Il latino del futuro

L’insegnamento dell’informatica nel Liceo delle Scienze Applicate

Relazione finale

di Enrico Terrone

Corso neoassunti, Anno scolastico 2011/2012

Liceo Scientifico “Avogadro”, Vercelli

Introduzione

Nei quadri orari del liceo delle scienze applicate c'è un'importante differenza rispetto al caso del precedente liceo tecnologico. Alle tre ore settimanali di informatica nel triennio si sono sostituite due ore settimanali articolate lungo l'intero quinquennio. L'informatica diventa così una materia caratterizzante che accompagna lo studente lungo l'intero percorso liceale, occupando una posizione paragonabile a quella che nel liceo scientifico tradizionale viene attribuita al latino.

Le analogie fra le due materie – informatica e latino – sono assai cospicue. Si tratta di due materie che vertono attorno a dei linguaggi "speciali" rispetto alle lingue correnti: una lingua "morta" caratterizzata da declinazioni e paradigmi da una parte, dei linguaggi di programmazione basati su strutture di sequenza, selezione, iterazione e ricorsione dall'altra. Si tratta in entrambi i casi di linguaggi con una forte struttura logica che non si apprendono per essere poi parlati quanto piuttosto, come si diceva una volta, "per imparare a ragionare". Infine, tanto il latino quanto l'informatica costituiscono forme linguistiche che hanno svolto (nel caso del latino) e svolgono o svolgeranno (nel caso dell'informatica) la funzione di lingua franca, di luogo di convergenza per una varietà di attività a forte valenza economica, sociale e culturale.

Le suggestive analogie fra informatica e latino non devono però fare passare in secondo piano le differenze fra i due ambiti disciplinari. Nel caso del latino abbiamo infatti a che fare con una vera e propria lingua storica, mentre nel caso dell'informatica abbiamo a che fare piuttosto con una categoria tecnologica che contempla al suo interno forme linguistiche. Tuttavia la funzione didattica che l'informatica è chiamata a svolgere all'interno del liceo delle scienze applicate ha molte affinità con quella svolta dal latino nel liceo scientifico tradizionale, e in tal senso ritengo che l'analogia fra le due materie meriti di essere approfondita.

Il documento ministeriale

Nelle “Indicazioni nazionali” fornite dal Miur relativamente al nuovo liceo delle scienze applicate, la sezione dedicato all’informatica si articola in due parti: la prima dedicata a “Linee generali e competenze”, la seconda riguardante gli “Obiettivi specifici di apprendimento”.

La prima parte insiste soprattutto sulla capacità dell’informatica di coltivare abilità di modellizzazione e *problem solving*, sulle potenzialità di interconnessione con le altre discipline e sulla rilevanza sociale e culturale dell’informatica nell’epoca contemporanea.

Si individuano come punti chiave del percorso di apprendimento la comprensione della struttura fisica e logico-funzionale del computer e delle reti, l’acquisizione di linguaggi di programmazione e l’uso dei più comuni strumenti software. Si insiste vigorosamente sul raccordo con gli altri insegnamenti, “con le discipline scientifiche, ma anche con la filosofia e l'italiano”, e si evidenzia l’importanza di un legame con le realtà attive sul territorio: “università, enti di ricerca, musei della scienza e mondo del lavoro”.

Resta il problema, che il documento ministeriale non affronta, di capire in che modo sia possibile realizzare questi auspici: come far acquisire agli studenti le abilità di modellizzazione e problem solving? Come interagire proficuamente con gli altri insegnamenti? Come instaurare relazioni reciprocamente gratificanti con le realtà attive sul territorio?

Di questo cercherò di occuparmi nel corso della mia relazione, ma per prima cosa, invertendo l’ordine di esposizione adottato dall’anonimo redattore ministeriale, ritengo che occorra considerare gli obiettivi di apprendimento e soprattutto la loro articolazione dei cinque anni del percorso scolastico.

Gli obiettivi principali del primo biennio individuati dal testo ministeriale si articolano nei seguenti punti: caratteristiche strutturali e architetturali di computer e reti; codifica binaria; nozione di sistema operativo; padronanza d’uso di documenti elettronici e servizi di rete; introduzione al concetto di algoritmo e alla pratica della programmazione.

Gli obiettivi del secondo biennio ricadono invece essenzialmente in tre aree: programmazione avanzata; database; progettazione per il web. In realtà all’insegnante viene lasciata ampia facoltà di impostare liberamente il programma, in base al principio didattico per cui: “la scelta dei temi dipende dal contesto e dai rapporti che si stabiliscono fra l’informatica e le altre discipline”. Questa impostazione flessibile e interdisciplinare si fa ancora più marcata quanto alle indicazioni per il quinto anno che si riduce a una generica esortazione a realizzare “percorsi di approfondimento, auspicabilmente in raccordo con le altre discipline”, con particolare riguardo agli algoritmi di calcolo numerico e ai software di simulazione.

Nella mia relazione, mi propongo di discutere e sostanziare – per ogni anno scolastico – questi obiettivi fissati dal testo ministeriale e di dettagliare alcune strategie per adeguate al loro raggiungimento. Sarà proprio riflettendo su queste strategie che emergeranno mano a mano le linee generali dalle quali il testo ministeriale prende avvio.

Il primo anno

Obiettivo principale del primo anno è la comprensione del significato dell'informatica come disciplina scientifica. Questo è a mio avviso possibile se si pone fin da subito con chiarezza l'oggetto di questa disciplina, che è *la distinzione fra hardware e software*, di cui il computer è soltanto l'occorrenza privilegiata, ma non l'unica (a maggior ragione oggi, con telefonini, tablet e reader che si rivelano ogni giorno che passa oggetti informatici a pieno titolo). Ritengo che l'intera didattica di informatica del primo anno debba essere orientata alla piana comprensione di questa fondamentale distinzione. Essa viene introdotta fin dalla prima lezione, e accompagna l'intero svolgimento dell'anno scolastico, presentato e scandito come un progressivo percorso di allontanamento dall'hardware e di avvicinamento ai livelli più astratti del software. Il protagonista del primo anno di informatica è in tal senso il modello a strati del calcolatore (cfr. Tanenbaum 1999). Alla base c'è l'hardware, a cui fanno seguito: linguaggio macchina, sistema operativo, linguaggi di programmazione e software applicativi.

L'evidenza da cui si parte è quella di oggetti fisici che almeno in parte tutti gli studenti conoscono e possono vedere e toccare: la tastiera, lo schermo, e soprattutto "la scatola" dentro la quale tutti sanno trovarsi le parti fisiche più importanti del computer. Questo è l'hardware: il corpo del computer, la sua parte fisica e materiale. Poi c'è un altro aspetto dell'informatica che tutti gli studenti già conoscono: i programmi come Windows, Word, Internet Explorer, Windows Movie Player e giochi di vario tipo.

