
# UNIVERSITÀ DEGLI STUDI DI L'AQUILA

**PERCORSO ABILITANTE SPECIALE (P.A.S.)**

Indirizzo: INFORMATICA

Classe di abilitazione: A042

# RELAZIONE FINALE

Relatore:

Prof. Luca Forlizzi

Abilitando

Mauro Gregori

ANNO ACCADEMICO 2013-2014

# INDICE

## Un'introduzione critica

Mi è stato chiesto di racchiudere in queste poche pagine un resoconto personale dell'esperienza di Percorso Abilitante Speciale (PAS) frequentato in codesta Università e di porre in luce pregi e criticità. Preferisco suggerire le criticità in questa breve introduzione per passare rapidamente nel seguito agli aspetti positivi.

Il PAS di L'Aquila ha mostrato all'inizio il suo lato peggiore (all'inizio utilizzavo il termine "punitivo") con un programma nutrito di discipline importanti e non banali compresse in orari al limite del possibile, in sovrapposizioni agli impegni di fine anno scolastico e un costo complessivo non indifferente: appariva una fatica erculea! Il notevole ritardo con cui si è avviato il PAS ha costituito indiscutibilmente una criticità oggettiva, che ha pesato tanto sugli abilitandi quanto sui docenti. Ma fa parte della maturità di una persona imparare ad affrontare ogni situazione della vita tirando fuori valore e questo ho cercato di fare in questo paio di mesi: alla fine tante buone cose sono emerse da questo percorso e nel proseguo relazionerò su questi valori.

Per descrivere meglio la seconda criticità devo premettere un po' di storia personale, così sarà più chiara la prospettiva da cui l'ho rilevata.

Da circa diciotto anni varco gli usci di scuole di ogni ordine e grado per svolgere attività di formazione e aggiornamento a docenti, consulenze a dirigenti, supporto tecnico (e spirituale!) agli addetti ai laboratori e, *last but not least*, educatore di giovani oltre che docente d'Informatica.

Nel 2000, anno del primo concorso abilitante per me disponibile, mi feci vincere da perplessità e disaccordi con la definizione del ruolo che l'allora Ministero della Pubblica Istruzione riservava a noi laureati di Scienze dell'Informazione e decisi di non approfittarne. Continuai in autonomia di vedute la mia personale campagna di evangelizzazione dell'Informatica (con la "I" maiuscola) nella scuola, cercando di contrastare mode e mitizzazione degli strumenti rispetto all'educazione delle menti.

Per molti anni ho potuto permettermi di vivere da "cane sciolto"! Pian piano però è emerso il bisogno di dare maggiore stabilità e ufficialità al mio rapporto con l'istituzione scolastica: ho deciso di "abilitarmi"! Il PAS si è quindi posto come una tappa *sine qua non* del mio percorso di educatore. Questa era la mia esigenza oggettiva.

Altra esigenza, maturata in queste esperienze, era di formarmi da un punto di vista pedagogico. Per tutti quegli anni l'istituzione scolastica mi ha consentito di svolgere il difficile ruolo di educatore, a contatto con giovani in una fase delicatissima del loro sviluppo. Io mi sono sforzato di farlo con sincera passione, imparando dagli errori e spronato dalle soddisfazioni professionali e umane. Mai però mi è stata richiesta una valutazione e certificazione delle competenze pedagogiche e psicologiche così essenziali nel nostro ruolo!

Essendo il PAS riservato a docenti con almeno tre anni di esperienza scolastica, ipotizzavo che il focus sarebbe stato principalmente sugli aspetti didattici e pedagogici quindi mi appariva come valida opportunità per rispondere a questi miei due bisogni: abilitarmi e migliorarmi come educatore. Gli aspetti disciplinari non mi preoccupavano più di tanto: mi sono laureato in un'ottima facoltà, forse le nozioni non sono più freschissime, ma l'impianto culturale generale e le esperienze professionali avute mi permettono di recuperare abbastanza velocemente i concetti dimenticati.

