmazione magari perché facevano programmi in Fortran 66 di qualche centinaio di righe.

L'ECDL ha introdotto un discrimine economico tra studenti: Per affrontare gli esami della certificazione, occorre acquistare delle skill card a pagamento, affrontare le prove sempre a pagamento, ci sono

studenti che sono costretti a rinunciare e si sentono discriminati, tutto all'interno della scuola pubblica che per la costituzione italiana dovrebbe contribuire a rimuovere gli ostacoli sociali della formazione dei cittadini, specialmente negli anni dell'obbligo scolastico.

Capisco che l'ECDL sia stata una manovra che ha generato tanti soldi per l'associazione che l'ha inventata. Per colpa dell'ECDL sono spariti nelle scuole i progetti di didattica della vera informatica. Ovviamente le multinazionali del software ringraziano per questo.

Per conto mio, da genitore, considero elemento di demerito per una scuola o un'università indicare l'ECDL nel proprio piano di offerta formativa o nei programmi dei corsi. Così come evito ogni bar che abbia video-poker, tenterò di non iscrivere i miei figli a quelle scuole.

Come informatici io penso dovremmo divulgare questa idea e batterci perché nelle scuole e nelle nostre università esca l'ECDL ed entrino corsi di Informatica vera. Sono interessanti per gli insegnanti e appassionanti per i ragazzi. Non ci può essere passione ed entusiasmo per una "stampa unione", mentre la costruzione di un videogioco, di un robot o di una animazione producono dipendenza dalla voglia di conoscere. È questo è il nostro ruolo: siamo "pusher" dell'entusiasmo per la conoscenza.

Bibliografia

[1] AL SWEIGART, *Invent with Python: learn to program by making Computer Games,* http://inventwithpython.com/.

[2] AA.VV., *Foss Sourcebook Standing on the shoulders of giants,* http://www.foss-sourcebook.org/.

[3] BBC CORP. UK, *Virtual Revolution 4: Homo Interneticus?* http://www.bbc.co.uk/virtualrevolution/makingofprog4.shtml

[4] BELL, T., ET AL., *Computer Science Unplugged* http://www.csunplugged.org.

[5] CMU (RANDY PAUSCH) *Alice* http://www.alice.org.

[6] COMPUTING AT SCHOOL, *The Raspberry Pi Education Manual* http://downloads.raspberrypi.org/Raspberry_Pi_Education_Manual.pdf.

[7] DAVOLI, R., *Vergognatevi,* http://www.bononia.it/~renzo/vergogna.html.

[8] DEPARTMENT OF EDUCATION (UK GOV): *'Harmful' ICT curriculum set to be dropped to make way for rigorous computer science,* 2012.

[9] GEORGE, D., *Christmas Lectures – Sparks will fly: How to hack your home,* BBC and The Royal Institution, 2014. http://www.rigb.org/christmas-lectures/sparks-will-fly.

[10] GOLDWEBER, M., BARR, J., CLEAR, T., DAVOLI, R., MANN, S., PATITSAS, E., PORTNOFF, S., *A Framework for Enhancing the Social Good in Computing Education: a Values Approach.* ACM INROADS, 4 (1), 2013 (Best Working Group Report ItiCSE 2012).

[11] GOLDWEBER, M., DAVOLI, R., LITTLE, J.C., RIEDESEL, C., WALKER, H., CROSS, G., VON KONSKY, B.R., *Enhancing the social issues components in our computing curriculum: computing for the social good,* ACM INROADS, 2011, 2 (1), 64 – 82, March 2011.

[12] GUO, P., *Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities,* Communications of the ACM (CACM) BLOG http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext.

[13] KUMAR, D. (editor), *Learning Computing with Robots,* Institute for Personal Robots in Education www.roboteducation.org.

[14] MILLER, B., RANUM, D., *How to think like a Computer Scientist/Problem Solving using Python* (interactive books), http://interactivepython.org/.

[15] MIT SCRATCH, http://scratch.mit.edu/.

[16] NICHOLAS, D., ROWLANDS, I., CLARK, D., WILLIAMS, P., *Google Generation II: web behaviour experiments with the BBC* Aslib Proceedings, 63 (1), 28 – 45.

[17] PROJECT EULER, http://projecteuler.net/.

[18] SAGE, *Open Source Mathematical Software,* http://www.sagemath.org/.

2. L'informatica come metodo di apprendimento e come linguaggio

Anna Labella[1]

Questo capitolo contiene alcune considerazioni, tutt'altro che esaurienti, sull'informatica in vista dello scopo principale di ogni insegnamento, cioè quello di educare l'allievo rendendolo capace di apprendere e orientarsi consapevolmente nel mondo che lo circonda.

2.1. Apprendimento ed esperienza sensoriale

Tutti siamo convinti che, dal punto di vista tecnologico, l'informatica abbia messo a disposizione dell'apprendimento e della formazione dell'individuo nuovi strumenti e ne abbia, di conseguenza, cambiato gli schemi. Un esempio fra i più significativi: l'uso di internet consente all'apprendimento una forma molto diversa da quella lineare tradizionalmente legata ad un libro di testo. Chiunque di noi abbia scritto finora un libro di testo si è reso conto della forzatura dovuta al mantenimento nel miglior modo possibile di quella sequenzialità, che rende il filo del discorso più corretto e, teoricamente, più semplice, ma più rigido e artificioso e che invece possiamo trascurare, quando facendo una lezione a viva voce, possiamo spostarci facilmente da una parte all'altra della lavagna creando collegamenti visuali e parallelismi con un semplice tratto di gesso. Questo permette di far capire molto più immediatamente certi concetti. Il libro costringe invece a una stretta sequenzialità della forma di apprendimento, legata alla diacronia tipica dell'udito (un libro si può leggere e ascoltare). La vista, anche se viene evidentemente usata, viene forzata ad uno "scorrimento" che non le è di

[1] Dipartimento di Informatica, Sapienza Università di Roma.

per sé naturale. Infatti si usano, almeno in un primo approccio, i libri illustrati, dove la vista può muoversi tranquillamente usando le sue regole proprie, che sono sincroniche. Da notare che negli individui sordi, nei quali la sequenzialità non è naturalmente presente, si hanno grosse difficoltà nell'apprendimento della lettura. L'apprendimento che si ottiene usando internet è molto diverso, perché, pur seguendo percorsi, può passare facilmente dall'uno all'altro cambiando, all'occorrenza, facilmente direzione e avviandosi su cammini a priori imprevisti. Si tratta perciò di una forma intrinsecamente più avanzata di quella di un libro e fondamentalmente interattiva.

A livello più profondo di sviluppo personale, possiamo considerare un altro esempio di influenza di un prodotto dell'informatica sulla formazione dell'individuo: "L'apparecchio (smartphone) è ormai un'appendice di sé. Attraverso il costante contatto seppur virtuale con il gruppo, i ragazzi mantengono una continuità nel loro senso di identità personale." [8].

Fin dal suo inizio l'apprendimento si basa sull'interagire attraverso diverse capacità sensoriali. Esiste una corrente di pensiero (cfr. [5]), con la quale tenderei a convergere su questo argomento, per la quale il significato ed i concetti si generano dall'esperienza globale dell'uomo, che potrebbe definirsi corporea, se con corpo intendiamo tutte le possibilità di interazione che abbiamo a disposizione e che cominciamo a sviluppare fin dalla nascita; come esempio di questi concetti primitivi che si vengono a formare, abbiamo: "dentro-fuori", "prima-dopo", ecc.

Già in questa prospettiva lo strumento informatico si presenta come quello capace di superare facilmente certe costrizioni sensoriali e mettersi al servizio dell'apprendimento in generale [1].

Su questo argomento esiste una letteratura sterminata ed innumerevoli sforzi di superare eventuali carenze sensoriali attraverso dispositivi creati ad hoc spostando le esperienze sensoriali di un certo tipo a quelle di un altro tipo [2].

2.2. Linguaggio astratto e analogia

Al di là di tutto questo, noi vogliamo qui sottolineare il fatto che, anche come disciplina teorica, l'informatica rappresenta un metodo

che può cambiare il modo di pensare e trattare i problemi a partire dall'esperienza comune in generale e scolastica in particolare. È nostra opinione che essa possa e debba fornire un metodo e un linguaggio comune per risolvere problemi.

È chiaro che, quanto più il linguaggio è astratto e "semplice", tanto più viaggia spedito nel conseguimento del suo scopo, ma rischia di lasciare a terra tutti coloro che non sono saliti sul suo treno, in altre parole, coloro che non riescono ad usarlo direttamente come modo di pensare.

Nelle considerazioni che sottoponiamo verrà elaborato principalmente il parallelo con la matematica che fino ad ora (almeno nei casi migliori) ha svolto questo ruolo. Non a caso, a partire dai Pitagorici, ogni cosiddetta scienza è tale perché è, almeno in parte, traducibile in termini matematici. Tuttavia è universalmente riconosciuto che esiste una difficoltà nell'apprendimento della matematica, sostanzialmente del suo linguaggio, e saremmo propensi a ritenere che ciò sia legato alla difficoltà di molti individui nel collegare tale linguaggio e il suo modo di procedere all'esperienza sensoriale. Di fatto anche la matematica nasce da un'esperienza corporeo-sensoriale, ma se ne allontana abbastanza velocemente. Abbiamo avuto epoche e culture nelle quali la matematica era sperimentale e strettamente legata all'esperienza fisica. Principalmente a partire dalla cultura greca e dal suo modo di domandarsi il perché di certi fatti, possiamo dire che il ruolo della matematica (oltre le applicazioni) è quello di insegnare a ragionare correttamente partendo da certe premesse, indipendentemente dalla loro connessione con la realtà sensibile. Si tratta quindi di una cosa molto astratta, che non si può dire facilmente a quale meta-meta…livello di linguaggio sia. Nei secoli si è fatto uso di paradigmi sensoriali diversi per il suo apprendimento, salvo rigettarli poi regolarmente quando volevano imporre le caratteristiche dovute alla loro concretezza:

- paradigma visivo-sincronico tipico della geometria messo in crisi ad esempio dal problema dell'incommensurabilità rispetto all'intuizione del punto fisico;
- paradigma sequenziale tipico dell'aritmetica e altri elementi diacronici come le successioni, messo in crisi dai paradossi dell'infinito; e, a un livello superiore, il continuo, messo in crisi dalle difficoltà di un comportamento diverso da quello dell'aritmetica, ecc.

Si è parlato addirittura di "matematiche" come se fossero campi diversi tra loro. In realtà è lo sviluppo di un unico linguaggio, che, sfruttando analogie provenienti da esperienze corporee diverse, tende a costituirsi come strumento di analisi e di ricerca.

Didatticamente, l'uso di certi particolari riferimenti concreti può far scattare l'astrazione meglio di certi altri. La teoria degli insiemi è sembrata in un certo periodo (seconda metà del ventesimo secolo) come molto adatta da questo punto di vista, perché parte appunto da un'esperienza primordiale: l'appartenenza. Essa usa il concetto basilare dentro-fuori (diagrammi di Eulero-Venn), legato all'esperienza corporea di chiunque, ma poi fa anche appello, ad esempio, a quello molto sofisticato che diremmo di "atomicità parametrica", cioè al fatto che qualcosa può essere considerato sia come un individuo, sia come una collezione di individui (si pensi ai numeri di von Neumann). Questo fa difficoltà a parecchi allievi ed è un esempio di concetto molto astratto.

Sembra quasi che ogni volta che si riesce a trovare un supporto di esperienza corporea che tende a semplificare la costruzione di certi concetti, occorra poi un salto per non banalizzare il tutto e diventare descrittivi e/o incoerenti.

Questa esperienza del "salto" è abbastanza evidente a noi docenti quando, dopo aver tentato con un allievo diverse metafore per arrivare a un concetto, lo vediamo illuminarsi: è finalmente riuscito a superare l'esperienza concreta per raggiungerne un'altra di livello superiore, ossia riconoscere il quid comune alle diverse esperienze che gli sono state proposte[2].

L'uso di analogie e metafore è essenziale nel costituirsi di linguaggi sempre più astratti e potenti, ma può rivelarsi anche fuorviante, se l'allievo non riesce a staccarsi dalla concretezza degli esempi.

La difficoltà maggiore che rileviamo come docenti universitari è la carenza di pensiero astratto nei nostri studenti che non permette loro di procedere speditamente nell'apprendimento, ma li costringe ad essere troppo legati agli esempi.

Pur non essendo esperti in materia, ci siamo più volte chiesti come si generi il "pensiero astratto", cioè la capacità di manipolazione del linguaggio, più o meno a prescindere dal suo significato[3].

---

[2] D'altra parte per Aristotele la scienza è scienza dell'universale.

[3] Viene attribuita a Socrate la ricerca dei concetti astratti.

Ormai nella nostra civiltà siamo abituati a pensieri molto specializzati. Seguendo un abbozzo di classificazione preso da Jeannette M. Wing [6], ben nota esponente della corrente americana di "Computational thinking", si può parlare di pensiero in molti sensi:

- astratto,
- analitico
- concettuale
- concreto
- costruttivo
- creativo
- critico
- deduttivo
- strategico
- sintetico
- tattico
- computazionale e procedurale.

Non c'è dubbio che in qualche modo tutte queste siano forme di pensiero astratto, anche se orientate e modellate in modo diverso.

Come avviene il passaggio dall'esperienza sensoriale a pensieri più elaborati e più specializzati come quelli in elenco? Un grosso lavoro viene svolto dallo strumento dell'analogia[4] che permette di trasferire esperienze da un campo all'altro (se vogliamo anche da una modalità sensoriale all'altra) e poi dalla metafora[5] che cerca di introdurre un significato usandone un altro molto lontano, ma correlato. Sono esempi di analogia: "seguire un ragionamento", "estrarre una radice".

Il ripetersi a livelli sempre più alti di questo meccanismo di trasferimento, porta ad una sempre maggiore astrazione. Si potrebbe pensare che da un certo punto in poi si lavori con la sola astrazione, cioè passando da casi simili ad un unico caso, cioè usando una forma di analogia astratta.

Un esempio di questo sono i parallelismi tra teorie (logica matematica e geometria algebrica fra i tanti).

---

[4] In generale, l'analogia è quel procedimento logico con cui si cerca di estendere l'applicabilità di talune proprietà o regole da un caso noto e definito ad altri casi che presentino aspetti di ragionevole somiglianza (cfr. Wikipedia).

[5] La metafora è una figura retorica che implica un trasferimento di significato. Si ha quando, al termine che normalmente occuperebbe il posto nella frase, se ne sostituisce un altro la cui "essenza" o funzione va a sovrapporsi a quella del termine originario creando, così, immagini di forte carica espressiva. La metafora non è totalmente arbitraria: in genere si basa sulla esistenza di un rapporto di somiglianza tra il termine di partenza e il termine metaforico, ma il potere evocativo e comunicativo della metafora è tanto maggiore quanto più i termini di cui è composta sono lontani nel campo semantico (cfr. Wikipedia).

Naturalmente ciò può essere usato sia in forma positiva che negativa. Un esempio di metafora: la parola "teorema", nonostante la sua etimologia che la riconduce alla "visione" di qualcosa di evidente, è usata dai matematici per indicare una proposizione correttamente dedotta da principi accettati almeno in un certo ambito. Essa ha acquisito nel linguaggio giuridico il significato di spiegazione astratta e ingiustificata di un fatto. L'aspetto di ragionamento astratto è stato mantenuto, passando da un caso all'altro, ma la connotazione positiva del rigore matematico è stata sostituita dalla connotazione negativa in campo giuridico di qualcosa completamente slegato dalla realtà.

Analogie e metafore sono alla base di ogni linguaggio: si pensi ad esempio all'arte figurativa che, anche se con fini diversi, si formalizza, per concretizzarsi, con un linguaggio e con una materia. Il suo bisogno di espressione la porta a incarnare i suoi concetti attraverso un'operazione metaforica che cambia secondo la cultura e l'epoca, ma le consente di raggiungere concetti ed esperienze universali e, in questo senso, astratti. Infatti la metafora nel linguaggio iconico rappresenta idee astratte attraverso figure simboliche; queste possono cambiare a seconda della cultura e, quindi, risultare più o meno efficaci. Perfino negli animali possiamo ritrovare una forma di astrazione funzionale: il gatto corre dietro alla palla come se fosse un topo, il bambino si succhia il dito come se fosse il seno materno (cfr. [3] pag. 14).

Dei tipi di pensiero enunciati sopra, a noi interessa in maniera particolare l'esperienza del Computational thinking. Il fatto che questo pensiero accetti di farsi materializzare proprio nell'uso di alcuni strumenti, potrebbe farci sperare di superare il problema di mancanza di corporeità della quale sembra spesso affetta la matematica con la conseguenza di essere capita soltanto da coloro che "sono portati".

Durante il percorso scolastico dovrebbe maturare questa specificità, fino ad arrivare al suo completamento al momento dell'accesso all'università, quando ci si dedica ormai, soprattutto nel settore scientifico, quasi esclusivamente a una conoscenza molto specializzata e, quindi, a linguaggi molto specifici, anche se mai del tutto completamente astratti.

Nelle diverse epoche alcuni di questi linguaggi hanno avuto, almeno parzialmente, la prevalenza, nel senso che sono risultati "metodologici" rispetto agli altri, costruendosi prima da metafore prese da comportamenti molto concreti, ma imponendo poi i propri schemi come metafore.

Come dicevamo sopra, il cammino della matematica può servire da paradigma, e anche da spunto, in questa direzione. Abbiamo parlato della metafora spaziale (geometria) e della metafora della successione (l'aritmetica), ma potremmo aggiungere la metafora della decisione (calcolo delle probabilità), la metafora economica (i numeri relativi), ma anche indovinelli e proverbi per introdurre la logica; in particolare il gioco per introdurre il concetto di regola. Infatti non sempre la matematica è stata un sistema ipotetico-deduttivo; come per le altre scienze si è avuta una fase empirica, come abbiamo detto sopra, poi diverse fasi sistematiche ed addirittura una fase che diremmo oggi di problem-solving.

Il "teorema di Pitagora", tipico risultato di un sistema ipotetico-deduttivo, è stato preceduto dalla conoscenza delle cosiddette "terne pitagoriche".

Nel medioevo abbiamo problemi matematici proposti come indovinelli, come il problema dei 17 cammelli, ma anche problemi tipicamente algoritmici: si pensi al modo di dire "salvar capra e cavoli". Nel tardo medioevo con l'introduzione di "tabulae" e, soprattutto della notazione posizionale dei numeri, che presuppone già un processo di astrazione, si introdussero una serie di algoritmi, ad esempio quello della divisione, operazione fino ad allora molto complessa. La matematica di quell'epoca sfocia dunque in un atteggiamento computazionale-procedurale, che secondo la classificazione sopra riportata, è tipico dell'informatica: infatti nel caso dell'informatica possiamo parlare appunto di pensiero computazionale-procedurale.

La matematica è ritornata ad essere poi, come abbiamo detto sopra, il pensiero "deduttivo" che sottende alle altre scienze. Sembra allora naturale supporre che questo pensiero computazionale-procedurale tenda a imporsi come pensiero metodologico all'inizio del XXI secolo pur nello stretto fecondo rapporto con gli altri, che certamente non possiamo ormai dimenticare.

2.3. Il pensiero computazionale

Saper "leggere" e "far di conto" sono due attività ormai imprescindibili nella nostra cultura (non è sempre stato così); forse dovremo aggiungere per gli uomini di questo secolo "seguire e formulare procedure".

Facciamo attenzione, però: non tutti coloro che sanno leggere e scrivere sono poeti, né quelli che sanno far di conto sono matematici, così non tutti coloro che riescono a pensare in modo procedurale sono informatici. Distinguerei perciò in maniera abbastanza accentuata l'insegnamento di queste metodologie nella scuola, cioè per tutti, dall'insegnamento di queste metodologie nell'università, cioè per coloro che devono fare gli informatici di professione, anche se i due insegnamenti devono essere in qualche modo collegati. Resta evidentemente da studiare questa connessione.

L'elaborazione dell'informazione può avvenire in tanti modi e sempre è stata praticata, ma adesso abbiamo tra le mani un linguaggio specifico, legato addirittura ad uno strumento (che, però, potrebbe cambiare); possiamo formulare i nostri problemi in un linguaggio rigoroso che ne consente la comunicazione, la soluzione e permette di confrontare le soluzioni, valutandone l'effettività, la complessità, ecc. Concetti assunti dall'informatica e resi quantitativi, fase che, in altri tempi, è avvenuta per la fisica e per la chimica.

Un aspetto nel quale l'informatica si pone nella scia della matematica è la relativizzazione dei concetti nel senso di un'indipendenza dei medesimi da un oggetto di riferimento. Già Aristotele si rendeva conto che la geometria non era una scienza di qualcosa che aveva una sua realtà, ma soltanto una scienza che da presupposti (per quanto evidenti) trae conseguenze, perciò è importante enucleare i presupposti (postulati) e stabilire le regole di inferenza. Così anche per l'aritmetica, anche se questo processo sarà più lungo, dopo un accenno da parte di Campano da Novara, bisognerà arrivare agli assiomi di Peano per avere una vera e propria assiomatizzazione, quando ormai esistono diverse aritmetiche, come anche diverse geometrie. Col passare dei secoli ci si accorge che c'è bisogno di una caratterizzazione, di una teorizzazione, perché non abbiamo una sola forma. Lo stesso accade per l'informatica ad esempio nei confronti degli algoritmi. Non so quanti dei nostri ragazzi siano coscienti del fatto che l'algoritmo che calcola la divisione tra due numeri non è unico. Per risolvere lo stesso problema possiamo usare algoritmi diversi: si sa, ma non se ne fa oggetto di riflessione finché l'informatica non se ne impossessa e sviluppa classificazioni e confronti di algoritmi.

L'apertura alla relativizzazione non è soltanto della scienza, ma è una fase cruciale della crescita: il bambino piccolo cerca di strutturare tutta la sua conoscenza e affettività in un paradigma unico; per questo talvolta ci fa sorridere. Con il crescere si accorge che questo paradigma unico non regge e gli orizzonti di riferimento devono essere molteplici. Quindi la relativizzazione accompagna e favorisce la crescita (v.[7]) .

2.4. Formalizzazione e astrazione

Un aspetto specifico dell'informatica è il risalto dato al problema della formalizzazione. Naturalmente questo era anche un problema della matematica, ma adesso ci si rende pienamente conto che si può fare con linguaggi diversi con difficoltà e risultati diversi. Il gioco con e tra i linguaggi diventa perciò cruciale, così come anche la relativizzazione dello strumento: l'algoritmo deve adattarsi alla macchina, ma la macchina può cambiare (e sta cambiando velocemente); c'è quindi un'interazione con lo strumento di calcolo a un livello mai prima conosciuto. Anche il "modo di ragionare" si deve adattare alle diverse situazioni e quindi ben vengano le diverse logiche che nel corso della storia, specialmente dell'ultimo secolo si sono venute configurando, ma anche, ahimé, i diversi livelli di astrazione.

Perciò, citando di nuovo la Wing [6], il processo di astrazione include:

- Scegliere le giuste astrazioni
- Operare simultaneamente a diversi livelli di astrazione
- Definire la relazione tra questi livelli

In questo l'informatica si pone in maniera speciale, perché: l'automazione è meccanizzare le astrazioni, i livelli di astrazione e le loro relazioni. La meccanizzazione è possibile grazie alla notazione precisa e ai modelli. C'è sotto qualche "computer" (umano o macchina, virtuale o fisico). Di nuovo, la logica, pensata come formalizzazione del modo di ragionare corretto, non è più qualcosa di unico, anzi, se ce ne fosse ancora bisogno, se ne viene a chiarire uno dei ruoli: riflettere sul meccanismo interno di certi linguaggi .

Il linguaggio diventa qualcosa di intermedio tra il naturale e l'artificiale. La relativizzazione dei linguaggi e delle logiche apre una strada teorica anche all'interdisciplinarietà. L'informatica di nuovo può presentarsi in due modalità: come strumento e come metodo. Anche qui

sappiamo tutti quali strumenti abbia fornito ai diversi campi della conoscenza l'informatica, ma ci interessa soprattutto notare come l'informatica ci ponga con occhi diversi di fronte ad altre discipline.

A mo' di esempio si consideri la musica: qui il vecchio detto che la musica è strettamente legata alla matematica può a maggior diritto essere applicato all'informatica, perché l'aspetto algoritmico è forse più insito e importante nella musica di quanto non lo siano i rapporti numerici. Ad esempio i canoni di vario tipo, così come l'accompagnamento di un brano o addirittura la sua generazione secondo uno stile prefissato, hanno una natura algoritmica [4].

D'altra parte, presentandosi come un metodo che analizza problemi complessi e li decompone in problemi più semplici, l'informatica può svolgere un ruolo fondamentale nella costituzione di una multidisciplinarietà, che consiste nella precisa individuazione delle competenze specialistiche, tutte importanti, necessarie per affrontare un problema e nel loro coordinamento. Si tratta di un primo passo verso la vera e propria interdisciplinarità che, sul piano dell'apprendimento, si pone come esigenza di ricomporre in senso comprensivo e intersettoriale i contenuti di apprendimento e di esperienza dell'alunno [9].

Bibliografia

[1] BOTTONI, P., CAPUANO, D., DE MARSICO, M., LABELLA, A., LEVIALDI, S., *Metaphor-based teaching for sensorially-critical subjects,* Proc. of ICSEC 2010, The 1st International Computer Science and Engineering Conference, Chiang Mai, 17-19 nov. 2010, Thailand, pp.17-22.

[2] CAPUANO, D., *Studio, analisi e applicazione didattica delle tecnologie E-Learning in contesti di apprendimento sensorialmente critici,* tesi magistrale, Sapienza Università di Roma, 2013.

[3] GOMBRICH, E. H., *A cavallo di un manico di scopa,* Torino, Leonardo Arte, 2001.

[4] LABELLA, A., SCOZZAFAVA, C., *Music and algorithms: a historical perspective,* Studi musicali 32 (2003), 3-50.

[5] LAKOFF, G., JOHNSON, M., *Metaphor We Live By,* University of Chicago Press. 1980.

[6] WING, J.M., *Computational Thinking,* Communications of the ACM 49 (3) March 2006.

[7] http://patriziamattioli.com/tag/relativizzazione/.
[8] http://patriziamattioli.com/2014/12/22/genitori-e-figli-uno-smartphone-come-costruzione-e-mantenimento-dellidentita/.
[9] http://www.rivistadidattica.com/fondamenti/fondamenti2.htm.

Ringraziamenti

Voglio qui ringraziare in particolare due amici con i quali ho cercato di chiarirmi i concetti esposti: Silvio Maracchia, storico della matematica, e Laura Gigli, storico dell'arte. Se sono restati fumosi è soltanto colpa mia.

3. Oltre l'arcano del "giuoco delle perle di vetro"

Claudio Mirolo[1]

3.1. Prologo

> Sotto l'alterna egemonia di questa o di quell'altra scienza o arte, il Giuoco dei giuochi era diventato una specie di linguaggio universale col quale i giuocatori erano in grado di esprimere valori mediante simboli e di metterli in vicendevole rapporto.
>
> *(Hermann Hesse, 1943, "Il giuoco delle perle di vetro")*

3.1.1. Il giuoco delle perle di vetro

Un tema molto dibattuto nell'ambito della didattica dell'informatica, a partire dagli anni '90 quando l'esercizio della programmazione ha cominciato ad essere via via sostituito dall'introduzione di strumenti applicativi preconfezionati, riguarda il ruolo educativo da attribuire a questa disciplina nel corso dell'istruzione scolastica. Ci si riferisce, in particolare, agli aspetti che possono avere valore formativo per *ogni* allievo, non solo per coloro che la scelgono come indirizzo specifico di studio. L'attualità del tema è recentemente riemersa a livello internazionale sull'onda della discussione attorno al *computational thinking* [23], il pensare in termini di elaborazione di informazioni, che tra l'altro ha determinato importanti interventi nei curricula di alcuni paesi europei (per una rassegna più dettagliata si rimanda a [12]).

In queste pagine vorrei quindi condividere con il lettore alcune riflessioni, e stimolarne di sue personali, che mi auguro possano contribuire a vedere l'informatica nella scuola in una prospettiva più

[1] Dipartimento di Matematica e Informatica dell'Università di Udine.

articolata, anche e specialmente da parte di insegnanti che hanno una diversa formazione culturale, scientifica o umanistica, ma che comunque si devono confrontare con essa in ragione della pervasività degli strumenti che ne applicano i principi.

Nell'organizzare il testo mi sono preso la libertà di accompagnare le mie riflessioni con alcuni brevi passi tratti dal romanzo di Hermann Hesse "Il giuoco delle perle di vetro" (*Das Glasperlenspiel* [9]), collegamenti senz'altro arbitrari dal punto di vista dell'opera letteraria, ma ai quali l'indeterminatezza del "giuoco" che fa da sfondo alla narrazione consegna un potere evocativo che lascia spazio all'immaginazione. Nel giuoco delle perle di vetro Hesse prefigura, infatti, una sorta di linguaggio universale per elaborare conoscenze scientifiche e artistiche, qualcosa di prossimo a una sintesi dei linguaggi della matematica, della logica (la *characteristica universalis* di Leibniz?) e della musica: attraverso questo strumento i monaci di Castalia interpretano la realtà, creano e contemplano—anche esteticamente—le *strutture* che il gioco consente loro di costruire. Nella mia rilettura, oggi, il "giuoco delle perle di vetro" rimanda al ruolo dell'informatica, sempre più inestricabilmente intrecciata allo sviluppo di nuove conoscenze in tutti gli ambiti. Inoltre, il titolo di un capitolo centrale del romanzo, "Magister ludi", ci fornisce un ulteriore spunto, che affido direttamente all'interpretazione del lettore: magister ludi, colui che dirige il gioco, è anche *ludi magister*, insegnante di scuola...

Ma in cosa consisterebbe questo "giuoco"[2]? Qual'è, cioè, l'essenza dell'informatica oggi? Il nodo della questione, dal nostro punto di vista, sta precisamente in questo: nel decidere quali aspetti della disciplina sono significativi nella formazione di ogni cittadino, almeno se riteniamo che ve ne siano di irrinunciabili, al di là delle ricadute tecnologiche e degli strumenti sempre più sofisticati che ci vengono profusamente messi a disposizione. Su questo problema ho avuto l'opportunità di confrontarmi in svariate occasioni, non solo con gli insegnanti di informatica in formazione, interessati a chiarire la propria funzione anche in contesti in cui la loro materia è piuttosto marginale, ma anche al fine di realizzare progetti interdisciplinari in collaborazione con scuole di ogni grado, a partire dalle elementari, in

[2] Qui e nelle pagine che seguono mutuo dalla traduzione di Ervino Pocar la forma letteraria 'giuoco' in sostituzione di quella più comune 'gioco'.

cui non sempre sono presenti insegnanti con adeguate competenze in merito alla didattica dell'informatica.

3.1.2. L'essenza del giuoco: indizi

È bene chiarire fin da ora che non si può parlare di un'essenza dell'informatica condivisa, in quanto convivono visioni molto diverse della disciplina anche all'interno della comunità che se ne occupa per vocazione. Di conseguenza, gli *indizi* raccolti in questa sezione per identificarne alcuni aspetti centrali sono filtrati dal mio punto di vista personale al riguardo, punto di vista condizionato da un interesse prevalente ad esplorare le implicazioni culturali di quanto si intende calare nella scuola. Mi appresto ad esprimere tali indizi, tuttavia, attraverso voci ben più autorevoli della mia e che sono riuscite a caratterizzarli in poche righe in modo particolarmente efficace.

Un primo indizio importante si riferisce alla necessità di dominare le corrispondenze fra *forma* e *significato*. Nelle incisive parole di Duchâteau [5]:

> L'informatica è una ricerca incessante per stanare il significato dietro la forma e per costringere il significato nella camicia di forza della forma: nessuno dovrebbe uscire dalla scuola senza averlo percepito.

In questo senso, i tipici strumenti tecnologici oggi in uso, anche nell'insegnamento, tendono a sottrarci questo compito che è invece cruciale per capirne la portata e i limiti.

Un secondo indizio ha a che fare con la comprensione della *natura dell'elaborazione formale*, di cui alcuni aspetti sono ben sintetizzati nei "miracoli" richiamati da Mazoyer [14], in particolare i primi due:

> Il primo miracolo è che combinare un gran numero di volte un piccolo insieme di operazioni elementari consente una potenza d'azione considerevole. […] Il secondo miracolo è che una grande varietà di insiemi di operazioni elementari (ragionevoli) conduce alla stessa potenza [di calcolo]. Il terzo miracolo è che i limiti di questa potenza sono esprimibili formalmente.

Ciò che caratterizza la natura dei processi di elaborazione non è la complessità di per sé stessa, ma piuttosto il fatto che la complessità di

quanto può essere perseguito è il risultato della combinazione di operazioni estremamente semplici, se non banali, e non è nemmeno particolarmente importante quali operazioni di base scegliamo come mattoni delle nostre costruzioni. Quello che conta, in ultima analisi, risiede dunque nelle nostre capacità organizzative e interpretative, non è intrinseco di qualche specifico dispositivo o sistema formale, e il concetto di *calcolabile* nasconde qualcosa di più profondo dei mezzi attraverso i quali ce lo rappresentiamo.

Un terzo indizio rimanda al ruolo centrale, sul piano introspettivo (in qualche modo) e organizzativo, che i linguaggi dell'informatica sono venuti ad assumere. Si tratta della particolare prospettiva che conduce Abelson e Sussman a parlare di *un'epistemologia procedurale* [1, 20], una novità sostanziale per il salto di qualità che si produce in termini di profondità di analisi di questa forma di conoscenza:

> [... Il] significato profondo [dell'informatica] ha poco a che vedere con i computer. La rivoluzione informatica è una rivoluzione nel modo di pensare e di esprimere quello che si pensa. L'essenza di questo cambiamento è l'emergere [di un']epistemologia procedurale—lo studio della struttura della conoscenza da un punto di vista procedurale [...]. La computazione fornisce una cornice per trattare precisamente con la nozione di "come?".

Sottolineo la radice introspettiva di questo tipo di conoscenza (come procedo esattamente, senza possibilità di essere elusivo, per risolvere il problema in esame?), perché mette in luce uno degli aspetti più significativi dal punto di vista pedagogico[3].