L'obiettivo principale del primo anno è di rispondere alla domanda: che cosa c'è in mezzo? Ovvero, quale sistema tecnologico fa sì che gli oggetti concreti (l'hardware) che abbiamo sotto i nostri occhi possano supportare i processi di comunicazione e interazione (il software) che impegnano le nostre menti? Rispondere a questa domanda significa intraprendere un percorso composto essenzialmente da sei tappe che di seguito mi occuperò a dettagliare. Prima però intendo ribadire l'importanza della distinzione hardware e software, di cui queste sei tappe costituiscono lo sviluppo e l'articolazione.

Ritengo che il modo migliore per introdurre agli studenti la distinzione fra hardware e software sia di insistere fin da subito sull'analogia che sussiste fra la coppia software/hardware da una parte, e la coppia mente/corpo dall'altra. Su questa analogia si fonda uno dei progetti di ricerca interdisciplinari più importanti dell'epoca contemporanea, quello delle scienze cognitive. Si tratta di un progetto che coinvolge e connette ambiti scientifici differenti quali in particolare la fisiologia, la neurologia, l'informatica, la linguistica, la logica, la psicologia e la filosofia. Le scienze cognitive sono insomma il riferimento ideale per un progetto scolastico che voglia fare dell'interdisciplinarietà il suo punto di forza. Purtroppo in ambito didattico l'importanza delle

scienze cognitive è stata colpevolmente e sciaguratamente sottovalutata. Nei programmi del vecchio liceo scientifico tecnologico si sono sporadicamente inseriti elementi di intelligenza artificiale, senza però mai riconoscere con chiarezza che è l'intelligenza artificiale può dare un contributo interdisciplinare veramente importante soltanto in virtù del suo inserimento nel quadro delle scienze cognitive. La questione delle scienze cognitive è ampia e complessa ed è sicuramente precoce come argomento del primo anno: saranno piuttosto il secondo biennio e soprattutto il quinto anno a costituire il contesto ideale per sviluppare simili discorsi. Tuttavia è importante che fin dal primo anno si chiarisca che la distinzione fra hardware e software è l'oggetto essenziale dell'informatica e che essa può essere compresa agevolmente riflettendo sull'analogia cognitiva fra la coppia software/hardware da una parte, e quella mente/corpo dall'altra. Questa riflessione deve necessariamente incorporare elementi critici: non si tratta insomma di convincere lo studente che l'essere umano e il computer sono fatti alla stessa maniera; si tratta piuttosto di invogliarlo a riflettere su alcuni importanti tratti comuni che li caratterizzano, ma anche su alcune differenze di cui è necessario tenere conto: ad esempio il fatto che il software può essere agevolmente duplicato o trasferito da un hardware all'altro, mentre la mente umana non può essere duplicata o trasferita da un corpo all'altro; qui si aprono prospettive fantascientifiche che di solito gli studenti accolgono con molta curiosità e interesse e che forse varrebbe la pena approfondire interdisciplinarmente con letture di racconti o visione di film di fantascienza.

L'analogia cognitiva permette anche di introdurre un'altra coppia di concetti fondamentali lo studio dell'informatica: input e output. L'idea che il computer sia un dispositivo la cui funzione principale consiste nel ricevere informazioni di input, elaborarle e produrre informazioni di output, può essere proficuamente introdotta notando che anche l'essere umano si comporta spesso e volentieri come un sistema di questo tipo, se consideriamo che i canali sensoriali sono i suoi input mentre le azioni e le parole costituiscono i suoi output.

Una volta chiarito di che cosa parla l'informatica e quali sono le coppie concettuali che la costituiscono (hardware/software, input/output), è il momento di iniziare a percorrere la strada in sei tappe che ci condurrà dall'hardware al software, cioè dalla concretezza all'astrazione, dalla materia all'intelligenza.

L'hardware

La prima tappa riguarda l'hardware medesimo: le parti fisiche che opportunamente interconnesse costituiscono il computer come macchina. Si tratta insomma di introdurre le nozioni di CPU, memoria centrale e periferiche, la cui rappresentazione in forma di schema a blocchi costituisce il celebre "modello di Von Neumann". Qui è importante iniziare a introdurre un altro aspetto

fondamentale dell'informatica: la sua dimensione storica. Gli studenti appartengono ormai alla generazione dei cosiddetti nativi digitali, ai quali il computer di sa come un oggetto che sembra esistere da sempre. La presentazione del modello di Von Neumann è il momento giusto per ricordare sul fatto che il computer è un'invenzione recente, che data grossomodo agli anni Quaranta del Novecento (ha insomma circa l'età dei nonni dei nostri attuali studenti), e che proprio questo signor Von Neumann che stiamo menzionando è stato uno dei grandi protagonisti di questa invenzione. Al tal proposito, è importante anche far notare – possibilmente con l'ausilio di materiali multimediali – che i computer all'inizio erano molto diversi da quelli che vediamo oggi, ma queste differenze in realtà sono meno profonde di quanto all'apparenza si possa credere. C'è infatti una struttura progettuale comune ai primi pachidermici sistemi di elaborazione dati e agli attuali agilissimi dispositivi portatili: questa struttura comune è data proprio dal modello di Von Neumann. Se invece si presta attenzione alle differenze fra i primi computer e i computer attuali, allora si finisce per parlare di usabilità e prestazioni, il che fornisce l'occasione per distinguere ciò che è fondamentale e ciò che è accessorio in un computer, ma soprattutto per introdurre le unità di misura e gli ordini di grandezza tipici dell'informatica (Bit, Byte, Hertz, Kilo, Mega, Giga, Tera). In questa fase ritengo che sia doverosa un'attività di laboratorio che porti – in collaborazione con l'ufficio tecnico – a un'apertura "chirurgica" del corpo del computer e a un'analisi dettagliata delle varie schede e dei dispositivi hardware che costituiscono la parte più importante dell'hardware ma che non sono normalmente visibili all'utente. Un'altra attività che potrebbe essere interessante, anche nell'ottica di una collaborazione con le realtà del territorio, consiste in un intervento esterno, a lezione, di qualche professionista che lavori nell'ambito dell'assemblaggio dei componenti alla manutenzione o alla riparazione di personal computer.

Infine, in chiusura di questo modulo introduttivo, si introducono le proprietà funzionali dell'hardware, che corrispondono a quelle di un automa a stati: un modello ideale di esecutore che trova nella Macchina di Turing la propria formulazione più compiuta. Questo nesso strutturale fra l'esecutore reale e l'esecutore ideale permette agli studenti di iniziare a intuire il legame fondamentale che sussiste fra la struttura dell'hardware, della quale ci siamo fin qui occupati, e le prestazioni intelligenti del software, che saranno il tema dei moduli successivi.