Ecco quindi l'elemento critico da me avvertito: complessivamente mi sarei aspettato una maggiore enfasi sulle problematiche didattiche nei moduli disciplinari e complessivamente più ore dedicate a esperienze pratiche, laboratoriali per gli aspetti pedagogici e psicologici. In parole semplici «sei laureato e già insegni; sul "cosa" dovresti essere già competente, il "come" te lo insegniamo noi!»

Ho avvertito tra i docenti una non uniforme e condivisa strategia rispetto agli obiettivi formativi. Non metto assolutamente in discussione la qualità dei contenuti e delle modalità di trasmissione (non mi arrogo certo le competenze per farlo!), che comunque dati i tempi ristretti sono stai eccellenti. Sollevo la questione di una non evidente unità d'intenti: in alcune materie è stata posta grande enfasi sulle nozioni e meno sui percorsi, modi e tecniche per insegnarle. Come già docente con esperienze pregresse avrei gradito una formazione che mi preparasse a utilizzare le discipline al meglio in ogni indirizzo scolastico,

a programmare più efficacemente la didattica, e mi indicasse strumenti e idee-progetto per portare in classe argomenti anche ostici (ma non impossibili se adattati al mondo degli alunni).

Come risulterà dalle mie conclusioni però il risultato finale è stato a mio parere soddisfacente anche perché ritengo che da ogni problema si possa ricavare valore e questo è stato sicuramente l'epilogo non solo personale ma di tanti colleghi con cui ho avuto l'onore e il piacere di condividere gioie e dolori di questo PAS 2014.

## Profilo personale ed esperienze educative

Le mie prime esperienze come educatore risalgono agli anni '70, quando poco più che adolescente operai come volontario in attività ludico-educative con l'Arciragazzi (http://www.arciragazzi.it) , associazione laica d'educazione extra-scolastica in ambito nazionale.

Dal 1984 e il 1986 svolsi il servizio civile come obiettore di coscienza presso l'Unione Italiana Ciechi di Pisa; tra le varie mansioni, oltre ad attività ludico-educative con bambini non vedenti, partecipai a un convegno nazionale a Treviso sulle prime tecnologie informatiche di assistenza ai non vedenti.

Tra il 1992 e il 1994 sviluppai una tesi di ricerca presso il CNUCE-CNR "Ricostruzione di schemi Entity-Relationship da analisi di programmi contenenti istruzioni SQL", da cui sono stati estratti alcuni articoli di Reverse Engineering presentati in vari convegni tra cui:

- "Reconstruction of ER Schema from Database Applications: a Cognitive Approach" (O.Signore, M.Loffredo, M.Gregori, M.Cima) - ER '94 Proceedings of the 13th International Conference on the Entity-Relationship Approach
- "Using procedural patterns in abstracting relational schemata" (O.Signore, M.Loffredo, M.Gregori, M.Cima) Program Comprehension, 1994. Proceedings., IEEE Third Workshop on., Washington, DC.

Laureatomi in Scienze dell'Informazione presso l'Università degli Studi di Pisa nel 1994 entrai subito nel mondo del lavoro professionale come tecnico e formatore presso varie aziende nazionali e multinazionali.

Trasferitomi in Abruzzo, parallelamente all'attività professionale iniziai quella di docente nella scuola pubblica con incarichi di vario genere. La prima esperienza in ambito scolastico risale al 1997, con un corso di alfabetizzazione informatica presso il Liceo Scientifico "Teofilo Patini" di Castel di Sangro (AQ), in cui avvicinai cinque classi del liceo e dell'Istituto Professionale dell'Industria e dell'Artigianato alle tecniche di *web author*ing (allora si parlava di costruzione d'*ipertesti multimediali*), il tutto realizzato in HTML con Blocco Note e Paintbrush di Windows 3.1. Il corso ebbe incredibile successo non solo per la partecipazione appassionata degli alunni ma anche per la qualità degli elaborati, che spaziavano da semplici ma avvincenti videogiochi a brevi ma ben strutturate enciclopedie tematiche.