Un ulteriore indizio, e qui mi fermerò anche se ovviamente avrei potuto optare per una collezione più estesa, si riferisce alle *proprietà degli algoritmi* ed è forse per certi versi il più scontato. Ne affido la formulazione alle parole di Knuth [11]:

> La definizione di informatica che prediligo è che si tratta dello studio degli algoritmi [...]. Forse la scoperta più significativa determinata

[3] È chiaro che il terzo *indizio* non può essere dissociato dai due precedenti, che ne circoscrivono la portata in termini di attribuzione di significato ai codici utilizzati e di pertinenza delle forme di proceduralità esprimibili.

> dall'avvento del computer sarà che gli algoritmi, come oggetti di studio, sono straordinariamente ricchi di proprietà interessanti; inoltre, un punto di vista algoritmico costituisce un modo utile di organizzare la conoscenza in generale.

In altri termini, il concetto di algoritmo non solo ci consente di pensare la proceduralità a un livello più astratto, ma gli algoritmi sono di per sé stessi interessanti oggetti di studio per le proprietà intrinseche che presentano, in particolare in relazione all'impiego di risorse di calcolo (tempi di elaborazione, spazio di memoria).

3.2. Babele (semantica)

L'introduzione nella scuola delle tecnologie dell'informazione e della comunicazione (ICT) ha portato a una proliferazione di attività accomunate sotto il cappello generico di "informatica". In realtà si è trattato e si tratta di attività di natura piuttosto differenziata, con il risultato che quando si parla di *informatica* non ci si intende più su quale sia il vero oggetto del discorso. Stando a quanto viene riferito, gli Inuit della Groenlandia si avvalgono di una dozzina di sostantivi per declinare le diverse sfumature della neve che pervade il loro ambiente (per quanto occorra prestare cautela nell'interpretare questa affermazione [13]). Noi, invece, spesso ci accontentiamo di un'unica parola per esprimere proprio ciò che appare più pervasivo nella nostra realtà quotidiana.

3.2.1. L'arcano

> La Castalia è un piccolo stato autonomo e il nostro Vicus Lusorum uno staterello entro quello stato [...]. Noi infatti abbiamo l'onore di custodire il vero sacrario della Castalia, il suo singolare simbolo e segreto, il Giuoco delle perle di vetro.
>
> *(H. Hesse, "Il giuoco delle perle di vetro": In carica)*

Arcanus è derivato di *arca*, intesa nella sua funzione di custodire, nascondere. Arcana è l'essenza della prospettiva informatica nella percezione del fruitore delle ICT che tale essenza nascondono per facilitarne l'operatività immediata. È l'obiettivo per cui tali strumenti sono stati concepiti. Ora si tratta però di chiedersi se è anche l'obiettivo che

intendiamo perseguire quando li caliamo nella scuola. Perché potrebbe benissimo esserlo da certi punti di vista, ma qualcosa non torna quando poi diciamo che semplicemente ponendo gli allievi di fronte a simili strumenti—probabilmente ciò che avviene nella maggior parte dei contesti scolastici—stiamo comunque insegnando l'informatica.

A questo proposito può essere istruttivo richiamare le conclusioni cui sono giunti alcuni ricercatori in merito alle implicazioni pedagogiche di un simile approccio, superficiale, all'informatica. Ho pertanto selezionato un passo significativo in relazione a ciascuno degli indizi identificati nella precedente sezione come elementi che contribuiscono a caratterizzare l'essenza dell'informatica.

1. Una caratteristica comune delle ICT è che si presentano attraverso interfacce che creano l'illusione di manipolazione diretta facendo svanire la distinzione fra *forma* e *significato* [4]:

   > Le interfacce di manipolazione diretta […] permettono un maggiore coinvolgimento nell'azione, al prezzo di un'illusione: l'utilizzatore opera direttamente su oggetti visibili sullo schermo, il cui funzionamento si basa su delle metafore [...].
   > Ma il trattamento operato, non visibile, tende a scomparire. Nessuna padronanza reale, ma del bricolage, senza comprensione. Lo si è potuto verificare sia su giovani allievi che sui futuri insegnanti. Atteso che le potenzialità dell'informatica risiedono nella possibilità di "far fare" dei trattamenti a una macchina, l'illusione del "fare", dell'azione diretta è certamente un ostacolo importante ai fini della padronanza e della comprensione di ciò che possono fare i computer.

2. Svanisce, inoltre, l'idea stessa di *elaborazione* [3]:

   > Uno dei risultati ottenuti [...] è rappresentato dalla quantità di informazioni conosciute dagli allievi senza un quadro per organizzarle. Più specificamente, la nozione di elaborazione il più delle volte era assente, mentre solo la parte visibile dei computer era considerata. […] Riassumendo, molti utilizzatori si impratichiscono per approssimazione, e non apprendono effettivamente.

3. Le tipiche modalità attraverso cui i ragazzi familiarizzano con l'uso delle ICT, sostanzialmente per imitazione, modalità che spesso non

riescono ad essere strutturate meglio nemmeno nell'ambito dell'istruzione scolastica, hanno come conseguenza, tra l'altro, il mancato sviluppo di un *linguaggio* adeguato per esprimere ciò che si sta facendo [8]:

> Di fronte alle difficoltà [...] incontrate, nella maggior parte dei casi gli allievi si ritrovano disarmati, non sapendo come descrivere quello che avviene sul monitor [...]. Questa mancanza di comprensione e di concettualizzazione si accompagna a una verbalizzazione piuttosto incerta [...]: gli allievi non sono in grado di esprimere né le loro azioni, né gli oggetti che all'apparenza manipolano facilmente.

4. Infine, per sviluppare una destrezza manipolatoria nell'uso delle ICT, l'dea di *algoritmo* risulta irrilevante [6]:

> [Con la didattica delle ICT] si è senza dubbio più prossimi all'insegnamento della pittura o all'iniziazione al tennis o alla pratica della chitarra che al problema di insegnare [...] l'algoritmica!

In conclusione, di cosa ci stiamo veramente occupando quando l'informatica si riduce all'addestramento all'uso di strumenti ICT? Ci sta bene che la sua essenza resti nascosta? Ci accontentiamo che l'Ordine castalio ne serbi gelosamente il segreto?

3.2.2. Partecipanti o spettatori del giuoco?

> Si era notato l'inconveniente che artisti della memoria sprovvisti di altre virtù eseguivano giuochi prestigiosi, sbalordendo e confondendo i partecipanti con la rapida successione d'innumerevoli idee.
>
> *(H. Hesse, "Il giuoco delle perle di vetro": Introduzione...)*

L'arcano si riverbera quindi in una babele semantica: al di fuori di *Vicus Lusorum* il giuoco delle perle di vetro può essere quasi qualsiasi cosa... E così non di rado finiscono per imporsi gli effetti speciali che strumenti sempre più sofisticati ci consentono di produrre con apparente naturalezza, anche se in realtà si tratta di mera imitazione: non siamo noi a condurre il giuoco.

Provando a schematizzare, e di conseguenza semplificando un po' la situazione, le accezioni dell'informatica prevalenti nei vari contesti formativi, si tratti di accezioni esplicite o implicite, possono essere classificate in tre categorie principali:

- *Accezione strumentale*, caratterizzata da una focalizzazione sul *prodotto* da realizzare, mentre i processi seguiti sono trattati a livello superficiale, se non elusi del tutto; in questa accezione tanto l'informatica quanto gli strumenti specifici che vengono applicati sono estranei agli obiettivi perseguiti.
- *Accezione operativa*, orientata all'*uso in generale* di certe tipologie di artefatti tecnologici, al di là da un contesto applicativo specifico; in questo caso interessa individuare schemi ricorrenti di interazione, invarianti cognitivi [22], ecc., che contribuiscano a sviluppare l'autonomia dell'allievo.
- *Accezione disciplinare*, laddove gli obiettivi perseguiti sono prettamente inerenti l'informatica, intesa nella sua dimensione culturale; secondo questa prospettiva, dal punto di vista didattico ci si propone di sviluppare competenze analitiche, metodologiche e pragmatiche rilevanti per la disciplina.

Ad integrazione delle annotazioni riportate sopra, la tabella seguente propone una caratterizzazione sintetica delle tre diverse accezioni dell'informatica, delineata attraverso alcuni termini chiave che dovrebbero essere autoesplicativi.

ACCEZIONE STRUMENTALE	ACCEZIONE OPERATIVA	ACCEZIONE DISCIPLINARE
Competenze contingenti	...	*Competenze stabili*
Saper fare	Saper generalizzare	Saper creare
Abilità manipolatorie	Intuizione analogica	Valutazione critica
Focus sui prodotti	Invarianti cognitivi	Focus sui processi
Addestramento	…	Formazione
"Black box"	"Translucent box"	"Glass/white box"

Tab. 3.1. Caratterizzazione delle accezioni dell'informatica.

3.3. Bussole

Da una caratterizzazione sommaria come quella abbozzata nel prologo sorge l'esigenza di un quadro in cui collocare gli "indizi" passati in rassegna (e/o altri eventuali) per dare loro un senso più compiuto. Anche in funzione delle peculiarità di un dato contesto, che include sensibilità e inclinazioni degli insegnanti presenti, può essere ragionevole proporre agganci con la riflessione sulla natura di discipline in qualche modo collegate, oppure con lo sviluppo storico-culturale per riconoscere i momenti in cui tali indizi hanno preso forma e le condizioni che hanno prodotto quel risultato.

3.3.1. Oltre l'arcano

> Plinio, pensa alla Castalia, non abbandonarci del tutto! Ci deve pure essere là fuori qualcuno che della Castalia ne sappia più dei motti di spirito che circolano sul nostro conto.
>
> *(H. Hesse, "Il giuoco delle perle di vetro": Waldzell)*

Di fronte a giovani allievi, per suggerire con una certa immediatezza che i principi fondativi dell'informatica prescindono dagli strumenti tecnologici, prendo frequentemente spunto da un confronto fra immagini relative al primo computer commerciale (il FERRANTI Mark I del 1951) e ad un semplice dispositivo mobile dei nostri giorni. A giudicare dall'apparenza esteriore dell'unità di elaborazione, della memoria di lavoro, di quella permanente, nonché dei mezzi che consentono l'interazione da parte dell'utente è ben difficile trovare qualcosa che accomuni le componenti corrispondenti.

Nella prospettiva dell'insegnante, un confronto del genere, ancorché superficiale, dovrebbe indurre a porsi delle domande. Innanzitutto c'è una questione che alla fine si rivela *ontologica*:

- Qual è il vero oggetto di studio che permette di spiegare in termini analoghi, sulla base degli stessi modelli, la funzione di artefatti così disparati per forma, dimensioni e principi fisici sfruttati? In altri termini, cosa accomuna in modo *essenziale* oggetti così diversi?

Il nodo di fondo, per affrontare una domanda come questa, sta nel fatto che l'oggetto di studio dell'informatica è *immateriale* in un senso sostanziale, al punto che si potrebbe caratterizzare la disciplina come

"scienza dell'immateriale"[4]. Appurato questo aspetto, che assegna un particolare rilievo ai processi cognitivi implicati, emergono alcune domande correlate:

- Se l'oggetto di studio è un'entità immateriale, attraverso quali strumenti *metodologici* siamo in grado di trattarla? E che tipo di conoscenza ne possiamo ricavare?
- Qual è inoltre il percorso storico-culturale che ha portato allo sviluppo dell'informatica come la conosciamo oggi? Quali sono le acquisizioni culturali *essenziali* lungo questo percorso?

In questa sede non c'è che lo spazio per un paio di brevi annotazioni in merito alle domande poste sopra. Mi limiterò quindi a suggerire delle direzioni verso cui eventualmente incanalare una riflessione.

3.3.2. Giuochi di specchi

> [La] contemplazione è per noi più importante dell'azione. […] A un tempo [il Giuoco] è, nel nostro tesoro, il gioiello più prezioso e più inutile, più amato e più fragile.
>
> *(H. Hesse, "Il giuoco delle perle di vetro": Il memoriale)*

Come prima annotazione, in relazione alla natura degli oggetti informatici, è interessante partire da questo passo di Abelson et al. [1], ribadito da Sussman [20], riguardo al ruolo dei programmi:

> Un linguaggio di programmazione non è semplicemente un modo per fare eseguire delle operazioni a un computer, ma è piuttosto un nuovo strumento formale per esprimere idee sulla metodologia. Pertanto i programmi devono essere scritti affinché le persone li possano leggere, e solo incidentalmente per essere eseguiti da macchine.

L'estratto contrappone due visioni divergenti di un *programma*, entità che si colloca al cuore dell'informatica:

- Un programma inteso come testo, scritto per agevolarne la lettura, cioè un prodotto culturale che in quanto tale può avere un impatto sulla realtà, ma solo indirettamente, attraverso la mediazione competente e consapevole di un attore umano.

---

[4] Mutuo questa felice espressione dal collega Furio Honsell, che l'ha spesso utilizzata nei suoi interventi sulla natura dell'informatica.

- Un programma inteso come processo, funzionale all'esecuzione su una macchina, cioè un'entità fenomenica, che può agire sulla realtà in modo diretto, indipendentemente da un intervento umano consapevole.

Da una parte, come prodotto culturale rivolto alle persone—un mezzo per condividere conoscenza procedurale—il programma (testo) può essere studiato con strumenti analitici assimilabili a quelli della matematica. Dall'altra, come entità fenomenica, il programma (processo) può diventare oggetto di sperimentazione scientifica per scoprirne le proprietà o per verificare delle ipotesi. Inoltre, una terza accezione del programma resta implicita per quanto detto fin qui; si tratta di quella tipica di un approccio ingegneristico:

- Un programma può essere concepito come prodotto di strategie di progetto e processi di sviluppo con l'obiettivo di crearne la struttura in relazione alla funzione da assolvere e di garantire l'affidabilità del risultato su un piano pragmatico.

Per approfondire la riflessione al riguardo può essere interessante considerare l'analisi di Eden [7] che, prendendo le mosse da queste tre caratterizzazioni contrastanti del concetto di programma, ne discute le implicazioni da un punto di vista filosofico[5]. In particolare, il quadro che viene a delinearsi consente di attribuire un significato più preciso—o meglio, significati alternativi—all'idea di *immaterialità* che contraddistingue l'essenza dell'informatica e al trattamento di entità immateriali. Le tabelle riportate nella pagina seguente riassumono in modo sommario la categorizzazione proposta da Eden in termini di natura attribuita agli oggetti studiati (ontologia), metodologia di indagine e natura delle conoscenze acquisite (epistemologia), mostrando inoltre come le accezioni corrispondenti siano concettualmente molto diverse.

Ontologia	*rappresentazioni semiotiche denotano oggetti astratti assimilabili ad oggetti matematici*
Metodologia	*analisi teorica, deduzione logica*
Epistemologia	*conoscenza* a priori *basata su caratterizzazioni formali complete*

Tab. 3.2. Prospettiva "matematica".

[5] Ai nostri fini è invece irrilevante condividere o meno la tesi sostenuta dall'autore in favore di un'accezione scientifica dell'informatica.

Ontologia	*processi di calcolo assimilati ad elaborazioni di informazioni in natura (DNA, reti neurali...)*
Metodologia	*inferenza da e verifica sperimentale di ipotesi*
Epistemologia	*conoscenza (inferenziale)* a priori *e (sperimentale)* a posteriori

Tab. 3.3. Prospettiva "scientifica".

Ontologia	*esistono solo pacchetti di dati rappresentati su supporti concreti*
Metodologia	*sviluppo pianificato e test di affidabilità*
Epistemologia	*conoscenza* a posteriori *basata su stime statistiche di affidabilità*

Tab. 3.4. Prospettiva "ingegneristica".

In sintesi, l'informatica è una disciplina multiforme e articolata in cui si integrano diverse anime, in particolare quelle condivise con le discipline degli ambiti matematico, scientifico e ingegneristico che hanno contribuito in modo sostanziale a farla nascere. Nelle parole di Hromkovič [10]:

> Le categorie disciplinari tradizionali sono insoddisfacenti: è interdisciplinare alla radice e si confronta con linguaggi e modi di pensare di diverse aree.

Privilegiando gli aspetti metodologici ed epistemologici rispetto alla pletora di nozioni accumulate nel corso della sua rapida evoluzione, ci si può avvicinare ad essa dalla prospettiva di ciascuno di questi ambiti, ponendo nello stesso tempo l'accento sui processi (p.es. cognitivi) messi in atto dagli allievi, processi che hanno maggiori probabilità di lasciare un segno stabile nel tempo. Per una discussione un po' più circostanziata delle implicazioni per la didattica dell'informatica si rimanda a [15, 21].

3.3.3. Flashback

> In fin dei conti dipende dall'arbitrio dello storico fin dove egli voglia far risalire gli inizi e la preistoria del Giuoco delle perle di vetro. Infatti, come tutte le grandi idee, esso non ha un vero e proprio inizio, ma come idea c'è sempre stato.
>
> *(H. Hesse, "Il giuoco delle perle di vetro": Introduzione...)*

Come seconda annotazione, le idee oggi associate all'informatica si possono rintracciare retrospettivamente nella storia rivolgendo l'attenzione a quei processi che hanno portato a trasferire al di fuori della mente umana attività che prima apparivano intrinsecamente legate ad essa. Possiamo provare a ricapitolare alcune tappe cruciali, in estrema sintesi, nei punti seguenti:

- *Segni*: uso di segni e sistemi semiotici al fine di registrare informazioni al di fuori dalla mente umana, abilità che si può far risalire al Paleolitico [19]. A questo livello interessa capire, per esempio, la natura convenzionale dei segni; la distinzione fra dati (forma) e informazioni (significato); la struttura e portata di un particolare codice.
- *Regole*: applicazione di regole per manipolare segni e rivelare nuove informazioni al di fuori della mente, competenze probabilmente sviluppate in Medio Oriente già nel corso del Neolitico [19]. A questo stadio ritroviamo alcuni aspetti importanti per capire la natura del trattamento formale, come le proprietà operative delle rappresentazioni (potenzialità di rivelare informazioni attraverso semplici manipolazioni).
- *Meccanismi*: l'applicazione di regole è affidata a meccanismi automatici per elaborare informazioni al di fuori della mente, inizio di un processo volto all'automazione che ha le radici nei dispositivi di calcolo e negli automi del periodo Ellenistico [18]. Il nodo concettuale di questo passaggio sta nell'interpretare dinamiche e fenomeni naturali in chiave computazionale, chiave di per sé non intrinseca ad essi.
- *Programmi*: codifica in qualche forma di programmi per controllare i passi dell'esecuzione di un piano al di fuori dalla mente umana. I primi esempi certi di simili artefatti appaiono negli automi programmabili di epoca Rinascimentale—anche se è possibile che delle versioni embrionali siano comparse già nel Medioevo arabo se non prima ancora [18]. Da un punto di vista

cognitivo ne sono prerequisito le capacità di introspezione e verbalizzazione di procedure algoritmiche.

- *Programmi-come-dati*: trattamento di programmi in quanto dati, che consente di realizzare conversioni fra livelli di astrazione al di fuori dalla mente umana. Si tratta dell'idea alla base della macchina universale concepita teoricamente da Turing intorno al 1936 e concretizzatasi negli anni immediatamente successivi al secondo conflitto mondiale, idea fondamentale per capire la natura e le potenzialità del computer moderno [16].
- *Sistemi complessi*: introduzione di varie strutture organizzative per gestire la complessità, a diversi livelli di astrazione, che caratterizzano l'evoluzione dell'informatica a partire dagli anni '50 del novecento. Questa evoluzione può essere vista come un processo che via via trasferisce fuori dalla mente prerogative che potevano sembrare peculiari dell'intelligenza umana; ovviamene rientrano in questo quadro anche i progetti di intelligenza artificiale [17].

Se si eccettuano le ultime due tappe, difficilmente istanziabili al di fuori del mondo del computer, molti principi fondativi possono essere ritrovati nella forma semplice—e meno aderente agli stereotipi correnti—che hanno assunto all'origine. Questo approccio si addice particolarmente alla scuola primaria, quando l'evoluzione culturale, soprattutto quella che si riferisce alle civiltà antiche, può essere presa come modello dello sviluppo cognitivo del bambino. Per esempio, i primi passi del percorso tracciato sono esplorati in [2], alla luce dei dati archeologici e antropologici, proprio in relazione all'istruzione elementare.

Vale inoltre la pena osservare che ciò che scandisce l'evoluzione cui si è accennato non sono particolari strumenti o dispositivi, di cui si potrebbero fare numerosi e differenziati esempi per ciascuna fase, ma delle acquisizioni cognitive, relative al modo di interpretare semplici oggetti e fenomeni. La predominanza degli aspetti cognitivi rispetto ai dispositivi fisici è messa chiaramente in risalto da Olley quando considera il significato del trattamento dei programmi come dati nella storia dell'informatica [16]:

[...] la mera esistenza dell'hardware necessario non conferisce l'essenza del computer, essendo piuttosto l'uso da parte di chi opera con la macchina a determinarne lo stato.

3.4. Epilogo: ludi magister

> Negli ultimi due anni del suo ufficio si definì due volte, nelle sue lettere, 'maestro di scuola', facendo notare che il titolo di Magister Ludi, che in Castalia ormai da generazioni significava 'Maestro del Giuoco', serviva in origine soltanto per designare il maestro di scuola.
>
> *(H. Hesse, "Il giuoco delle perle di vetro": In carica)*

A questo punto torniamo alle circostanze prefigurate all'inizio del presente capitolo: l'informatica calata in un contesto dove lo spazio è insufficiente per affrontarla in modo approfondito e dove eventualmente gli insegnanti con la responsabilità o la propensione ad occuparsene non hanno una sufficiente familiarità con la disciplina. È comunque ragionevole proporsi di fare qualcosa che non eluda l'essenza della disciplina? Da questo punto di vista, infatti, il mero utilizzo delle ICT può avere rilevanza educativa in qualche (altro) ambito, o anche in generale, pur restando sostanzialmente marginale nel senso messo in luce in queste pagine.

Potremmo sospettare che le opzioni che si presentano corrispondano alle situazioni estreme: fare informatica "sul serio", oppure "fare finta" di affrontarla attraverso un bricolage con le ICT. Ebbene, forse una strada intermedia la si può trovare se ci si sforza di ampliare la riflessione al di là delle pratiche più scontate. Per concludere, mi arrischio a suggerire degli spunti che fungano da linee guida in una situazione didattica generica e indipendentemente dagli strumenti che si ritiene opportuno utilizzare, siano ICT o altro, interpretando gli indizi introdotti nel prologo... (nel consueto ordine):

1. Per quanto possibile è meglio aggirare gli strumenti che danno l'illusione di un trattamento a livello di significato, per esempio rappresentando i dati in modi inconsueti per costringere a distinguere i piani della forma e del significato.
2. È più formativo sforzarsi di costruire a partire da pochi mattoni di partenza, per esempio limitando la disponibilità di strumenti utilizzabili alle funzionalità più elementari, in modo da stimolare lo sviluppo di competenze organizzative.
3. È utile esplicitare e verbalizzare i procedimenti risolutivi nei dettagli, senza ambiguità, anche quando può apparire pedante.
4. È opportuno pianificare diligentemente la struttura delle azioni

volte a perseguire un certo obiettivo, anticipandone i possibili sviluppi prima di cominciare a metterle in opera, evitando di procedere in modo incrementale per tentativi ed errori.

...Nonché, più in generale, in base alla discussione che ne è seguita:

- La fiducia nell'interpretazione dei dati in esame deve essere riposta esclusivamente sulla base dalla padronanza del trattamento soggiacente.
- Non è opportuno curarsi troppo del prodotto, poiché conta molto di più il processo che lo produce.
- È bene investire tempo e attenzione nelle fasi di progetto, che spesso tendono invece ad essere eluse.

Infine, non posso che accomiatarmi dal lettore che ha avuto la pazienza (e ha fatto la fatica) di seguire il filo intrecciato della trama fin qui, augurandomi che mi vorrà perdonare per l'arbitrio con cui vi ho incastonato le parole di Hermann Hesse, decontestualizzate dall'opera in cui le aveva collocate con tanta cura. Tanto più che la struttura che ne risulta, tutta giocata sull'ambiguità da questo punto di vista, ha ben poco a che spartire con le strutture di cui si occupa l'informatica...

Bibliografia

[1] ABELSON, H., SUSSMAN, G.J., SUSSMAN, J., *Structure and Interpretation of Computer Programs,* MIT Press (1984).

[2] BITTO, D., MIROLO, C., *"Archaeology of Information" in the Primary School,* Proc. of ISSEP 2013, LNCS 7780 (2013), 115-126.

[3] BRUILLARD, É., *From the didactics of computer science towards the didactics of instrumental activities with ICT*, 2nd Greek Conference on Didactics of Informatics (2004).

[4] BRUILLARD, É., *Informatique en contexte scolaire, enseignement, diffusion*, Séminaire de didactique des sciences expérimentales et des disciplines technologiques, 14 (2006), 115-128.

[5] DUCHÂTEAU, C., *Peut-on définir une "culture informatique"?*, Journal de Réflexion sur l'Informatique, 23-24 (1992), 34-39.

[6] DUCHÂTEAU, C., *Mais qu'est la didactique de l'informatique devenue?, Le technologies en éducation: Perspectives de recherches et questions vive*, Actes du symposium international francophone (2002), 33-42.

[7] EDEN, A.H., *Three Paradigms of Computer Science,* Minds and Machines, 17 (2007), 135-167.
[8] FLUCKIGER, C., *Internet et ses pratiques juvéniles,* Medialog, 69 (2009), 42-45.
[9] HESSE, H., *Das Glasperlenspiel,* Fretz & Wasmuth (1943).
[10] HROMKOVIČ, J., *Contributing to General Education by teaching informatics*, Proc. of ISSEP 2006, LNCS 4226 (2006), 25-37.
[11] KNUTH, D.E., *Computer Science and Its Relation to Mathematics,* The American Mathematical Monthly, 81 (1974), 323-343.
[12] MANNILA, L., DAGIENE, V., DEMO, B., GRGURINA, N., MIROLO, C., ROLANDSSON, L., SETTLE, A., *Computational Thinking in K-9 Education,* Proc. of ITiCSE-WGR '14, ACM (2014), 1-29.
[13] MARTIN, L., *Eskimo Words for Snow: A case study in the genesis and decay of an anthropological example,* American Anthropologist, 88 (1986), 418-23.
[14] MAZOYER, J., *L'enseignement de l'informatique de la maternelle à la terminale,* Institut de France - Académie des sciences (2005).
[15] MIROLO, C., *A Present-Day "Glass Bead Game": A Framework for the Education of Prospective Informatics Teachers Inspired by a Reflection on the Nature of the Discipline,* Proc. of ISSEP 2014, LNCS 8730 (2014), 138-149.
[16] OLLEY, A., *Existence Precedes Essence : Meaning of the Stored-Program Concept,* in History of Computing: Learning from the Past, Springer (2010), 169-178.
[17] O'REGAN, A., *A Brief History of Computing,* Springer, 2012.
[18] ROSSI, C., RUSSO, F., RUSSO, F., *Ancient Engineers' Inventions: Precursors of the Present,* Springer, 2009.
[19] SCHMANDT-BESSERAT, D., *How Writing Came About,* The University of Texas Press, Austin, 1996.
[20] SUSSMAN, G.J., *The Legacy of Computer Science, Computer Science: Reflections on the Field, Reflections from the Field,* The National Academies Press (2004), 180-183.
[21] TEDRE, M., SUTINEN, E., *Three traditions of computing: what educators should know,* Computer Science Education, 18 (2006), 153-170.
[22] VANDEPUT, É., *Milestones for Teaching the Spreadsheet Program,* Proc. of EuSpRIG (2009), 133-143.
[23] WING, J.M., *Computational Thinking,* Communications of the ACM, 49 (2006), 33-35.

4. La formazione degli insegnanti della classe 42/A – Informatica: l'esperienza dell'Università degli Studi di Milano

Carlo Bellettini, Violetta Lonati, Dario Malchiodi, Mattia Monga, Anna Morpurgo, Federico Pedersini[1]

In Italia la formazione universitaria all'insegnamento nella scuola secondaria ha una tradizione piuttosto recente: alle scuole di specializzazione attive negli anni 1999-2008 dovrebbero – secondo quanto previsto dal Decreto Ministeriale del 10 settembre 2010, n. 249 – sostituirsi lauree magistrali innestate sulla corrispondente formazione disciplinare triennale; nel transitorio sono stati attivati corsi annuali riservati a laureati di secondo livello selezionati tramite esami (Tirocinio formativo attivo, TFA) o titolari di un'esperienza professionale di insegnamento di almeno 3 anni (Percorsi abilitanti speciali, PAS). Questo capitolo descrive l'esperienza dell'Università degli Studi di Milano nel progettare e gestire i corsi 42/A TFA e PAS, nel triennio 2012-2015.

4.1. Il contesto

A partire dal settembre 2010 i requisiti e le modalità della formazione dei futuri insegnanti della scuola secondaria sono stati oggetto di vari provvedimenti ministeriali[2]. L'obiettivo, tuttora previsto dal regolamento del 2010, di attivare apposite lauree magistrali orientate all'insegnamento sembra per il momento piuttosto lontano. L'idea di "formazione universitaria all'insegnamento" presupporrebbe due importanti condizioni di partenza: da una parte gli abilitandi dovrebbero costruire la loro professionalità didattica sulla base di una solida competenza disciplinare, dall'altro le università dovrebbero essere in

[1] Dipartimento di Informatica, Università degli Studi di Milano.

[2] http://hubmiur.pubblica.istruzione.it/web/universita/offertaformativa/formazione-iniziale-degli-insegnanti-corsi-universitari.

grado di fornire corsi con obiettivi formativi sia pedagogici che di didattica disciplinare.

Non sono molte, però, le discipline per le quali già esiste un diffuso interesse scientifico volto a comprendere il modo migliore per insegnarle nelle scuole: l'area delle scienze fisiche prevede uno specifico settore disciplinare (FIS/08 – Didattica e storia della fisica), la matematica ha una consolidata tradizione di riflessione didattica, così come l'insegnamento delle lingue straniere (anche in questo caso è previsto uno specifico settore disciplinare L-LIN/02 – Didattica delle lingue moderne).

Per quanto riguarda la classe concorsuale 42/A – Informatica[NDR], la situazione di partenza pare invece, almeno in Italia, piuttosto insoddisfacente: gli abilitandi provengono spesso da formazioni di base molto eterogenee e la ricerca nell'ambito della didattica dell'informatica[3] ha diffusione piuttosto limitata negli atenei italiani: per esempio, una ricerca di pubblicazioni posteriori al 2000 classificate come "K.3.2 Computer and Information Science Education" fra quelle edite dall'ACM, fornisce soltanto 55 risultati con un'affiliazione italiana (su un totale di 9223), contro i 207 della Germania (su 19513) e i 196 della Finlandia (su 3136).

A ciò si aggiunge un ulteriore fattore, la cui criticità è ormai riconosciuta internazionalmente [12] [4]: la ricorrente confusione su ciò che debba essere associato al termine informatica [11], equivocata con l'uso delle applicazioni e delle tecnologie digitali. Nonostante la formazione preveda un consistente numero di crediti INF/01 (perciò chiaramente orientati agli aspetti scientifici e metodologici dell'informatica), le linee guida ministeriali spesso mancano di marcare in modo inequivoco la distinzione. Ciò disorienta formatori, docenti in formazione e in definitiva gli studenti, che rischiano – soprattutto nel caso in cui non vengano esposti a studi specialistici – di identificare un intero ecosistema scientifico con una serie di abilità destinate a invecchiare piuttosto in fretta.

NDR La classe concorsuale chiamata qui 42/A è detta altrove in questo volume A042; lo stesso dicasi per altre classi menzionate.

3 Si noti che la didattica dell'informatica è tutt'altra cosa rispetto alla tecnologia di supporto alla didattica (informatica per la didattica): purtroppo però le due cose sono spesso confuse.

4.2. I contenuti dei corsi d'informatica nella scuola superiore

Le riforme della scuola degli anni '90 hanno introdotto il principio della autonomia scolastica: ogni istituto redige annualmente un "Piano dell'offerta formativa" che può riorganizzare liberamente i percorsi didattici, a condizione che si rispettino le indicazioni e gli obiettivi d'istruzione fissati a livello nazionale.

Nel caso della scuola secondaria di secondo grado, i documenti di riferimento sono le "Indicazioni nazionali"[4] per i Licei e le "Linee guida"[5] per gli istituti tecnici e professionali.

4.2.1. Indicazioni nazionali (licei)

Iniziamo subito a notare come si fissi fra i risultati di apprendimento comuni a tutti i percorsi liceali di "Essere in grado di utilizzare criticamente strumenti informatici e telematici nelle attività di studio e di approfondimento; comprendere la valenza metodologica dell'informatica nella formalizzazione e modellizzazione dei processi complessi e nell'individuazione di procedimenti risolutivi." A ciò però non sembra seguire un coerente disegno volto a presentare gli aspetti metodologici e scientifici dell'informatica: la materia con questo nome è infatti prevista solo per i licei scientifici "Opzione scienze applicate", il che fa pensare che gli estensori delle indicazioni abbiano sottovalutato il potenziale epistemologico, e in generale culturale, della rivoluzione informatica iniziata nel '900, anche per le scienze "pure".

Per l'indirizzo "Opzione scienze applicate", invece, si delinea un percorso di tutto rispetto, da svolgere però in sole 66 ore all'anno: "Dal punto di vista dei contenuti il percorso ruoterà intorno alle seguenti aree tematiche: architettura dei computer, sistemi operativi, algoritmi e linguaggi di programmazione, elaborazione digitale dei documenti, reti di computer, struttura di Internet e servizi, computazione, calcolo

---

[4] http://nuovilicei.indire.it/content/index.php?action=lettura_paginata&id_m=7782&id_cnt=10497 [5].

[5] Per gli istituti tecnici: http://www.indire.it/lucabas/lkmw_file/nuovi_tecnici///INDIC/_LINEE_GUIDA_TECNICI_.pdf [6], per gli istituti professionali: http://www.indire.it/lucabas/lkmw_file/nuovi_professionali///linee_guida/_LINEE GUIDA ISTITUTI PROFESSIONALI_.pdf [7].

numerico e simulazione, basi di dati.". Vale la pena riportare alcuni tratti del profilo dello studente modello descritto dalle indicazioni: "Ha una sufficiente padronanza di uno o più linguaggi per sviluppare applicazioni semplici, ma significative, di calcolo in ambito scientifico. Comprende la struttura logico-funzionale della struttura fisica e del software di un computer e di reti locali, tale da consentirgli la scelta dei componenti più adatti alle diverse situazioni e le loro configurazioni, la valutazione delle prestazioni, il mantenimento dell'efficienza. L'uso di strumenti e la creazione di applicazioni deve essere accompagnata non solo da una conoscenza adeguata delle funzioni e della sintassi, ma da un sistematico collegamento con i concetti teorici ad essi sottostanti. Il collegamento con le discipline scientifiche, ma anche con la filosofia e l'italiano, deve permettere di riflettere sui fondamenti teorici dell'informatica e delle sue connessioni con la logica, sul modo in cui l'informatica influisce sui metodi delle scienze e delle tecnologie, e su come permette la nascita di nuove scienze."