Il linguaggio macchina

La seconda tappa parte dalla considerazione che l'hardware parla una propria lingua che non è la nostra lingua. La "lingua" parlata dall'hardware si chiama "linguaggio macchina" e soltanto imparando a nostra volta questa lingua potremmo riuscire a utilizzare proficuamente questo hardware. Qui l'obiezione degli studenti è prevedibile: io riesco a usare benissimo il computer

senza sapere il linguaggio macchina. Ma la contro-obiezione è altrettanto immediata: questo può avvenire soltanto perché altre persone conoscono il linguaggio macchina e hanno sfruttato questa conoscenza per far sì che il computer potesse interagire anche con persone che non lo conoscono. Ecco una prima plateale manifestazione della differenza fra un utilizzatore di computer e un informatico: il primo può usare il computer nonostante la propria ignoranza, facendo leva proprio sulle conoscenze del secondo. Quel che si richiede agli studenti a questo punto non è certo di imparare a programmare in linguaggio macchina, ma di prendere consapevolezza e dimestichezza con la sua caratteristica fondamentale: la codifica binaria di dati e istruzioni. Dunque affrontare il linguaggio macchina a questo livello significa considerare le varie categorie di informazione (insiemi numerici, testi, istruzioni, immagini, suoni, video) e per ciascuna categoria capire come rappresentarla facendo ricorso alle sole cifre zero e uno.

Il sistema operativo

La terza tappa ha per tema il sistema operativo, considerato come il programma più importante presente sul computer: quello che si fa carico della mediazione fra il linguaggio macchina e l'hardware da una parte, e il software di alto livello dall'altra. Il sistema operativo viene inizialmente presentato genericamente come quel programma che fa sì che anche le persone che non conoscono la struttura dell'hardware e le regole del linguaggio macchina possano comunque usare il computer. Come caso esemplare di sistema operativo viene naturalmente proposto Windows, ma si insiste fin da subito sul fatto che Windows è soltanto un sistema operativo fra i tanti che si sono storicamente succeduti e che attualmente si contendono il mercato.

Questa presentazione di massima viene poi dettagliata considerando i vari livelli in cui il sistema operativo si articola al suo interno: gestore del processore, gestore della memoria, gestore delle periferiche, gestore delle informazioni, interfaccia utente. L'idea principale di questa introduzione ai sistemi operativi è di risalire da quella che è l'esperienza già nota agli studenti, cioè l'uso dell'interfaccia utente, alla pluralità di strati software sottostanti che rendono possibili le funzionalità offerte dall'interfaccia all'utente.

I linguaggi di programmazione

La quarta tappa del nostro percorso dall'hardware al software riguarda i linguaggi di programmazione che, sfruttando le funzionalità offerte dal sistema operativo e da altri software di mediazione (in particolare, compilatori e interpreti), permettono di utilizzare le potenzialità di calcolo dell'hardware senza dover fare ricorso al linguaggio macchina, al posto del quale si possono usare linguaggi artificiali molto più vicini al modo di ragionare degli esseri umani e quindi molto

più adatti a sfruttare il computer per risolvere efficacemente problemi e perseguire scopi specifici. L'introduzione alla programmazione che avviene in questa fase è di estrema importanza perché pone le basi per quella che sarà l'attività principale dello studente nel corso dei cinque anni. La scrittura di un programma sarà l'esercizio che accompagnerà lo studente in tutto il suo percorso scolastico nel liceo delle scienze applicate così come l'attività di traduzione dal latino all'italiano accompagna lo studente del liceo tradizionale. Per prima cosa si tratta di chiarire che la programmazione si basa su due nozioni fondamentali: quella di *problem solving* e quella di *modellizzazione*. Programmare significa essenzialmente *far risolvere una famiglia di problemi al computer* (questo il problem solving), e per ottenere questo scopo occorre che i dati e i risultati del problema siano adeguatamente individuati e rappresentati (questa la modellizzazione). L'individuazione del problema e la modellizzazione delle informazioni costituisce la prima fase del lavoro di programmazione: la cosiddetta fase di *analisi*.

All'analisi fa seguito una seconda fase imperniata sulla fondamentale nozione di algoritmo. L'algoritmo è la sequenza di passaggi che permette di passare dai dati ai risultati, cioè dal problema alla soluzione. L'algoritmo è un concetto generale che non vale solo in ambito informatico ma nei settori più disparati, e che spesso ha ricadute importanti anche nella vita di tutti i giorni (ad esempio quando usiamo la ricetta di una torta oppure diamo indicazioni a un passante su come raggiungere la stazione). Proprio per la sua generalità, l'algoritmo non è vincolato a un particolare sistema di rappresentazione ma può essere espresso sia usando il linguaggio ordinario sia adottando un particolare linguaggio grafico, quello dei flow-chart (diagrammi di flusso). La mia idea a questo proposito è che ogni studente debba scegliere il sistema di espressione dell'algoritmo che gli è più congeniale, alla sola condizione che esso risulti comprensibile a chi legge. Nel corso del tempo ho notato che ci sono studenti ai quali i flow-chart risultano molto utili ai fini della costruzione di algoritmi e altri per i quali essi rappresentano soltanto un fastidioso ingombro. Ritengo quindi che i flow-chart vadano presentati come un utile strumento di supporto al problem solving ma non debbano essere oggetto di "accanimento didattico".

La terza fase del processo di programmazione, dopo l'analisi e la scrittura dell'algoritmo, è quella dell'implementazione. Qui si impone al docente una scelta delicata, quella del primo linguaggio di programmazione da presentare agli studenti. Sicuramente il discorso va introdotto insistendo sulla grande varietà dei linguaggi di programmazione, sia in prospettiva storica, sia nel contesto dell'informatica attuale. Ugualmente importante è distinguere fra linguaggi di programmazione di basso livello (i cosiddetti *assembly*, che ricalcano il linguaggio macchina depurandolo del codice binario) e di alto livello, e fra le differenti tipologie di linguaggio che si possono riscontrare fra questi ultimi. Tuttavia, dopo questa fase introduttiva, una decisione va presa e un primo linguaggio

di programmazione va presentato agli studenti. A questo proposito ci sono essenzialmente due scuole di pensiero: la prima sostiene che sia meglio partire da linguaggi grafici progettati appositamente per l'insegnamento dell'informatica, come ad esempio Scratch e Alice (cfr. Cooper, Dann, Pausch 2006), mentre le seconda ritiene opportuno mettere fin da subito gli studenti davanti a linguaggi di programmazione di livello professionale come il C o il Java. La mia idea a questo proposito è che almeno il primo contatto con la programmazione debba comportare il contatto diretto con un linguaggio professionale, per evitare equivoci, fraintendimenti e pericolose illusioni. Se si vuole far capire innanzitutto a uno studente che cosa vuol dire programmare, è con C o con Java che occorre cimentarsi, non con i cartoni animati di Scratch e Alice. Questo non vuol dire che questi software non possano avere un ruolo utile nell'insegnamento dell'informatica, ma a mio avviso solo in seconda battuta, dopo che almeno un primo contatto con la dura realtà della programmazione è già avvenuto. Questo "primo contatto", nel corso del primo anno di liceo, può tranquillamente limitarsi alle strutture elementari delle programmazione: istruzioni di input e output, creazione di variabili numeriche e testuali, assegnazione, sequenza, selezione e iterazione. Insomma, non si tratta tanto di imparare già a programmare appropriatamente, questo piuttosto di iniziare a capire che cosa vuol dire programmare.