A seguire fui coinvolto in vari corsi di alfabetizzazione informatica per docenti e personale non docente di alcune scuole dell'Alto Sangro. In contemporanea continuai a essere reclutato per docenze in corsi d'informatica extracurricolari presso scuole di vari indirizzi: Istituto Professionale Alberghiero di Roccaraso, liceo Scientifico e Istituto Professionale per l'Agricoltura e l'Ambiente di Castel di Sangro. Svolsi anche corsi di formazione per enti quali l'Università "Federico II" di Napoli, la Camera di Commercio di Benevento e l'Agenzia Regionale per i Servizi di Sviluppo Agricolo abruzzese.

Mi preme ricordare un incarico presso il Liceo Scientifico di Castel di Sangro che curai con particolare passione: riguardava la progettazione e l'attuazione di un corso d'informatica per una classe che includeva un alunno con grave handicap uditivo-verbale e motorio, progetto che aveva come finalità collaterale il consolidamento dell'atteggiamento collaborativo del gruppo classe. In quell'ambito insieme alla docente di sostegno, promotrice del corso, nacque l'idea di sviluppare esperienze educative per alunni con disturbi specifici dell'apprendimento e con handicap tramite l'utilizzo di metodi e strumenti informatici. Purtroppo a causa del rapporto precario con l'istituzione scolastica di entrambi il progetto non si concretizzò più.

Per circa otto anni (1998-2006) ho condotto il progetto d'informatica "Insieme verso il futuro" presso le

scuole medie dell'Istituto Comprensivo di Pescocostanzo in cui, oltre a percorsi di alfabetizzazione informatica, ho realizzato con gli alunni delle classi seconda e terza l'ipertesto multimediale "Vita negli Altipiani Maggiori d'Abruzzo", una documentazione storico-ambientale del territorio,.

Contemporaneamente iniziò il periodo di docenze a tempo determinato sia in Informatica (A042) sia in Matematica (classe A047), che perdura a tutt'oggi.

### Esperienze didattiche nella classe di concorso A042 (Informatica)

Ho insegnato in molti indirizzi con classi di diverso livello: istituti professionali, istituti tecnici e licei scientifici. In particolare vorrei citare alcune esperienze per me significative:

- ITIS "Amedeo d'Aosta" di L'Aquila (indirizzo Abacus) classi 4ª e 5ª, a.s. 2010-11.
- Progetti Europei "Scuole e nuovi apprendimenti 1 e 2"
  - A.s. 2011-2012 presso il Liceo Scientifico "E.Fermi" di Sulmona"
  - A.s 2012-2013 presso l'ITC "De Nino" di Sulmona su "CSM ed eCommerce".
- Liceo Scienze Applicate IIS "T.Patini" di Castel di Sangro (due classi 3ª) e ITC "De Nino" (quattro classi del I biennio) di Sulmona, a.s. 2013-2014.

In particolare la docenza nelle classi terze del Liceo Scienze Applicate è stata molto importante per comprendere il "che fare", poiché svolta in un indirizzo in cui a mio parere l'informatica deve ancora trovare il giusto equilibrio tra la mera suggestione culturale, la formazione scientifica, l'addestramento all'uso delle tecnologie e il semplice supporto al *problem solving*.

Tutte esperienze molto significative, purtroppo tutte caratterizzate negativamente dalla *non continuità didattica* che tende a vanificare ogni progettualità curricolare, rende caotico il contesto scolastico e sfiducia il rapporto con l'istituzione.

Al termine di questo sommario resoconto del mio curricolo vorrei sottolineare la positiva influenza che ha avuto la mia esperienza professionale (sistemistica, di sviluppo di applicazioni, d'assistenza, di formazione aziendale e di sviluppo web), per cui ritengo fondamentale per un docente informatico mantenere legami col mondo del lavoro, per la costante e rapida evoluzione di questo settore.