L'obiettivo non può che risultare gradito a orecchie informatiche, ma – posto che sia alla portata degli studenti della scuola secondaria – è impossibile non notare come ciò sia in contrasto con il ridotto numero di ore a disposizione (analogo a quelle riservate a "Disegno e storia dell'arte" e "Scienze motorie e sportive").

4.2.2. Linee guida (istituti tecnici e professionali)

Per gli istituti tecnici e professionali, materie informatiche sono proposte nel biennio a tutti gli indirizzi.

Per gli istituti professionali gli obiettivi previsti, molto strumentali, riguardano per lo più l'uso delle applicazioni dell'informatica, con particolare riferimento agli ambiti della grafica e della multimedialità.

Negli istituti tecnici del settore economico, è prevista una materia denominata "Informatica". Nonostante si individui come obiettivo prioritario "una formazione tecnologica rivolta all'innovazione, che richiede sia la capacità di risolvere problemi sia quella di riflettere sui modelli e sui fondamenti concettuali", l'elenco proposto per conoscenze e abilità della disciplina Informatica non brilla per limpidezza dei riferimenti fondamentali: si mischiano livelli di astrazione e impostazioni molto differenti. Per esempio fra le conoscenze si citano: informazioni, dati e loro codifica; software di utilità e software

gestionali; funzioni e caratteristiche della rete Internet e della posta elettronica; normativa sulla privacy e sul diritto d'autore. Per le abilità l'elenco comprende: riconoscere e utilizzare le funzioni di base di un sistema operativo; analizzare, risolvere problemi e codificarne la soluzione; riconoscere le principali forme di gestione e controllo dell'informazione e della comunicazione specie nell'ambito tecnico-scientifico-economico. Insomma c'è un po' di tutto, manca invece un filo conduttore evidente.

Per gli istituti tecnici del settore tecnologico (escluso l'indirizzo "Informatica e telecomunicazioni") si parla invece di "Tecnologie informatiche". Qui il disegno didattico sottostante è più chiaro: le conoscenze e abilità indicate sono sempre piuttosto vaghe, ma meno variegate: ad esempio si citano le "Funzioni e caratteristiche della rete internet", senza lo strano riferimento esplicito alla posta elettronica. La programmazione è descritta in maniera coerente ed esplicita.

Un discorso finale a parte meritano invece le materie informatiche dell'indirizzo "Informatica e telecomunicazioni". In questo caso si mira chiaramente a formare degli specialisti e i contenuti sono quindi orientati a formare professionalità spendibili immediatamente nel mondo del lavoro. Le aree tematiche classiche dell'informatica e delle sue ramificazioni professionali, fra "Informatica", "Sistemi e reti", "Tecnologie e progettazione di sistemi informatici e di telecomunicazioni", "Gestione progetto, organizzazione d'impresa" sono senz'altro presenti. Anche qui va rilevato che "Informatica" è la disciplina in cui il disegno didattico è meno esplicito, e fra le conoscenze appare il concetto di "File di testo" insieme a quello di ben altra caratura di "Programmazione ad oggetti", oltre a una strana distinzione fra linguaggi di programmazione lato client e lato server. Ciò che pare più difficile da comprendere, però, è la distinzione fra le discipline previste dall'articolazione "Informatica" e quella "Telecomunicazioni". Anche se denotate da insegnamenti con gli stessi nomi ("Sistemi e reti", "Tecnologie e progettazione di sistemi informatici e di telecomunicazioni", "Gestione progetto, organizzazione d'impresa") hanno obiettivi diversi. Si nota inoltre come i principi architetturali dei sistemi di calcolo sono praticamente assenti nella versione "Informatica".

4.3. L'insegnamento di materie informatiche affidato ai docenti 42/A

Ai docenti di ruolo nella classe 42/A possono essere affidati insegnamenti in corsi di studio molto diversi tra loro. La Tabella 4.1 riassume la situazione dopo i riordini del 2012 e 2013 (vedi Nota MIUR 2916 del 21/3/2013). In verde sono indicati gli insegnamenti che dovrebbero essere affidati esclusivamente a 42/A.

È da notare come l'insegnamento dell'informatica nei licei, benché contempli obiettivi didattici di ampio respiro, non risulti affidato a docenti della classe 42/A (salvo per l'opzione "Scienze applicate"); in genere spetta invece, come parte della loro materia, a docenti di matematica, quindi non necessariamente con una formazione specifica in informatica.

Gli insegnamenti di Sistemi e reti, Tecnologie e progettazione di sistemi informatici e di telecomunicazioni, Gestione progetto, organizzazione d'impresa negli istituti tecnici a indirizzo "Informatica e telecomunicazioni" sono affidati a docenti 42/A se l'articolazione è informatica, altrimenti (articolazione telecomunicazioni) a docenti 34/A (Elettronica).

In generale va notata la compresenza di professionalità piuttosto differenti eppure incaricate di perseguire i medesimi obiettivi formativi: è difficile immaginare che una tale varietà nel corpo docente riesca a trasmettere un'idea coerente delle discipline informatiche. Il problema è tutto sommato di secondaria importanza nei percorsi scolastici in cui l'informatica ha chiaramente un ruolo di servizio, ma diventa invece critico dove si suppone invece che essa diventi modello metodologico e chiave interpretativa della realtà, come dovrebbe avvenire in un percorso liceale.

Abbastanza sorprendente, infine, risulta l'esame della disciplina "Scienze e tecnologie applicate" per la quale è difficile capire il legame con la classe 42/A (e veramente anche con la 34/A – Elettronica). Le conoscenze elencate sono: I materiali e loro caratteristiche fisiche, chimiche, biologiche e tecnologiche; Le caratteristiche dei componenti e dei sistemi di interesse; Le strumentazioni di laboratorio e le metodologie di misura e di analisi; La filiera dei processi caratterizzanti l'indirizzo e l'articolazione; Le figure professionali caratterizzanti i vari settori tecnologici. Le abilità: riconoscere le proprietà dei materiali e le

funzioni dei componenti; utilizzare strumentazioni, principi scientifici, metodi elementari di progettazione, analisi e calcolo riferibili alle tecnologie di interesse; analizzare, dimensionare e realizzare semplici dispositivi e sistemi; analizzare e applicare procedure di indagine; riconoscere, nelle linee generali, la struttura dei processi produttivi e dei sistemi organizzativi dell'area tecnologica di riferimento.

In definitiva, quindi, l'analisi complessiva delle assegnazioni alle diverse classi di concorso lascia qualche dubbio di coerenza.

ISTITUTO	INDIRIZZO	DISCIPLINA	ORE SETTIMANALI					CLASSE DI CONCORSO
	anno		1	2	3	4	5	
Tecnico	*Amministrazione, finanza e marketing*	*Informatica*	*2*	*2*				*42/A + 75/A + 76/A*
Tecnico	*Amministrazione, finanza e marketing*	*Informatica*			*2*	*2*		*42/A*
Tecnico	*Amministrazione, finanza e marketing: Relazioni internazionali per il marketing*	*Tecnologie della comunicazione*			*2*	*2*		*42/A*
Tecnico	*Amministrazione, finanza e marketing: Sistemi informativi aziendali*	*Informatica*			*4*	*5*	*5*	*42/A*
Tecnico	*Turismo*	*Informatica*	*2*	*2*				*42/A + 75/A + 76/A*
Tecnico	*Meccanica, meccatronica ed energia*	*Tecnologie informatiche*	*3*					*34/A + 35/A + 42/A*
Tecnico	*Trasporti e logistica*	*Tecnologie informatiche*	*3*					*34/A + 35/A + 42/A*
Tecnico	*Elettronica ed elettrotecnica*	*Tecnologie informatiche*	*3*					*34/A + 35/A + 42/A*
Tecnico	*Informatica e telecomunicazioni*	*Tecnologie informatiche*	*3*					*34/A + 35/A + 42/A*
Tecnico	*Informatica e telecomunicazioni*	*Scienze e tecnologie applicate*		*3*				*34/A + 42/A*
Tecnico	*Informatica e telecomunicazioni: Informatica*	*Informatica*			*6*	*6*	*6*	*42/A*

Tecnico	*Informatica e telecomunicazioni: Informatica*	*Sistemi e reti*			4	4	4	*42/A*
Tecnico	*Informatica e telecomunicazioni: Informatica*	*Tecnologie e progettazione di sistemi informatici e di telecomunicazioni*			3	3	4	*42/A*
Tecnico	*Informatica e telecomunicazioni: Informatica*	*Gestione progetto, organizzazione d'impresa*					3	*42/A*
Tecnico	*Informatica e telecomunicazioni: Telecomunicazioni*	*Informatica*			3	3		*42/A*
Tecnico	*Informatica e telecomunicazioni: Telecomunicazioni*	*Sistemi e reti*			4	4	4	*34/A*
Tecnico	*Informatica e telecomunicazioni: Telecomunicazioni*	*Tecnologie e progettazione di sistemi informatici e di telecomunicazioni*			3	3	4	*34/A*
Tecnico	*Informatica e telecomunicazioni: Telecomunicazioni*	*Gestione progetto, organizzazione d'impresa*					3	*34/A*
Tecnico	*Grafica e comunicazione*	*Tecnologie informatiche*	3					*34/A + 35/A + 42/A*
Tecnico	*Grafica e comunicazione*	*Scienze e tecnologie applicate*		3				*42/A + altri*
Tecnico	*Grafica e comunicazione*	*Progettazione multimediale*			4	3	4	*42/A + altri*
Tecnico	*Chimica, materiali e biotecnologie*	*Tecnologie informatiche*	3					*34/A + 35/A + 42/A*
Tecnico	*Sistema moda*	*Tecnologie informatiche*	3					*34/A + 35/A + 42/A*
Tecnico	*Agraria, agroalimenare e agroindustrale*	*Tecnologie informatiche*	3					*34/A + 35/A + 42/A*
Tecnico	*Costruzioni, ambiente e territorio*	*Tecnologie informatiche*	3					*34/A + 35/A + 42/A*
Professionale	*Servizi per l'agricoltura e lo sviluppo rurale*	*Tecnologie dell'informazione e della comunicazione*	2	2				*42/A + altri*
Professionale	*Servizi commerciali*	*Informatica e laboratorio*	3	3				*42/A + 75/A + 76/A*

Professionale	*Produzioni industriali e artigianali*	*Tecnologie dell'informazione e della comunicazione*	*2*	*2*					*42/A + altri*
Professionale	*Manutenzione e assistenza tecnica*	*Tecnologie dell'informazione e della comunicazione*	*2*	*2*					*42/A + altri*
Professionale	*Manutenzione e assistenza tecnica*	*Tecnologie e tecniche di installazione e manutenzione*			*3*				*42/A + altri*
Liceo	*Opzione scienze applicate*	*Informatica*	*2*	*2*	*2*	*2*	*2*		*34/A + 35/A + 42/A*

Tab. 4.1. Affidamento degli insegnamenti di area informatica alle classi di concorso: 42/A Informatica, 34/A Elettronica, 35/A Elettrotecnica e applicazioni, 76/A Trattamento testi, calcolo, contabilità.

4.3.1. Chi ha titolo per insegnare l'informatica nella scuola superiore?

Per rispondere occorre districarsi fra gli ordinamenti dei corsi di studio, in costante evoluzione da almeno un ventennio. Per questo motivo il Ministero mette a disposizione un'applicazione[6] per conoscere quali titoli di studio danno accesso ai concorsi nella classe 42/A – Informatica. Occorre possedere una laurea in:

- Scienze dell'informazione
- Informatica
- Ingegneria informatica
- Matematica
- Fisica
- Ingegneria elettronica
- Ingegneria aerospaziale
- Ingegneria delle telecomunicazioni
- Ingegneria gestionale
- Discipline nautiche, purché il piano di studi seguito abbia compreso i corsi annuali (o due semestrali) di: calcolo numerico e programmazione, complementi di matematica per le applicazioni, teoria dei sistemi o esami ritenuti omogenei a questi, secondo un'apposita tabella di corrispondenze.

6 http://hubmiur.pubblica.istruzione.it/PRTATitoliAccesso/ricercatitoliperclasse.action.

Come si vede, una platea piuttosto variegata, con qualche presenza sorprendente. Ancora più sorprendente è l'elenco degli esami considerati omogenei: per "Teoria dei sistemi" sono Programmazione, Sistemi di elaborazione, Sistemi di elaborazione dell'informazione, Sistemi operativi. Nessuna difficoltà nel considerare questi ultimi come necessari per una formazione informatica, ma non è chiaro cosa significhi considerarli "omogenei" allo studio dei sistemi dinamici e retroazionati che è invece l'ambito classico della teoria dei sistemi. Le "omogeneità" sembrano improntate più ad assonanze terminologiche (la presenza della parola "Sistemi") che a vere relazioni semantiche: anche Calcolo numerico è ritenuto omogeneo a Calcolo parallelo!

La classe di concorso 42/A – Informatica prevede che i concorrenti possano essere messi alla prova secondo un programma ministeriale che, abbastanza sorprendentemente, e, al contrario di quanto accaduto per la maggior parte delle altre discipline, non è stato aggiornato dal Decreto Ministeriale 21 settembre 2012, n. 80. Bisogna pertanto riferirsi al precedente Decreto Ministeriale 11 agosto 1998, n. 357:

1. *Modelli dell'informatica*
   - Soluzione dei problemi: processi euristici e processi algoritmici
   - Proprietà degli algoritmi: costrutti fondamentali, complessità
   - Algoritmi notevoli: ordinamento, ricerca, fusione
   - Sistemi logico-deduttivi
   - Linguaggi formali. Sintassi e semantica

2. *Programmazione e linguaggi*
   - Rappresentazione dei dati e delle procedure, linguaggi e tecniche di programmazione secondo i diversi paradigmi:
     - programmazione imperativa
     - programmazione rivolta agli oggetti
     - programmazione non procedurale: funzionale e logica
   - Proprietà dei linguaggi di programmazione in relazione ai diversi paradigmi
   - Metodologia di costruzione dei programmi. Modularità.
   - Ingegneria del software, tecniche di documentazione e di manutenzione dei programmi.

3. *Architettura dei sistemi di elaborazione*
   - Sistemi digitali e programmabili. I microprocessori. Programmazione a livello macchina e con linguaggi orientati alla macchina.
   - Componenti di un sistema di elaborazione. Unità centrale. Unità periferiche. Memorie e loro gerarchia
   - Elaboratori con un solo processore: tipologie di architetture e loro caratteristiche funzionali.
   - Architetture parallele. Sistemi multiprocessori. Sistemi a matrice.

4. *La struttura dei programmi di base*
   - Sistemi operativi. Tipologie, struttura e funzioni. Tipologie di interfaccia con l'utente icone e comandi.
   - La gestione delle risorse fisiche e dei programmi da parte del sistema operativo. Analisi delle prestazioni.
   - Problemi di parallelismo e concorrenza.
   - Programmi di elaborazione dei linguaggi: interpreti e compilatori.
   - Programmi applicativi di utilità generale.

5. *Reti di elaboratori e reti di comunicazione*
   - Fondamenti di comunicazioni: segnali, canali, mezzi e metodi di trasmissione (analisi funzionale). Modem.
   - Protocolli. Standard di interfaccia, livelli e modelli.
   - Reti locali e reti geografiche: architettura fisica, sistemi operativi e programmi di comunicazione.
   - Servizi telematici.

6. *Gestione delle informazioni*
   - Analisi e progetto dei sistemi informativi. Archivi.
   - Gestione degli archivi con linguaggi di programmazione.
   - Basi di dati: struttura, progetto, linguaggi per la realizzazione e per l'interrogazione.

7. *Sistemi multimediali*
   - Rappresentazione dei diversi tipi di informazione: simboli, suoni, disegni, immagini.
   - Componenti fisici per i sistemi multimediali.
   - Strumenti di programmazione per i sistemi multimediali: linguaggi speciali orientati alle immagini, sistemi ipertestuali.

8. *Elementi di didattica*
   - La lezione frontale
   - Il problem solving
   - La scoperta guidata
   - L'analisi di caso e l'analisi tecnica
   - L'indagine
   - Il metodo dei progetti
   - La funzione del laboratorio nella didattica delle discipline tecniche e nelle attività progettuali
   - L'organizzazione del lavoro didattico: classe, gruppi, lavori individuali

9. *La programmazione e la progettazione didattica*
   - Analisi disciplinare e definizione degli obiettivi
   - Piani di lavoro
   - Moduli e unità didattiche: scelta dei metodi e delle risorse
   - Verifiche e valutazione.

4.4. La proposta di Unimi per la formazione degli insegnanti

Quando, durante l'anno accademico 2011/12, fu necessario segnalare l'interesse a erogare un corso TFA 42/A (ritenuto all'epoca del tutto transitorio, in attesa dell'attivazione delle lauree magistrali tuttora previste dalla legge per l'anno 2012/13), come gruppo ("ALaDDIn"[7]) di docenti del Dipartimento di Informatica dell'Università degli Studi di Milano che si occupa di didattica e divulgazione dell'informatica interpretammo l'occasione come lo spunto per stabilire una cinghia di trasmissione stabile con le scuole secondarie del territorio. Sembrava soprattutto una buona opportunità per acquisire esperienza da spendere al momento dell'attivazione delle lauree magistrali.

Fin dall'inizio ci fu chiaro che sarebbe stato necessario rincorrere pianificazioni approssimative (disattese innanzitutto dagli organismi ministeriali che le emanavano) e districarsi fra incertezze ed evoluzioni

[7] http://aladdin.di.unimi.it/.

normative e organizzative[8]. Ciò nonostante decidemmo di provare a mettere in piedi un'offerta formativa originale che sfruttasse i 18 CFU a disposizione del settore INF/01 per riflettere seriamente sulla didattica della disciplina, con un chiaro aggancio con le parallele attività di ricerca in quest'ambito ([1] [2] [3] [7] [8] [10]), senza mutuazioni dai corsi curriculari, per loro natura focalizzati su obiettivi più schiettamente disciplinari.

Ad oggi abbiamo erogato tre edizioni del percorso di formazione, riprogettato più volte, in parte in risposta a riorganizzazioni e vincoli posti dall'esterno (la seconda edizione, per esempio, si rivolgeva a docenti già titolari di un'esperienza triennale di insegnamento, e prevedeva perciò una regolamentazione peculiare), ma soprattutto per adattarsi meglio all'obiettivo dichiarato di rendere più efficace l'insegnamento delle materie informatiche nella scuola secondaria di secondo grado. Per questo crediamo possa essere utile ripercorrere qui la storia delle tre edizioni.

4.4.1. TFA 2012/13

L'avventura del TFA 2012/13 inizia con la primavera 2012, quando il MIUR fissa i contingenti per la classe 42/A: verranno accettati 315 candidati in tutta Italia. La prima sorpresa è che i 25 posti riservati alla Lombardia sono divisi fra Università degli Studi di Milano e Università degli Studi di Milano – Bicocca (rispettivamente 10 e 15 posti). Che vi siano nella stessa regione due sedi, per di più nella stessa città, per un numero così esiguo di candidati, pare a chi scrive uno straordinario esempio di spreco di risorse umane e organizzative[9]. Dopo la selezione nazionale a luglio (con 60 domande a risposta multipla, di cui 16 valutate a posteriori ambigue o scorrette e le cui risposte verranno perciò considerate corrette in ogni caso) 31 candidati possono partecipare all'esame locale (50 a Bicocca)[10], si presentano in 20 e solo 6 superano la prova.

---

[8] Vale la pena ricordare che la legge che istituisce i corsi TFA, incardinandone il controllo a livello di Facoltà universitarie, è parte della stessa riforma della governance degli atenei, che, fra le altre cose, svuota le Facoltà del loro ruolo tradizionale, affidandone i compiti di organizzazione didattica ai Dipartimenti.

[9] Per l'edizione 2014/15 le sedi lombarde sono diventate tre per 38 posti: alle due precedenti si è aggiunta l'Università Cattolica del Sacro Cuore di Brescia.

[10] A livello nazionale, su 1505 candidati, le sufficienze con almeno 42 risposte esatte saranno 1004 (erano 451 prima della rettifica delle domande ambigue/errate).

Per la prova locale si è scelto di focalizzarsi sulla programmazione, chiedendo ai candidati di implementare (sulla carta) un algoritmo di ordinamento[11] a scelta fra Merge, Heap o Quick-sort in un linguaggio, di nuovo, a scelta. Si chiedeva inoltre di dare ragione della correttezza dell'implementazione, identificando casi di test ritenuti "significativi". Chiudeva il compito una domanda a scelta fra diagrammi E/R, cache e frammentazione della memoria.

Dei 6 ammessi, solo tre si presenteranno effettivamente a seguire i corsi. In totale il corso partirà con quattro studenti, tenuto conto di un'ammissione "extranumeraria" dovuta a situazioni particolari previste per legge. Solo due riusciranno a ottenere l'abilitazione nel luglio 2013, per i rimanenti (alla fine abilitati) sarà necessario ricorrere a una sessione suppletiva a febbraio 2014.

I corsi di informatica[12] previsti ricalcavano uno schema ereditato dalle precedenti scuole di specializzazione, anche se con contenuti del tutto rinnovati. Tre corsi da 6 CFU (Architetture, Programmazione, Sistemi operativi e Reti) di cui 2 CFU di ripasso e identificazione di nodi concettuali fondamentali e il resto focalizzato su strategie per l'insegnamento, con particolare riguardo alle esperienze laboratoriali.

La scelta dei tre corsi incentrati sulle "Architetture", la "Programmazione" e i "Sistemi operativi e reti" ha lo scopo di trasmettere una rappresentazione inequivoca dell'informatica come disciplina che studia l'elaborazione automatica dell'informazione: servono quindi mezzi di calcolo, e occorre saperli programmare e gestire in un'ottica sistemica e di rete. Questo nucleo essenziale dovrebbe far parte dell'esperienza di chiunque si avvicini allo studio dell'informatica a prescindere dall'eventuale (e in molti casi senz'altro consigliabile) acquisizione di specifiche abilità nell'uso di applicazioni di settore.

---

[11] Gli algoritmi di ordinamento sono espressamente citati nel programma ministeriale dei concorsi 42/A. Un candidato ha scritto di non conoscere nessun linguaggio di programmazione: ha perciò fornito una soluzione tramite un diagramma di flusso, peraltro sbagliato, dell'algoritmo Bubble-sort.

[12] Per i crediti di scienze dell'educazione, l'Università degli Studi di Milano ha erogato corsi comuni per tutte le classi dell'area scientifica.

4.4.2. PAS 2013/14

Nel 2013/14 è stato attivato un Percorso Abilitante Speciale (PAS) riservato a coloro i quali potessero vantare 3 anni di servizio nella classe 42/A. Nonostante non fosse prevista alcuna prova d'ingresso, il decreto istitutivo del PAS richiamava la necessità di verificare la solidità delle competenze di base degli abilitandi.

Per questo motivo abbiamo deciso di iniziare con un corso introduttivo pluridisciplinare di informatica (3CFU), in cui si riepilogassero i concetti fondamentali su cui costruire una successiva riflessione didattica. Il corso ha insistito in particolare sui rudimenti della programmazione, oltre che sulle nozioni fondamentali di architetture, sistemi e reti.

L'ufficio scolastico regionale ha assegnato all'Università degli Studi di Milano 30 studenti aventi titolo alla partecipazione. Alla prima lezione se ne sono presentati soltanto 17 (ulteriori due verranno poi assegnati con un paio di settimane di ritardo a seguito di ricorsi), con questa composizione: 5 laureati in matematica, 4 in ingegneria informatica, 3 in ingegneria elettronica, 1 in scienze dell'informazione, 1 in ingegneria delle telecomunicazioni, 1 in fisica, 1 in ingegneria aerospaziale e 1 non laureato, ma esplicitamente autorizzato a partecipare dall'ufficio scolastico. Due erano già in possesso di abilitazione in altre classi.

Come prima attività abbiamo chiesto di fornire, in forma anonima, una concisa definizione di informatica, la disciplina per la quale i partecipanti aspiravano a una abilitazione (e che, non va dimenticato, avevano in forme più o meno precarie già insegnato per almeno 3 anni). Le risposte ottenute sono riportate di seguito.

- Comunicazione, risoluzione di problemi.
- Scienza che ha per oggetto gli studi teorici dell'informazione e ne permette la computazione e l'elaborazione.
- Lo studio della scienza dell'informazione.
- Un mezzo per migliorare la vita.
- Una materia multidisciplinare che permette di toccare più realtà e più discipline. Permette di astrarre ogni realtà e di farne un modello.
- Disciplina che studia le modalità di comunicazione tra uomo e PC utilizzando linguaggi di programmazione e reti di comunicazione.
- L'informatica è il progresso. E` la scienza che si propone di studiare tutto ciò che si può definire tecnologico e le modalità in cui si crea automazione.

- Scienza che si occupa del trattamento, memorizzazione, elaborazione e trasmissione dell'informazione, in maniera automatizzata.
- La scienza che studia l'elaborazione e la trasmissione dell'informazione.
- Informazione automatica. Computer hardware/software programmazione, algoritmi. Linguaggi. Linguistica computazionale. Codici. 0 e 1.
- La scienza che si occupa dell'elaborazione dell'informazione (dati) mediante metodi che sono propri dell'automatica, in particolare mediante l'ausilio del computer.
- Letteralmente significa "Tecnologia dell'informazione", cioè parte della scienza che studia la comunicazione e l'elaborazione di dati digitali.
- Informazione tecnologica delle conoscenze acquisite.
- L'informatica nasce dalla necessità di meccanizzare le informazioni. Da qui nasce un nuovo mondo che permette di ampliare i metodi di comunicare e lavorare.
- Informazione automatica. Scienza che studia l'informazione, la sua trasmissione e il suo utilizzo all'interno di sistemi automatici; inoltre sviluppa sistemi automatici in grado di prendere decisioni in modo autonomo secondo una determinata programmazione.
- Strumenti e linguaggi che aiutano a vivere meglio.
- È una materia multidisciplinare perché con i suoi strumenti permette di spaziare in molti campi lavorativi.

Dei 19 studenti che hanno perfezionato l'iscrizione, 9 si sono ritirati durante l'anno (la maggioranza durante il corso introduttivo pluridisciplinare) e 1 non ha conseguito l'abilitazione perché respinto agli esami di profitto.

I 15 CFU rimanenti, tolti i 3 dedicati al corso pluridisciplinare, sono stati focalizzati sulla didattica attiva e laboratoriale, con l'organizzazione descritta di seguito.

4.4.2.1. Metodi per l'insegnamento della programmazione

Obiettivo del corso era presentare un approccio metodologico alla programmazione basato su una serie di analogie con il metodo scientifico, che costituisce il paradigma culturale centrale delle discipline scientifiche e tecniche insegnate nella scuola secondaria di secondo grado. Il corso era stato progettato attorno a tre moduli concettuali:

- Formulare una teoria. Le prime due lezioni di natura più “motivazionale” hanno avuto per argomento gli elementi concettuali

della programmazione e sono state incentrate sullo strumento visuale di programmazione Scratch; l'ultima lezione ha offerto un'introduzione al (segmento imperativo della versione 3 del) linguaggio Python.
- Dedurre comportamenti. Grazie all'introduzione delle funzionalità offerte dalla libreria standard di Python, sono stati sviluppati alcuni progetti di programmazione, scelti anche in funzione della loro attrattività interdisciplinare.
- Osservare. Le ultime lezioni hanno avuto per argomento debugging e testing che, tra gli argomenti collegati alla programmazione, sono quelli che più naturalmente si pongono in relazione con il metodo scientifico.

È stato purtroppo necessario ridimensionare i contenuti e gli obiettivi del corso, a causa dello scarso livello generale di competenze disciplinari pregresse degli abilitandi.

4.4.2.2. Metodi per l'insegnamento delle architetture degli elaboratori

Sull'opportunità di includere nella didattica dell'informatica una parte sulle architetture degli elaboratori esistono pareri discordanti. Effettivamente, se i contenuti di architetture sono intesi (come in alcuni casi i programmi ministeriali sembrerebbero suggerire) come una carrellata nozionistica e necessariamente superficiale sulla struttura e il funzionamento dei componenti hardware di un elaboratore, la loro rilevanza didattica è sicuramente discutibile. Viceversa, partendo dalla definizione di informatica come scienza che ha come oggetto l'elaborazione automatica delle informazioni, diviene chiaro l'obiettivo didattico dell'architettura: la comprensione del funzionamento dei dispositivi fisici in grado di effettuare tali elaborazioni automatiche. Riteniamo che la missione di una didattica delle architetture sia, in sintesi, la comprensione del fatto che l'elaboratore non è un "cervello elettronico" in grado di pensare o scegliere, bensì nient'altro che una gran quantità di porte logiche opportunamente collegate fra loro, ciascuna in grado soltanto di effettuare automaticamente un'operazione logica elementare come NAND o NOR[13].

[13] Un esempio illustre di questo approccio è il corso "From NAND to Tetris"

In virtù di tali considerazioni, la proposta didattica è partita dalla definizione e progettazione della porta logica come dispositivo elementare di elaborazione di informazioni binarie. Si è passati quindi alla definizione di dispositivi via via più complessi (decoder, mux, ALU, flip-flop, registri, ecc.), sempre visti come composizione di porte logiche o di moduli precedentemente sviluppati. Ad ogni definizione è seguita la progettazione pratica del modulo, avvalendosi di strumenti di simulazione di circuiti digitali (Logism, http://www.cburch.com/logisim). Così procedendo, si è giunti a definire la struttura e il funzionamento di un elaboratore e quindi a progettarne uno funzionalmente completo.

4.4.2.3. Didattica dell'informatica

Il fulcro di questo insegnamento è stato quello di ragionare sulla progettazione di esperienze didattiche volte all'insegnamento dell'informatica, sulle modalità con cui queste esperienze possono essere presentate agli studenti e sulle tecniche che permettono di valutare la loro efficacia. Si è posta una specifica enfasi su argomenti che potessero essere proposti agli studenti sfruttando una metodologia attiva di apprendimento, sfruttando la pluriennale esperienza del gruppo dei docenti del corso nell'organizzare workshop introduttivi a argomenti di carattere informatico rivolti a studenti delle scuole secondarie. In particolare, le lezioni hanno riguardato:

- un'introduzione metodologica alla progettazione di percorsi didattici, anche alla luce delle menzionate linee guida e indicazioni nazionali ministeriali;
- la proposta di percorsi didattici attivi legati alla codifica dei testi formattati, della rappresentazione delle immagini, delle macchine di Turing, degli algoritmi greedy, della progettazione di algoritmi e delle tecniche crittografiche di base;
- una serie di suggerimenti didattici legati alla sicurezza informatica, ai sistemi operativi, alle reti e all'utilizzo di tecnologie specifiche per gli studenti con bisogni educativi speciali (BES).

---

http://nand2tetris.org/.

4.4.3. TFA 2014/15

Per la seconda edizione del TFA alla Lombardia vengono assegnati 38 posti, divisi su tre atenei: Università degli Studi di Milano (12), Università degli Studi di Milano – Bicocca (13), Università Cattolica del Sacro Cuore – Sede di Brescia (13). Alla selezione nazionale di luglio partecipano 125 candidati lombardi, di cui risultano sufficienti 68[14]. Alla selezione locale si presentano presso Unimi 21 candidati, di cui 18 vengono giudicati idonei. Il numero finale degli studenti partecipanti ai corsi sarà 17: ai 12 previsti si aggiungeranno 2 "extranumerari" ammessi senza esami e 3 ammessi alla partecipazione con un mese di ritardo in base a una redistribuzione degli idonei (ma esclusi per punteggio) fra i tre atenei effettuata dall'ufficio scolastico regionale. Al momento della scrittura di questo articolo il percorso non si è ancora concluso, ma è già possibile contare un ritiro e uno studente respinto agli esami di profitto.

Questa seconda edizione del TFA è stata profondamente riprogettata, alla luce dell'esperienza accumulata e cercando anche di tenere conto delle indicazioni e linee guida ministeriali. Seguendo le indicazioni del coordinamento d'ateneo volte a uniformare le modalità organizzative fra tutte le classi, la parte disciplinare del corso ha poi dovuto essere divisa in due insegnamenti, rispettivamente da 11 e 7 CFU, ciascuno comprendente 1 CFU esplicitamente rivolto ad attività laboratoriali. Abbiamo perciò deciso di attivare due corsi: "Didattica 1 – Progettazione di sistemi informatici e pensiero computazionale", più legato agli aspetti metodologici della disciplina; "Didattica 2 – Tecnologie per la realizzazione di sistemi informatici", che si focalizza maggiormente sugli aspetti tecnologici.

4.4.3.1. Didattica 1 (Aspetti metodologici)

Il corso "Didattica 1" è organizzato in quattro moduli prevalentemente metodologici: (Programmazione, Didattica della programmazione in laboratorio, Metodi per la didattica del pensiero computazionale, Progettazione di sistemi informatici), che sono senz'altro la parte più originale della nostra offerta formativa.

---

[14] A livello nazionale le sufficienze saranno 683 su 1244 candidati. In questa seconda tornata l'ateneo della regione dove effettuare la prova locale veniva scelto in un secondo momento.

Oltre ai contenuti relativi alla programmazione, già proposti nel PAS, il corso propone strumenti utili a presentare, a partire dal biennio, l'informatica come disciplina scientifica, senza limitarsi agli aspetti strumentali, come invece spesso succede perfino negli istituti a chiaro indirizzo tecnologico. Nei contesti in cui è più fragile la motivazione degli studenti nei confronti dell'informatica (magari proprio a causa di una percezione distorta) suggeriamo di proporre in classe occasioni coinvolgenti di avvicinamento a temi informatici significativi, con una valenza formativa di carattere generale (es: rappresentazione delle informazioni, problem solving, modularità e astrazione, ecc.).

Durante il corso vengono dunque proposti percorsi didattici basati sull'uso di metodologie attive di apprendimento, che mettano gli allievi in condizione di esplorare in prima persona il tema in questione. Lavorando ad esempio suddivisi in piccoli gruppi, o partendo da materiali o contesti non convenzionali, il tema potrà essere indagato sotto diversi punti di vista, si potranno costruire modelli interpretativi e fare ipotesi che possano essere messe alla prova nel contesto guidato dell'attività proposta. I percorsi didattici proposti sono frutto di un lavoro di progettazione e sperimentazione in numerose scuole che il gruppo ALaDDIn svolge con regolarità dal 2012 [1].