Le applicazioni

La quinta tappa, con cui si conclude il percorso didattico del primo anno, riguarda i software applicativi. Qui per il momento non occorre dilungarsi molto perché si tratta di applicazioni con cui lo studente ha già una buona famigliarità che gli deriva dei programmi di tecnica delle scuole medie o da esperienze personali di uso del computer. Si tratta soltanto di introdurre le nozioni generali di applicazione come software con finalità specifiche, e di considerare all'interno del pacchetto Office i casi emblematici di World, Powerpoint ed Excel, consolidandone l'uso da parte degli studenti attraverso qualche esercitazione mirata.

Il secondo anno

La didattica del secondo anno verte su quattro temi principali che si possono considerare come approfondimenti all'interno del quadro generale impostato nell'anno iniziale. I temi in questione sono i seguenti: logica, programmazione, teoria delle reti e norme tecniche di scrittura.

Logica

Dato il ruolo cruciale che la logica matematica svolge sia storicamente sia strutturalmente nella programmazione informatica, ritengo necessario dedicare una sezione apposita del programma a un'introduzione alla logica proposizionale e alla logica dei predicati (che aggiunge variabili, funzioni e quantificatori). Questa parte del programma, che costituisce un momento di interdisciplinarietà con l'insegnamento di matematica, nel quadro dell'informatica è utile soprattutto per il supporto che fornisce all'attività di programmazione. Si tratta infatti di rendere consapevole lo studente dell'esistenza di speciali entità "matematiche" (le proposizioni, i predicati, i connettivi, i quantificatori) che non fanno riferimento a "valori numerici" ma a "valori di verità". Questa consapevolezza risulta un prerequisito fondamentale per una piena comprensione e un efficace utilizzo delle *condizioni* all'interno delle istruzioni algoritmiche di selezione e iterazione. Inoltre l'introduzione alla logica permette di riflettere sul ruolo della nozione di verità all'interno dell'uso del linguaggio e sulle analogie e le differenze fra le lingue parlate e i linguaggi formalizzati come quelli di cui si occupa l'informatica.

Programmazione

Una volta terminata l'introduzione alla logica, si può tornare a occuparsi della programmazione informatica con rinnovato vigore. La programmazione, che nel corso del primo anno era stata introdotta soprattutto allo scopo di esemplificare il penultimo livello del modello a strati del computer, adesso diventa finalmente oggetto privilegiato di studio e di pratica di laboratorio. Da una settimana all'altra – per tutto l'anno scolastico e poi anche per gli anni a venire – gli studenti sono chiamati a svolgere un esercizio di programmazione assegnato, introdotto e successivamente corretto e discusso in classe. Questa pratica dovrà accompagnare costantemente gli studenti per tutti e cinque gli anni. A tal scopo, è necessario predisporre un repertorio di esercizi che siano in grado di interessare e invogliare gli studenti all'attività di modellizzazione e problem solving.

Questo è a mio avviso un punto cruciale dell'insegnamento della programmazione non abbastanza considerato da chi si occupa di didattica dell'informatica. Lo studente è di solito invogliato a risolvere un problema che egli ritiene utile e interessante, mentre lo stesso studente affronta

controvoglia un problema che ha come unica finalità la risoluzione del problema medesimo. Ad esempio “Scrivere un programma che permetta di simulare una serie di partite a blackjack”, oppure “scrivere un programma che permetta di giocare all’impiccato” sono problemi di programmazione che coinvolgono gli studenti e li invogliano a trovare una soluzione, mentre “scrivi un programma che visualizzi tutti i divisori dispari di 97 compresi fra 20 e 35” oppure “calcola la somma di tutti i numeri pari compresi fra 1 e 100” sono esercizi autoreferenziali che generalmente annoiano e deprimono chi li deve svolgere. Eppure sui libri si trovano principalmente esercizi di quest’ultimo tipo. Né l’argomento interessante deve aver a che fare per forza con i giochi o con altre frivolezze: anche “scrivi un programma che permetta di risolvere equazioni di primo e secondo grado” è un programma interessante per lo studente, perchè gli fa vedere che l’informatica può far svolgere al computer il lavoro che nelle esercitazioni di matematica tocca allo studente medesimo. Più in generale, la scelta del repertorio di esercizi può rappresentare un momento di forte interdisciplinarietà collegando l’informatica a una pluralità di materie, ma questo può avvenire soltanto a condizione che l’insegnante dedichi una parte consistente della sua attività di progettazione didattica alla costruzione del repertorio di esercizi. Il criterio con cui gli esercizi devono essere assegnati deve sempre essere quello del “valore aggiunto”: oltre a permettere di applicare una qualche nozione informatica, l’esercizio deve dare una soddisfazione aggiuntiva allo studente, sia di ordine pratico, sia di tipo ludico, sia di natura interdisciplinare.

A questo scopo, è importante che siano introdotti il prima possibile alcuni strumenti di programmazione che permettono di ampliare notevolmente la gamma dei problemi affrontabili e risolvibili per mezzo di un programma. A tale scopo non solo si riprendono i concetti chiave di assegnazione, selezione, iterazione, ma anche si introducono le istruzioni per la generazione dei numeri casuali, si estende il campo delle istruzioni di input e output considerando non solo tastiera e monitor, ma anche i files, per poi arrivare a trattare anche di vettori e matrici in modo da poter sottoporre agli studenti esercizi che abbiano a che fare non solo con numeri, ma anche con testi e con strutture grafiche.

Reti

Un altro tema fondamentale del secondo anno è l’introduzione alle reti di calcolatori. Qui si ha finalmente la possibilità di tornare a discutere l’analogia cognitiva fra computer e essere umano. Come le persone non solo pensano ma anche comunicano fra loro, così possono fare i computer. Come le persone si connettono per formare una comunità, così i computer si connettono per formare una rete. In entrambi i casi perché la cooperazione e la comunicazione funzionino occorre definire e rispettare delle regole. Gli insiemi regole che governano le reti di computer si chiamano protocolli.

La scienza che studia le reti di computer è una branca dell'informatica che riceve un nome specifico: telematica.

Al pari del singolo computer, la rete si organizza in una serie di livelli gerarchicamente articolati. Anche in questo caso, la distinzione fra hardware e software si rivela cruciale. Il primo livello su cui si basano le reti è infatti il livello delle connessioni fisiche, mentre i livelli successivi procedono verso gradi di una progressiva astrazione per descrivere i quali si fa solitamente riferimento al modello ISO-OSI, che prevede i seguenti livelli: data-link, rete, trasporto, sessione, presentazione, applicazione. Questi livelli verranno introdotti e spiegati uno per uno, in analogia a quanto era stato fatto nel corso del primo anno per i livelli che costituiscono il singolo computer. Inoltre per rendere percepibile direttamente che cosa significa costruire e configurare una rete di computer si predispongono lezioni di laboratorio in cui – in collaborazione con l'ufficio tecnico – si illustra, con visione diretta di dispositivi e di software, il funzionamento delle rete scolastica.