## Ripensare al ruolo dell'informatica nella scuola

Un primo riscontro positivo di questo percorso abilitante l'ho avuto durante le lezioni di Metodi e Strumenti per la Didattica dell'Informatica (MSDI), con la conferma di tante mie ipotesi sulle incongruenze della missione affidata al docente di questa materia nella scuola italiana: per dirla con una metafora, troppo spesso l'informatica è la "Cenerentola" confinata in "cucina" (leggi laboratorio) a "riscaldar minestre" (leggi ECDL)! A parte gli scherzi personalmente mi sono trovato spesso a dover difendere la dignità del ruolo scientifico e culturale dell'informatica contro l'interpretazione puramente tecnologica in cui la si vuole relegare!

Il corso di MSDI mi ha innanzitutto incoraggiato nel perseguire il ruolo di evangelizzatore della *Computer Science* nella scuola e nella società. Oltre ad aspetti metodologici su cui riflettere, di cui dirò nel seguito, ho trovato incoraggiamento in opinioni e suggestioni autorevoli in questa direzione.

Ecco, ad esempio, cosa sostiene il prof. Giuseppe Bizzarri, ricercatore e docente di matematica, fisica e informatica:

«L'informatica ha natura triplice: scienza, tecnologia e strumento trasversale[…]

- **Strumento trasversale alle varie discipline** per l'insegnamento e come ambiente di apprendimento, piuttosto che semplice addestramento all'utilizzo di alcuni pacchetti software o dell'uso di strumenti. […] consideriamo necessario l'utilizzo delle tecnologie per la didattica e l'insegnamento di quelle attitudini informatiche utili sia per l'apprendimento che per la vita sociale e le professioni lavorative.
- Per **informatica come tecnologia** intendiamo lo studio degli aspetti tecnici di uno strumento e del

suo funzionamento. Ad esempio di uno smartphone, della CPU, delle reti ma anche la sintassi, la semantica e i dettagli di un linguaggio di programmazione. Questo è l'aspetto più presente nei programmi e, in alcuni indirizzi, trattato in maniera approfondita.

- **Informatica come scienza**. Si dovrebbe affrontare in maniera più approfondita il pensiero informatico dai punti di vista epistemologico e storico. Far comprendere che l'informatica non è nata insieme ai calcolatori ma molto prima. Insistere sulla trattazione nella scuola superiore di quei concetti che rendono l'informatica una scienza che ha cambiato il modo di pensare e affrontare problemi quotidiani della vita reale, come ad esempio:
    - il **concetto di indecidibilità di alcuni problemi** […]
    - la **tesi di Church-Turing** […]
    - dato un **problema computazionale**, **classificarne la complessità intrinseca**, e progettare un algoritmo risolutivo efficiente, ove possibile
    - le **classi dei problemi P e NP** […]» (Bizzarri)

Bizzarri è convinto che questi argomenti «suscitano interesse e affascinano i giovani» (Bizzarri).

A patto che il docente stesso ne abbia colto l'essenza e sappia trasferirla nella sfera cognitiva degli alunni! Per fare chiarezza dovremmo noi docenti per primi distinguere ciò che attiene alla *Digital Literacy* (alfabetizzazione informatica), alla *Information Technology* e ciò che invece dovrebbe essere il nostro campo d'azione principale: la ***Computer Science***.

Bizzari poi cita la professoressa Jeannette Wing, docente di Computer Science alla Carnegie Mellon University, che ha coniato l'espressione ***computational thinking*** «per indicare il nucleo fondamentale dei concetti alla base dell'informatica. Il *computational thinking* descrive l'attività mentale di formulare problemi in modo che essi abbiano una soluzione computazionale. La soluzione può quindi essere calcolata da un umano, da una macchina, o più in generale da combinazioni di uomini e macchine. Il *computational thinking* merita un posto centrale nell'educazione moderna. L'informatica, evidenziando il *computational thinking*, può assumere un ruolo centrale nell'educazione scolastica in quanto:

- stimola la creatività, illustrando una varietà di modi per affrontare e risolvere un problema;
- è costruttiva: progettare algoritmi è un lavoro ingegneristico, che produce artefatti visibili (sebbene virtuali);
- aiuta a maneggiare la complessità: imparare a risolvere problemi informatici aiuta a risolvere problemi in altre aree e discipline;
- enfatizza accuratezza e precisione nel ragionamento: scrivere programmi che risolvono in modo corretto ed efficace i problemi richiede esattezza in ogni dettaglio.» (Bizzarri)