In coerenza con quanto proponiamo di fare in classe, il corso stesso è sviluppato usando questo metodo di insegnamento: gli abilitandi sono quindi chiamati a farsi coinvolgere in maniera partecipe nelle attività proposte e nelle discussioni che ne seguono, che tipicamente si concentrano sul ruolo che l'insegnante deve avere in classe durante lo svolgimento di tali attività: da trasmettitore di conoscenza, a facilitatore/mediatore nel processo cognitivo di ciascun allievo.

Gli stessi docenti del corso sono titolari da quest'anno anche di un insegnamento di "Didattica dell'informatica" attivato presso il corso di laurea magistrale in informatica, e alcuni dei laboratori proposti agli abilitandi del TFA sono stati seguiti anche dagli studenti di laurea magistrale. L'iniziativa si è rivelata un successo: oltre ad aumentare il numero degli studenti, che diventa, grazie all'inserimento di una decina di nuovi frequentanti[15], simile a quello di una classe delle scuole

[15] Hanno partecipato alle lezioni, come uditori, anche tre professori di ruolo nella scuola secondaria di primo grado.

superiori rendendo particolarmente realistico il lavoro di gruppo, l'interazione fra docenti in formazione e studenti freschi di competenze disciplinari si è rivelata molto positiva.

Il corso di Didattica 1 include inoltre un modulo di progettazione di sistemi informatici, utile in particolare a fornire spunti e riflessione didattica per i corsi di Gestione di progetto e organizzazione d'impresa, cui sono chiamati (in esclusiva!) i docenti 42/A. Il modulo, tenuto da un docente esperto di ingegneria del software, dopo una classica presentazione delle qualità peculiari del software e che influenzano particolari scelte nella gestione dei processi di sviluppo, si è focalizzato sulle metodologie "agili" in quanto ritenute portatrici di elementi didatticamente interessanti per introdurre i problemi della pianificazione e gestione delle attività di sviluppo. In particolare due elementi tipici caratterizzanti le metodologie agili, quali la struttura poco gerarchica con figure interscambiabili e la autorganizzazione dei team nella schedulazione e pianificazione di processi iterativi di sviluppo, possono essere a nostro avviso applicati efficacemente anche in lavori di gruppo da proporre agli studenti. È stato inoltre realizzato un laboratorio "agile" specifico per comprendere e sperimentare tecniche collaborative di stima dei tempi e modalità correttive in itinere delle stime stesse.

4.4.3.2. Didattica 2 (aspetti tecnologici)

Il corso di Didattica 2 è organizzato in tre moduli più tecnologici: Architettura dei sistemi informatici, Sistemi operativi e reti, Basi di dati.

La maggiore novità in questo caso è l'introduzione delle Basi di dati e la conseguente riduzione della parte di Architetture, che le linee guida di fatto eliminano dal dominio dei docenti 42/A. Riteniamo che sia comunque importante trasmettere il legame indissolubile che l'elaborazione delle informazioni ha con lo strumento di calcolo e abbiamo deciso perciò di mantenere almeno una piccola parte in cui si discute come da dispositivi puramente combinatori è possibile derivare elementi sequenziali.

4.4.4. Considerazioni

La prima esperienza del TFA e quella del PAS, in definitiva, sono state molto deludenti: tralasciando le difficoltà gestionali, va notato

che il considerevole sforzo progettuale è stato proposto a pochi abilitandi (solo quattro nel caso del TFA), tutti già inseriti nel sistema scolastico. Sulla carta questi percorsi formativi dovrebbero corrispondere a un impegno a tempo pieno: lezioni, tirocinio diretto e indiretto e attività di scrittura della relazione finale, compresse in un arco temporale limitato, sarebbero più che sufficienti a saturare le energie degli studenti. Nei fatti, il caso più comune è che gli abilitandi affrontino il percorso di abilitazione senza rinunciare all'insegnamento che molti di loro già praticano: l'alternativa sarebbe sommare alla non irrisoria tassa d'iscrizione (circa 2500 EUR) il mancato stipendio. Si capisce quindi come una delle maggiori fonti di tensione fra studenti e docenti sia il tema del riconoscimento di attività pregresse, volto a ridurre l'obbligo di attività TFA. Il risultato è che un'offerta formativa pensata con obiettivi complessivi coerenti viene fruita in modo frammentato, spesso accompagnata da un contenzioso strisciante capace di frustrare la buona volontà di tutti.

L'esperienza del TFA 2014/15, ancora in corso, invece ci appare soddisfacente, forse anche grazie ad un meccanismo di selezione più efficace: gli abilitandi hanno dimostrato in generale competenze disciplinari pregresse più solide e maggiore motivazione e impegno alla riflessione didattico-metodologica.

4.5. Conclusioni

La prospettiva dell'insegnamento dell'informatica nella scuola secondaria italiana dal punto d'osservazione della formazione nella classe 42/A desta più di una preoccupazione. Da una parte indicazioni e linee guida mancano di chiarire senza ambiguità il ruolo delle discipline informatiche nella scuola secondaria, confondendo aspetti scientifici e tecnologici con il piano più meramente strumentale e di servizio: siamo ben lontani da una formulazione coerente dell'informatica capace di chiarire il suo "influsso sui metodi delle scienze e delle tecnologie, e su come permetta la nascita di nuove scienze". In alcuni casi si ha l'impressione che gli estensori di programmi e mansionari concepiscano l'informatica come un'accozzaglia di parole chiave, il cui insegnamento può essere affidato a chiunque abbia avuto a che fare con un computer (tutti, al giorno d'oggi).

Dall'altra parte c'è poi il problema della preparazione del personale docente: specie nelle zone in cui le aziende del settore informatico assorbono senza problemi tutti i tecnici di valore, il rischio che si dedichino all'insegnamento elementi scartati dal mondo del lavoro purtroppo è alto.

Non bisogna disperare, però, e negli anni ci sembra di avere messo a punto una proposta che, nel rispetto di vincoli normativi non sempre coerenti né del tutto condivisibili, crediamo sia in grado di rispondere alle esigenze di formazione per una classe di concorso cui sono affidati incarichi didattici decisamente variegati.

Bibliografia

[1] BELLETTINI, C., LONATI, V., MALCHIODI, D., MONGA, M., MORPURGO, A.,TORELLI, M., *Exploring the processing of formatted texts by a kynesthetic approach,* pp. 143-144., 2012.

[2] BELLETTINI, C., LONATI, V., MALCHIODI, D., MONGA, M., MORPURGO, A., TORELLI, M., ZECCA, L., *Extracurricular activities for improving the perception of Informatics in Secondary schools,* 8730 pp. 161-172, 2014.

[3] BELLETTINI, C., LONATI, V., MALCHIODI, D., MONGA, M., MORPURGO, A., TORELLI, M., ZECCA, L., *Informatics Education in Italian Secondary School,* ACM Transactions on Computing Education 14: 15:1-15:6., 2014.

[4] HROMKOVIVC, J., *Contributing to General Education by Teaching Informatics,* pp. 25-37, 2006.

[5] *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'art. 10, comma 3, del DPR 15/3/2010, n. 89, in relazione all'art. 2, commi 1 e 3, del medesimo regolamento,* 2010.

[6] *Istituti professionali, linee guida per il passaggio al nuovo ordinamento,* http://www.indire.it/lucabas/lkmw_file/nuovi_professionali/linee_guida/_LINEE%20GUIDA%20ISTITUTI%20%20PROFESSIONALI_.pdf, 2010.

[7] *Istituti tecnici, linee guida per il passaggio al nuovo ordinamento,* http://www.indire.it/lucabas/lkmw_file/nuovi_professionali/linee_guida/_LINEE%20GUIDA%20ISTITUTI%20%20PROFESSIONALI_.pdf, 2010.

[8] LISSONI, A., LONATI, V., MONGA, M., MORPURGO, A.,TORELLI, M., *Working for a leap in the general perception of computing,* pp. 134-139, 2008.

[9] LONATI, V., MALCHIODI, D., MONGA, M., MORPURGO, A., *Is coding the way to go?,* 2015.

[10] LONATI, V., MONGA, M., MORPURGO, A.,TORELLI, M., *What's the Fun in Informatics? Working to Capture Children and Teachers into the Pleasure of Computing,* 7013 pp. 213-224, 2011.

[11] MIROLO, C., *Quale informatica nella scuola?,* 2003.

[12] THE ROYAL SOCIETY, *Shut down or restart? The way forward for computing in UK schools,* 2012.

Ringraziamenti

Gli autori desiderano ringraziare Ottavio D'Antona, Alfio Ferrara, Massimo Santini, colleghi docenti di corsi TFA e PAS per la classe 42/A presso l'Università degli Studi di Milano e i tutor che hanno collaborato con ALaDDIn nella messa a punto e nella sperimentazione dei percorsi didattici svolti presso le scuole.

5. Informatica per la classe A033: ricadute sull'aggiornamento degli insegnanti del primo ciclo

G. Barbara Demo[1]

Le Indicazioni nazionali per la disciplina Tecnologia nella scuola del primo ciclo (pubblicate nel novembre 2012) recitano "Quando possibile, gli alunni potranno essere introdotti ad alcuni linguaggi di programmazione particolarmente semplici e versatili che si prestano a sviluppare il gusto per l'ideazione e la realizzazione di progetti (siti web interattivi, esercizi, giochi, programmi di utilità) e la comprensione del rapporto che c'è tra codice sorgente e risultato visibile" [10]. Gli insegnanti della disciplina citata, abilitandi o in servizio, hanno spesso un discreto livello di alfabetizzazione digitale ma per lo più non hanno competenze informatiche. Questo rende poco probabile che si possano realizzare le Indicazioni. Viene qui descritto un corso di Informatica che ha avvicinato alla programmazione gli abilitandi TFA e PAS della classe A033 (Tecnologia per le scuola secondaria di primo grado) con esito molto positivo. Chiavi del successo sono state l'uso di un ambiente di sviluppo programmi semplice come Scratch, e l'esperienza maturata nel gruppo di lavoro Teachers for teachers (T4T) da anni attivo presso il Dipartimento di Informatica dell'Università di Torino. T4T ha come obiettivo l'aggiornamento degli insegnanti coinvolgendoli in attività proponibili direttamente in aula e con contenuti interessanti per varie discipline. Tali attività multidisciplinari si propongono di catturare l'interesse di più insegnanti nelle singole scuole secondarie di secondo grado convincendoli ad un graduale aggiornamento delle loro competenze informatiche infine diffondendo il pensiero

[1] Dipartimento di Informatica, Università di Torino.

computazionale. La proposta è stata adottata anche in successive iniziative di aggiornamento che hanno suggerito la messa a punto di declinazioni diverse delle proposte di cui si dà brevemente conto.

5.1. Introduzione

Nelle discussioni sull'informatica nella scuola è ormai accettata la distinzione tra informatica e alfabetizzazione digitale quali sono definite ad esempio nel rapporto congiunto ACM e Informatics Europe pubblicato nell'aprile 2013 [6] o nel rapporto della Académie des Sciences pubblicato nel maggio dello stesso anno [1]. Nel primo rapporto si legge: "Alfabetizzazione digitale è saper usare dei programmi e navigare in internet. Invece informatica è la scienza su cui si fonda la tecnologia del digitale. Informatica è una scienza a sé, caratterizzata da propri concetti, metodi, corpo di conoscenze e problemi aperti. È emersa in un ruolo simile a quello della matematica, come un'area interdisciplinare che oggi contribuisce ai progressi scientifici, dell'ingegneria e dell'economia". Sempre il primo dei rapporti citati auspica che l'alfabetizzazione digitale raggiunga un buon livello entro il primo anno della scuola secondaria di primo grado e che le competenze riguardo l'informatica siano costruite durante l'intero iter scolastico con un processo e con contenuti adatti, naturalmente, alle diverse età ed esperienze degli studenti. A proposito delle diverse accezioni con cui il termine informatica viene percepito si veda anche il "Manifesto dell'Informatica per la scuola" scritto a più mani da ricercatori GRIN e GII nel maggio 2010 e pubblicato in [8].

Varie nazioni europee sono attivissime nella definizione di curricula scolastici che integrino competenze informatiche nei vari livelli e tipi di scuole. L'iniziativa inglese CaS (Computing at School) ha trascinato il grande aumento di attività informatiche nelle scuole inglesi http://www.computingatschool.org.uk/,. In tutte le nazioni si sta discutendo il problema: nei ministeri, nelle scuole, nelle associazioni di informatici e insegnanti, sui media.

Attualmente il problema impellente da risolvere in Italia e in altri paesi è quello dell'aggiornamento degli insegnanti di ruolo e di quelli che, già laureati, ambiscono insegnare. In Italia presso le varie università si sono svolti nel 2013 i corsi TFA (Tirocini Formativi Attivi), nel 2014 i corsi per i PAS (Percorsi Abilitanti Speciali) e sono ancora in

corso successive edizioni dei corsi che avevano un alto numero di studenti. Tra le classi di concorso per cui hanno preso il via i corsi disciplinari e trasversali troviamo la classe A033 che riguarda i docenti della disciplina Tecnologia (l'insegnamento un tempo chiamato "Applicazioni tecniche") insegnata nella scuola secondaria di primo grado. Per il Piemonte nei corsi del TFA 2012/13 si è trattato di circa un centinaio di professori, per lo più con laurea magistrale in Architettura, qualche ingegnere civile o ambientale, e qualche altra laurea ammessa a quel tipo di insegnamento. Analoga provenienza ma più di trecento iscritti per i corsi PAS ancora in corso (distribuiti su tre anni proprio per il numero di studenti). Il Politecnico di Torino, affiancato dalle Università di Torino e del Piemonte orientale, ha avuto il compito di organizzare i corsi disciplinari afferenti al tipo di argomenti trattati dalla disciplina Tecnologia con i contenuti che si evincono dalle linee guida del MIUR.

La grossa novità è stata quella che, tra gli altri corsi disciplinari, fin dal TFA 2012/13 si è scelto di organizzare un corso di 40 ore di Didattica e laboratorio dell'Informatica. Una scelta innovativa, isolata dalle scelte operate a livello nazionale, tenendo conto che le linee guida ministeriali della disciplina Tecnologia per la scuola media inferiore recitano "Quando possibile, gli alunni potranno essere introdotti ad alcuni linguaggi di programmazione particolarmente semplici e versatili che si prestano a sviluppare il gusto per l'ideazione e la realizzazione di progetti (siti web interattivi, esercizi, giochi, programmi di utilità) e la comprensione del rapporto che c'è tra codice sorgente e risultato visibile" [10]. Dunque è previsto che, tra le tecnologie, si studi anche l'Informatica, pur tenendo conto della condizione dei laboratori in molti istituti scolastici più poveri e del fatto che, come gli abilitandi hanno evidenziato più volte durante le lezioni, si è passati da un monte ore di 3 ore settimanali a sole 2 ore settimanali con il conseguente taglio dei contenuti trattati in classe. Questi tagli normalmente finiscono per escludere le già poche attività di informatica, anche perché sono rare le iniziative di aggiornamento dei docenti che hanno spesso un discreto livello di alfabetizzazione digitale mentre per lo più non hanno competenze informatiche.

Il programma del corso di Didattica e laboratorio dell'Informatica è stato definito tenendo conto anche delle esperienze condotte da vari anni per la diffusione del pensiero computazionale nelle scuole dalle

elementari alle secondarie di II grado (dette k-12 nel mondo anglo-sassone) realizzate col progetto Teachers for teachers (T4T) del Dipartimento di Informatica dell'Università di Torino, t4t.di.unito.it [5]. Per contribuire in modo concreto all'aggiornamento dei futuri insegnanti si è deciso di focalizzare il corso sull'insegnamento dell'Informatica. Dunque non alfabetizzazione digitale con insegnamento ai futuri docenti dell'utilizzo delle applicazioni informatiche più comuni (videoscrittura, foglio di calcolo, presentazioni, ecc.), bensì introduzione dei concetti di base della programmazione perché questa possa essere offerta agli alunni della scuola secondaria di primo grado come un nuovo strumento con cui esprimere la loro creatività. Una sfida assolutamente non facile da vincere per vari motivi ardui da superare: da una parte ci sono le perplessità, la diffidenza e i dubbi dei docenti di Tecnologia nell'affrontare un argomento per loro nuovo che si differenzia totalmente dal loro modo di insegnare l'Informatica intesa come puro addestramento all'uso del computer. Dall'altra dobbiamo affrontare il timore delle difficoltà che si possono presentare nell'introdurre concetti complessi ed astratti a giovanissimi studenti (nativi digitali, ricordiamolo) nel pochissimo tempo-scuola a disposizione. Ma una sfida altrettanto importante da vincere per chi crede nel ruolo dell'Informatica nell'istruzione di base, come ci viene ormai detto e ripetuto da tempo nelle linee guida per la scuola tratte dalle direttive europee e a cui molti paesi dell'Unione Europea si sono poi adeguati adattando o cambiando i loro curricoli scolastici. Volendo seguire queste linee guida si è riproposto e ancora si sta proponendo lo stesso approccio adottato nei corsi TFA e PAS per la classe A033 nel già menzionato progetto T4T, in incontri di aggiornamento di insegnanti del primo ciclo (per esempio insegnanti nelle scuole coinvolte nel progetto Scuola 2.0 del Comune di Torino) e in insegnamenti dei corsi di laurea in Scienze dell'Educazione dell'università di Torino.

Nel paragrafo 5.2 si dice di come nelle scuole, in particolare del primo ciclo, si abbia ancora una situazione molto carente riguardo il digitale in genere per mancanza di aggiornamento e quindi si sottolinea la necessità di avviare iniziative che favoriscano l'aumento delle competenze digitali. Segue il paragrafo 5.3 dedicato a Scratch come nuovo strumento usando il quale i bambini esprimono la loro creatività. Si descrivono in breve varie attività di avvio degli insegnanti all'uso di Scratch. Il paragrafo 5.4 raccoglie soprattutto esempi di come

la proposta, ripresa in altre iniziative di aggiornamento di più breve durata, sia stata adattata ricorrendo a esempi di algoritmi in esperienze quotidiane che per la loro familiarità rendono in genere più rapido avvicinarsi all'informatica per chi non l'abbia mai frequentata, e rendono più facile entrare in sintonia con nuove formulazioni superando parte degli eventuali problemi. Infine sono citati esempi di come sono stati introdotti anche concetti basilari dell'informatica.

5.2. Situazione nelle scuole del primo ciclo

Normalmente nelle scuole secondarie di primo grado non ci sono tecnici (come invece spesso troviamo nelle secondarie di secondo grado, anche se non è previsto l'insegnamento dell'informatica). Questo fa sì che spesso nelle scuole ci siano problemi di varia natura rispetto alla presenza del digitale in senso lato. Si sono sentiti giudizi molto negativi sugli eccessivi costi dell'abbonamento alla rete di una scuola o del tipo di contratto con un'azienda di consulenza informatica. Tra l'altro può esserci il pericolo che scuole con contratti di consulenza con società private si trovino in situazioni di pura illegalità qualora, rilevato l'eccessivo costo, il contratto venga rescisso. In un tal caso la società privata potrebbe non concedere alla scuola qualcuno dei prodotti digitali che gestiva per conto della scuola: ci riferiamo per esempio al sito web ma anche a documenti più delicati quali registri elettronici, pagelle o simili.

Spesso altrettanto negativamente è giudicato lo scarso uso di strumenti digitali costosi lasciati diventare obsoleti. Accade che questi giudizi siano accompagnati da osservazioni sullo spreco di denaro pubblico che essi rappresentano: "fossero denari privati nulla da eccepire, ma essendo risorse pubbliche…" è una facile conclusione.

Conclusione facile ma con, a sua volta, delle pecche, specie se venisse da qualcuno che all'interno di una amministrazione pubblica, o vicino a quella, potrebbe avere ruolo nel concorrere a formare (gruppi di) esperti cui le scuole ed i loro dirigenti o personale in genere possano rivolgersi per le situazioni cui si è accennato. In questo caso lo sperpero di denaro pubblico sarebbe doppio: all'interno delle scuole e all'interno dell'amministrazione, che non ha provveduto a mettere anche le scuole del primo ciclo nelle condizioni di operare senza sprechi e aiutarle a dotarsi di soluzioni tecniche favorevoli.

Noi crediamo ci debba essere partecipazione da entrambe le parti a migliorare la situazione: di chi è nelle scuole e di chi sta fuori delle scuole ma nella pubblica amministrazione. Le scuole però possono contribuire soltanto se aumentano le competenze digitali di chi opera al loro interno, in modo che questi possa andare a cercare informazioni quando necessario sapendo cosa chiedere, possa quindi riconoscere situazioni negative, sappia quali aspetti debbano fare parte, per esempio, di contratti. Devono aumentare le competenze digitali di chi opera nella scuola sia in quanto a contenuti didattici sia quanto ad aspetti amministrativi perché le due componenti vanno di pari passo, si stimolano positivamente in modo vicendevole.

Uno dei motivi che venivano addotti in Francia alla introduzione della disciplina Science du numérique nelle scuole secondarie di II grado era preparare chi sarebbe stato chiamato in Parlamento a decidere su leggi inerenti il digitale. E si faceva l'esempio di una legge in discussione qualche anno fa. Riteniamo che un tale ragionamento sia da estendere almeno alla preparazione di chi potrà trovarsi ad operare nella amministrazione pubblica in genere: che vuol dire in realtà estenderlo alla preparazione di tutti gli studenti.

Ma torniamo alla preparazione o all'aggiornamento degli insegnanti e torniamo ad occuparci, nei paragrafi che seguono, delle competenze digitali degli insegnanti per gli aspetti inerenti la didattica.

5.3. Forniamo agli studenti un nuovo strumento per esprimere la loro creatività

La programmazione non è l'unica competenza informatica fondamentale cui dovrebbero essere introdotti un po' tutti ma, a patto di usare strumenti opportuni, è la più caratteristica, anche per chi non ha nessuna o pochissime conoscenze di informatica, e insieme la più concreta, facile e anche gradevole per introdurre al pensiero computazionale (http://www.programmaIlFuturo.it). Può essere infatti acquisita in modo "amichevole" perché permette di produrre risultati soddisfacenti con poco sforzo e anche utili: basta usare ambienti di sviluppo programmi adatti allo scopo.

Scratch è uno di questi strumenti: è un ambiente di programmazione creato con l'obiettivo di introdurre gli studenti nella fascia dell'obbligo scolastico ai concetti di base della programmazione e del

problem solving attraverso uno strumento che ad un primo approccio colpisce per l'aspetto ludico ma che in realtà esercita i discenti alla logica e al ragionamento. Infatti permette di elaborare variabili e liste di valori, offre controlli per la selezione e l'iterazione di istruzioni con cicli di vario tipo, permette di realizzare animazioni via via più complesse, offre la possibilità di far eseguire più processi contemporaneamente e di farli interagire, permette l'implementazione del paradigma ad eventi e molto altro ancora [2, 9, 11].

Scratch è un ambiente di sviluppo programmi in cui produrre attività molto diverse [4]. È costruito in modo da invitare a scoprire le varie azioni che si possono fare e come si può aumentare la conoscenza del sistema: in genere è bene cominciare gli incontri facendo vedere agli studenti una attività semplice come la Farfalla-Blu in Fig. 5.1. Se necessario incoraggiamoli a non aver timore di pigiare i vari tab per vedere a cosa conducono (sarà buona abitudine per il digitale in genere) e a provare i comandi che in Scratch possono essere eseguiti anche uno per volta o in brevi sequenze separate dal resto per vederne subito l'effetto. Si possono creare storie con uno o più personaggi che agiscono su un palcoscenico con uno, in genere più, fondali e suoni di vario tipo (voci, musiche, rumori) [12]. I personaggi hanno comportamenti definiti attraverso programmi quindi sequenze di codice, in ambiente Scratch chiamati script (cioè "parte" nel senso in cui il termine parte è usato in teatro). La specifica del codice avviene impilando blocchi-comando di forma e colore diverso, a seconda della funzione e della categoria di appartenenza, che vanno ad incastrarsi (vedi Fig. 5.1) come nel gioco dei mattoncini Lego.

Naturalmente i personaggi ed i comportamenti degli attori possono essere i più diversi: si può raccontare un viaggio, un episodio accaduto a chi racconta (quelle che chiamiamo "personal stories") o possono essere le istruzioni per fare qualcosa tipo "istruzioni per l'uso" [3]. Oppure possono proporre risoluzioni di problemi di vario tipo e quindi apparire più vicini ai programmi dell'informatica tradizionale: negli incontri introduttivi è bene ricorrere molto raramente a questo tipo di attività che comunque può essere utile per presentare o consolidare concetti fondamentali del pensiero computazionale.

Fig. 5.1. Farfalla-Blu.

Scratch è scaricabile gratuitamente dal sito scratch.mit.edu, stabile, è un ambiente versatile e ideale per lo sviluppo di applicazioni ludiche, animazioni grafiche, ipertesti ma anche per la realizzazione delle classiche applicazioni che si sviluppano quando si impara la programmazione da zero. Il sito di Scratch ospita manuali gratuiti e in molte lingue, gallerie di progetti, materiali informativi, video esplicativi, forum di discussione, e permette di scaricare più di 4 milioni progetti completamente gratuiti con licenza Creative Commons o di caricare i propri progetti condividendoli con gli altri utenti sparsi per il mondo. Componente importante del mondo Scratch è proprio la comunità in cui chi è agli inizi trova attività altrui da vedere e provare per trarre ispirazione, magari poi modificare per farle diventare una propria storia.

Nella scuola del primo ciclo (ed in particolare nella scuola primaria) è importante pensare al digitale come a uno degli strumenti che forniamo ai bambini per esprimere la loro creatività. A questo riguardo è molto utile il documento preparato da Martina Kabatova e Katarína Mikolajová per il workshop “Fostering creativity through programming” [7] da loro curato durante ISSEP 2011.

All'indirizzo http://wiki.scratch.mit.edu/wiki/Scratch_Support_Materials si trovano un manuale di introduzione e le ben note "Carte Scratch per cominciare" tradotti da Alessandro Rabbone autore anche del blog http://bambinicheimparanoaprogrammare.blogspot.it/ da cui si può risalire alle tante attività condivise negli anni dalle sue classi sul sito https://scratch.mit.edu/.

5.4. Le esperienze e gli algoritmi di tutti i giorni

Dopo un'analisi dell'ambiente di sviluppo di Scratch, semplice ed amichevole, colorato e stimolante, agli abilitandi dei TFA e dei PAS A033 sono state illustrate attività concordate tra i formatori che li hanno accompagnati nell'imparare ad utilizzare lo strumento e nel capire le basi della logica degli algoritmi. Dal classico "story telling", visto in diverse modalità di realizzazione, si è poi passati allo sviluppo di programmi per la costruzione di questionari, passando per animazioni grafiche di vario tipo sino allo sviluppo di videogiochi e di programmi di utilità varia [3].

In attività di aggiornamento di minor durata del corso di Informatica nel TFA A033 (le più frequenti) è importante che nel primo incontro si sia molto pratici per appassionare gli insegnanti anche perché gli aggiornamenti sono attività volontarie che in genere non portano a particolari riconoscimenti. A questo obiettivo concorre bene:

- il far vedere subito una attività Scratch semplice come la Farfalla-Blu di Fig. 5.1 e chiedere di cambiare frasi e/o personaggi: questo permette di vedere l'ambiente già facendo qualcosa,
- poi chiedere di comporre una attività usando un certo insieme di comandi, per esempio otto comandi scelti tra quelli che possono produrre un effetto estetico piacevole (per esempio cambiare colore ad uno sprite o farlo ruotare)
- trasporre in Scratch attività quotidiane come la ScuolaPane-o-GiochiCasa in cui si riproduce in Scratch una attività di tutti i giorni (con una selezione tra due differenti attività) che quindi chi partecipa conosce bene e sa leggere anche quando la vede formulata in uno script (Fig. 5.2).

Analogamente alla descrizione di attività quotidiane sono importanti anche esempi di algoritmi che si conoscono dalla scuola primaria quali l'esecuzione di operazioni tra numeri a più cifre o, di nuovo, quelli che usiamo in esperienze quotidiane, ad esempio trovare il minimo in un insieme di valori. È un algoritmo di questo tipo lo scegliere il prezzo minore tra prodotti dello stesso genere quando si va a fare la spesa. Per la loro familiarità questi algoritmi rendono più facile avvicinarsi all'informatica per chi non l'abbia mai frequentata, poiché rendono più facile entrare in sintonia con nuove formulazioni superando almeno parte degli eventuali problemi.

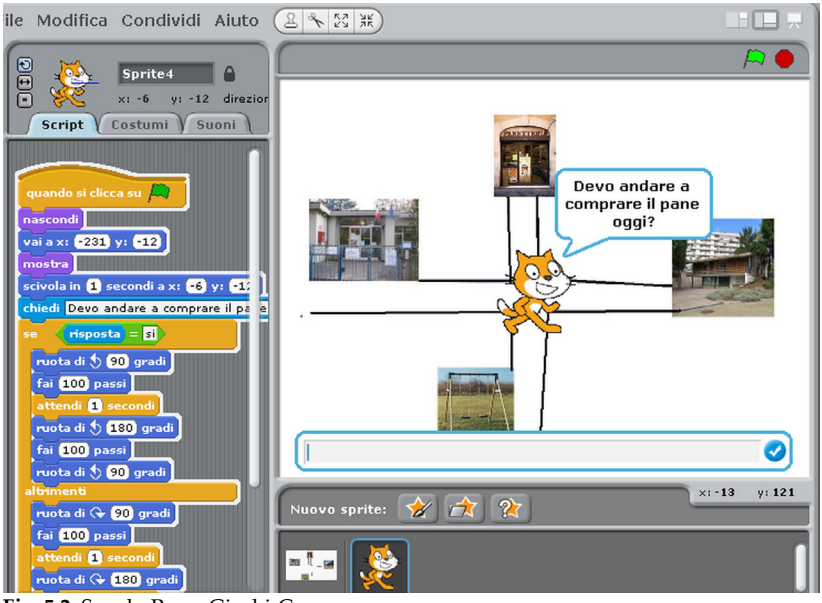


Fig. 5.2. Scuola-PaneoGiochi-Casa.

Con Scratch i corsisti abilitandi della classe A033 così come i partecipanti alle altre iniziative hanno avuto modo di essere introdotti alla programmazione e all'utilizzo di un semplice linguaggio di programmazione e sperimentare l'ideazione e la realizzazione di progetti, comprendendo il rapporto esistente tra codice sorgente e risultato visibile, proprio come richiesto dalle linee guida.

Nel caso del corso Informatica nel TFA A033 in poche lezioni la diffidenza e le perplessità della maggior parte dei corsisti sono state superate e si è passati da un clima in laboratorio teso o, comunque, poco sereno a un clima più rilassato e partecipe. I dubbi che potevano esserci all'inizio del corso sono stati per lo più fugati man mano che si procedeva e sostituiti da interesse per approfondimenti e curiosità. Alcuni docenti più temerari hanno voluto e potuto già in corso d'opera sperimentare nelle loro classi le esercitazioni loro proposte per registrare le reazioni dei loro studenti. Nella maggior parte dei casi queste reazioni sono state positive: il che ha dato ancor maggior fiducia ai docenti nell'uso e nelle potenzialità dello strumento informatico. Inoltre i corsisti hanno condiviso le reazioni dei loro studenti in brevi dibattiti con interventi sulle modalità didattiche utilizzate allo scopo.

5.5. Introduzione di concetti fondamentali

Si noti che sono stati introdotti anche concetti quali la complessità degli algoritmi discutendo esercizi che adottano strategie quali la ricerca binaria in giochi da proporre in classe come il gioco "Indovina il numero (più alto più basso)" (vedi Fig. 5.3).

Fig. 5.3. AltoBasso.

Come si è detto Scratch è stato molto ben accolto dai corsisti che hanno apprezzato soprattutto:

- l'uso di un ambiente semplice, ma attraverso il quale si ottengono risultati gratificanti,
- la proposta di attività da presentare direttamente in aula e con contenuti interessanti per le altre discipline e quindi per i colleghi insegnanti dei medesimi studenti.

Esempio di attività interdisciplinare sono i programmi "pensa un numero ed io lo indovino" in cui ogni gruppo di allievi inventa il proprio indovinello attraverso un'attività sperimentale su equazioni lineari (Fig. 4). L'abbiamo anche indicata come un'attività in cui gli studenti inventano, realizzano e risolvono una equazione: processo che diventa un indovinello quindi una attività costruttiva. Con questa esperienza gli studenti manipolano un concetto che normalmente vedono soltanto scritto sul libro di testo come esercizio predefinito.

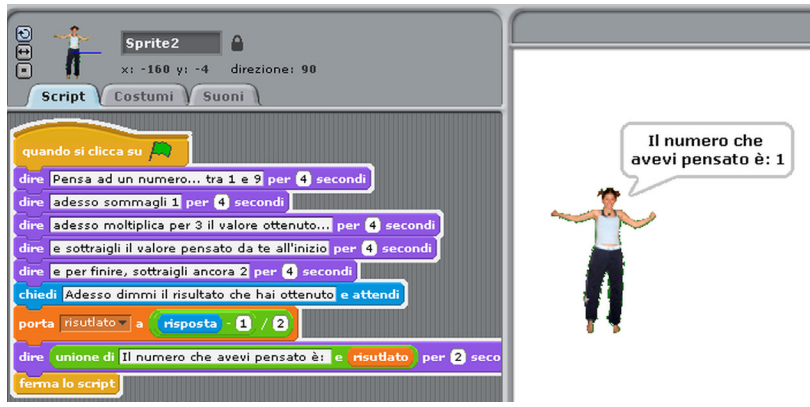


Fig. 5.4. Indovina un numero.

Una discussione con i corsisti ha poi messo in evidenza come un tipo di esercizio molto semplice quale un quiz possa da una parte avere valenza informatica nell'introduzione graduale e motivata all'uso delle variabili, ma anche proporre modalità di apprendimento originali. Per esempio ogni gruppo di allievi crea un proprio quiz dove vengono proposte domande sugli aspetti di una lezione che hanno più interessato il gruppo. Gli altri gruppi possono giudicare della mancanza di attenzione per altri aspetti e quindi comporre insieme una visione integrata dell'argomento. Questi sono esempi tra tanti di attività che favoriscono il coinvolgimento degli insegnanti delle altre discipline della scuola secondaria di primo grado, quindi, il progressivo aggiornamento delle competenze informatiche anche di questi docenti: coinvolgimento essenziale se si vuole arrivare ad utilizzare il digitale come strumento attraverso il quale imparare in modo attivo e costruttivo le varie discipline.