L'ultimo livello del modello ISO-OSI, il livello di applicazione, riguarda l'uso della rete da parte dell'utente per degli scopi ben precisi: navigare sul web, mandare un messaggio di posta elettronica, caricare o scaricare file da un sito. In tal senso il lavoro di analisi svolto nel corso di questo modulo didattico è isomorfo a quello svolto nel primo anno: inizialmente lo studente si rende conto di conoscere soltanto il primo e l'ultimo livello del modello in questione: da una parte le componenti hardware direttamente visibili (per esempio il modem o il router), dall'altra le componenti software direttamente utilizzabili (per esempio i programmi di navigazione). Il lavoro didattico consiste allora nel fargli prendere consapevolezza e acquisire dimestichezza con gli strati software intermedi che permettono di realizzare le funzioni desiderate sui dispositivi fisici disponibili.

Scrittura scientifica

L'ultimo punto che ritengo importante affrontare in conclusione del primo biennio concerne le norme di scrittura scientifica. Si tratta di un insieme di regole fondamentali per la redazione di un documento elettronico (sia questo un file word o pdf, o una presentazione powerpoint) che rispetti alcuni principi elementari di correttezza per quanto riguarda sia la dimensione sintattica sia quella semantica. Si tratta insomma di sfruttare l'informatica come forma privilegiata di accesso a una modalità di scrittura che è di fondamentale importanza in ambito tecnico e scientifico ma che riceve di solito scarsissima attenzione negli attuali ordinamenti didattici. L'idea è di educare gli studenti a svolgere con adeguata contezza le seguenti fondamentali attività: cercare appropriatamente informazioni sul web (usando oltre all'utile ma non sempre affidabile Wikipedia, anche uno strumento molto più potente quale Google Scholar); citare le fonti anziché copiare abusivamente testi altrui (la violazione del diritto d'autore è reato, questo è un principio che la scuola non

ribadisce con sufficiente durezza); costruire bibliografie e sitografie; inserire appropriatamente le note; rispettare le norme tecniche di scrittura (punteggiatura, spaziatura, titolatura, uso appropriato di corpo tipografico, grassetto, corsivo e sottolineato, inserimento di figure e immagini, giustificazione e interlinea); imparare a conteggiare battute e cartelle di un testo; costruire presentazioni powerpoint efficaci e funzionali che non siano semplicemente pagine di testo proiettate su schermo (la regola aurea dello slide show è "una frase per diapositiva, non una pagina per diapositiva").

Chiaramente in questa fase è fondamentale che l'esercitazione sulla scrittura scientifica avvenga su temi che risultano interessanti e rilevanti per lo studente. Qui il ruolo dell'interdisciplinarietà si rivela nuovamente cruciale. L'ideale sarebbe che lo studente costruisse un proprio testo e una propria presentazione che poi verranno esposti, discussi e valutati di fronte all'intera classe alla presenza dell'insegnante di informatica e degli insegnanti delle altre materie coinvolte.

Il terzo anno

I due grandi temi della didattica del terzo anno sono il web design e la programmazione avanzata. I due discorsi principali sviluppati nel corso del secondo anno – le reti e la programmazione strutturata – vengono così rilanciati a un livello di complessità e di professionalità superiore. Dopo aver appreso come è fatta e come funziona una rete di computer, si impara finalmente a progettare contenuti che possano essere fruiti attraverso la rete stessa. Dopo aver imparato come è fatto e come funziona un linguaggio di programmazione, si impara a sfruttarlo al meglio attraverso le strategie avanzate di analisi dei problemi (top-down, bottom-up, gerarchie concettuali) e la conseguente articolazione del programma in sottoprogrammi (paradigma funzionale) oppure in oggetti (paradigma object-oriented). Vediamo ora nel dettaglio come si possono sviluppare questi due discorsi didattici.

Web design

L'introduzione al web design passa attraverso l'apprendimento dei linguaggio HTML e CSS per la progettazione grafica dei siti e di un linguaggio lato server come PHP, JSP o ASP (a seconda che nel primo biennio si sia privilegiato come linguaggio di programmazione C, Java o Visual Basic) per la creazione di pagine dinamiche in grado di interagire con l'utente e di effettuare elaborazioni. Agli studenti viene richiesto di realizzare inizialmente semplici pagine e poi di renderle interattive attraverso l'introduzione di form e il collegamento con una pagina dinamica. La programmazione lato server permette allo studente di mettere a frutto le competenze basilari di programmazione acquisite nel primo biennio all'interno di un nuovo ambito di applicazione, quello del web design. Anche in questo caso, è fondamentale che il tipo di esercizio proposto (nella fattispecie, il tipo di pagine web che si richiede allo studente di realizzare) non sia fine a se stesso ma abbia una forte componente interdisciplinare, offrendo la possibilità di creare un piccolo sito che non serva soltanto a esercitarsi nell'uso dei linguaggi di web design ma abbia anche un valore intrinseco quanto ai contenuti e al servizio offerto dal sito. In questa fase sarebbe ugualmente importante che gli studenti potessero dare piena attuazione alla propria attività di web designer caricando effettivamente il proprio sito in uno spazio dedicato sul web, attraverso software ftp come ad esempio Filezilla.

Programmazione avanzata

Dopo l'ampia e proficua digressione nell'ambito dei linguaggi lato server, la programmazione avanzata riporta l'attenzione ai linguaggi di programmazione tradizionali, con un portentoso salto di scala in termini di complessità e di potenzialità. L'idea della programmazione avanzata è infatti

quella di passare da semplici programmini, mirati a risolvere piccoli problemi isolati, a programmi complessi che coinvolgono spesso una molteplicità di programmatori e che puntano a risolvere problemi di grande portata. Si tratta insomma di passare dal semplice “saper programmare” che era l’obiettivo principale del primo biennio al “saper programmare come dio comanda”. Volendo richiamarci al paragone con il latino, possiamo dire che si tratta di passare dal fare brevi versioni al tradurre intere opere di letteratura.

Gli argomenti principali della programmazione avanzata sono: sottoprogrammi, menu, funzioni, procedure, ricorsione, puntatori, strutture, classi, ereditarietà e polimorfismo. La maggiore difficoltà didattica in questo caso non riguarda l’esposizione dei concetti, ma l’assegnazione e lo svolgimento degli esercizi. Il problema è infatti che la programmazione avanzata è fatta per risolvere problemi complessi e che quindi per dare un senso al suo utilizzo occorrerebbe assegnare esercizi complessi, la cui taglia però eccede le tempistiche delle esercitazioni, delle verifiche e della valutazioni scolastiche. La tipica obiezione degli studenti più intelligenti quando si presenta loro la programmazione avanzata è la seguente: perché dobbiamo usare funzioni, procedure oppure le classi quando potremmo risolvere molto più agevolmente lo stesso problema con un unico programmino? L’unico modo davvero persuasivo per rispondere a questa obiezione consisterebbe nell’assegnare alla classe un programma di una complessità tale da rendere indispensabile l’uso di sottoprogrammi o di oggetti. Questo si può fare, ed è anzi un’ottima occasione per impostare un lavoro di gruppo, ma richiede estrema attenzione da parte dell’insegnante nella scelta del progetto, nella divisione del lavoro e nel coordinamento delle varie attività. L’apprendimento della programmazione avanzata può essere in tal senso un momento decisivo per il passaggio da un approccio solitario e individualistico alla programmazione a una forma più realistica e più professionale di programmazione intesa come attività cooperativa.