La citazione completa era d'obbligo perché definisce in modo preciso un concetto che vorrei sostenere in queste pagine:

*il ruolo dell'informatica, anzi di un docente d'informatica, onesto e solido nelle conoscenze, dotato di spirito di ricerca, ricco di esperienze, consapevole delle responsabilità della scienza e della tecnologia sui destini umani, è centrale nel processo educativo dei giovani poiché insegnare informatica vuol dire fornire conoscenze, ambiente culturale, metodi di lavoro e strumenti operativi per potenziare lo sviluppo metacognitivo degli studenti.*

Una particolare attenzione vorrei porre su un argomento che mi sta molto a cuore: nella mia esperienza didattica di questi anni, che include anche supplenze nella classe di concorso A047, Matematica, ho rilevato a livello qualitativo una significativa carenza nelle capacità di **astrazione** degli alunni che si affacciano alla scuola secondaria di secondo grado. L'**astrazione** è il processo mentale con cui si formano idee, immagini mentali, su oggetti, utilizzando informazioni ritenute caratterizzanti, prescindendo dai dettagli irrilevanti. Aggiungerei che è un processo che può essere sviluppato a vari livelli: dalla realtà percepita, dettagliata, ad astrazioni via via più generalizzate (in Informatica il risultato di questo processo conduce appunto alle generalizzazioni di entità o di classi).

L'analisi di Bizzarri del concetto di astrazione utilizza diverse prospettive:

- quella linguistica (de Saussure) attraverso lo studio della genesi dei "segni" in forme atte alla

comunicazione (condivisione di significati);
- quella informatica in cui il fine è creare «un modello concettuale della realtà , definito come l'insieme dei processi che […] permette di comprendere le logiche di funzionamento del sistema stesso». (Bizzarri)

Le due visioni confluiscono nell'**ontologia informatica**: specifica formale (machine-readable) ed esplicita (non ambigua) di una concettualizzazione condivisa.

In Informatica l'astrazione porta quindi a "una rappresentazione (schema concettuale) esaustiva e rigorosa di un dato dominio", cioè uno schema concettuale composto da classi (o dalle loro istanze individuali) e dalle relazioni fra di esse. Ogni istanza è caratterizzata da proprietà e può essere primitiva o derivare da altre (ereditarietà) in uno schema tassonomico.

Queste considerazioni sono fortemente correlate al computational thinking che ha definito il campo di azione di queste conoscenze. Si pensi allo studio e utilizzo dei linguaggi di programmazione (anche solo come strumento descrittivo di algoritmi): linguaggi formali, con cui possono essere espressi risultati "realizzabili, comunicabili e riutilizzabili".

«L'educazione al *computational thinking* oltre che disciplinare il pensiero, introdurre il rigore e sollecitare la creatività, contribuisce a migliorare la percezione della tecnologia e degli strumenti a disposizione e, in definitiva, a migliorare la percezione del mondo e quindi la cultura [Casadei and Teolis, 2011]» (Bizzarri)

Ed è proprio con questi mezzi di crescita culturale che vorrei incidere maggiormente sulla *Weltanschauung* dei miei studenti, sia che essi decidano di diventare medici, architetti, economisti o informatici.

## Metodologie

Per quanto concerne la metodologia di lavoro invece desidero porre l'accento su tre efficaci approcci: l'***active/constructionist learning*** da un lato e il ***cooperative learning*** dall'altro; indissolubilmente collegati tra loro in quanto l'ultimo sostiene i primi e questi motivano il secondo.