5.6. Conclusioni

In questo capitolo si sono raccolte le considerazioni che hanno portato ad includere un corso di Informatica per gli studenti del TFA A033. Si è dato conto dell'uso di Scratch per introdurre alla programmazione futuri insegnanti della disciplina Tecnologia (della scuola secondaria di primo grado) con nessuna o pochissime competenze informatiche. I responsabili del TFA avevano già proposto l'uso di Scratch in contesti molto diversi e hanno continuato a farlo: in attività con

bambini del quarto e quinto anno della scuola primaria, con ragazzi e insegnanti della scuola secondaria di primo grado, con studenti del primo biennio di istituto tecnico.

In ognuno di questi contesti Scratch si è rivelato uno strumento adatto a stimolare la voglia di proseguire, adatto a produrre in relativamente poco tempo materiale utile in classe o sentito come prodotto finito e da far vedere agli altri dagli studenti. Nel contempo è strumento usando il quale si possono introdurre elementi fondamentali dell'informatica ed arrivare ad una competenza di programmazione che rende più veloce e sicuro passare all'utilizzo di altri linguaggi ed ambienti in un percorso evolutivo di acquisizione delle competenze informatiche che vengono da più parti indicate come indispensabili per gli studenti e non soltanto.

Le esperienze descritte sono state proposte in diverse attività di aggiornamento per insegnanti o direttamente nelle classi. Gli insegnanti con pochissime o nessuna conoscenza di informatica sono stati accompagnati a vedere un poco tutte le facce di Scratch perché potessero poi da sé adattare le attività ai loro studenti. Gli insegnanti di informatica hanno imparato che si possono raccontare storie, pensare come indovinello la ricerca binaria e quasi si sono stupiti quando hanno avuto allievi che hanno reso come storia anche trovare il minimo e il massimo di un insieme di numeri ("Il vincitore del gran premio"). Gli insegnanti di matematica hanno costruito insieme agli studenti attività in cui si sperimentano parti della programmazione didattica, ad esempio le equazioni ad una variabile, o semplicemente indovinelli di "allenamento" per esempio al calcolo, come hanno fatto insegnanti di altre discipline per le loro materie. A tutte le età l'uso di Scratch ha permesso di iniziare a vedere il calcolatore e la programmazione come uno strumento per esprimere l'inventiva di ciascuno.

È infine da evidenziare che il corso qui descritto si presta non soltanto ad un avvio alla programmazione ma piuttosto ad "introdurre all'informatica" poiché risulta naturale contestualizzare le varie attività introducendo, pur brevemente, cosa è un sistema operativo, cosa vuol dire linguaggio formale e di programmazione, cosa vuol dire efficienza di calcolo (e complessità) e simili. Si vedano a questo proposito i materiali didattici per il PAS pubblicati in http://pas.i-learn.unito.it/. Gli esercizi sono invece disponibili all'indirizzo: http://scratch.mit.edu/users/barbero/.

Al termine dei corsi il giudizio dell'esperienza non può che essere positivo con la certezza che Scratch, "quando possibile", entrerà a far parte degli strumenti usati nel primo ciclo anche grazie ai docenti di Tecnologia che, dopo averlo sperimentato con i ragazzi, ne hanno colto la validità da molti punti di vista: un buon passo in avanti per la scuola italiana.

Bibliografia

[1] ACADEMIE DES SCIENCES, *L'enseignement de l'informatique en France - Il est urgent de ne plus attendre,* maggio 2013.

[2] ARMONI, M., BEN-ARI, M., *Computer Science Concepts in Scratch,* free book can be download from http://stwww.weizmann.ac.il/g-cs/scratch/scratch_en.html; http://www.academie-sciences.fr/activite/rapport/rads_0513.pdf.

[3] BARBERO, A., DEMO, G. B., *Informatica per le scuole secondarie di primo grado,* Atti DIDAMATICA 2014, Napoli, Maggio 2014.

[4] DEMO, G. B., WILLIAMS. L., *The Many Facets of Scratch,* Atti Convegno ISSEP, Istanbul, LNCS 8730, Springer, Settembre 2014.

[5] DEMO, G. B., *T4T: A Model for In-service Teachers' Training,* in Atti del Convegno WiPSCE 2014, Berlin, Novembre 2014.

[6] GANDER, D. ET AL., *Informatics Education. Europe cannot afford to miss the boat,* Joint Report ACM & Informatics Europe, aprile 2013. http://europe.acm.org/iereport/index.html.

[7] KABATOVA, M., MIKOLAJOVA, K., *Fostering creativity through programming,* Scratch workshop, ISSEP 2011, Bratislava, http://www.issep2011.org/files/workshops/w07.pdf.

[8] MANIFESTO DELL'INFORMATICA PER LA SCUOLA (SECONDARIA) //www.grininformatica.it/opencms/opencms/grin/infoescuola/riformascuola/.

[9] MEERBAUM-SALANT, O., ARMONI, M., BEN-ARI, M., *Learning computer science concepts with Scratch,* ICER '10 Proceedings of the Sixth international workshop on Computing education research, pp 69-76, ACM New York, NY, USA ©2010, ISBN: 978-1-4503-0257-9.

[10] MIUR, *Indicazioni Nazionali per il curricolo della scuola dell'Infanzia e del primo ciclo d'istruzione,* novembre 2012, http://hubmiur.pubblica.istruzione.it/web/istruzione/prot7734_12.

[11] RESNICK, M., MALONEY, J., MONROY-HERNANDEZ, A., RUSK, N., EASTMOND, E., RESNICK, M., BRENNAN, K., MILLNER, A., ROSENBAUM, E., SILVER, J., SILVERMAN, B., KAFAI, Y., *Scratch: Programming for All,* Communications of the ACM, Vol.52 n.11, 2009.

[12] WILLIAMS, L., CERNOCHOVA, M., *Literacy from Scratch,* in Atti della X IFIP WCCE (World Conference on Computers in Education), Torun, Polonia, 2-5 Luglio, 2013.

6. Cambiare la didattica dell'informatica attraverso la conoscenza del suo contenuto pedagogico

Luca Forlizzi[1]

I percorsi di formazione universitaria degli insegnanti di scuola secondaria sono oggetto, sin dalla loro introduzione, di forti critiche provenienti dal mondo della scuola e in particolare da coloro che, pur essendo a vario titolo già impegnati come docenti nelle scuole, scelgono di frequentarli. Alcune delle critiche attaccano il modello organizzativo dei percorsi, il fatto che siano a pagamento, i problemi logistici e le scarse risorse che le università a essi dedicano. Altre invece sono dirette ai contenuti dell'offerta formativa, e pertanto ci riguardano più da vicino. Possono essere riassunte così:

i contenuti dei percorsi sono inutili in quanto completamente avulsi da quello che si insegna realmente nelle scuole.

Nel caso dell'informatica, quest'affermazione è probabilmente vera nella maggior parte dei casi. Ma ciò è, a nostro giudizio, un fatto positivo. L'assunzione implicita nella critica, infatti, è che i programmi e/o le pratiche scolastiche siano "la cosa giusta", da preservare e perpetrare per le successive generazioni di allievi. Com'è noto, per quanto attiene l'insegnamento dell'informatica, tale assunzione non è affatto condivisa al di fuori del mondo della scuola, come ampiamente argomentato da voci provenienti dal mondo dell'accademia, dell'industria e da associazioni di esperti ([11], [5], [1]). Quindi il vero problema è cambiare ciò che realmente si insegna nelle scuole, e lo stimolo a operare in tal senso ci ha animati durante il compito di progettare, organizzare e condurre i percorsi formativi che il nostro ateneo ha attivato per la classe d'insegnamento A042.

---

[1] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila.

Cambiare un sistema complesso come quello scolastico, non è un processo semplice, per via dei meccanismi e delle abitudini consolidate che si va ad intaccare. È indispensabile, per riuscire, ottenere il consenso degli attori coinvolti, in questo caso soprattutto degli insegnanti che devono accogliere favorevolmente nuovi contenuti e metodi, per tradurli in un reale innalzamento della qualità dell'insegnamento dell'informatica.

Per fare fronte alla difficoltà del compito, abbiamo guardato alle esperienze di paesi in cui già da tempo la comunità accademica interviene nella formazione degli insegnanti di informatica. Il sistema di formazione costruito in Israele, documentato in numerose pubblicazioni ([4], [6]), gode di ampia considerazione a livello internazionale [11] ed è stato il riferimento principale che abbiamo adottato nel progettare e realizzare il TFA ed il PAS nel nostro ateneo. Nei prossimi paragrafi descriviamo le principali caratteristiche del modello fornito dall'esperienza israeliana, dapprima in generale e poi fornendo ulteriori informazioni a proposito del corso Methods of Teaching Computer Science (MTCS), che costituisce lo strumento principale attraverso il quale viene costruita la competenza relativa alla didattica della disciplina nei futuri insegnanti di informatica. In seguito discutiamo in che modo, e con quali adattamenti, abbiamo trasferito il modello nei percorsi di formazione da noi realizzati, soffermandoci in particolare sul PAS 2014. In particolare presentiamo obiettivi specifici, struttura e contenuti di un corso, in parte ispirato da MTCS, che attraverso l'illustrazione di strumenti per la didattica dell'informatica di applicabilità generale, serve a fornire motivazioni, incentivi e ausilio nel processo di rinnovamento del modo di insegnare informatica. L'efficacia del corso, in relazione a questi scopi, è stata valutata sottoponendo dei questionari agli studenti neoabilitati.

6.1. Il modello israeliano per l'insegnamento dell'informatica nelle scuole secondarie

La qualità dell'insegnamento dell'informatica in Israele è il prodotto di un sistema educativo articolato [6]. È possibile identificare cinque componenti principali:

1. *Curriculum nazionale ben definito.* Definito da un comitato istituito nel 1990, in cui erano coinvolti insegnanti di provata esperienza, esponenti del mondo dell'accademia e rappresentanti del ministero competente, è stato adottato nel 1995 e in seguito aggiornato. Sin dalla sua prima versione poggia su alcuni princìpi più che condivisibili, che stentiamo a ritrovare nelle rinnovate Indicazioni/Linee Guida del MIUR emanate tra il 2010 e il 2012:
   - l'informatica è una disciplina scientifica a tutti gli effetti;
   - i programmi devono vertere attorno ai concetti chiave e ai fondamenti disciplinari;
   - questioni concettuali e sperimentali devono essere intrecciate;
   - è ideale insegnare due diversi paradigmi di programmazione;
   - un laboratorio ben equipaggiato e mantenuto è indispensabile.

   Il curriculum esiste in due versioni, una basilare e una più estesa per studenti maggiormente interessati, ma anche nella prima vengono enfatizzati gli aspetti teorici della materia accanto a quelli applicativi, e gli studenti sono esposti a due diversi paradigmi di programmazione. Per tutti gli argomenti oggetto di insegnamento sono disponibili testi e materiali didattici scritti da esperti.
2. *Titoli specifici e obbligatori, per l'insegnamento dell'informatica nelle scuole secondarie.* Per diventare insegnanti di informatica in Israele è necessario un Bachelor in computer science seguito da una Abilitazione all'insegnamento dell'informatica o in alternativa un Bachelor of Education in Computer Science Education.
3. *Formazione universitaria degli insegnanti.* Nella maggior parte dei casi, i programmi di formazione per gli insegnanti sono di tipo universitario. Vi si ritrovano gli stessi "ingredienti" tipici di un TFA: corsi di pedagogia generale e specifici corsi per l'insegnamento dell'informatica, tra i quali Methods of Teaching Computer Science (MTCS) su cui forniremo ulteriori dettagli nel seguito, in aggiunta a un tirocinio effettuato presso le scuole. Anche i programmi di aggiornamento per gli insegnanti già in servizio sono organizzati dalle università.
4. *Centro nazionale per gli insegnanti di informatica.* Si tratta di una struttura centralizzata di coordinamento che fa da ponte tra gli insegnanti di informatica e la comunità accademica, organizzando conferenze, workshop, seminari e corsi e che pubblica un sito web e

una rivista dedicata. In particolare, la struttura promuove la costruzione di una comunità professionale di insegnanti di informatica e il coinvolgimento degli stessi insegnanti negli sviluppi del curriculum.
5. *Ricerca in didattica dell'informatica.* I ricercatori israeliani che si occupano di didattica dell'informatica, sono direttamente coinvolti nella scrittura dei libri di testo scolastici. Le ricerche didattiche vengono condotte sperimentalmente in classi selezionate e con il supporto degli insegnanti. I risultati sono applicati nella revisione o produzione di nuovo materiale didattico.

6.2. Methods of Teaching Computer Science

I programmi di formazione degli insegnanti sono di solito basati su conoscenze pedagogiche generali, conoscenze disciplinari, conoscenza del contenuto pedagogico della disciplina oggetto di insegnamento, e attività di tirocinio svolte presso classi reali. La conoscenza del contenuto pedagogico (da Pedagogical Content Knowledge, PCK) è una delle categorie del modello delle conoscenze di un insegnante proposto da Shulman [10] e consiste nell'insieme delle conoscenze necessarie e/o utili per insegnare una specifica disciplina (quindi distinte dalle conoscenze disciplinari). Il programma israeliano per la formazione degli insegnanti di informatica, attribuisce grande importanza alla PCK, cui viene dedicato [7] il corso Methods of Teaching Computer Science (MTCS). La centralità che gli viene attribuita nel processo di formazione degli insegnanti israeliani è testimoniata dalla durata (112 ore, articolate in due semestri) suggerita per un'implementazione appropriata [6].

Alcuni obiettivi specifici del corso MTCS sono:
1. aiutare gli studenti nella costruzione di un'identità professionale di insegnanti di informatica;
2. rendere consapevoli delle specificità dell'insegnamento dell'informatica;
3. discutere delle difficoltà tipiche che i discenti incontrano nell'apprendimento di determinati argomenti del curriculum in informatica;
4. presentare tipici metodi insegnamento dell'informatica;
5. sviluppare la padronanza di abilità e strumenti pedagogici utili nell'insegnamento dell'informatica, con riferimento ai bisogni di diverse tipologie di discenti;

6. esporre gli studenti alla ricerca sulla didattica dell'informatica e alle sue applicazioni nell'insegnamento a livello scolastico.

Un importante aspetto caratterizzante del corso, è che esso viene proposto come modello didattico, ovvero non si limita a presentare princìpi e a descrivere metodi e strumenti didattici, ma li implementa nella sua stessa struttura, mostrando ai futuri docenti come essi vengano effettivamente applicati. MTCS si configura pertanto come un laboratorio didattico che include lezioni, attività volte allo sviluppo di materiale didattico o di specifiche abilità, simulazioni di lezioni, esperienze pratiche di uso di strumenti. Questa impostazione si sposa con la convinzione, che anche nella comunità israeliana trova ampio consenso [2], che l'insegnamento dell'informatica debba essere condotto attraverso forme di active learning, che favoriscono negli studenti processi di costruzione di modelli mentali corretti degli oggetti di studio, in modo particolare dei processi computazionali.

6.3. Metodi e strumenti per la didattica dell'Informatica nei PAS e TFA

In assenza di precedenti esperienze di formazione rivolta agli insegnanti di informatica, sin dall'avvio presso l'Università dell'Aquila del primo TFA per la classe A042 (formalmente relativo all'a.a. 2011/12, ma erogato nel 2013) si è guardato a realtà già consolidate. Il modello israeliano è stato preso come punto di riferimento principale, in virtù dei risultati da esso ottenuti e della qualità della documentazione disponibile.

Tuttavia la carenza di risorse (docenti esperti e tempo per preparare i corsi) ci hanno obbligato ad adottare un approccio più conservativo, in cui un corso di Metodi e Strumenti per la Didattica dell'Informatica (MSDI), ispirato a MTCS ma di durata e ampiezza molto più limitate è stato affiancato ad altri corsi dedicati alla didattica di specifici argomenti quali la Teoria della Computabilità (Calcolabilità e Complessità Computazionale degli Algoritmi), le Basi di Dati, la Programmazione, l'Ingegneria del Software e le Reti. La strategia che questo approccio misto si proponeva di attuare, era quella di fornire attraverso MSDI un riferimento metodologico entro cui inquadrare i contenuti più specifici trattati negli altri corsi, i quali sono stati affidati a docenti che avevano insegnato tali contenuti a livello universitario.

Alla prima applicazione, nel TFA 2011/12, l'esito, rilevato analizzando le relazioni prodotte dagli abilitandi, ci ha soddisfatto solo in parte. Da un lato, la versione di MSDI offerta nel TFA 2011/12, ha avuto carattere sperimentale e il programma si è limitato a trattare poche questioni epistemologiche, etiche e relative alla pratica professionale dell'informatica (licenze software). Questi temi, sebbene di indubbio interesse e ritenuti parte indispensabile del bagaglio di conoscenze di un aspirante docente di informatica ([3], [11]), sono stati percepiti come accessori rispetto al nucleo dei contenuti disciplinari dell'informatica e non hanno fornito l'occasione di introdurre metodi di insegnamento più generali.

I docenti dei corsi specifici, d'altro canto, hanno trattato alcune questioni, nell'ambito dei rispettivi argomenti, che ritenevano importanti in base alla loro personale esperienza didattica. In mancanza di un coordinamento forte, non sono mancate duplicazioni e disomogeneità evidenziate nelle relazioni degli abilitandi. Nonostante queste pecche, abbiamo valutato positivamente la riuscita complessiva del TFA 2011/12, anche per merito degli studenti che hanno mostrato impegno e un'ottima preparazione di base.

Nel progettare il piano didattico per il PAS 2014, abbiamo naturalmente fatto tesoro dell'esperienza acquisita, ma abbiamo anche dovuto tener conto delle diverse caratteristiche degli studenti cui era rivolto. Essi, infatti, oltre a non essere stati selezionati in base alle loro competenze disciplinari, nella maggioranza dei casi avevano alle spalle percorsi di studio universitari in cui l'informatica aveva avuto un ruolo di secondo piano: per l'esattezza, degli undici abilitandi PAS, solo uno aveva una laurea in Scienze dell'Informazione e nessuno in Informatica o Ingegneria Informatica. A fronte di questa situazione, abbiamo ritenuto opportuno riproporre l'organizzazione in corsi dedicati ad argomenti specifici, particolarmente adatta "alla verifica e al consolidamento della conoscenza delle discipline oggetto di insegnamento della classe di concorso" prevista dalla normativa Ministeriale [8] come obiettivo del PAS complementare a quello della costruzione di competenze didattiche.

Al corso MSDI, interamente riprogettato per il PAS in modo da seguire molto più da vicino sia la metodologia che la successione degli argomenti previsti in MTCS, è stata demandata la presentazione di:

1. metodi di insegnamento e strumenti pedagogici di pertinenza non limitata ad argomenti particolari;
2. spunti di approfondimento su temi, ancorché non tecnici, ritenuti importanti nel bagaglio culturale di un docente di informatica, come quelli suggeriti in [3].

A questi obiettivi, che risultano in linea con quelli della controparte israeliana, ne abbiamo aggiunto uno in funzione delle caratteristiche dei fruitori del PAS. Essi, infatti, pur non avendo svolto studi universitari con prevalenza di contenuti propri dell'informatica, avevano già maturato una certa esperienza di insegnamento in materie di pertinenza della classe A042. Nella maggioranza dei casi, com'era naturale attendersi, la loro esperienza di docenza si era saldamente collocata nel solco della tradizione di insegnamento dell'informatica che risulta prevalente nella scuola italiana. I limiti e gli errori di tale tradizione sono noti e ampiamente discussi, anche da altri contributi in questo volume. Condividendo tale giudizio negativo, attraverso l'intero PAS e soprattutto tramite MSDI abbiamo tentato non solo di presentare un modo diverso di insegnare informatica, ma anche di invogliare gli studenti a provare ad applicarlo nella loro attività di docenti. L'opera di persuasione è stata improntata al massimo rispetto verso la loro professionalità, astenendosi da commenti negativi o polemici diretti al loro consueto modo di lavorare e rivolgendo la critica sempre nei confronti del sistema complessivo.

Gli argomenti trattati nel corso MSDI per il PAS sono riassunti nel seguente sillabo, e sono stati svolti in 15 ore complessive:

1. *L'informatica nella scuola italiana:* Partendo dal resoconto delle esperienze di docenza degli studenti, è stata svolta una discussione critica sul modo di insegnare informatica nella scuola italiana, aprendo spiragli su approcci alternativi, approfonditi nel seguito del corso. Sono state poi analizzate le indicazioni ministeriali relative alle materie scolastiche di pertinenza della classe A042. Infine si è parlato delle competenze necessarie ad un insegnante di informatica.
2. *La natura dell'informatica:* Cenni sul dibattito relativo alla natura e ai confini dell'informatica quale campo di conoscenze e disciplina scientifica. Presentazione delle diverse aree in cui si articola il corpo di conoscenze dell'informatica, in base alle classificazioni operate da organizzazioni internazionali.

3. *Discenti, astrazione e competenze:* Discussione su competenze e concezioni dell'informatica proprie dei "nativi digitali", in relazione alle capacità di astrazione di soggetti in età adolescenziale.
4. *Strumenti per la Didattica Attiva dell'informatica:* Cenni su costruttivismo, costruzionismo e active learning. Giochi, competizioni ed artefatti tangibili: discussione dei benefici pedagogici ed alcuni esempi (Olimpiadi di Informatica, CS Unplugged, robotica educativa).
5. *Ambienti di apprendimento basati su visualizzazione e animazione:* uso di visualizzazione e animazione nella didattica dell'informatica: discussione ed esempi (Alice, Greenfoot, Karel the robot, Logo e derivati, Squeak e derivati). Approfondimento su UCB Logo.
6. *Metodi di insegnamento per l'informatica:* Presentazione di metodi didattici di comune impiego in didattica dell'informatica: metodo dei progetti, pair programming, metodo Jigsaw.
7. *Pianificazione didattica:* Metodi di pianificazione e sequenze didattiche proposte e analizzate in letteratura.
8. *Concezioni alternative nei discenti:* Metodi per l'individuazione di modelli concettuali, costruiti dai discenti, non conformi al reale funzionamento dei sistemi informatici.
9. *Soft ideas nell'informatica*: Concetti che non è possibile definire formalmente o in maniera rigida, e che sono presenti in numerosi argomenti disciplinari (ad esempio astrazione, modularità, paradigma).

Oltre che parte dei contenuti, da MTCS è stato mutuato il metodo di insegnamento adottato in MSDI, basato sullo svolgimento di un cospicuo numero di attività didattiche. Esse, oltre a supportare la formazione della conoscenza in base alle teorie costruttiviste, sono servite a far sperimentare in prima persona, ai nostri studenti, attività didattiche da proporre ai loro allievi, a dare occasione di produrre materiale didattico e applicare diversi metodi presentati nel corso, e a fornire spunti per riflessioni individuali e/o collettive sul proprio operato didattico, sia nell'ambito di MSDI che nella loro attività professionale di insegnamento.

6.4. Valutazione didattica

Una volta concluse le lezioni del corso MSDI nel PAS, e le relative sessioni d'esame, abbiamo chiesto agli studenti di rispondere, usando una scala di Likert a 5 punti, al questionario di valutazione riportato in Tabella 6.1. I valori medi delle risposte sono mostrati in Tabella 6.2.

Considerate le peculiarità del corso, che non presentava particolari difficoltà concettuali o operative (pur tenendo conto delle caratteristiche dei discenti), la valutazione della capacità di trasmissione di conoscenze da parte del docente è sostanzialmente racchiusa nel giudizio complessivo di gradimento che deriva dal confronto tra le prime due risposte (ASPA e ASPD).

NUM	ETICHETTA	DOMANDA
1	*ASPA*	*Come giudicheresti le tue aspettative sul corso, prima di frequentarlo?*
2	*ASPD*	*Il corso si è rivelato all'altezza delle aspettative?*
3	*SOR*	*In che misura i contenuti del corso sono stati inattesi?*
4	*CONT*	*In che misura ritieni importanti i contenuti del corso?*
5	*METODO*	*In che misura ritieni efficace il metodo di insegnamento utilizzato dal docente nel corso?*
6	*INFLG*	*In che misura ritieni che il corso abbia influenzato la tua concezione generale dell'informatica?*
7	*INFLI*	*In che misura ritieni che il corso abbia influenzato la tua concezione dell'insegnamento dell'informatica nelle scuole secondarie?*
8	*CAMBIOC*	*Dopo aver seguito il corso quanto pensi di cambiare i contenuti che insegni nelle tue classi ?*
9	*CAMBIOM*	*Dopo aver seguito il corso quanto pensi di cambiare i tuoi metodi di insegnamento?*
10	*CAMBIOS*	*Dopo aver seguito il corso quanto pensi di cambiare gli strumenti (ad es. linguaggi di programmazione, ambienti di apprendimento, robot) che adotti per l'insegnamento?*

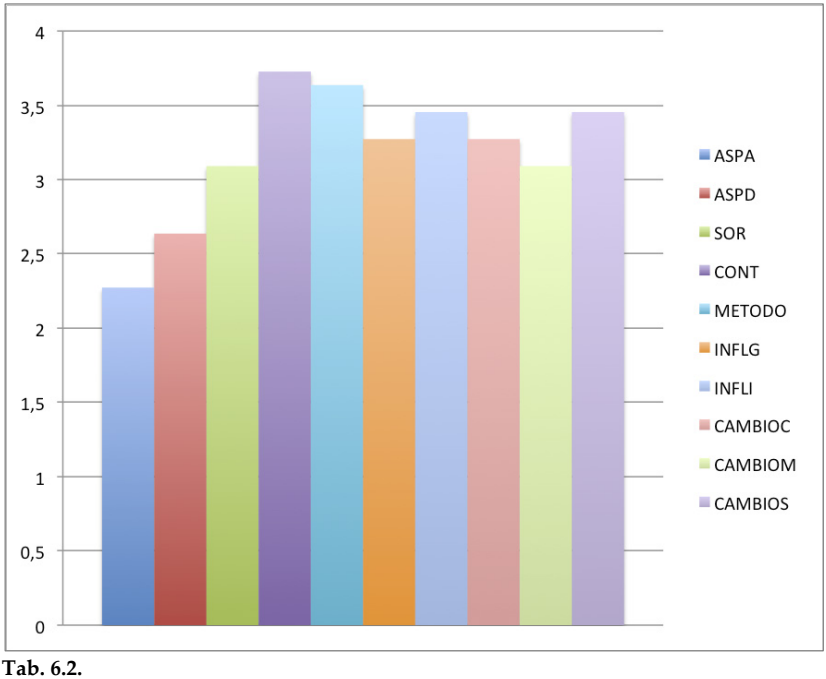
Tab. 6.1.

Il fatto che il corso abbia superato le aspettative iniziali dei discenti, è senz'altro indice di una buona riuscita, anche sotto il profilo della chiarezza del modo di esporre i contenuti.

È convinzione di chi scrive che l'inerzia che determina l'attuale stato dell'insegnamento dell'informatica nelle scuole sia il risultato di profonde carenze, presso gli operatori scolastici, non tanto di conoscenze tecniche, quanto di prospettiva sulla disciplina. Abbiamo voluto verificare, su scala ridottissima la fondatezza di quest'opinione, attraverso la domanda 3 e, sebbene statisticamente irrilevante vista la dimensione del campione, un valor medio superiore a 3 sembra avallarla.

La domanda 4 chiedeva agli studenti la loro opinione sulla importanza dei contenuti del corso e ha ottenuto un riscontro molto positivo, che testimonia come sia sentita l'esigenza di strumenti metodologici adeguati, anche da parte di insegnanti che possono già vantare una esperienza didattica significativa. Il metodo di insegnamento adottato è stato ritenuto adeguato, ma ciò non sorprende in quanto era stato già ampiamente validato dalle esperienze israeliane.

Le domande da 5 a 10 costituiscono il gruppo su cui si concentrava maggiormente il nostro interesse, in quanto tese a valutare l'influenza che il corso può avere sullo sviluppo professionale degli insegnanti. Anche in questo caso gli studenti hanno fornito un riscontro ampiamente positivo, affermando di avere modificato la propria visione della disciplina (domande 5-6) e, soprattutto, affermando di voler applicare quanto appreso nelle loro classi (domande 7-10).



Tab. 6.2.

6.5. Conclusioni

Percorsi di formazione degli insegnanti, organizzati dalle università su solide basi scientifiche e tecniche, possono essere il primo passo verso un cambiamento radicale del modo in cui l'informatica viene insegnata [6] e del ruolo che le viene attribuito nella scuola italiana. In tal senso, la nostra esperienza nell'ambito del PAS 2014 è promettente e suggerisce che anche insegnanti già in servizio potrebbero essere favorevoli a rivedere il loro modo di lavorare, purché siano loro forniti stimoli, motivazioni, strumenti e supporto adeguati. Resta comunque molto lavoro da fare, a cominciare dal TFA 2015, in corso di svolgimento, che sta richiedendo numerosi adattamenti rispetto a quanto descritto in questo contributo. Riteniamo che per un decisivo salto di qualità sia importante, da un lato tener conto delle esperienze estere già consolidate, ma anche intensificare gli scambi di opinione e i momenti di confronto tra le diverse iniziative messe in atto nel nostro paese, in modo da delineare e mettere a punto le soluzioni più adatte alle peculiarità del sistema scolastico italiano. L'auspicio, resta quello di poter giungere alla formulazione di sillabi condivisi per le attività di formazione degli insegnanti di informatica.

Bibliografia

[1] ACM – CSTA, *Running on empty:The failure to Teach K-12,* Computer Science in the digital age , http://www.acm.org/Runningonempty/ , last accessed: 2013.

[2] BEN ARI, M., *Constructivism in computer science education,* J. of Comput. In Math. And Sci. Teach. 20 (1) 2001, 45-73.

[3] GAL-EZER, J., HAREL, D., *What (else) should CS educators know?*, Comm. of ACM 41 (9) 1998, 77-84.

[4] GAL-EZER, J., HAREL, D., *Curriculum for a high school computer science curriculum,* Computer Science Education 9 (2) 1999, 114-147.

[5] GANDER W., PETIT A., BERRY G., DEMO B. ET AL., *Europe cannot afford to miss the boat,* Report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education, 2013.

[6] HAZZAN, O., LAPIDOT, T., RAGONIS N., *Guide to Teaching Computer Science: An Activity-Based Approach,* Springer, 2011.

[7] LAPIDOT, T., HAZZAN, O., *Methods of Teaching Computer Science course for prospective teachers,* Inroads – SIGCSE Bull. 38 (2) 2003, 29-34.

[8] MIUR, *Decreto Dipartimentale 45,* del 22 Novembre 2013, all. A

[9] MERRIT, S. M. ET AL., *ACM model of high school computer science curriculum,* Communications of the ACM 36 (5) 1993, 87-90.

[10] SHULMAN, L.S., *Those who understand: knowledge groth in teaching,* Educ. Teach. 15(2) 1986, 4-14.

[11] TUCKER, A.B., *K-12 Computer Science: Aspirations, Realities, and Challenges,* Proc. of the 4th International Conference on Informatics in Secondary Schools - Evolution and Perspectives: Teaching Fundamentals Concepts of Informatics, 22-34, January 13-15, 2010, Zurich, Switzerland.

7. La teoria degli algoritmi nella scuola secondaria di secondo grado

Luca Forlizzi[1], *Guido Proietti*[2]

Grande è l'insoddisfazione della comunità accademica internazionale che si occupa di didattica dell'informatica nei riguardi dello stato in cui versa l'insegnamento della disciplina nelle scuole. Si ritiene infatti che non si insista a sufficienza sui fondamenti scientifici della disciplina, e che molto spesso, trascurando financo importanti contenuti tecnologici, ci si riduca a mero addestramento all'utilizzo delle tecnologie ICT, l'aspetto più superficiale della cosiddetta *Digital Literacy*. Questa situazione viene denunciata sia in numerosi articoli scientifici [2], [9], [7], [10], sia in report di organizzazioni accademiche [6], [1]. Questa situazione sembra comune un po' in tutto il mondo, tranne in alcune nazioni che si distinguono per aver intrapreso riforme dei curricula informatici fin dalla scuola primaria [4].

Nel caso italiano, cambiare questo stato di cose richiede sia un adeguamento delle indicazioni ministeriali sui programmi di insegnamento, sia una diversa formazione dei futuri docenti di informatica. La recente istituzione di percorsi universitari abilitanti all'insegnamento dell'informatica nella scuola secondaria di secondo grado, ovvero i TFA e i PAS, costituisce una preziosa occasione per la comunità accademica italiana per intervenire nella definizione delle competenze dei futuri docenti.

In questo lavoro prendiamo in esame l'insegnamento di due pilastri caratterizzanti dell'informatica: la teoria della calcolabilità e quella

[1] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila.

[2] Dipartimento di Informatica, Università degli Studi dell'Aquila e Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", Consiglio Nazionale delle Ricerche.

degli algoritmi. In particolare, partendo da una riflessione sulla percezione culturale dell'informatica, descriviamo l'approccio che abbiamo utilizzato al fine di consapevolizzare i futuri docenti in merito al ruolo formativo peculiare che l'informatica in generale, e l'algoritmica in particolare, può svolgere sugli studenti delle scuole secondarie di secondo grado. Infine, presentiamo i risultati di un'indagine di gradimento del corso che abbiamo condotto tra i discenti dei corsi TFA e PAS, i cui risultati si sono e si stanno dimostrando utili per i successivi affinamenti dei contenuti proposti.

7.1. Algoritmi nelle scuole: aspirazioni e realtà

Gli algoritmi sono naturalmente uno dei concetti fondanti della disciplina e non sorprende che il loro studio sia centrale in numerose proposte di curricula e sperimentazioni didattiche [8], [3], [5], [7]. Nelle attuali indicazioni ministeriali essi sono sì presenti, pur tuttavia non con la centralità loro dovuta.

La nostra esperienza di confronto con i docenti della scuola ci suggerisce che gli algoritmi vengono in genere considerati come una sorta di esercizio propedeutico alla programmazione. Lo studio degli algoritmi viene spesso inteso come:

- la formalizzazione dei diagrammi di flusso
- l'esposizione di alcuni esempi di algoritmi notevoli (ordinamento su tutti) il cui scopo unico è veicolare agli allievi l'idea che prima di cominciare a scrivere codice devono fermarsi a pensare.