Il quarto anno

I due grandi temi della didattica del quarto anno sono le basi di dati e il mark-up semantico. Si tratta di due argomenti fortemente correlati, nella misura in cui il mark-up semantico si può considerare come una generalizzazione a dati generici delle tecniche di descrizione e modellizzazione che le basi di dati utilizzano relativamente a insiemi di dati circoscritti. In altre parole, le basi di dati modellizzano e descrivono soltanto *i propri dati*, mentre i linguaggi di mark-up permettono di realizzare descrizioni valide *per qualsiasi dato*. In tal senso i linguaggi di mark-up possono funzionare come "lingua franca" attraverso la quale database differenti possono sia comunicare fra di loro, sia mettere a disposizione i propri dati per un'efficace ricognizione da parte dei motori di ricerca.

Database

La progettazione di basi di dati si può considerare come il punto di arrivo del discorso sulla programmazione svolto nei tre anni precedenti. I database rappresentano infatti un decisivo potenziamento dei linguaggi di programmazione quanto alle modalità di rappresentazione e di archiviazione delle informazioni. Laddove i linguaggi tradizionali possono ricorrere soltanto a variabili, vettori, strutture e file, o al limite a gerarchie di classi, il database, attraverso un software apposito denominato DBMS, mette a disposizione una modalità di definizione dei dati molto più potente, flessibile, e soprattutto molto più vicina al modo di ragionare e di classificare dell'intelligenza umana. In questo senso la metodologia di progettazione dei database è di fondamentale importanza per il perfezionamento delle abilità di modellizzazione dello studente. Essa comprende una fase di progettazione concettuale (modello ER), una fase di progettazione logica (modello relazionale) e una fase di implementazione (linguaggio SQL). Una volta realizzato, il database può essere proficuamente interrogato mediante richieste formalizzate (query), anch'esse espresse in linguaggio SQL, che permettono di estrarre dalla base di dati le informazioni utili per determinati scopi pratici.

Le esercitazioni sui database sono proposte agli studenti con le stesse modalità delle esercitazioni sui linguaggi di programmazione. Anche in questo caso è importantissimo costruire un repertorio di esercizi appropriato, che abbiano sempre un valore non solo informatico ma anche interdisciplinare o più in generale sociale e culturale. Allo studente viene fornita una descrizione di situazione a partire dalla quale egli deve produrre una modellizzazione adeguata che conduca sino alla realizzazione, al caricamento e all'interrogazione di un database. Un altro parametro importante in quest'ottica è la scelta del DBMS. In base al principio per cui è inopportuno far sembrare allo studente le cose più semplici e banali di quello che in realtà sono, si ritiene che sia molto meglio

partire fin da subito con un database professionale come MySql (magari nella versione con interfaccia grafica di PhpMyAdmin), lasciando l'uso facilitato di prodotti come Access a una fase successiva.

Infine è importante notare che il database non è tanto alternativo alla programmazione tradizionale, quanto piuttosto complementare. Il database è infatti essenzialmente uno strumento di rappresentazione dei dati che integra significativamente l'apparato rappresentazionale dei linguaggi di programmazione, ma che non può certo competere con la loro potenza di calcolo. In tal senso i linguaggi di programmazione sono spesso forniti di apposite librerie che permettono loro di interfacciarsi con il DBMS di un database, eseguire delle query e poi utilizzare i dati estratti dal database per ulteriori elaborazioni. L'esempio che ritengo più proficuo proporre agli studenti è quello dell'integrazione fra database e linguaggi di programmazione lato server, i quali sono in grado di eseguire query per estrarre informazioni dal database, per poi elaborarne i risultati e presentarli all'interno di pagine web.

Mark-up semantico

L'introduzione ai linguaggi di mark-up e al web semantico costituisce un autentico cambio di marcia nella didattica, in grado di preparare lo studente al programma ad alto grado di interdisciplinarietà e sperimentazione che dovrebbe caratterizzare il quinto anno. L'idea del mark-up semantico è infatti quella di uno strumento di descrizione e modellizzazione in grado di applicarsi a qualsiasi ambito disciplinare, e che permetta ai programmi informatici di *capire* il significato dei dati che stanno manipolando. Il linguaggio fondamentale in tal senso è XML, che viene presentato allo studente come un'estensione di HTML che muove dal mero ambito grafico fino a una molteplicità di possibili contenuti. In vista del lavoro da svolgere nel corso del quinto anno, vengono poi introdotti linguaggi come RDF e OWL che estendono ulteriormente le potenzialità di rappresentazione di XML, fino ad arrivare alla nozione di "ontologia applicata", cioè di rappresentazione formalizzata (e quindi comprensibile per i programmi informatici) di interi domini della conoscenza umana.

Il quinto anno

L'obiettivo principale del quinto anno consiste nell'esplicitare il più possibile il carattere interdisciplinare dell'insegnamento dell'informatica, che si rivela così un autentico paradigma di *scienza applicata*. L'insegnante e gli studenti sono chiamati a mettere finalmente a frutto le conoscenze apprese nel corso dei due bienni, e a dare piena attuazione a fondamentali strumenti di progettazioni quali i linguaggi di programmazione strutturata, il web design, le basi di dati e il mark-up semantico, nonché le abilità di scrittura scientifica. Con questo, la didattica del quinto anno si prefigge sostanzialmente due scopi: da una parte, utilizzare il potenziale di rappresentazione dell'informatica per esplorare una varietà di campi di sapere, in modo da preparare gli studenti alla scelta e alla frequentazione dei corsi universitari; dall'altra, realizzare una relazione di fine anno che non sia la solita tesina abborracciata che troppo spesso viene propinata ai malcapitati commissari di esame, e che abbia invece i dignitosi connotati di una genuina relazione scientifica, come si confà a un candidato al diploma del liceo delle scienze applicate. A tale scopo, la didattica del quinto anno dovrebbe porsi come obiettivo primario l'esplorazione dei più significativi punti di contatto fra l'informatica e gli altri insegnamenti del piano di studi, oltre che i possibili contatti col il mondo universitario e professionale, con particolare attenzione ai soggetti economici e culturali attivi sul territorio.

Lingua e letteratura italiana

In questo ambito disciplinare ci sono almeno due punti di contatto di notevole interesse. Il primo è il discorso sulla formalizzazione della grammatica, che porta ad affrontare la questione del rapporto fra lingue naturali e linguaggi artificiali, e della possibilità per il computer di comprendere e utilizzare lingue simili alle lingue naturali (cfr. Ausiello, D'Amore, Gambosi 2003). Il secondo punto riguarda invece la possibilità di rappresentare e modellizzare testi narrativi attraverso l'individuazione degli elementi strutturali (personaggi, conflitti, intenzioni, azioni) e la loro rappresentazione in termini di basi di dati e linguaggi di mark-up che rendono possibili procedimenti di "annotazione semantica" (cfr. http://www.cadmos-project.org/).