### Active learning (apprendimento attivo)

«Per imparare, gli studenti devono fare di più che ascoltare: devono leggere, scrivere, discutere, o impegnarsi a risolvere problemi. Ci si riferisce ai tre settori di apprendimento indicati come conoscenze, abilità e attitudini (KSA), e che questa tassonomia dei comportamenti di apprendimento può essere pensata come "gli obiettivi del processo di apprendimento" (Bloom, 1956). In particolare, gli studenti devono impegnarsi in tali attività del pensiero d'ordine superiore come l'analisi, la sintesi e la valutazione. L'*active learning* impegna gli studenti in due aspetti - fare le cose e pensare alle cose che stanno facendo (Bonwell Eison, 1991).» (Wikipedia)

### Constructionist learning

«Il *costruttivismo* è una teoria della conoscenza (epistemologia), che sostiene che gli esseri umani generano la conoscenza e il significato grazie ad una interazione tra le loro esperienze e le loro idee. Esso è stato sviluppato a partire dal lavoro di Jean Piaget ed è alla base di diverse pedagogie e numerosi metodi di insegnamento.» (Forlizzi, Teorie e Strumenti per la Didattica dell'Informatica, 2014) (Wikipedia)

La ricerca pedagogica sull'insegnamento dell'informatica ha ampiamente verificato l'efficacia di metodi didattici basati su *active learning* e *costruttivismo*. Ciò probabilmente è dovuto al fatto che intimamente connaturati all'informatica: un algoritmo è la formalizzazione di un'attività o meglio di una serie di attività definita in un processo costruttivo di logica sequenzializzazione di operazioni.

Seymour Papert, matematico, informatico e pedagogista collaboratore di Piaget, padre del ***constructionist learning***, parla di "artefatti cognitivi", cioè di strumenti con cui esplorare i modi per cui costruire da soli i propri progetti, provare schemi e manipolare nozioni e idee, modificando lo *status* di "consumatore" di informazioni in quello di "produttore" di conoscenza. Per il costruzionismo la vita

stessa è un processo cognitivo che nasce dall'esperienza individuale. In tale processo ogni essere vivente costruisce il proprio mondo, alternando fasi di disequilibrio e assestamento. Mentre per Karl Popper "vivere è risolvere problemi" e per Konrad Lorenz "vivere è imparare", per Papert potremmo dire che "vivere è imparare costruendo soluzioni". Papert osservò che in alcune civiltà africane i bambini costruivano case in scala o manufatti in giunco. La mente ha bisogno di materiali da costruzione appropriati, esattamente come un costruttore: il prodotto concreto può essere esaminato, discusso, valutato e goduto.

Infine per Papert il computer è un supporto efficace per l'istruzione e un ambiente d'apprendimento che aiuta a costruirsi nuove idee, come strumento per creare simulazioni.

### Cooperative learning

In un ottica costruttivista la classe/laboratorio diventa un *knowledge-building community* (comunità di apprendimento) in cui gli alunni si relazionano e imparano attraverso la discussione, condividono le conoscenze e abilità e costruiscono insieme nuove conoscenze. Si parla quindi di *flipped classroom* (classe capovolta): "capovolgere" la classe significa invertire il tradizionale schema di insegnamento e apprendimento, in cui l'aula "non [è] più il luogo di trasmissione delle nozioni ma lo spazio di lavoro e discussione dove si impara ad utilizzarle nel confronto con i pari e con l'insegnante" (Paolo Ferri, professore associato all'Università di Milano-Bicocca).

Il docente non interpreta più il ruolo del "disseminatore di conoscenze" ma di "facilitatore" o di "coach", che dirige il processo programmando gli obbiettivi, stabilendo i gruppi, assegnando i ruoli e sostenendo i singoli e il collettivo, valorizzando e armonizzando le abilità, alla scoperta di nuove conoscenze e costruzione di nuove competenze.