Nulla o poco si dice quindi in merito al significato più profondo di algoritmo come vero e proprio idioma di una scienza a tutto tondo, ancorché moderna, ovvero l'informatica. A nostro avviso, tale distorsione ha radici antiche, che rimandano addirittura alla radice linguistica del termine informatica. Tale termine deriva infatti dal tedesco *Informatik*, contrazione di *Automatische Informationsverarbeitung*. Nell'accezione italiana, dunque, il termine viene associato sia alla pratica di utilizzo di dispositivi automatici di elaborazione dati (computer, in senso lato: PC, tablet, smartphone, etc.), sia alla ben più profonda attività legata alla loro *programmazione*, vero e proprio connubio di ingegno e tecnica, l'educazione alla quale richiede un approccio pedagogico sistematico e di carattere scientifico. Si noti invece che nella

lingua inglese tale dicotomia è esplicitata nel lessico, in quanto il termine informatica si traduce con:

- *Information and Communication Technology (ICT)*, quando ci si riferisce ai dispositivi tecnologici e al loro utilizzo pratico;
- *Computer Science (CS)*, quando ci si riferisce allo studio sistematico dei processi computazionali (e quindi ai principi di programmazione di un elaboratore).

La prima declinazione rimanda quindi all'*empirismo* in filosofia: l'esperienza (cioè l'uso dello strumento) è alla radice della conoscenza. È questo l'aspetto che tuttora viene privilegiato nei corsi di formazione continua e in quelli scolastici primari di alfabetizzazione informatica in Italia (locuzione espressamente utilizzata nella Riforma Moratti del 2003): il discente viene istruito al mero uso di alcune delle applicazioni digitali di largo consumo, con ampie distorsioni di natura commerciale (in alcune circolari ministeriali viene addirittura esplicitamente previsto l'addestramento all'utilizzo di prodotti proprietari!). Tale approccio ha diverse implicazioni negative, tra le quali citiamo:

- l'incapacità di utilizzo consapevole dei mezzi di calcolo, ovvero mancanza di una vera cultura (o educazione) digitale, la quale consente invece di padroneggiare gli strumenti tecnologici e di muoversi a proprio agio nella comunità virtuale (la rete);
- la rapida obsolescenza delle informazioni acquisite, stante il livello di continua innovazione (hardware e software) in ambito tecnologico.

La seconda declinazione del termine inglese rimanda invece al *razionalismo* in filosofia: la ragione pura (conoscenza *a priori*) prevale sull'esperienza sensoriale (conoscenza *a posteriori*). In questo senso, la programmazione diventa quindi assimilabile ad un'attività matematica, con le tutte le implicazioni di carattere epistemologico che ne conseguono.

L'informatica con quest'ultima accezione è quindi, a nostro avviso, una disciplina scientifica pervasiva e con caratteristiche peculiari, che però stenta a trovare un suo ruolo nel panorama scolastico italiano (e mondiale), anche per via dei richiamati dissensi sulla sua identità culturale.

Nella comunità accademica, c'è invece ampio consenso riguardo al fatto che l'obiettivo principale dell'informatica sia quello di definire degli strumenti metodologici atti a sviluppare, sul piano psicologico, comportamentale ed operativo, le abilità necessarie a risolvere efficientemente ed automaticamente problemi computazionali per i quali non si

possiede a priori una procedura risolutiva. Questo approccio conduce quindi allo sviluppo nel discente del cosiddetto *pensiero computazionale*, che completa e complementa il classico sviluppo del pensiero logico-matematico. Il pensiero computazionale è un *modus cogitandi* finalizzato ad automatizzare la risoluzione di un dato problema mediante la definizione di una *soluzione algoritmica*, ovvero una sequenza accuratamente descritta di passi, ognuno dei quali appartenente ad un catalogo ben definito di operazioni di base ammissibili. L'algoritmo diventa quindi, a tutti gli effetti, il concetto centrale attorno al quale ruotano tutti gli aspetti fondanti dell'informatica intesa come *computer science*.

7.2. Teoria della computabilità per i futuri docenti di informatica

L'obiettivo fondamentale dei percorsi TFA e PAS che abbiamo costruito è stato proprio quello di aumentare la sensibilità dei futuri docenti rispetto all'importanza dello sviluppo del pensiero computazionale nei discenti delle scuole. Per costruire dei sillabi coerenti con le premesse, ci siamo ispirati all'insegnamento di due numi tutelari dell'informatica:

- "Il ragionamento matematico può essere considerato piuttosto schematicamente come l'esercizio di una combinazione di due capacità, che possiamo chiamare intuizione e ingegnosità." (Alan Turing (1912-1954)
- "Se è vero che un problema non si capisce a fondo finché non lo si deve insegnare a qualcun altro, a maggior ragione nulla deve essere compreso in modo più approfondito di ciò che si deve insegnare ad una macchina, ovvero di ciò che va espresso tramite un algoritmo." (Donald Knuth, autore di The Art of Computer Programming).

Per risolvere un problema computazionale, bisogna quindi coltivare e sviluppare l'intuito e l'ingegno del discente attraverso la comprensione iniziale della natura del problema stesso (ovvero del suo nucleo matematico), seguita poi dalla progettazione di una appropriata procedura di risoluzione algoritmica (laddove possibile!), ovvero una sequenza di istruzioni formali che potranno poi essere tradotte in un linguaggio intellegibile al computer (programma), il tutto concluso da una meticolosa verifica della correttezza del risultato.

Abbiamo dunque ritenuto opportuno, dedicare, sia nel TFA che nel PAS, un corso specifico sull'anima algoritmica dell'informatica, o più propriamente sulla *teoria della computabilità*, che a sua volta può essere suddivisa in due grandi filoni:

- La *teoria della calcolabilità*, ovvero lo studio della (ir)risolubilità dei problemi computazionali mediante un procedimento algoritmico;
- La *teoria degli algoritmi e della complessità computazionale*, ovvero lo studio delle risorse di calcolo (principalmente tempo di esecuzione e spazio di memoria utilizzato) necessarie e sufficienti ad un algoritmo (esatto, approssimato, randomizzato) per risolvere un problema computazionale.

Nella prima parte ci si poneva dunque un obiettivo ambizioso: trasmettere integralmente il significato di *calcolabilità*, tuttora relegato ai corsi di livello universitario. A nostro avviso si tratta invece di un concetto imprescindibile nel momento stesso in cui si introduce il principio della risoluzione *automatica* (o *algoritmica*) di un problema. Per inciso, si ritiene che la presentazione del paradosso insito nel *problema dell'arresto*, utilizzato allo scopo di mostrare l'esistenza di problemi non risolubili algoritmicamente, sia pienamente trasferibile ai discenti della scuola secondaria di secondo grado.

La seconda parte sulla teoria degli algoritmi e della complessità computazionale si poneva invece obiettivi ugualmente ambiziosi, ma diversificati: da un lato, molti dei concetti esposti erano pensati per essere trasferiti ai discenti, dall'altro lato alcuni contenuti sono stati pensati soprattutto in veste di arricchimento culturale dei docenti, al fine di sensibilizzarli circa l'importanza di trattare questi argomenti nella scuola e di costituire quell'*humus* propizio allo sviluppo di competenze più avanzate in materia.

Il tutto è stato illustrato cercando di utilizzare un linguaggio rigoroso, ma senza eccedere nel formalismo, con l'obiettivo quindi di fornire delle idee e del materiale da riutilizzare in classe (opportunamente adattato alle esigenze di ciascuno). Concretamente, tale approccio ha portato allo sviluppo di tre tematiche principali, le prime due delle quali riutilizzabili in classe, mentre la terza a carattere prettamente divulgativo e formativo verso i docenti, ovvero:

1. Facile, difficile, impossibile: spunti di teoria della calcolabilità e principali classi di complessità computazionale dei problemi.
   - Che cos'è un algoritmo?
   - È sempre possibile risolvere (algoritmicamente) un dato problema? Il problema dell'arresto.
   - Caratterizzazioni dei problemi in funzione della loro "difficoltà" computazionale. Macchine di Turing deterministiche e non deterministiche. Le classi di complessità P, NP, EXPTIME.
2. Essere algoritmista: progettare un algoritmo corretto, efficiente, e possibilmente ottimo.
   - Un esempio istruttivo: il problema dell'ordinamento.
   - Algoritmi di ordinamento elementari a confronto: prestazioni nel caso migliore, medio e peggiore.
   - Quanto è difficile ordinare? La delimitazione inferiore alla complessità computazionale di un problema.
   - Algoritmi di ordinamento ottimi.
3. Quando il problema è troppo arduo e tutto il resto fallisce: gli algoritmi di approssimazione e il potere della randomizzazione.
   - Un altro modo di definire la classe NP: il concetto di certificato.
   - La classe dei problemi NP-completi.
   - Il problema da 1 Milione di Dollari: P versus NP.
   - Un modo divertente di parlare di complessità computazionale: puzzle, matematica e algoritmi ricorsivi.
   - Algoritmi di approssimazione per problemi su grafi: Vertex Cover e Travelling Salesman Problem.
   - Protocollo di comunicazione randomizzato per stabilire la consistenza di due database.

Infine, è giusto ricordare la variegata composizione, in termini di *curriculum studiorum* e di esperienza di insegnamento, dei discenti del TFA e PAS. Gli iscritti al TFA avevano in prevalenza lauree in Informatica o Ingegneria informatica e manifestavano una buona conoscenza degli argomenti trattati. Decisamente diversa la situazione della maggioranza degli iscritti al PAS, che non avevano affrontato nei loro studi universitari insegnamenti specifici sulla teoria della computabilità. Nel caso del PAS, pertanto, come previsto dalla normativa, il nostro corso ha affiancato alla didattica della disciplina, anche una necessaria attività di verifica e consolidamento delle relative conoscenze.

7.3. Valutazione didattica

Non essendoci una tradizione consolidata nella formazione dei docenti di informatica in Italia, abbiamo ritenuto particolarmente importante valutare l'esito della nostra azione didattica. Attraverso l'indagine condotta abbiamo voluto soprattutto capire l'opinione, al termine del corso, dei futuri docenti in merito all'importanza e alla fattibilità di introdurre argomenti di teoria della computabilità nei programmi scolastici delle materie riferibili alla classe A042.

N.	ETICHETTA	DOMANDA
1	*PRE*	*Quale era la tua conoscenza in ingresso sui contenuti del corso?*
2	*SOR*	*In che misura i contenuti del corso sono stati inattesi?*
3	*DIFF*	*In che misura i contenuti del corso sono stati difficili?*
4	*CONT*	*In che misura ritieni importanti i contenuti del corso?*
5	*METODO*	*In che misura ritieni efficace il metodo di insegnamento utilizzato dal docente nel corso?*
6	*INFLG*	*In che misura ritieni che il corso abbia influenzato la tua concezione generale dell'informatica?*
7	*LSa*	*In che misura ritieni che i contenuti del corso possano essere oggetto di insegnamento in un Liceo Scientifico - opzione Scienze Applicate?*
8	*LSb*	*In che misura ritieni importante insegnare i contenuti del corso in un Liceo Scientifico - opzione Scienze Applicate?*
9	*ITola*	*In che misura ritieni che i contenuti del corso possano essere oggetto di insegnamento in indirizzi o articolazioni orientati all'informatica di Istituti Tecnici?*
10	*ITolb*	*In che misura ritieni importante insegnare i contenuti del corso in indirizzi o articolazioni orientati all'informatica di Istituti Tecnici?*
11	*ITnola*	*In che misura ritieni che i contenuti del corso possano essere oggetto di insegnamento in indirizzi o articolazioni NON orientati all'informatica di Istituti Tecnici?*
12	*ITnolb*	*In che misura ritieni importante insegnare i contenuti del corso in indirizzi o articolazioni NON orientati all'informatica di Istituti Tecnici?*

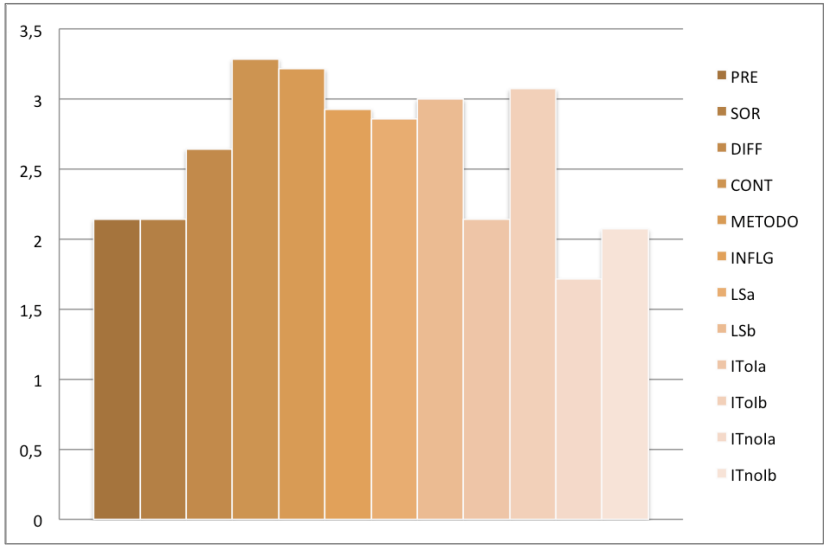
Tab. 7.1.

Una volta concluse le lezioni del corso e le relative sessioni d'esame, abbiamo chiesto ai discenti di rispondere, usando una scala di Likert a 5 punti, al questionario riportato in Tabella 7.1.

Le domande 1-2 sono volte a indagare la conoscenza in ingresso degli argomenti del corso, mentre la 3 e la 5 a valutare l'efficacia della trasmissione dei contenuti. Le domande 4 e 6, invece, servono a capire

quanto il corso abbia avuto successo nel sottolineare l'importanza culturale dei contenuti. Attraverso le domande 7-12 abbiamo chiesto l'opinione dei futuri docenti sulla fattibilità e sull'importanza di insegnare la teoria della computabilità, distinguendo le scuole in cui esistono insegnamenti di pertinenza della classe A042 in 3 categorie: liceo scientifico (opzione scienze applicate), indirizzi di istituti tecnici in cui l'informatica è tra gli insegnamenti caratterizzanti, altri indirizzi di istituti tecnici.

Hanno risposto al sondaggio, complessivamente 14 discenti, in prevalenza frequentanti del PAS. La Tabella 7.2 mostra i valori medi delle risposte.



Tab. 7.2.

Innanzitutto le risposte ci dicono che la conoscenza in ingresso della teoria della computazione supera di poco la media, e questo non ci sembra un buon viatico per l'insegnamento dell'informatica nelle scuole secondarie. D'altronde il risultato non è sorprendente, visti i titoli di studio degli aspiranti docenti di informatica che hanno svolto il PAS. In base all'esperienza fatta nella formazione degli insegnanti, riteniamo che l'elenco dei titoli di studio che consentono l'accesso alla classe A042 sia troppo ampio. I valori medio-alti delle risposte alle domande 3-6 ci confortano sul successo del corso, in modo particolare in

riferimento alla sensibilizzazione circa l'importanza dei contenuti in relazione alla disciplina. Le risposte alle domande 8, 10, 12 indicano che i discenti sono in media convinti che sia importante parlare di teoria della computazione nelle scuole, compresi gli indirizzi di istituti tecnici in cui l'informatica ha un ruolo di contorno. Questo è, a nostro avviso, il risultato migliore del corso, perché valida l'idea che il Pensiero Computazionale sia un insieme di competenze importanti per il cittadino del XXI secolo. Tuttavia, il compito viene percepito come arduo, come attestano le risposte alle domande 7, 9, 11, in particolare negli istituti tecnici. Sorprende soprattutto il risultato della domanda 9, che tendiamo a interpretare come scarsa fiducia nella propensione degli studenti degli istituti tecnici verso gli aspetti teorici di una disciplina.

Abbiamo ricavato ulteriori riscontri dalla lettura delle relazioni finali prodotte dai discenti al termine dei percorsi TFA e PAS, in cui, come previsto dalla normativa, veniva richiesto agli abilitandi di esprimere un giudizio critico su quanto appreso durante il percorso. C'è un consenso quasi unanime sul fatto che il concetto di complessità computazionale inteso come numero di passi richiesti per l'esecuzione di un algoritmo sia alla portata degli studenti di scuola superiore, e che costituisca un forte arricchimento della nozione di algoritmo. Alcuni abilitandi hanno anche provato ad ipotizzare una collocazione precisa nell'ambito di un programma di studi: terzo o quarto anno in un istituto tecnico (assumendo che nel corso del terzo anno si studino algoritmi notevoli), mentre in un liceo tratterebbero l'argomento nel corso del quinto anno, traendo vantaggio dallo studio parallelo dell'analisi matematica. Altrettanto favorevolmente vengono visti gli algoritmi elementari su grafi, sebbene vengano considerati più difficili rispetto ad altri algoritmi elementari, come quelli di ordinamento di vettori. Diverse tesi esprimono un giudizio positivo anche su algoritmi randomizzati e algoritmi di approssimazione, specie in virtù della disponibilità di esempi concreti, tratti dal materiale del corso, in problemi in cui essi possono risultare più efficaci rispetto ad algoritmi deterministici esatti. Relativamente alla teoria della calcolabilità e allo studio delle classi di complessità, le opinioni sono maggiormente dissonanti. La maggioranza dei candidati, in effetti preferisce non esporsi, definendole molto importanti ma guardandosi bene dall'inquadrarle in un possibile programma di studio. I più ottimisti si dichiarano fiduciosi

sul fatto che tali argomenti siano alla portata degli studenti dell'ultimo anno di un liceo, a loro dire più abituati alla speculazione teorica, mentre per l'istituto tecnico propongono di "selezionare un sottoinsieme degli argomenti relativi alla dicotomia P/NP". Altri abilitandi si esprimono invece in senso negativo, ritenendo le nozioni di calcolabilità e classe di complessità "troppo complesse per poter essere comprese da studenti di una scuola superiore" in quanto "comprendere davvero il non determinismo richiede delle capacità di astrazione che gli studenti di questa fascia di età non hanno". Infine, anche per i più pessimisti, la nozione dell'esistenza di problemi irrisolvibili algoritmicamente potrebbe essere data, con un taglio molto informale.

7.4. Conclusioni

Gli algoritmi e il complesso di teorie che ne studiano le proprietà fondamentali, sono al cuore dell'informatica. Secondo molti punti di vista epistemologici è proprio la centralità degli algoritmi nell'attività di ricerca a caratterizzare l'informatica come disciplina scientifica autonoma. Troviamo quindi paradossale il modo in cui l'idea stessa di algoritmo venga travisata da ampie porzioni del mondo della scuola, dai docenti che li identificano con i diagrammi di flusso fino agli esperti del MIUR che in un quesito della prova di accesso al TFA 2012/13 hanno ridotto la portata del concetto al puro determinismo. Il desiderio di superare questa situazione ci ha spinti a dedicare un corso agli algoritmi nei percorsi di formazione degli insegnanti, incentrato sui concetti fondamentali di calcolabilità e complessità computazionale. L'obiettivo era quello di sensibilizzare i futuri docenti sull'importanza di impartire ai loro allievi una corretta preparazione di base sugli algoritmi, ma il nostro intento è stato anche quello di fornire loro un aiuto concreto sotto forma di materiale potenzialmente adattabile all'uso nelle scuole.

Sulla base dei dati che abbiamo raccolto, tramite questionari e analisi delle relazioni finali, riteniamo di aver ottenuto risultati soddisfacenti, quantomeno rispetto all'obiettivo di dare consapevolezza della centralità agli algoritmi nell'ambito dell'informatica, sia intesa come disciplina scientifica che come materia scolastica.

Bibliografia

[1] ACM – CSTA, *Running on empty: The failure to Teach K-12 Computer Science in the digital age,* http://www.acm.org/Runningonempty/, 2013.

[2] CALZAROSSA, M.C., CIANCARINI, P., MICH, L., SCARABOTTOLO, N., *Informatics Education in Italian High Schools,* Proc. of 5th Int. Conference in Informatics in Secondary Schools – Evolution and Perspectives (ISSEP), 31-42, 2011.

[3] GAL-EZER, J., *A Pre-Programming Introduction to Algorithmics,* Mathematics and Computer Education, 1996, 30, 1, 61-69.

[4] GAL-EZER, J. AND HAREL, D., *Curriculum for a high school computer science curriculum,* Computer Science Education 9(2), 114-147, 1999.

[5] GAL-EZER, J., ZUR E., *The Concept of 'Algorithm Efficiency' in the High School CS Curriculum,* Proc. of Frontiers in Education (FIE) 2002.

[6] GANDER W., PETIT A., BERRY G., DEMO B., ET AL., *Europe cannot afford to miss the boat,* Rep. of the joint Informatics Europe & ACM Europe Working Group on Informatics Education, http://www.informatics-europe.org/services/reports.html, 2013.

[7] HROMKOVIC J., *Contributing to General Education by Teaching Informatics,* Proc. of 4th Int. Conference in Informatics in Secondary Schools – Evolution and Perspectives (ISSEP), 25-37, 2006.

[8] MERRIT, S. M. ET AL., *ACM model of high school computer science curriculum*, Communications of the ACM 36 (5), 87-90, 1993.

[9] TUCKER, A.B., *K-12 Computer Science: Aspirations, Realities, and Challenges,* Proc. of the 4th International Conference on Informatics in Secondary Schools - Evolution and Perspectives: Teaching Fundamentals Concepts of Informatics, 22-34, January 13-15, 2010, Zurich, Switzerland.

[10] WING, J.M., *Computational Thinking*, Communications of the ACM 49 (3) March 2006.

8. L'insegnamento dell'Informatica e i linguaggi di programmazione

Maurizio Boscaini, Ugo Solitro e Margherita Zorzi[1]

In questo capitolo si propongono alcune riflessioni sull'esperienza dei percorsi *TFA (Tirocinio Formativo Attivo)* e *PAS (Percorsi Abilitanti Speciali)* per gli insegnanti di Informatica della Scuola Secondaria di II grado che si sono tenuti presso l'*Università degli Studi di Verona* in collaborazione con gli altri Atenei veneti dall'anno accademico 2012/13. Si esamina in particolare l'esperienza veronese ponendo l'attenzione sui temi della *Programmazione* e dei *Linguaggi* e della loro valenza nella didattica. Si prosegue con alcune considerazioni di carattere generale sul valore della collaborazione tra Scuola e Università e si conclude con alcune riflessioni sull'attività svolta e sulle prospettive future.

8.1. L'esperienza di Verona

Il percorso formativo "disciplinare" è articolato in più insegnamenti che corrispondono ad una suddivisione dei temi dell'informatica globalmente intesa come disciplina in macroaree tematiche: i fondamenti e la programmazione, i sistemi informativi, i sistemi operativi e le reti. Inoltre, nei percorsi rivolti agli insegnanti tecnico-pratici, un insegnamento è in modo specifico dedicato alle attività di laboratorio informatico.

Questa struttura è stata inizialmente proposta per il I Ciclo del TFA ed è stata mantenuta, con gli opportuni adattamenti, anche nei percorsi successivi. L'intento è duplice: da una parte si vogliono affrontare le

[1] Dipartimento di Informatica, Università degli Studi di Verona e ITIS "G. Marconi" di Verona.

questioni inerenti la didattica a partire da una rassegna sui temi fondamentali della disciplina; dall'altra si vuole proporre una riflessione sulla natura dell'informatica che consenta di integrare le eterogenee conoscenze pregresse dei futuri insegnanti. Nei diversi moduli che costituiscono gli insegnamenti, sono stati affrontati alcuni temi collegati alle aree suddette con il proposito di esaminare gli aspetti didattici e metodologici richiamando, quando necessario, i principi fondamentali [5, 8].

In questa relazione riportiamo quanto è emerso dall'esame delle attività didattiche e dai questionari con riferimento specifico all'area dei linguaggi e della programmazione.

Prima di proseguire con la descrizione dell'esperienza, è utile riportare qualche considerazione sulla preparazione pregressa dei futuri insegnanti. Chi affronta un percorso di abilitazione possiede conoscenze tecniche che derivano da studi universitari diversi; alcuni corsisti hanno inoltre seguito corsi di formazione e specializzazione orientati alla didattica, e qualcuno ha conseguito un dottorato. In molti casi i candidati hanno alle spalle un'effettiva attività di docenza, spesso corposa e consolidata.

Stabilire le modalità e i termini del percorso di abilitazione è risultata essere una questione non banale. La sfida nasce dal riuscire a definire un percorso il più possibile costruttivo raccordando formazioni eterogenee e conoscenze talvolta solide ma orientate alla pratica e più carenti dal punto di vista dei fondamenti.

Le lezioni di didattica dell'informatica, infine, richiedono di riferirsi ai contenuti senza poterli affrontare frontalmente. Pertanto nello svolgimento dell'attività didattica si è scelto di presentare una selezione significativa dei contenuti e utilizzarli come "running examples", per richiamare al contempo i saperi minimi (e per tentare di allineare le competenze di base) senza perdere di vista la preparazione metodologica, che deve essere offerta ai corsisti nel miglior modo possibile.

Il fine, oltre al conseguimento dell'abilitazione, è quello di migliorare competenza e consapevolezza didattica, permettendo al futuro insegnante di assumere il ruolo di guida che gli compete nel processo di apprendimento.

8.1.1. I Ciclo di TFA

Nel A.A. 2012/13 a Verona si è svolto il percorso per la classe A042 (Informatica). Alla selezione hanno partecipato oltre cento candidati per

una decina di posti, ai quali si sono aggiunti alcuni "soprannumerari" (ovvero candidati provenienti da percorsi precedenti non completati).

Superata una certa "prudenza" iniziale, il rapporto tra docenti del percorso e corsisti si è sviluppato secondo una modalità rilassata e costruttiva, aperta allo scambio e alla discussione. Riguardo all'insegnamento dei linguaggi di programmazione sono emersi alcuni punti critici. Infatti, accanto ad una padronanza dei principi fondamentali della programmazione generalmente adeguata alle necessità della didattica, si è riscontrata una mancanza di "ampiezza" nella conoscenza dei linguaggi di programmazione: pochi linguaggi conosciuti (a volte solo uno o due) e scarsa percezione delle caratteristiche espressive dei diversi linguaggi e paradigmi e della rilevanza di questi aspetti nelle attività di risoluzione computazionale dei problemi.

Alcune vivaci discussioni in aula hanno avuto effetti positivi: superate alcune resistenze iniziali, i corsisti hanno accolto favorevolmente l'allargamento d'orizzonte proposto. Nonostante ciò, i tempi ristretti a disposizione non hanno consentito di acquisire una buona padronanza dei nuovi punti di vista e gli elaborati prodotti dai corsisti come attività conclusiva per l'insegnamento di *"Fondamenti e Programmazione"* hanno spesso mostrato un'attitudine conservativa.

8.1.2. PAS - Classi A042 e C310

Nell'anno accademico seguente, 2013/14, l'Ateneo veronese ha ospitato i percorsi PAS per le classi *A042 (Informatica)* e *C310 (Laboratorio di Informatica Gestionale)* con una trentina di partecipanti in totale, con una netta prevalenza della prima classe di concorso.

I due percorsi, pur avendo strutture e finalità diverse, hanno condiviso alcuni momenti di formazione: in particolare, i fondamenti della disciplina e i linguaggi di programmazione.

Ai corsisti è stato chiesto di rispondere ad un questionario informativo anonimo e volontario e di svolgere alcune attività in itinere che sintetizziamo qui di seguito.

Nel questionario proposto (al quale hanno risposto circa i 2/3 dei corsisti) erano richieste alcune informazioni generali e specifiche nelle diverse aree corrispondenti agli insegnamenti del percorso disciplinare.

Come previsto, i titoli di studio sono piuttosto eterogenei, con una netta prevalenza delle lauree magistrali (o equivalenti) in Informatica,

Ingegneria Elettronica e Matematica, ma compaiono anche Fisica, Ingegneria delle Telecomunicazioni e Aeronautica.

I corsisti "mediamente" dichiarano di conoscere circa 3 linguaggi di programmazione, ma al più uno come "esperti"[2]. Prevalgono nettamente C++, C e Pascal, ma emergono anche Visual Basic, Java, Javascript e Python.

I paradigmi imperativo e ad oggetti sono conosciuti dalla gran parte dei corsisti, ma meno della metà si dichiara esperto del paradigma imperativo e una minima parte di quello ad oggetti. Gli altri paradigmi risultano di fatto trascurabili.

Le attività in itinere assegnate ai corsisti nel periodo di lezione hanno fornito informazioni più ampie, anche se più complesse da riportare in sintesi. Riguardo ai linguaggi e alla programmazione è stato chiesto ai corsisti di descrivere in chiave didattica e introduttiva

- le caratteristiche essenziali di un linguaggio imperativo e dei tipi di dati fondamentali,
- gli aspetti "funzionali" e "ad oggetti" presenti nei linguaggi di programmazione a loro noti.

Dall'analisi degli elaborati emerge una conoscenza non sempre adeguata dei linguaggi imperativi e una certa difficoltà a sviluppare l'argomento in modo didatticamente semplice ed efficace. Inoltre i linguaggi presi come riferimento, che sono probabilmente quelli meglio noti ai corsisti, spesso mal si prestano a spiegazioni introduttive per la loro intrinseca complessità. È il caso, ad esempio, di linguaggi come C e C++.

Emergono una percezione del linguaggio di programmazione come mero strumento e una limitata coscienza dei principi fondamentali che ne ispirano la concezione. Di conseguenza, frequentemente appaiono trascurate le potenzialità espressive e il valore formativo dei diversi linguaggi [6, 7, 8].

È particolarmente significativo inoltre che, malgrado la grande diffusione del paradigma "orientato agli oggetti", le caratteristiche essenziali dei linguaggi che a questo modello si ispirano non siano pienamente comprese e, di conseguenza, siano generalmente ignorate le relative peculiarità e le problematicità.

---

[2] Le opzioni di risposta, volte all'autovalutazione delle proprie competenze erano le seguenti: nulla, base, intermedio, esperto.

Infine, si osserva che i corsisti, forse nuovamente per i tempi ristretti in cui si sono trovati ad operare, non sempre sono in grado di rielaborare autonomamente e criticamente i contenuti proposti.

8.1.3. II Ciclo TFA (classe A042) e PAS (classe C310)

Consideriamo per finire i percorsi dell'anno accademico corrente: il PAS della *classe C310 (Laboratorio di Informatica Industriale)* con circa trenta corsisti e il TFA della *classe A042 (Informatica)* con più di venti corsisti. Per questi ci limitiamo ad alcune considerazioni preliminari basate su una parziale analisi dello svolgimento dei percorsi stessi.

Si confermano, pur con le dovute cautele, le considerazioni già fatte per gli anni precedenti riguardo la conoscenza concreta dei linguaggi di programmazione e dei paradigmi, la padronanza dei principi fondamentali della programmazione e dei linguaggi.

Nella classe A042 si nota una maggiore coerenza dei titoli di studio con la disciplina che si vuole insegnare: prevalgono nettamente le lauree magistrali in Informatica (e varianti) e la laurea magistrale in Ingegneria Elettronica.

Tra i linguaggi di programmazione conosciuti si evidenziano maggiormente quelli con una ricaduta "lavorativa" più importante e rispetto ai paradigmi non si notano differenze significative.

Non è stato possibile finora analizzare le attività svolte in itinere. Si osserva tuttavia una generale maggiore padronanza della disciplina, probabilmente grazie a percorsi di studio più recenti e maggiormente attinenti la materia. Alcuni aspetti fondamentali e didatticamente rilevanti riguardanti i linguaggi di programmazione emergono ancora con fatica. Ci si riferisce, ad esempio, ai diversi modelli cui fanno riferimento i paradigmi di programmazione e l'influenza che questi esercitano sui linguaggi stessi e sulle metodologie di risoluzione dei problemi.

8.2. Altre osservazioni

8.2.1. Sui linguaggi e i paradigmi

Come abbiamo già osservato i corsisti hanno esperienza di un insieme piuttosto ristretto di linguaggi di programmazione e la conoscenza di un linguaggio a un buon livello (almeno intermedio) si riscontra in un numero limitato di casi, e spesso è correlata direttamente

agli insegnamenti offerti agli studenti oltre che alla propria formazione scolastica e accademica.

È interessante notare che tra i linguaggi ai quali viene solitamente attribuito un particolare valore didattico solo il *Pascal* appare nella "parte alta della classifica" dei linguaggi utilizzati. *Python*, *Scratch*, *Ruby* sono scarsamente considerati o del tutto trascurati.

Meno sorprendente è la marginalità del paradigma funzionale. Anche se questi linguaggi sono scarsamente diffusi, questa lacuna nella preparazione degli insegnanti di scuola può avere una ripercussione sottile sulle potenzialità dell'insegnamento. Il paradigma funzionale, proprio per l'assenza della sequenzialità nella scrittura del programma, offre una prospettiva molto diversa rispetto al "programmare per istruzioni", e d'altra parte è fondamentale perché fornisce la struttura concettuale di molte tecniche fondamentali della programmazione e del *problem solving*.

8.2.2. Sui linguaggi per la didattica

La maggioranza dei candidati (quasi il 90%) ha già esperienza nell'insegnamento della programmazione, ma solo una minima parte di essi ha avuto la possibilità di esprimere un parere sulla scelta del linguaggio di programmazione da adottare nelle attività didattiche.

Dai risultati dei questionari emerge che, nelle scuole, il problema del "linguaggio didattico" è sentito, ma sembra che operativamente si privilegi una conservazione dello *status quo*. L'utilità didattica affiora solo in pochi casi: la precarietà, le docenze brevi, gli standard conservativi della scuola, non lasciano margine e possibilità di fare delle scelte libere e ragionate e il docente deve adattarsi alla situazione che incontra. Forse è anche per questo motivo che la riflessione critica sulla natura dei linguaggi fatica a farsi strada.

8.2.3. Sulla didattica della programmazione

Concludiamo queste osservazioni con i rilievi riportati dai corsisti sulle difficoltà che incontrano gli studenti nell'apprendimento della programmazione.

L'insegnamento della programmazione viene spesso introdotto attraverso una descrizione informale delle soluzioni per mezzo di *flowchart* e pseudo-codice per passare in un momento successivo all'implementazione nel linguaggio di programmazione adottato.

Gli aspiranti insegnanti mettono in relazione le difficoltà di questo processo con aspetti quali: difficoltà legate al ragionamento astratto, procedurale e logico, difficoltà nel passaggio dal problema all'algoritmo, scarsa attitudine alla risoluzione dei problemi.