Lingua e cultura straniera

Per quanto riguarda la lingua inglese, va notato innanzitutto che si tratta della lingua nella quale sono formulati i principali linguaggi di programmazione, e sono scritti i più importanti volumi e articoli scientifici riguardanti l'informatica, per molti dei quali – soprattutto i più recenti che spesso, in una disciplina così dinamica, sono anche i più importanti – non è disponibile una traduzione italiana. In tal senso, un fondamentale momento di interdisciplinarietà, consiste nel lavoro di lettura

e comprensione di articoli scientifici in inglese su temi riguardanti l'informatica. Un altro importante elemento di connessione fra la lingua straniera e l'inglese riguarda la questione dei software di traduzione automatica che possono essere analizzati e valutati sia sotto il profilo tecnico sia da un punto di vista linguistico. Infine, va tenuta in conto la possibilità di avvalersi della metodologia CLIL (Content and Language Integrated Learning), e di tenere alcune lezioni di informatica direttamente in lingua inglese.

Storia

L'informatica, rispetto alle altre scienze, ha per oggetto un artefatto umano, il computer, che ha un'origine storica e che si evolve storicamente. In tal senso l'informatica ha un rapporto privilegiando con la storia, e in particolare con la storia della seconda metà del Novecento nel corso della quale il computer si impone progressivamente come autentico protagonista. Più in generale, l'informatica può essere situata all'interno della storia delle tecnologie e dei media (cfr. Ortoleva 2002, Briggs e Burke 2003), portando lo studente a riconoscere e rintracciare l'origine storica di una serie di artefatti e di pratiche talmente radicate nella nostra società da sembrare ormai suoi tratti essenziali. Si tratta quindi, da una parte, di tracciare una storia dell'informatica e di collocarla nel quadro più ampio della storia delle tecnologie e dei media; dall'altra, di considerare in che modo questa storia della tecnica interagisca con la storia canonicamente intesa, e in particolare con le sue dimensioni economiche, politiche e culturali.

Filosofia

Con una semplificazione forse eccessiva ma tutto sommato veritiera, la filosofia del secondo Novecento viene normalmente suddivisa di due grandi ambiti, quello analitico e quello continentale. Mentre la filosofia continentale si muove nel solco dell'idealismo, dell'esistenzialismo e dell'ermeneutica, la filosofia analitica trova i suoi punti di riferimento nelle riflessioni di pensatori come Frege, Russell e Wittgenstein sul funzionamento del linguaggio, della logica, e della mente umana. In tal senso la filosofia analitica ha un potenzialmente rapporto privilegiato con l'informatica, rapporto che storicamente trova la sua piena attuazione nel dibattito sull'intelligenza artificiale e nel programma di ricerca delle scienze cognitive (cfr. Hofstadter 1979, Hofstadter e Dennett 1981). L'analogia fra mente e corpo da una parte e hardware e software dall'altra è il tema di un'ampia e importante discussione filosofica, imperniata sulle nozioni di rappresentazione, di intenzionalità e di coscienza, che può essere peraltro agevolmente introdotta ed esemplificata agli studenti mediante la visione di film come *Blade Runner* o *A.I. – Intelligenza artificiale*.

Un'altra area filosofica che ha uno stretto legame con l'informatica è quella dell'ontologia, intesa come disciplina che si occupa di analizzare e catalogare le entità che costituiscono l'intero dominio dell'essere (cfr. Varzi 2001). La saldatura fra informatica e filosofia sul tema dell'ontologia ha dato vita negli ultimi decenni a un importante programma di ricerca che va sotto il nome di "ontologia applicata" (cfr. Smith 2008), che è strettamente legato al tema del web semantico, e che può pertanto essere agevolmente e proficuamente sottoposto agli studenti.

Matematica

Il più evidente punto di contatto fra informatica e matematica, per quanto riguarda il programma del quinto anno, consiste nell'ambito del calcolo numerico, per mezzo del quale l'informatica è in grado di offrire soddisfacenti soluzioni approssimate a problemi matematici. Un caso esemplare è quello del calcolo delle radici di equazioni attraverso il metodo delle tangenti e il metodo delle secanti. Agli studenti si chiede di sfruttare le conoscenze nel campo dei linguaggi di programmazione per implementare questi algoritmi con appropriate inferacce utente. Un altro caso paradigmatico di applicazione del calcolo numerico che può essere considerato riguarda il metodo dei rettangoli e il metodo dei trapezi per il calcolo degli integrali.

Più in generale, anche argomenti come i limiti, le successioni e le serie possono valere da spunto per proporre agli studenti programmi in grado di rappresentare queste entità matematiche attraverso algoritmi di calcolo. Infine l'informatica permette di approfondire il discorso sulla probabilità realizzando programmi che mettono in evidenza il legame fra la definizione classica di probabilità (casi favorevoli fratto casi possibili) e la definizione statistica (numero di successi fratto numero di esperimenti).

Fisica

La relazione fra informatica e fisica può essere approfondita facendo leva sulla nozione di simulazione, e sulla sua esemplificazione attraverso programmi che, utilizzando le leggi fisiche e le condizioni iniziali di un sistema, ne rappresentino l'evoluzione permettendo di fare previsioni sui suoi stati futuri. Un altro aspetto in base al quale può essere considerata la relazione fra informatica e fisica riguarda il ruolo della fisica nelle tecnologie che permettono la realizzazione dell'hardware dei computer (elettronica dei semiconduttori, magnetismo, raggi laser) e del livello fisico delle reti informatiche (onde elettromagnetiche, fibre ottiche, satelliti). Infine, è importante mettere in evidenza anche attraverso l'ausilio di materiali multimediali e di esempi specifici il ruolo decisivo svolto nella ricerca della fisica contemporanea dalla potenza di calcolo che l'informatica è in grado di mettere a disposizione.

Scienze naturali

La relazione fra informatica e scienze naturali ci porta ancora una volta a considerare il tema basilare delle scienze cognitive, e cioè l'analogia fra l'hardware e il software da una parte, e il corpo e la mente umana dall'altra. Nella fattispecie, si tratta di evidenziare analogie e differenze fra il computer e il cervello umano, mettendo a confronto l'ambito dell'informatica con quell'area delle scienze naturali che va sotto il nome di neuroscienze. Da una parte l'informatica fornisce schemi di implementazione del ragionamento che possono risultare utili per formulare ipotesi sul funzionamento del cervello umano (approccio funzionalista). Dall'altra, le evidenze empiriche delle neuroscienze portano alla luce modalità di funzionamento specifiche del cervello (approccio connettivista) che differiscono da quelle tipiche del computer ma che possono aprire la strada per la realizzazione di strutture informatiche di nuova generazione (l'ipotesi delle reti neurali).