8.3. Il dialogo tra Scuola e Università

La scelta di affidare ad una collaborazione tra Scuola e Università i percorsi di formazione degli insegnanti della scuola secondaria ha, o dovrebbe avere, un significato profondo.

Nella Scuola operano insegnanti impegnati ad accompagnare studenti diversi per carattere, indole e predisposizione, in un percorso che può essere impegnativo, ma anche stimolante e divertente. Un insegnante si trova ad affrontare frequentemente situazioni nuove, inaspettate, difficili e nelle quali deve intervenire mettendo in gioco preparazione, esperienza, creatività.

L'Università è il teatro di percorsi e attività volte al consolidamento e all'avanzamento del sapere. I docenti e i ricercatori, persone attive nella ricerca, specializzate in settori all'avanguardia nelle applicazioni e nei fondamenti, possono proporre prospettive non scontate anche per l'insegnamento di base. Anche se l'insegnamento è spesso considerato un impegno accessorio nell'attività complessiva di un accademico, esiste tra ricerca e didattica un rapporto dialettico. È difficile concepire una buona didattica, finalizzata quindi non ai soli contenuti, ma alla formazione metodologica, senza coinvolgere capacità di formalizzazione rigorosa, di versatilità mentale e di *problem solving*, attitudini che si sviluppano propriamente soprattutto facendo ricerca attiva.

Il confronto, la collaborazione e lo scambio di esperienze tra Scuola e Università creano un terreno prolifico per offrire migliori strumenti per la formazione dei nuovi insegnanti. L'Università si propone come il luogo privilegiato per favorire questo incontro e sperimentare e ricercare nuove soluzioni e metodi.

Qui di seguito descriviamo alcune situazioni dove è possibile sviluppare un positivo scambio tra le due istituzioni.

8.3.1. La meta-lezione: insegnare ad insegnare

Le persone coinvolte nelle lezioni frontali dei percorsi abilitanti sono esse stesse insegnanti, con esperienza sia nell'ambito della docenza universitaria, sia nell'ambito scolastico.

L'insegnamento accademico rispetto a quello scolastico presenta differenze strutturali e culturali, che riguardano in buona parte lo snaturamento della lezione frontale. Con grande variabilità da situazione a situazione, non sempre le condizioni al contorno (numero di studenti, frequenza delle lezioni, numero delle ore di lezione) sono in grado di ricostruire la situazione "classe" in ambiente accademico, ovvero la situazione in cui il docente offre il proprio insegnamento in modo personalizzato, costruendo nel tempo un dialogo con l'identità globale degli studenti (che si riconoscono e identificano in un gruppo di apprendimento) e con lo studente specifico. Vengono così a mancare le condizioni per seguire il percorso di apprendimento dello studente che, uscito dalle scuole superiori, viene generalmente considerato autonomo. I contenuti assumono un maggior rilievo e si perde attenzione per la metodologia pedagogica. Sul versante Scuola, negli ultimi anni si osserva la volontà di modernizzare e mitigare la lezione tradizionale con l'introduzione parziale di nuovi stili educativi come la *flipped classroom*, l'apprendimento cooperativo e altri approcci didattici, dove lo studente gioca un ruolo attivo.

Non sorprende dunque che le differenze di metodo, oltre a quelle nei contenuti e negli obiettivi, possano contribuire a rendere difficile il passaggio dalla scuola superiore all'università.

Avanziamo un'ipotesi. La collaborazione nei percorsi per la formazione degli insegnanti può fornire anche l'occasione per favorire un naturale e proficuo filo conduttore tra scuola superiore e università, integrando aspetti metodologici e valorizzando in particolare il contributo proprio della Scuola, dove il rapporto tra docente e studente è basato su una continua e costante interazione.

Un'interessante spunto operativo ci viene offerto in ambito universitario dall'esperienza della *didattica trasversale*.

Insegnamenti di Informatica sono, di fatto, impartiti in contesti diversi dai corsi di laurea di tipo tecnico e scientifico. In questi casi si incontrano studenti con formazione non tecnica, spesso puramente

umanistica; per riuscire a trasmettere conoscenze e competenze significative in questi contesti è necessario da parte del docente procedere a una vera parafrasi del metodo di spiegazione per declinare il più possibile la comunicazione dei contenuti, senza ovviamente snaturarli, in direzione della formazione dello studente.

Tale lavoro sul metodo è correlato a quello che deve mettere in atto il docente della scuola secondaria per rivolgersi a studenti più giovani che, pur avendo scelto studi tecnico-scientifici, non hanno una formazione adeguata o per fattori cognitivi legati all'età o per fattori legati ad un particolare curriculum scolastico, per poter affrontare alcuni argomenti avanzati. In entrambi i casi, esiste la necessità di insegnare materie "ostiche" in termini "amichevoli".

La condivisione di questa esperienza di didattica trasversale con i nostri candidati rappresenta una prima efficace metodologia di metalezione volta al fine di spiegare il metodo didattico stesso.

8.4. Conclusioni

Le esperienze che abbiamo descritto in questa relazione ci permettono di trarre alcune conclusioni e soprattutto di fare delle riflessioni *a posteriori* al fine di perfezionare la nostra metodologia in questo ambito.

I percorsi per la formazione degli insegnanti che si sono avviati in tutta Italia a partire dall'anno accademico 2012/13 si sono svolti in un contesto non particolarmente favorevole. Procedure d'avvio difficoltose, incertezze nella normativa, problemi di coordinamento tra le diverse istituzioni coinvolte, ristrettezza di tempi e risorse, incertezza nelle prospettive. Questa situazione ha reso particolarmente impegnativo lo sviluppo delle attività didattiche e di tirocinio. Di conseguenza in alcune situazioni gli aspetti formali e organizzativi hanno in parte messo in ombra gli aspetti positivi e sostanziali dei percorsi.

Pur tenendo conto della complessità e problematicità del contesto, ci sembra di poter affermare che i percorsi si sono sviluppati in modo generalmente positivo. Anche se molti spunti interessanti, questioni stimolanti, aspetti innovativi e sperimentali non si sono potuti espandere in modo soddisfacente nell'immediato, speriamo che possano "dar frutto" in un momento successivo.

8.4.1. Aspetti didattici

La didattica disciplinare è stata fortemente condizionata dalla eterogeneità nella conoscenze pregressa dei partecipanti. A questo proposito, riprendendo quanto precedentemente esposto in questo stesso documento e semplificando, possiamo individuare due scenari prevalenti nella formazione dei corsisti:

I. da una parte percorsi di laurea non specifici, dove la disciplina ha un ruolo strumentale, a volte marginale, e manca solitamente un'efficace visione fondazionale dell'informatica come disciplina indipendente,
II. dall'altra percorsi universitari specifici, non sempre recenti, nei quali l'aspetto fondazionale è stato in qualche misura studiato, ma è mancata la possibilità di approfondire gli aspetti didattici.

Di conseguenza la definizione dei contenuti e dei metodi da sviluppare nei corsi disciplinari ha presentato non poche difficoltà.

I docenti hanno dovuto in più di un'occasione adattare (e a volte reinventare) la didattica per adattarsi alle situazioni che man mano emergevano. E ai corsisti è stata richiesta flessibilità e disponibilità per affrontare in tempi brevi (almeno a livello introduttivo) temi fondamentali che nella loro formazione non avevano trovato spazio.

Più volte docenti e corsisti si sono trovati d'accordo nell'esprimere il desiderio di dare più ampio respiro nello svolgimento delle attività didattiche.

8.4.2. Prospettive future

Alla luce delle problematicità e delle potenzialità incontrate in questi anni, ci siamo chiesti come e dove sia necessario intervenire per riuscire a formare insegnanti non soltanto preparati tecnicamente, ma anche in grado di affrontare una realtà critica ed in continuo divenire come la Scuola italiana dove la disciplina Informatica non ha ancora trovato una collocazione soddisfacente.

Da quanto abbiamo osservato "quantitativamente" (attraverso l'esperienza diretta, i questionari, le attività in itinere) emerge la necessità di predisporre un retroterra fondazionale e didattico che permetta al futuro insegnante di sostenere con sicurezza, competenza e flessibilità lo studente nell'esplorazione del sapere informatico.

Nasce l'urgenza di individuare nei curricula universitari che mirano alla formazione del futuro insegnante, percorsi che permettano di creare un terreno culturale più adeguato alla professione.

Ma per poter realizzare concretamente queste prospettive è necessario dare stabilità strutturale ai percorsi di formazione degli insegnanti e consentire una più ampia collaborazione tra *Scuola* e *Università* negli ambiti dell'aggiornamento, della ricerca e della sperimentazione.

Infine, riferendoci in modo più preciso all'*Informatica*, ci auguriamo che in tempi brevi anche in Italia, come già avviene in molti contesti internazionali, sia riconosciuto a questa disciplina quel ruolo fondamentale e autonomo che è indispensabile per tenere il passo in una società pervasa dalle *tecnologie dell'informazione e della comunicazione* delle quali l'*Informatica* è la scienza fondante [4].

Ringraziamenti

La presente relazione ha tratto spunti, arricchimenti e stimoli dagli incontri con i diversi componenti del Gruppo di Lavoro "Informatica e Scuola" del GRIN in particolare in occasione dei workshop. Il lavoro è stato possibile anche grazie al sostegno del Coordinamento dei percorsi PAS e TFA dell'Ateneo veronese, alla collaborazione dei docenti che hanno contribuito agli stessi e, ovviamente, alla disponibilità attiva dei partecipanti.

Bibliografia

[1] AA. VV., *Psychology of Programming Interest Group Annual Conference,* 2014 .

[2] DIJKSTRA, E, W., *On the Cruelty of Really Teaching Computing Science* (EWD-1036). E.W. Dijkstra Archive. Center for American History, University of Texas at Austin. (1989).

[3] FLORES D'ARCAIS, N., *La ricerca Pedagogica,* Ed. Università di Padova (1975).

[4] GANDER, W., ET AL., *Informatics education: Europe cannot afford to miss the boat,* ACM, http://europe.acm.org/iereport/ie.html (2013).

[5] HAZZAN, O., LAPIDOT, T., RAGONIS, N., *Guide to Teaching Computer Science,* Springer (2014).

[6] MARTINI, S., *Informatica: Elogio di Babele,* Studi offerti a Alessandro Perutelli. Aracne editrice (2008).

[7] MARTINI, S., *Lingua Universalis,* Annali della Pubblica Istruzione, 4: 65—70 (2012).

[8] MIROLO, C., *A Present-Day "Glass Bead Game": A Framework for the Education of Prospective Informatics Teachers Inspired by a Reflection on the Nature of the Discipline,* In: Informatics in Schools. Teaching and Learning Perspectives. Springer International Publishing, p. 138-149 (2014).

[9] MENDELSOHN, P., GREEN T., BRNA, P., *Programming Languages in Education: The Search for an Easy Start,* In Psychology of Programming, pp. 175-200 (1990).

9. Come motivare i nativi digitali all'uso della linea di comando

Vincenzo Del Fatto, Gabriella Dodero[1]

Immaginate di dover imparare a mungere una mucca,
e di avere 15 giorni di tempo per farlo.
Vi vengono offerte due alternative:
1. Un corso di anatomia comparata
2. Un tirocinio presso un contadino
Quale sceglierete?

Quanti di noi posti di fronte a questa alternativa sceglierebbero la prima proposta, e quanti la seconda? Noi non abbiamo dubbi. Per una "abilità" pratica, nulla vale quanto apprenderla sotto la guida di un esperto, lavorando dal contadino come apprendista mungitore; nessun corso teorico, nemmeno se fosse svolto dai migliori docenti, potrebbe dare risultati confrontabili.

È così che, nella bottega dove l'apprendista affianca il maestro, si sono tramandati i mestieri artigianali, ma non solo: è così che le "botteghe" italiane del Cinquecento hanno sfornato capolavori e creato scuole artistiche, con Raffaello che impara dal Perugino, e Tiziano dal Bellini. Anche l'arte quindi si impara praticandola, a lungo, e sotto la supervisione di un maestro.

E l'informatica? Da chi conviene impararla? E soprattutto, come si fa a imparare l'informatica?

Ora, Donald Knuth ha intitolato la sua opera "The Art of Computer Programming", la monografia ritenuta uno dei migliori 100 lavori scientifici dello scorso secolo, e questo potrebbe farci riflettere che,

[1] Libera Università di Bolzano

forse, converrà imparare l'informatica come l'arte, andando a bottega da un maestro, e facendo gli apprendisti, come fecero a suo tempo i grandi artisti del Cinquecento.

Non so quanti informatici, compresi noi che scriviamo e voi che leggete, abbiano imparato a suo tempo "l'arte" facendo gli apprendisti da un maestro. Concorderete comunque che avere di fronte 25 adolescenti, in classe per 50 minuti, o anche 100 studenti universitari, in un'aula ad anfiteatro per 45 minuti, non crea le condizioni di partenza per un proficuo rapporto maestro/apprendista. L'insegnante è distante, gli studenti sono svogliati ed annoiati, o peggio, sono demoralizzati dai continui messaggi di errore che ricevono dal computer, quando si va finalmente in laboratorio ad usarlo.

Se mettiamo in discussione i processi di insegnamento dell'informatica, anche quelli impartiti a suo tempo a noi stessi, è perché abbiamo iniziato un percorso alternativo, da esploratori, e nel tempo abbiamo raggiunto risultati che vale la pena di condividere. Non siamo partiti da soli, c'è una letteratura pedagogica e di "Computer Science Education" che ci ha mostrato la direzione, e soprattutto ci sono colleghi e amici, partiti prima di noi, che ci hanno fatto inizialmente da guida. Poi, la strada si è allargata, ed abbiamo proseguito con le nostre gambe. Vi invitiamo a seguirci in queste pagine, e poi, forse, vorrete partire ad esplorare anche voi.

Nei paragrafi seguenti, troverete la descrizione della metodologia seguita, le risorse didattiche "open access" che abbiamo predisposto per le scuole, e le prime esperienze di loro utilizzo in alcune scuole della Provincia Autonoma di Bolzano. Condivideremo infine alcune considerazioni conclusive su questa esperienza, e su come contiamo di portarla avanti.

9.1. Extreme apprenticeship, apprendistato estremo

La pedagogia ha definito una metodologia specifica, detta apprendistato cognitivo [2], che prevede per l'appunto di organizzare i processi di apprendimento cercando di ricreare le condizioni per l'apprendimento come tra maestro ed apprendista. Originariamente proposta con esempi riferiti alle abilità di “leggere, scrivere e far di conto”, si basa sulla ricerca ed esplicitazione di ciò che Scardamalia aveva

chiamato in [12] "knowledge-telling strategies" ovvero strategie per raccontare la conoscenza. Collins individua le seguenti fasi chiavi dell'apprendistato, da riportare nell'insegnamento:

- modeling, l'osservazione del maestro al lavoro;
- scaffolding, il supporto offerto dal maestro all'apprendista;
- fading, la progressiva assunzione di responsabilità da parte dell'apprendista; e infine
- coaching, che riassume in se tutte le attività di supervisione, compresa la valutazione del lavoro, il feedback relativo a tale valutazione, ed ogni forma di incoraggiamento nella prosecuzione dello stesso.

Un adattamento di questi concetti all'insegnamento della programmazione alle matricole universitarie è stato effettuato dal gruppo di ricerca RAGE dell'università di Helsinki, definendo una metodologia specifica, l'Extreme Apprenticeship (abbreviata XA), che è impiegata dal 2010 con successo a Helsinki nei primi anni universitari [14]. Con l'XA si sono ottenuti risultati eccellenti nell'apprendimento della programmazione da parte degli studenti, raffinando la metodologia nel tempo e ottenendo analoghi successi anche nell'insegnamento della matematica [5].

Il gruppo di Helsinki riprende i principi dell'apprendistato cognitivo e li declina intorno alle attività di insegnamento/apprendimento della programmazione, vediamo come:

La fase di modeling serve a mostrare allo studente-apprendista come l'esperto svolge il compito: è infatti noto da studi pedagogici [2] che l'osservazione rappresenta la base per la creazione di un modello dell'attività da svolgere, nella mente dell'apprendista. Perché questo modello si costruisca correttamente, occorre presentare esempi concreti e portarli in fondo, magari ragionando ad alta voce ("thinking aloud") ed esplicitando così i processi mentali e i meccanismi che portano a compiere le scelte migliori.

La fase di scaffolding si realizza facendo svolgere all'apprendista compiti di difficoltà inizialmente semplice, e via via crescente, sotto la supervisione del maestro. Riprendendo concetti di Vygotskiĭ [16] sullo scaffolding, il maestro fornisce all'apprendista, quando è in difficoltà, solo le informazioni necessarie a proseguire, ad esempio indicando in che cosa consiste l'errore, ma mai fornendo la soluzione esplicitamente. Presuppone quindi un feedback bidirezionale e continuo tra ciò che l'apprendista sta facendo, e come il maestro lo valuta. Sempre

seguendo le indicazioni di Vygotskiĭ, la difficoltà graduale e progressiva degli esercizi mantiene l'apprendista sempre nella "zona di sviluppo prossimale" che è quella in cui avviene l'apprendimento. Fornire esercizi difficili, che esulano dalla zona di sviluppo prossimale, renderebbe impossibile la loro soluzione, ed avrebbe come risultato la caduta della motivazione da parte dell'apprendista.

La fase di fading si realizza progressivamente, rendendo l'apprendista via via più autonomo e in grado di lavorare a compiti di difficoltà maggiore, anche senza la supervisione costante del maestro.

Queste fasi portano ad identificare due nuovi "valori" alla base dell'XA nell'insegnamento dell'informatica:

1. Si impara attraverso la pratica. Bisogna continuare a fare pratica, per tutto il tempo necessario a impadronirsi dell'abilità. In altri termini, non esiste un tempo da trascorrere in laboratorio che vada bene per tutti, ciascuno deve poter fare tutta la pratica di cui ha bisogno.
2. Feedback continuo tra maestro ed apprendista e viceversa. L'apprendista riceve feedback sul proprio progresso, e il maestro osserva il progresso dell'apprendista, e si congratula per i suoi successi. Non deve esistere un lavoro svolto dall'apprendista su cui il maestro non abbia espresso la sua valutazione.

Da questi valori di base, i colleghi finlandesi hanno derivato un decalogo di buone pratiche su cui dal 2010 hanno basato la didattica dell'informatica nei primi anni di corso (compreso l'insegnamento nei cosiddetti corsi di servizio, cioè a quegli studenti che non sono iscritti a un corso di studio in Informatica):

1. L'efficacia delle lezioni teoriche per insegnare a programmare è discutibile, le eventuali lezioni dovrebbero limitarsi al minimo, per poter iniziare gli esercizi al più presto.
2. Se si fanno lezioni teoriche, il collegamento con gli esercizi deve essere esplicitato e ben visibile.
3. Gli esercizi devono iniziare alla prima settimana di lezione, e già nella prima settimana si devono risolvere un numero significativo di esercizi. Questo stimola fin dall'inizio la motivazione a seguire il corso.
4. Gli esercizi si completano in laboratorio, alla presenza del maestro, che fa scaffolding. Ci deve essere a disposizione un maestro, fino a quando tutti gli apprendisti abbiano completato gli esercizi.

5. Il lavoro deve essere spezzato in molti, piccoli passi, svolti in altrettanti esercizi. In questo modo ogni apprendista riesce a misurare lo stato di avanzamento del proprio processo di apprendimento.
6. Gli esercizi sono la cosa più importante, la maggior parte degli studenti deve riuscire a risolvere tutti o quasi gli esercizi.
7. Gli esercizi devono essere numerosi, al punto da essere talvolta ripetitivi.
8. Il testo degli esercizi deve essere chiaro, in particolare riguardo a come si deve iniziare il lavoro, e in quali circostanze si è sicuri di averlo concluso correttamente.
9. Gli apprendisti sono stimolati a cercare ulteriore documentazione, al di là di quanto fornito dal maestro.
10. Nella fase di scaffolding il maestro sottolinea gli aspetti di qualità, mentre sostiene gli apprendisti – sviluppare un prodotto di qualità non richiede nessuno sforzo aggiuntivo.

Sottolineiamo che alcuni di questi punti rappresentano una vera e propria rivoluzione rispetto al modo in cui si è insegnata l'informatica, per lo meno in molte università italiane, e invitiamo gli scettici a controllare la posizione nei ranking internazionali degli informatici di Helsinki. Non stiamo evidentemente parlando di un dipartimento che per la didattica ha sacrificato la ricerca!

Il segreto sta, a nostro avviso, nel punto 10 dell'elenco di cui sopra: il docente è responsabile della qualità dell'apprendimento, e se insiste perché siano prodotte soluzioni di qualità, gli studenti impareranno a produrle; se si accontenta che in qualche modo "il codice giri", gli apprendisti si adegueranno, e nessun insegnamento, nei semestri successivi, potrà insegnare la produzione di software di qualità, se sono state prese abitudini ed interiorizzati processi mentali che non la prevedono. Tra le molte obiezioni che sono state sollevate al precedente decalogo, la più consistente si riferisce alla praticabilità ed alla "scalabilità" della metodologia per quanto riguarda i punti 4 e 6. Come garantire la presenza di un docente in laboratorio, pronto a fare scaffolding e fornire feedback, per "tutto il tempo che serve" al meno veloce tra gli apprendisti, per completare un carico di lavoro significativo? Quante ore di apertura di laboratorio devono essere garantite, e quanti docenti, assistenti, tecnici qualificati devono essere presenti? In breve: si può scalare questa metodologia, dalla bottega artigiana, con

rapporto maestro/apprendista pari al più a ¼, a una situazione scolastica o universitaria dove il rapporto, nei casi più favorevoli, sarà 1/20?

La soluzione data a Helsinki prevede di fornire scaffolding da parte di studenti degli anni superiori, adeguatamente istruiti, e che riceveranno crediti universitari come "soft skills" in base al lavoro di scaffolding effettivamente svolto. Una trattazione dettagliata di come è stata impostata l'organizzazione didattica si trova in [15], ed esula dagli scopi di questo lavoro, ma ne consigliamo vivamente la lettura a chiunque debba assumersi responsabilità organizzative nell'insegnamento di materie informatiche (dirigente scolastico, o direttore di corso di studio in Informatica).

Dato il costo in termini di risorse che viene ipotizzato dalle buone pratiche 4 e 6, abbiamo sperimentato a Bolzano l'efficacia di una forma di scaffolding blended, che preveda

a) l'utilizzo di una piattaforma di e-learning per la consegna degli esercizi, ovviando così al numero limitato di ore in cui il docente può essere presente in laboratorio per fornire scaffolding faccia a faccia;
b) la correzione frequente degli esercizi, via via che vengono consegnati, fornendo un feedback di tipo valutazione formativa, sempre sulla piattaforma, che poi lo inoltra automaticamente allo studente via email;
c) la possibilità di riconsegnare un esercizio svolto non correttamente, per un numero illimitato di volte e senza che la votazione finale ne risulti penalizzata, fino a una data di scadenza prefissata (in genere circa due settimane).

Il problema più significativo che si incontra passando da una didattica in presenza ad una didattica blended, è la mancanza di tutte quelle interazioni sociali che la "bottega artigiana" utilizza implicitamente per migliorare il processo di apprendimento [2]. Tra queste, ci preme sottolineare l'aspetto motivazionale, che nei giovani universitari, e ancora di più negli studenti della scuola superiore, spesso è determinante per l'abbandono degli studi. Il crollo della motivazione a proseguire è addirittura drammatico nei percorsi MOOC, dove si stima che circa il 10% degli iscritti riesca a concluderli con successo [8], e dove si è verificato che la percentuale aumenta qualora si riesca ad instaurare una forma di socializzazione, virtuale o reale, tra i partecipanti [13].

Per quanto riguarda i contenuti di un insegnamento come Sistemi Operativi, che prevede in particolare un'attività di laboratorio sullo

shell scripting, c'è un ulteriore ostacolo motivazionale da tenere presente. Tra i cosiddetti "nativi digitali" [10], che comprendono ormai la popolazione delle scuole secondarie di secondo grado, e tutti o quasi gli studenti universitari, l'interazione con il computer/tablet/smartphone è immediata, ma si realizza quasi esclusivamente con interfacce grafiche, dando gli input sul touch screen e tuttalpiù, su un portatile, con il mouse. Passare ad una modalità di interazione con il computer esclusivamente testuale, come imposto dall'uso di una shell, e come fanno abitualmente i sistemisti, rappresenta qualcosa di faticoso, molto tecnico e quindi noioso (prima che succeda qualcosa, occorre digitare un bel po'...). Il rischio del rifiuto dell'argomento in toto è quindi da tenere ben presente.

Abbiamo preso in considerazione il problema della caduta della motivazione, e del conseguente abbandono, cercando di contrastarlo con alcuni accorgimenti, descritti in [3]. Questi accorgimenti fanno capo al concetto di "gamificazione" dell'ambiente di apprendimento, che andremo a descrivere nella prossima sezione.

9.2. Il canale YouTube di Sistemi Operativi

L'esperienza didattica maturata a Unibz nel corso di Sistemi Operativi con la metodologia Extreme Apprenticeship "blended", è stata dapprima descritta in [4]. Negli anni accademici successivi, il materiale didattico è stato raffinato, ma senza modificarlo in maniera significativa rispetto alla prima edizione del corso, mentre si è progressivamente ampliato l'aspetto di "gamificazione" dell'ambiente di apprendimento.

Si è per prima cosa verificato che l'approccio XA blended risultava praticabile, anche e soprattutto per quei gruppi di studenti che tradizionalmente trovano più difficoltà nel percorso universitario, quali studenti lavoratori, giovani extracomunitari (purtroppo spesso carenti nella preparazione di base linguistica e matematica), pendolari che non riescono a frequentare tutte le ore di lezione. La scansione settimanale degli esercizi obbligava questi studenti a lavorare sul materiale didattico fin dal primo giorno di lezione, prevenendo l'accumularsi del materiale da studiare in prossimità dell'esame. La difficoltà crescente a piccoli passi degli esercizi rendeva possibile risolverne alcuni, poi alla prima difficoltà chiedere un semplice feedback del docente via email, e così superare la difficoltà e risolvere i rimanenti esercizi.

Le statistiche di superamento del corso, oltre che le impressioni soggettive del docente, parlano chiaro in proposito: rispetto alla metodologia di insegnamento tradizionale in università, fatta di lezioni frontali, ed assegnazione di progetti complessi di fine corso, XA riesce ad innalzare il livello medio degli studenti, agendo in particolare sull'innalzamento delle prestazioni degli studenti "meno bravi".

Viceversa, per tenere "agganciati" al corso i più bravi, già esperti fino dalla scuola superiore nell'uso della shell e nello scripting, ed evitare che si annoiassero, il docente metteva in palio un premio simbolico settimanale, destinato a chi avesse risolto per primo tutti gli esercizi giusti, la "medaglia Speedy Gonzales" d'oro, argento e bronzo. Senza nessun valore sul punteggio finale del corso, ha rappresentato comunque un premio ambito, che alcuni ex studenti ricordano anche a distanza di anni!

In Figura 9.1 mostriamo il testo di un esercizio, in cui sono stati evidenziate con colori diversi le parti di testo in cui si illustra un concetto teorico nuovo (giallo), in cui si chiede di svolgere una azione (azzurro) ed in cui si forniscono spiegazioni supplementari e consigli (arancio). Si nota come la parte teorica risulti diluita all'interno del testo degli esercizi, ed inframmezzata con esempi, che si chiede di completare o sviluppare ex novo. Con il colore verde, infine, si identifica la formula conclusiva dell'esercizio.

Il passaggio successivo, per supportare al meglio la motivazione e l'aspetto di "learning by doing" negli studenti non frequentanti, è stato di predisporre una serie di brevi video didattici, uno per esercizio. Infatti, secondo Peters [9], visualizzare una azione attiva le stesse aree cerebrali che si attiverebbero compiendo l'azione stessa. Il maestro quindi è il protagonista del video, e l'apprendista, guardandolo, è incoraggiato a ripetere quanto vede e a proseguire svolgendo nuovi esercizi.

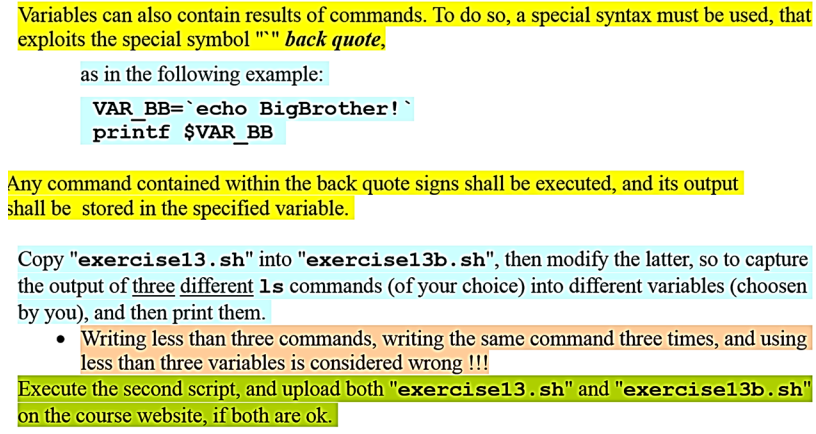


Fig. 9.1. Testo di un esercizio.

I concetti essenziali intorno ai quali si è sviluppato il progetto del video sono basati sulle euristiche di Nielsen [7]. Una voce narrante illustra l'esercizio, mentre in basso si evidenziano tab di navigazione corrispondenti alla caratterizzazione del testo attualmente letto (i tab sono quindi Theory, Exercises, Tips, e in conclusione Wrap-up). L'immagine che compare sul terminale bash mostra il risultato delle operazioni descritte, e una colonna sulla sinistra visualizza i concetti chiave dell'esercizio: titolo, prerequisiti (resources), parole chiave. Il tono "fumettistico" in cui il docente, ed un gruppo di collaboratori, si trasformano a turno in avatar, e la presenza di elementi esteticamente accattivanti per aiutare la memorizzazione e ridurre la noia, sono stati introdotti appositamente per aumentare la motivazione.

Le Figure 9.2, 9.3, 9.4 illustrano alcune schermate dei video realizzati. Si noti che ciascun video è stato realizzato nella lingua scelta dal personaggio-guida, che ha letto il testo e ha fornito alcune proprie immagini. Circa un terzo dei video è quindi in italiano, un terzo in inglese, ed un terzo in tedesco. Ciò è tipico della Libera Università di Bolzano e del territorio circostante, in cui a scuola si imparano fin dalla scuola elementare le due lingue italiano e tedesco, aggiungendo successivamente l'inglese.

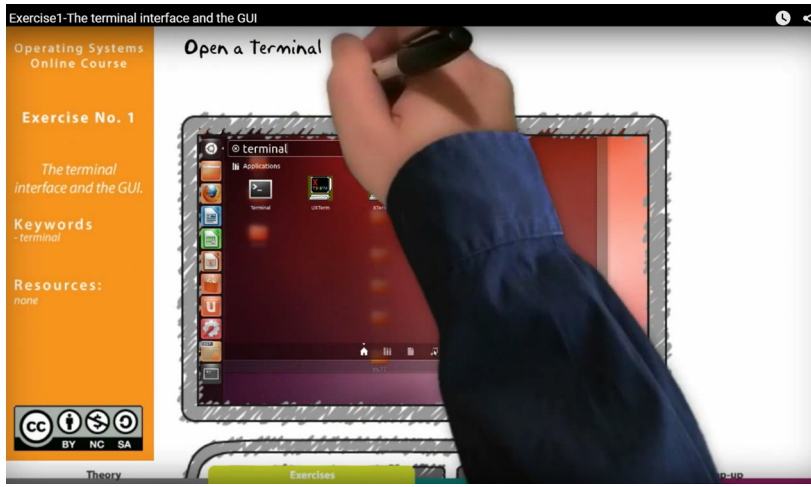


Fig. 9.2. Una sequenza con una animazione (la mano che scrive).

Fig. 9.3. L'avatar ed il fumetto.

Fig. 9.4. Elementi grafici, avatar, ed un personaggio di fantasia (il "Grande Fratello").

Infine i video realizzati (ad oggi 27) sono stati raccolti nel canale YouTube E3 OS VIDEOS, dove E3 sta per Easy, Effective, Enjoyable: le tre parole chiave del progetto. Il canale si trova alla URL https://www.youtube.com/user/e3osvideos/videos. I video sono distribuiti con licenza Creative Commons (CC-BY-NC-SA) e durano mediamente 3-4 minuti l'uno.

9.3. L'uso didattico: alcune esperienze a scuola

In collaborazione con alcune scuole dell'Intendenza Scolastica Italiana della Provincia Autonoma di Bolzano, ad inizio 2014 è partito un progetto di utilizzo di queste risorse in due Istituti Tecnici ed in un Liceo Scientifico opzione Scienze Applicate. Ogni insegnante ha usufruito dei materiali nella maniera ritenuta più idonea per le classi coinvolte, in particolare scegliendo solo alcuni video o proponendoli tutti, a seconda degli obiettivi specifici.

Al Liceo "Cantore", i video sono stati utilizzati come supporto nelle attività di laboratorio; negli istituti tecnici, invece, si è preferita una modalità didattica che avvicinasse maggiormente gli alunni all'ambiente "MOOC". Si riporta in Tabella 1.1 la griglia di progettazione, presentata in [11], elaborata dagli insegnanti dell'ITE "Battisti" e dell'ITSS "Galilei".

Possibile modello di operatività: Flipped classroom	*1.Fruizione dei contenuti del Mooc outisde-class da parte dei singoli studenti* *2.Relazione alla classe da parte dei singoli sui contenuti appresi* *3. Condivisione dell' esperienza con la classe in aula*	*Apprendimento individuale auto-organizzato* *Metacognizione e chiarimento* *Fase di peer-learning*
Metodologia	*1. Mooc – video online*	*Fruizione dei contenuti in modo individuale outside-class*
	2. Uso veicolare lingua L2 e straniera (CLIL)	*Fruizione di contenuti disciplinari in lingue diverse*
	3. Didattiche innovative	*Oltre il libro di testo: utilizzo del video*
	4. Peer-learning	*Condivisione, spiegazione e aiuto reciproco*
Obiettivi	*1. Metodologici*	*Apprendimento autonomo Learning by doing Revisione, metacognizione Autovalutazione*
	2. Contenuti: conoscenze e abilità della disciplina informatica	*Bash – basic shell di Linux*
	3. Linguistici	*Utilizzo di tre lingue in contesto*
	4. Etico-critica	*Comprensione delle possibili "manipolazioni" determinate dall'uso del coding*
	4. Orientamento	*In relazione alle possibili scelte universitarie*
Valutazione e autovalutazione (dopo l'esperienza)	*1. Focus group per l'analisi dell'esperienza in classe*	*Formalizzazione punti di forza e debolezza (studente)*
	2. Scheda di osservazione, tempi ed efficacia dell'apprendimento	*Valutazione da parte del docente*
	3. Analisi e conclusioni	*Docenti/ricercatori*

Tab. 9.1. La griglia di progettazione delle unità didattiche relative ai video.