Un caso paradigmatico che può essere presentato agli studenti per esemplificare analogie e differenze fra computer e cervello è quello della visione artificiale (cfr. Marr 1983), in cui i programmi informatici sono finalizzati alla realizzazione di prestazioni di riconoscimento paragonabili a quelle svolte del sistema visivo.

Disegno e storia dell'arte

Portando a termine una rivoluzione visiva avviata dalla fotografia e dal cinema (cfr. Benjamin 1955), l'informatica ha fortemente influenzato l'evoluzione delle arti visive nell'era contemporanea, in particolare per quanto concerne l'architettura, il cinema e più in generale il mondo dell'arte. In particolare l'uso dell'informatica a fini artistici ha portato alla concezione di opere interattive, come ad esempio *Façade* (http://www.interactivestory.net/), che possono essere gratuitamente fruite su Internet e che potrebbero diventare oggetto di analisi e di approfondimento da parte degli studenti. Il tal senso uno studioso come Lev Manovich (2001, 2010) ha parlato di un "linguaggio dei nuovi media" e di una "software culture" di cui ritengo proficuo mostrare agli studenti i più originali esiti disponibili on-line. Parlare oggi di disegno e arte ignorando il contributo dell'informatica significa confinarsi in un'epoca passata, rinunciando a essere cittadini del proprio tempo.

Conclusioni

Nella mia relazione mi sono proposto di mostrare quale ruolo l'informatica può svolgere nel nuovo liceo delle scienze applicate, proponendo un piano didattico che permetta di realizzare le indicazioni ministeriali. Ho dunque cercato di mostrare in che modo sia possibile sviluppare i principali aspetti dell'informatica nel corso dei due bienni, in modo da utilizzare l'ultimo anno per insistere soprattutto sugli aspetti di interdisciplinarietà sulla cui importanza il documento ministeriale insiste fortemente.

Ritengo che la principale peculiarità dell'insegnamento dell'informatica nel liceo delle scienze applicate debba essere il suo ruolo formativo e culturale, di contro alla predominante funzione professionalizzante che giustamente caratterizza l'insegnamento dell'informatica nelle scuole tecniche e professionali. In tal senso ho insistito molto sul paragone fra il ruolo dell'informatica nel liceo delle scienze applicate e il ruolo del latino nel liceo scientifico tradizionale.

Rileggendo quanto ho scritto, mi rendo conto che il programma che propongo è molto ambizioso e molto impegnativo, e richiederebbe un lavoro ingente agli insegnanti ma soprattutto agli studenti. Un critico pessimista potrebbe agevolmente obiettare che, soprattutto nel contesto della scuola attuale – caratterizzata da classi sempre più sovraffollate, studenti sempre più distratti e demotivati, insegnanti sempre più demotivati e depressi – proporre un programma così culturalmente ambizioso è un atto di un velleitarismo intollerabile, come uno che proponesse di organizzare un aperitivo a base di caviale per un gruppo di persone che sta morendo di fame.

L'obiezione non è del tutto peregrina, ma ritengo che la mia proposta abbia i mezzi per fronteggiarla. L'informatica nel liceo delle scienze applicate può essere l'avamposto congeniale in cui combattere la battaglia per una scuola superiore più seria e più dignitosa. Una scuola che non tratti gli studenti come dei bambini sciocchi e viziati, ma attribuisca loro un appropriato livello di razionalità e di responsabilità, pretendendo un comportamento consequenziale.

A tal scopo, è fondamentale impedire che passi l'idea assurda che le ore di informatica si possano ridurre, come talvolta ho l'impressione che accada, a un momento di svago prolungato, a uno spazio di intrattenimento in cui gli studenti navigano liberamente su internet o usano programmi come Windows e Word che in realtà sanno già usare per conto proprio, spesso in modo più rapido ed efficace di quanto i loro stessi insegnanti non sappiano fare. La difesa del valore scientifico e culturale dell'insegnamento dell'informatica, in tal senso, è una battaglia emblematica se si vuole una scuola che sappia sì stare al passo con i tempi senza, ma senza sacrificare la propria autorevolezza. Una scuola che sappia proporre una visione critica del mondo contemporaneo, senza ridursi a ratificare gli aspetti più fatui e corrivi dell'epoca in cui ci troviamo a vivere.

Bibliografia

Ausiello G, D'Amore F., Gambosi G, *Linguaggi, modelli, complessità***,** Milano, Franco Angeli, 2003

Benjamin, W., *Das Kunstwerk im Zeitalter seiner technischen Reproduzierbakeit* in *Schriften*, Frankfurt am Main, Suhrkamp Verlag, 1955; tr. it. *L'opera d'arte nell'epoca della sua riproducibilità tecnica – Arte e società di massa*, Einaudi, 1966

Berto, F., *Logica da Zero a Gödel*, Laterza, 2007

Briggs, A., Burke, P., *A Social History of the Media: From Gutenberg to the Internet*, Polity Press, 2003; tr. it. *Storia sociale dei media. Da Gutenberg a Internet*, Il Mulino, 2007

Cooper, S., Dann, W. P., Pausch, *R., Learning to Program with Alice*, Prentice Hall, 2006

Dennett, D., Hofstadter, D., *The Mind's I: Fantasies and Reflections on Self and Soul*, 1981; tr. it. *L'io della mente. Fantasie e riflessioni sul sé e sull'anima*, Adelphi 1993

Hofstadter, D., *Gödel, Escher, Bach: an Eternal Golden Braid*, 1979; tr. it. *Gödel, Escher, Bach –Un'eterna ghirlanda brillante*, Adelphi 1984

Manovich, L., *The Language of New Media*, MIT Press, 2001; tr. it. *Il linguaggio dei novi media*, Olivares , 2002

Manovich, L., *Software takes Command*, Creative Commons License, 2008; tr. it. *Software Culture*, Olivares, 2010

Marr, D., *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W. H. Freeman and Company, 1983

Ortoleva, P., *Mediastoria*, Il Saggiatore, Milano 2002

Smith, B., "Informatica" in Ferraris, M., (a cura di), *Storia* dell'*ontologia*. Milano, Bompiani, 2008

Tanenbaum, A.S., *Structured Computer Organization*, New York, Prentice-Hall, 1999; tr. it., *Architettura del computer: un approccio strutturato*, Torino, UTET, 2000

Varzi, A., *Parole, oggetti, eventi e altri argomenti di metafisica*. Carocci, 2001

Sitografia

Cadmos Project
http://www.cadmos-project.org/

Indicazioni ministeriali per l'informatica nel liceo delle scienze applicate
http://www.indire.it/lucabas/lkmw_file/licei2010///INFORMATICA_Scientifico%20opz.%20scienze%20applicate.pdf

Façade
http://www.interactivestory.net/

Lev Manovich
new media | digital humanities | cultural analytics | software studies
http://manovich.net/

The Semantic Web by Tim Berners-Lee, James Hendler and Ora Lassila | Thursday, Scientific American, 2001, 6
http://www.scientificamerican.com/article.cfm?id=the-semantic-web&print=true