La disponibilità dei video ha reso possibile l'utilizzo della cosiddetta "flipped classroom", in cui l'apprendimento individuale (la visione dei video e lo svolgimento degli esercizi, realizzato autonomamente a casa) viene seguito da una rielaborazione condivisa in classe.

Alcuni insegnanti di lingue, inizialmente non coinvolti, hanno manifestato interesse per la prospettiva di utilizzo dell'inglese e tedesco veicolare (metodologia CLIL), e per l'opportunità di sperimentare la lingua della professione informatica, che lo studente dovrà parlare sul posto di lavoro. La presenza del testo trascritto (nel fumetto) permette

all'allievo di concentrarsi sull'esercizio, senza ulteriori difficoltà linguistiche nella comprensione del parlato.

Da sottolineare anche la riflessione critica che il personaggio di "BigBrother" stimola sull'utilizzo consapevole della Rete: cosa potrà fare un "BigBrother" attraverso gli script?

9.4. Conclusioni

Le esperienze sviluppate ed ancora in corso che prevedono di utilizzare la metodologia Extreme Apprenticeship nel corso del 2014 e del 2015 stanno rapidamente aumentando.

In parallelo all'esperienza sopra descritta, una tesi di laurea di primo livello ha affrontato l'utilizzo dell'XA in una scuola professionale [1]. La scuola professionale, in Alto Adige come nel resto d'Italia, costituisce spesso un ambiente molto difficile, in cui alunni demotivati e frustrati da bocciature in altri tipi di scuola rappresentano la maggioranza. L'insegnamento della programmazione Javascript in queste scuole è difficilmente realizzato con successo, ciononostante l'esperienza si è conclusa più che positivamente, con quattro dei sette allievi partecipanti che si sono appassionati alla programmazione, al punto di sceglierla come propria professione futura.

Nell'autunno 2014 la Libera Università di Bolzano ha organizzato i corsi PAS per la classe di abilitazione 42/A Informatica, frequentati da 18 insegnanti delle Province Autonome di Trento e Bolzano (per la lingua italiana).

Gli Autori hanno riportato, in due corsi di didattica disciplinare, le esperienze realizzate con la metodologia XA in università e nelle scuole, incoraggiando gli insegnanti presenti a sperimentare con le rispettive classi qualche unità didattica con questa metodologia. Si sono quindi raccolti numerosi progetti di unità didattiche, erogate nelle varie scuole secondarie di provenienza dei corsisti. Le attività si prolungheranno fino al termine dell'anno scolastico (giugno 2015).

Si sono verificati i punti di forza dell'XA in numerosi contesti e al fine di sviluppare abilità informatiche di vario tipo (costrutti di selezione ed iterazione in diversi linguaggi, pascal, c#, java, nelle formule di Calc e MS Excel, sviluppo di pagine statiche in HTML e dinamiche in Ajax, insegnamento delle basi di dati, eccetera). L'aumento della

motivazione intrinseca, e la possibilità offerta a tutti i partecipanti di sviluppare le abilità previste nel tempo previsto, sono state riscontrate in tutti i contesti didattici.

Punti di forza verificatesi nelle scuole superiori, e mai riscontrati nelle esperienze universitarie, sono stati:

- La possibilità di relazionare agevolmente sui risultati scolastici con le famiglie: gli insegnanti hanno per ogni allievo un percorso chiaro e molto strutturato, dal quale evidenziare immediatamente "quanto" progresso ha fatto l'allievo e "quanto manca" per concluderlo;
- La possibilità di gestire senza difficoltà classi particolarmente vivaci, assegnando un insieme di task da eseguire che mantenesse concentrati sull'attività di laboratorio anche i più vivaci per tutta l'ora, senza annoiarli con spiegazioni.
- La possibilità di gestire in tempi diversi allievi di diverse velocità, ad esempio nelle scuole serali, dove si incontrano allievi di 20-30 anni, perfettamente a loro agio con le tecnologie, ed "esodati" over 50 rientrati in formazione e digiuni di uso del computer. Per questi ultimi è stato possibile, con tempi molto dilatati rispetto ai primi, ma senza particolari intoppi, raggiungere risultati insperati di alfabetizzazione tecnologica.

Punti di debolezza, peraltro noti, sono dati dalla "pesantezza" dell'attività di coaching e scaffolding, specialmente in classi numerose. La si è realizzata talvolta con l'aiuto in laboratorio di insegnanti tecnico-pratici, talvolta in modo blended, con l'uso di LMS come Moodle o Google Classroom. Sovente l'insegnante ha dovuto abbandonare l'obiettivo di dare un feedback per ogni attività, valutando a campione gli allievi "bravi" e concentrandosi su quelli meno bravi che necessitavano di maggiore supporto.

Il problema della scarsità di risorse (in questo caso, risorse umane) a disposizione nel mondo della scuola e dell'università rende da una parte pessimisti sulla possibilità che questa metodologia si diffonda su larga scala, ma d'altra parte stimola la realizzazione di sistemi automatici di auto-correzione, che alcuni insegnanti stanno sviluppando, ad esempio per i contenuti di ECDL, al fine di diminuire la necessità di continui feedback per gli esercizi di base.

I feedback testuali ricavati dai questionari di valutazione degli studenti in merito alle attività proposte sono stati, e sono tuttora, la

molla motivazionale che ha portato gli autori, ed a cascata gli insegnanti con loro formatisi, ad affrontare sempre nuove proposte didattiche con la metodologia XA.

Non abbiamo ancora un campione di utilizzatori statisticamente significativo, ma ci sembra che si possa già affermare che, se un gruppo di persone accetta consapevolmente di passare diversi fine settimana a correggere esercizi, la rispettiva motivazione intrinseca deve essere davvero superiore al valor medio della classe docente 42/A.

Bibliografia

[1] BARAZZUOL, L., *Extreme apprenticeship method in teaching programming in a Vocational course,* Tesi di Laurea in Informatica Applicata, Libera Università di Bolzano, ottobre 2014.

[2] COLLINS, A., SEELY BROWN, J., HOLUM, A., *Cognitive apprenticeship: Making thinking visible.* American educator, 15(3):6-11, 1991. http://www.21learn.org/archive/cognitive-apprenticeship-making-thinking-visible/.

[3] DEL FATTO, V., DODERO, G., GENNARI, R., MASTACHI, N., *Extreme Apprenticeship Meets Playful Design at Operating Systems Labs: A Case Study*, In: Methodologies and Intelligent Systems for Technology Enhanced Learning, Di Mascio T., Gennari R., Vittorini P., Vicari R., De La Prieta F. (eds). Volume 292, Advances in Intelligent Systems and Computing Series, Springer International Publishing, 19-26, (2014).

[3] DODERO, G., DI CERBO, F., *Extreme Apprenticeship Goes Blended: An Experience,* ICALT 2012, IEEE International Conference on, Advanced Learning Technologies, 2012, 324-326.

[4] HAUTALA, T., ROMU, T., RÄMÖ, J., VIKBERG, T., *Extreme apprenticeship method in teaching university-level mathematics,* Proceedings of the 12th International Congress on Mathematical Education, July 8-15, 2012, Seoul, Korea. International Commission on Mathematical Instruction (ICMI), 10 p.

[5] KNUTH, D.E., *The Art of Computer Programming,* Addison-Wesley, 1968.

[6] NIELSEN, J., *10 Usability Heuristics for User Interface Design*. http://www.nngroup.com/articles/ten-usability-heuristics/.

[7] PARR, C., *Mooc completion rates 'below 7%,* Times Higher Education, 9 may 2013 (online) http://www.timeshighereducation.co.uk/news/mooc-completion-rates-below-7/2003710.
[8] PETERS, D., *Interface Design for Learning: Design Strategies for Learning Experiences,* New Riders (2013).
[9] PRENSKY, M., *Digital Natives, Digital Immigrants*. On the Horizon 9 (5), 1-6, (2001).
[10] RAVANELLI F., DEL FATTO V., DODERO G., GENNARI R., MASTACHI N., FRANCESCHINI B., MACOLA C., *Un "Mooc a scuola", approcci e prospettive,* Atti del convegno DIDAMATICA 2014: Nuovi processi e paradigmi per la didattica, Napoli, 7-9 maggio 2014.
[11] SCARDAMALIA, M., BEREITER, C., *Fostering the development of self-regulation in children's knowledge processing,* In S. F: Chipman, J. W. Segal, & R. Glaser (Eds.), Thinking and learning skills: Vol. 2. Research and open questions , 563–577, (1985).
[12] THOMAS, L., HERBERT, J., *Even in a MOOC, students want to belong,* Social Science Space, September 4, 2014 (online) http://www.socialsciencespace.com/2014/09/even-in-a-mooc-students-want-to-belong/.
[13] VIHAVAINEN, A., PAKSULA, M., LUUKKAINEN, M., *Extreme apprenticeship method in teaching programming for beginners,* SIGCSE '11 Proceedings of the 42nd ACM technical symposium on Computer science education, 93-98, 2011, ACM New York, USA.
[14] VIHAVAINEN, A., PAKSULA, M., LUUKKAINEN, M., KURHILA, J., *Extreme apprenticeship method: key practices and upward scalability,* ITiCSE '11 Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, Pages 273-277, ACM New York, USA (2011).
[15] VYGOTSKIĬ, L.S., *Mind in society: The development of higher psychological processes,* Harvard Univ. Pr., 1978.

Ringraziamenti

Un grande ringraziamento a tutte le nostre "cavie", quelle che hanno affrontato Sistemi Operativi alla Laurea in Scienze ed Ingegneria dell'Informazione di Bolzano negli ultimi 4 anni accademici, e quelle dei corsi PAS 42/A del 2013/14. Ci siamo reciprocamente trasmessi l'entusiasmo per apprendere, per apprendere ad apprendere, per apprendere ad apprendere ad apprendere....

Grazie agli insegnanti che ci hanno aiutato nelle sperimentazioni didattiche nelle scuole di Bolzano e Brunico, all'Ispettore Valer e al prof. Bonani, che ci hanno sempre supportato, qualsiasi cosa proponessimo di sperimentare.

Grazie anche alle Co-autrici delle pubblicazioni citate, Rosella, Naomi e Francesca.

Infine, a Francesco e Nabil, compagni di avventure didattiche nel favoloso mondo dei Sistemi Operativi, un ringraziamento in finlandese: kiitos!

10. Progettare giochi interattivi a scuola, includendo tutti e divertendosi insieme

Rosella Gennari, Alessandra Melonio, Santina Torello[1]

I bambini di oggi sono "consumatori digitali" fin dalla più tenera infanzia, da quando nelle case è presente un computer, e ancora di più da quando nelle tasche di ogni genitore, fratello maggiore o anche nonno, è presente uno smartphone o un tablet, p.es., vedasi [5]. L'interazione è assolutamente naturale, e il tablet sta assumendo la funzione di "babysitter elettronica" che fino a non molti anni or sono svolgeva la televisione. Senza voler discutere pro e contro di questo fenomeno, a noi preme sottolineare che esso rende, da una parte, estremamente familiare l'esistenza di videogiochi (nota: useremo gioco, video-gioco e gioco digitale come sinonimi), di cui ogni bambino ha esperienza diretta, anche in quelle famiglie dove si vigila attentamente sul loro uso e se ne evita accuratamente l'abuso. Dall'altra parte, la necessità di alfabetizzare bambini della scuola all'uso di strumenti informatici sembra in via di sparizione, forse prima ancora che alcuni insegnanti abbiano pienamente acquisito l'uso della stessa strumentazione a fini didattici?

Se ci limitassimo a considerare l'informatica come puro "consumo della tecnologia", il ruolo dell'informatica nella scuola potrebbe quindi dirsi concluso. E si potrebbe poi discutere se e quanto ha senso avviare alla programmazione le classi quarte o quinte primarie, usando ambienti di sviluppo quali Scratch [16]. Nel contempo, però sono emerse anche linee di pensiero diverse, che tendono a rivalutare l'informatica come "forma mentis" [2]. Per citarne una, consideriamo il "pensiero computazionale" [17]: non richiede l'uso o la produzione di artefatti

[1] Libera Università di Bolzano.

tecnologici; mira ad insegnare concetti e metodi legati a decomposizione, ad astrazione, a divide-et-impera, tipici dell'informatica, utilizzabili in quasi ogni contesto.

Per quanto riguarda il mondo della scuola, in particolare quella primaria, i fenomeni di globalizzazione e al contempo la mutata demografia della società italiana rendono ogni classe un caleidoscopio di culture, lingue e colori della pelle. La classe omogenea, fatta di bambini tutti di analoga estrazione sociale o culturale, non esiste quasi più. L'eterogeneità non deve diventare uno svantaggio, bensì un'opportunità di crescita a vari livelli. In particolare, l'eterogeneità può essere funzionale a suscitare la creatività di gruppo, dove le idee degli studenti vengono arricchite da prospettive differenti [11], nonché a promuovere le tanto citate competenze trasversali.

Per gli insegnanti è però una sfida coinvolgere tutta la classe, senza annoiare né frustrare, al fine di stimolare le potenzialità di ogni bambino, perché cresca pronto per il mondo di oggi, e possa diventare un cittadino di domani. L'importanza del coinvolgimento emotivo attivo e positivo del bambino non va mai sottovalutata, fosse solo per il legame che sussiste fra emozioni e rendimento del bambino, p.es., si veda [9]. Nello specifico, il modello di Pekrun [12] distingue le emozioni in attivanti/deattivanti e positive/negative; esso postula che le emozioni sono strettamente interconnesse con le prestazioni scolastiche, anche tramite meccanismi funzionali, p.es., la motivazione. L'evidenza empirica basata su questo modello ha dimostrato che le prestazioni degli studenti sono positivamente legate alle emozioni positive attivanti, come il godimento, e negativamente per le emozioni negative disattivanti, come la noia, mentre sono spesso variabili per le emozioni positive attivanti, come il rilassamento, e le emozioni negative attivanti, come l'ansia, p.es., si veda [13].

Considerando le linee di ricerca odierne nell'ambito della didattica dell'informatica e calandole nella realtà scolastica di oggi, tratteggiata brevemente sopra, un'attività di informatica alle primarie dovrebbe soddisfare i seguenti requisiti:

- esercitare aspetti di "pensiero computazionale" e competenze trasversali;
- essere inclusiva, considerando gruppi eterogenei, che richiedano abilità diverse, in modo che ogni bambino possa trovare il proprio modo di esprimersi e contribuire al lavoro comune;

- essere emotivamente coinvolgente, con un fine chiaro, concreto, ed un grado di competizione che non sfoci in aggressività e frustrazione "negativa", con ricompense e valutazioni condivise nonché funzionali a migliorare il lavoro di gruppo.

Progettare un gioco digitale è un'attività di informatica che può essere portata a scuola ed essere strutturata in modo tale da soddisfare tutti i requisiti elencati [1]. Innanzitutto, ha un chiaro inizio ed una altrettanto chiara conclusione, prevede modalità di successo e di fallimento, ha regole precise e richiede concentrazione. Le relazioni temporali di sequenzialità, con "prima" o "dopo", e quelle di causa-effetto, con "se allora" e "altrimenti", nonché l'iterazione, per esempio "ripeti finché", emergono quando si stabiliscono le regole del gioco e, più in generale, la meccanica del gioco (se non raccogli la spada, non potrai sconfiggere il drago, ecc.) e può essere legata ad una qualsiasi attività scolastica tradizionale. Per esempio, la lettura di una storia può essere collegata alla creazione di un gioco, laddove la storia funge da "storyline" del gioco.

Partendo dall'approccio alla co-progettazione per i bambini [10; 14], si può progettare un gioco digitale, che si presenta dunque come un banco di prova ideale per sperimentare diverse modalità di sollecitazione di abilità diversificate e complesse. Se la co-progettazione è poi fatta in maniera ludica (*gamified*), promette di essere coinvolgente come giocare con il gioco stesso [15]. Se fatto in maniera cooperativa, considerando l'eterogeneità come un vantaggio ed un'opportunità di crescita, allora aiuta anche a sviluppare le cosiddette competenze trasversali per lavorare in gruppo [4]. Possiamo così progettare insieme, in maniera inclusiva e ludica, un gioco digitale, come suggerito in [7].

Immaginiamo quindi una scuola primaria dove c'è una maestra che ha un'infarinatura di progettazione di giochi, o che conosce un progettista di giochi disponibile ad aiutarla in classe. Cosa serve per progettare un gioco elettronico? Dipende da dove ci si vuole fermare nella lunga strada tra progetto e realizzazione. Con i materiali presenti a scuola, carta, forbici e matite colorate, si può già realizzare un prototipo cartaceo, con una scena, i personaggi, gli oggetti, ed "animarlo" simulando come si svolge l'azione. Per quest'ultima fase di animazione, i bambini non hanno problemi a raccontare o mimare le azioni dei giocatori. A questo punto non abbiamo un gioco completo, ma un prototipo di gioco, a bassa fedeltà, e in una scuola elementare ci potremmo fermare qui.

Se nello stesso edificio, come spesso capita, ci fosse anche una scuola media, potremmo andare ancora avanti, coinvolgendo un insegnante di "tecnica" ed una classe della scuola media, per realizzare un prototipo del gioco di media fedeltà. Ci sono software semplici da usare, con cui i ragazzi della scuola media possono realizzare brevi animazioni, partendo da prototipi a bassa fedeltà e dalla loro presentazione, fatta dagli scolari delle elementari.

E infine, nello stesso complesso scolastico, non è impensabile che ci sia anche una scuola secondaria superiore, in cui lavora un insegnante di informatica, che ha insegnato ad esempio JavaScript, Java o Python ad una classe terza o quarta, e che potrebbe proporre agli studenti di realizzare un prototipo ad alta fedeltà dei giochi stessi, a partire dalle animazioni realizzate nella scuola media.

Un simile progetto "verticale" oltre a stimolare sinergie tra allievi di diversi ordini di scuola, tra i rispettivi insegnanti e dirigenti scolastici, aiuterebbe a rendere i giovani non soltanto consumatori ma progettisti di tecnologie. Non è fantascienza, noi lo abbiamo fatto più volte ed in modi diversi [8]. Per esempio, abbiamo attuato una collaborazione fra scuola primaria ed università che andiamo a descrivere nel seguito. In fondo a questo lavoro, presenteremo alcune considerazioni sulla prosecuzione di questa esperienza, e sulla sua riproducibilità.

10.1. La co-progettazione ludica e cooperativa va a scuola

Nel 2013, abbiamo iniziato a sperimentare la metodologia di co-progettazione ludica e cooperativa denominata GaCoCo [7] in quattro classi della scuola primaria. Prima della sperimentazione, gli insegnanti erano stati formati e informati sui fini del lavoro, nonché al protocollo di lavoro; successivamente le classi con i rispettivi insegnanti avevano letto e commentato una storia, scelta come base per la progettazione, e parte dell'attività scolastica regolare. Quindi progettisti della Facoltà di Scienze e Tecnologie Informatiche della Libera Università di Bolzano sono "andati a scuola". Le classi, con l'aiuto dei progettisti, hanno realizzato prototipi a bassa fedeltà, su carta, di giochi che analizzavano la storia letta. Nel seguito faremo riferimento soprattutto all'esperienza del 2013, precisando che nel 2014 e nel 2015 sono state apportate alcune variazioni di dettaglio alla sperimentazione, che è tuttora in corso mentre scriviamo.

10.2. Partecipanti e ruoli

Le quattro classi coinvolte erano: due terze elementari (una della scuola M.L.King di Bolzano, l'altra della scuola G.Galilei di Brunico) e due quinte elementari (una per ciascuna delle due scuole sopra citate). In tutto sono stati coinvolti 56 allievi, 4 insegnanti e 4 progettisti.

Gli *allievi* erano i veri e propri ideatori dei giochi, lavorando in gruppi di 3-5 bambini, secondo strategie dell'apprendimento cooperativo. Tutti dovevano rispettare regole di base per l'apprendimento cooperativo, quali il rispetto del turno di parola. Per ogni gruppo di allievi, c'era un esperto del prodotto da progettare, ovvero un gioco digitale. Ognuno di questi esperti illustrava al proprio gruppo l'organizzazione del lavoro, seguiva e supportava lo sviluppo del progetto nel gruppo (ad esempio rispondeva ai dubbi sollevati, ed assisteva il progetto al fine di evitare il rischio di fallimento), osservava le attività e, al momento della valutazione dei prototipi, assegnava un voto alla cooperazione ed al coinvolgimento. Gli *insegnanti* avevano la responsabilità di formare i gruppi, eterogenei in termini di abilità sociali e di profitto scolastico, usando una scheda ad-hoc fornita dai progettisti, di moderare il comportamento della classe, e di stimolare le discussioni in classe quando previsto dal protocollo di co-progettazione.

10.2.1. Protocollo di co-progettazione

Lo sviluppo del prototipo ha richiesto quattro sessioni, una per classe, sempre alla presenza dell'insegnante di classe e dei quattro progettisti. In dettaglio ecco quanto realizzato.

Missioni. Ogni sessione è stata spezzata in tre missioni ludiche, ciascuna da svolgersi in tempi prefissati. Questa tempistica è stata sostanzialmente dettata dall'organizzazione dell'orario scolastico. Nella prima missione, sotto la guida dell'insegnante, veniva letta e commentata in classe una storia. Nella seconda e nella terza missione si lavorava a gruppi: nella seconda, gli stessi gruppi co-progettavano il prototipo di gioco; nella terza missione, i prototipi venivano discussi in classe, visitando una "galleria", come in Figura 1.1. La seconda e terza missione hanno richiesto dalle due alle tre ore per classe. In Figura 1.2 si vedono due prototipi, sviluppati rispettivamente da una classe terza ed una quinta.

Fig. 10.1. La galleria dei prototipi.

Fig. 10.2. Prototipi di giochi divisi in due livelli: di una terza (sopra) e di una quinta (sotto).

Progressione e competenze. Nella prima missione l'interazione avveniva tra l'insegnante e la classe; nella seconda missione, l'interazione avveniva all'interno di ciascun gruppo, e nella terza missione, tutta la classe interagiva a turno con ciascun gruppo. Ogni missione aveva proprie regole e sfide di difficoltà crescente, sollecitando abilità diverse. Ogni sfida si basava su quanto prodotto dalla sfida precedente, in una progressione lineare, che richiedeva di completare ogni sfida prima di passare alla successiva. Vediamo, come esempio, lo svolgersi

della seconda missione, dedicata al lavoro di gruppo. La prima sfida richiedeva di discutere ed accordarsi sulla cosiddetta "idea di gioco" [1], sulla base dell'obiettivo assegnato; quindi, si esercitavano soprattutto le abilità verbali. Nelle altre sfide, realizzando lo scenario del gioco su carta, si attivavano prevalentemente abilità visuali e motorie.

Cooperazione, competizione e premi. Alla fine della prima missione, prima di suddividere la classe in gruppi per la seconda missione, gli insegnanti chiarivano l'obiettivo di competizione tra gruppi, ovvero, ogni gruppo avrebbe dovuto cercare di costruire il prototipo di gioco migliore. Quale fosse il gioco migliore sarebbe stato scelto sulla base di quanto efficace era la cooperazione di gruppo (secondo il giudizio di progettisti ed insegnanti), della valutazione di esperti dei giochi, e di una votazione da parte di tutti i bambini di altre scuole. Il premio previsto per questo gioco vincente sarebbe stato la realizzazione del gioco su tablet, messo a disposizione di tutti i bambini della classe "per giocare davvero". Per tutti i gruppi era comunque previsto un premio: la presentazione del loro gioco, registrata durante la terza missione, sarebbe stata resa disponibile via internet, con accesso protetto.

10.3. Il ciclo di vita dei prodotti

Seguendo un modello di ciclo di vita "snello" (lean), i prodotti sono stati valutati da gruppi misti. Tutti i prototipi a bassa fedeltà di giochi creati dai bambini sono stati dapprima valutati dai 4 progettisti e dai 4 insegnanti, come segue.

Innanzitutto insegnanti e progettisti valutavano la coerenza del prototipo rispetto all'idea di gioco iniziale. Poi, i progettisti valutavano ogni gioco secondo euristiche di "giocabilità'" [6], e secondo il livello di cooperazione all'interno del gruppo. Partendo dai risultati di questa valutazione, i progettisti hanno selezionato 4 prototipi, poi trasformati in prototipi a media fedeltà, ovvero dei video, atti a simulare l'interazione del giocatore con il prototipo.

I prototipi a media fedeltà sono poi stati valutati da altre due scuole primarie, stavolta nel Centro Italia: una terza ed una quarta della scuola Oriente di Pescina, ed altre due classi, sempre una terza ed una quarta, ma nella scuola Cerchio di Cerchio. Questa fase di valutazione ha quindi coinvolto altri 64 bambini, 2 insegnanti e 2 progettisti: ogni

classe, assistita da un progettista, ha valutato ciascuno dei quattro prototipi di media fedeltà, ovvero i video. Infine, una studentessa di Scienze e Tecnologie Informatiche della Libera Università di Bolzano ha sviluppato i prototipi ad alta fedeltà sotto forma di app per tablet Android, tenendo conto dei suggerimenti emersi dalla valutazione. In questo sviluppo, la studentessa di Informatica è stata seguita da due docenti dell'università, esperti di progettazione di giochi digitali. In tutte queste fasi, le idee e i prodotti dei bambini sono stati conservati quanto più possibile, come si vede in Figura 1.3.

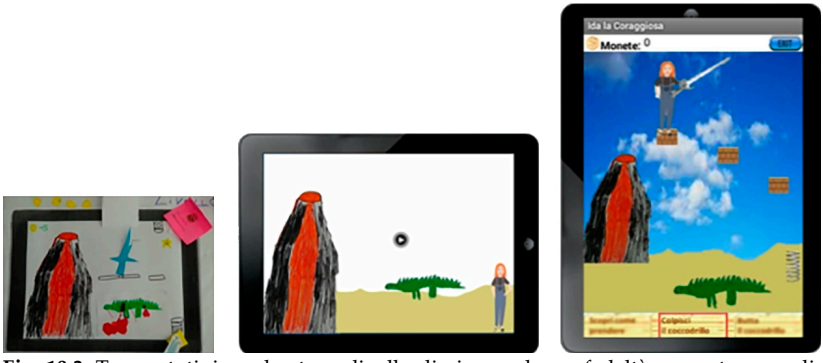


Fig. 10.3. Tre prototipi per lo stesso livello di gioco: a bassa fedeltà, su carta; a media fedeltà, un video; ad alta fedeltà, una app per tablet Android.

10.4. Conclusioni

Il percorso di progettazione ludica, con apprendimento cooperativo, non ha ancora raggiunto il suo "game over". È continuato e maturato nel corso degli anni, ottenendo risultati positivi anche per inclusione e coinvolgimento emotivo [3]: gli elementi di ludicizzazione e di cooperazione sono aumentati, introducendo componenti elettroniche quali Arduino per migliorare l'interattività; il percorso ha coinvolto altre classi, venendo maggiormente strutturato e snodandosi in una progettazione più lunga e variegata, ed ora sta valicando l'Arco Alpino, coinvolgendo—speriamo—anche bambini di altri paesi.

Bibliografia

[1] ADAMS, E., *Fundamentals of Game Design,* 3rd Ed., Pearson 2013.

[2] BOARD OF INFORMATICS EUROPE & ACM EUROPE, *Informatics education: Europe cannot afford to miss the boat,* Online (2013), http://europe.acm.org/iereport/ACMandIEreport.pdf.

[3] BRONDINO, M., DODERO, G., GENNARI, R., PASINI, M., RAC-CANELLO, D., TORELLO, S., *Emotions and Inclusion in Co-Design at School: Let's Measure Them!,* in Proc. Methods and Intelligent Sys-tems for Technology Enhanced Learning, in print, 2015.

[4] COHEN, E., *Making Cooperative Learning Equitable,* Educational Leadership 1998.

[5] COMMON SENSE MEDIA RESEARCH STUDY, *Zero to Eight: Children's Media Use in America 2013,* Tech. Rep., 2014.

[6] DESURVIRE, H., WIBERG, C., *Game Usability Heuristics (PLAY) for Evaluating and Designing Better Games; the Next Iteration,* in Proc. 3rd Interntional Conference OCSC, HCI International, 2009.

[7] DODERO, G., GENNARI, R., MELONIO, A., TORELLO, S., *Gamified Co-design With Cooperative Learning,* in CHI '14 Extended Abstracts on Human Factors in Computing Systems, New York, (2014), 707-718.

[8] DODERO, G., GENNARI, R., MELONIO, A., TORELLO, S., *Towards Tangible Gamified Co-design at School: Two Studies in Primary Schools,* in SIGCHI annual symposium on Computer-human interaction in play (CHI PLAY '14), New York, (2014), 77-86.

[9] FREDERICKS, J., BLUMENFELD, P., PARIS, A., *School Engagement: Potential of the Concept, State of the Evidence*, Review of Educational Research, 2004.

[10] MAZZONE, E., *Designing with Children: Reflections on Effective Involvement of Children in the Interaction Design Process,* Phd thesis, University of Central Lancashire, 2012.

[11] PAULUS, P.B., NIJSTAD, B.A., (EDS.), *Group Creativity: Innovation through Collaboration,* Oxford University Press, 2003.

[12] PEKRUN, R., *The Control–value Theory of Achievement Emotions: Assumptions, Corollaries, and Implications for Educational Research and Practice performances,* Educational Psychology Review, 18 (2006), 315–341.

[13] PEKRUN, R., ELLIOT, A., MAYER, M., *Achievement Goals and Achievement Emotions: Testing a Model of Their Joint Relations With Academic Performance,* Journal of Educational Psychology 101 (2009), 115–135.
[14] SANDERS, E.B., STAPPERS, P.J., *Co-creation and the New Landscapes of Design, CoDesign: International Journal of CoCreation,* in Design and the Arts 4, (2008), 5–18.
[15] SEABORN, K., FELS, D., *Gamification in Theory and Action: A Survey,* International Journal of Human-Computer Studies 74 (2015), 14–31.
[16] SCRATCH, Online (2015) https://scratch.mit.edu/.
[17] WING, J.M., *Computational Thinking,* CACM viewpoint, 49 (2006), 33-35.

Ringraziamenti

Gli Autori ringraziano: bambini, genitori, dirigenti di tutte le scuole coinvolte nella sperimentazione, a Bolzano, Brunico, Merano, Cerchio e Pescina e gli insegnanti "pionieri", ovvero, Carla Campanella, Birgit Daniel, Daniela Marina, Stefania Marsich, Stefano Rento, Anna Santon, Paolo Soldani, Ambra Tarter, diventati "giocatori" attivi ed attivanti in questa ricerca; la Soprintendenza Scolastica Italiana di Bolzano, nonché i colleghi Gabriella Dodero, Vincenzo Del Fatto (Libera Università di Bolzano) e Tania Di Mascio (Università dell'Aquila) per aver svolto il ruolo di progettisti in alcune scuole; il progetto di ricerca DARE (http://www.inf.unibz.it/dare/schools.html), finanziato della Provincia di Bozen-Bolzano.

Ringraziamenti

Ringrazio i colleghi dei diversi atenei italiani che hanno contribuito a questo volume dopo aver partecipato ai diversi incontri del Gruppo di Lavoro "Informatica e Scuola" del GRIN specie in occasione dei workshop, cui hanno preso parte attiva anche abilitandi e tutor provenienti dalla scuola.

L'architetto Marco Setti ha interpretato graficamente l'idea portante del volume tramite il disegno di copertina.

Anna Labella

Collana Manuali

1. Il Tempo e la Verità
   Ernesto Capanna
2. Aerodinamica
   Giorgio Graziani
3. Medicina Nucleare in Oncologia
   Francesco Scopinaro e Maria Gemma Parisella
4. Impianti Nucleari
   Maurizio Cumo
5. Accounting for Equity and Other Comprehensive Income
   Francesco Bellandi
6. Gasdinamica
   Filippo Sabetta
7. Segnali - Processi Aleatori - Stima - Vol. I
   Gaetano Scarano
8. Segnali - Processi Aleatori - Stima - Vol. II
   Gaetano Scarano
9. Esercizi di analisi matematica
   Paola Loreti e Daniela Sforza
10. Meccanica dei fluidi sperimentale
    Antonio Cenedese e Monica Moroni
11. Impianti Nucleari - 2ª edizione
    Maurizio Cumo
12. Lettere in classe
    Percorsi didattici del TFA di area letteraria della Sapienza
    a cura di Paola Cantoni e Silvia Tatti
13. Gastrostomia Endoscopica Percutanea (PEG)
    Massimo Chiaretti, Alessandra Santiloni, Giovanna Angela Carru Annalisa Italia Chiaretti
14. E questo tutti chiamano Informatica
    L'esperienza dei TFA nelle discipline informatiche
    a cura di Anna Labella

Questo volume è il punto di arrivo di una serie di incontri del Gruppo di Lavoro "Informatica e Scuola" del GRIN presso diverse università italiane, riguardanti i TFA di tipo informatico (classe A042 e A033). L'ultimo di questi incontri si è tenuto il 21-22 febbraio 2014 presso il dipartimento di Informatica della Sapienza, ma da allora tale esperienza si è ulteriormente arricchita anche attraverso i relativi PAS. Esso contiene riflessioni generali sul ruolo che potrebbe svolge-re l'informatica nella società di oggi e nella preparazione dei giovani per la società di domani, riferendo l'esperienza della preparazione degli insegnanti nelle diverse sedi italiane alla luce delle normative vigenti sia peri TFA che per il PAS, anche con riferimenti a quanto si fa all'estero. Si approfondiscono poi alcuni temi specifici della didattica dell'informatica con le loro possibilità e difficoltà.

Anna Labella, ordinario di Informatica presso Sapienza università di Roma, ha cominciato la sua professione come algebrista e docente di Matematiche complementari. Avendo ereditato dal suo maestro, Lucio Lombardo Radice, la passione per la didattica, ha sempre coltivato questo campo a fianco della ricerca più propriamente scientifica. L'istituzione dei TFA ha costituito l'occasione per conciliare queste due attività e per mettere a frutto le competenze filosofiche acquisite negli anni.

ISBN 978-88-98533-63-3

www.editricesapienza.it