«L'apprendimento cooperativo è quindi una nuova visione pedagogica e didattica che utilizza il coinvolgimento emotivo e cognitivo del gruppo come strumento di apprendimento ed alternativa alla tradizionale lezione accademica frontale.» (Wikipedia) (Forlizzi, Metodi e Strumenti per la Didattica dell'Informatica )

### Metodo dei progetti

Presentiamo infine un esempio di applicazione di queste metodologie attraverso un metodo che le contempla tutte. Nell'ambito dell'informatica «consiste nel progettare e realizzare un software relativamente complesso, tale da richiedere uno o più giorni di lavoro e rendere opportuna una attività di progettazione e pianificazione prima di procedere nella scrittura dei programmi. Adottando questo metodo didattico, il docente svolge la funzione di supervisore, o mentore, dell'attività svolta dagli allievi.» (Forlizzi, Metodi e Strumenti per la Didattica dell'Informatica )

### La metafora

In un passo della *Poetica* (21, 1457b, 1-10) Aristotele definisce la metafora come «*il trasferimento ad una cosa di un nome proprio di un'altra o dal genere alla specie o dalla specie al genere o dalla specie alla specie o per analogia.*» 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. E nella *Retorica* (III, 1041ob, 6 ss) afferma «*Noi apprendiamo soprattutto dalle metafore.*» (Wikipedia)

La metafora nella didattica dell'informatica è fondamentale: noi trattiamo concetti astratti, intangibili (il software) che modellano e gestiscono informazione. Dobbiamo sollecitare le capacità astrattive dei discenti, ma dobbiamo farlo partendo dal loro vissuto, dalla conoscenza concreta del loro mondo. Bisogna gettare un ponte tra queste due sfere e questo ponte è la metafora:

- una scatola per rappresentare una variabile;
- una cassettiera per l'array;
- lo smartphone con la sua interfaccia, le sue funzionalità e i meccanismi interni, misteriosi e nascosti che lo animano, per rappresentare un'oggetto;
- la sequenza dei gesti per vestirsi, la decisione di prendere o no l'ombrello a seconda del tempo,

l'attesa che spiova per uscire di casa, la ricerca di un paio di guanti in un comò non sono che semplici metafore per rappresentare le strutture di controllo nella programmazione strutturata.

L'uso della metafora è proprio il fulcro del processo di ***scaffolding***.

### Scaffolding

Il termine **scaffolding** (dall'inglese *scaffold*, "impalcatura" o "ponteggio") in psicologia e pedagogia indica l'aiuto dato da una persona esperta ad un'altra meno esperta nello svolgere un compito, risolvere un problema, raggiungere un obiettivo. La metafora dell'impalcatura suggerisce il sostegno che un esperto (adulto o pari) offre ad un apprendista durante la costruzione attiva del suo processo di apprendimento. Per comprendere il termine scaffolding bisogna introdurre il concetto di "zona di sviluppo prossimale" (Lev Semënovič Vygotskij). Lo psicologo distingue due aree che concernono lo sviluppo individuale di un soggetto:

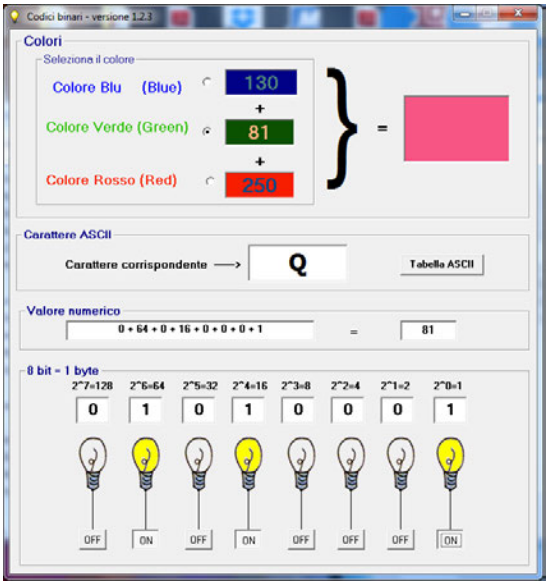
- "area effettiva di sviluppo", si tratta delle competenze effettivamente acquisite ad un certo momento dello sviluppo cognitivo di un individuo;
- "area potenziale di sviluppo", la stessa cosa, per le competenze potenzialmente acquisibili in un futuro ravvicinato o che potrebbe già raggiungere attraverso l'aiuto di una persona esperta.

L'attività didattica dell'insegnante si svolge in quella che viene detta ***zona di sviluppo prossimale*** , collocata tra l'area effettiva di sviluppo e quella potenziale:

- operando un'attività di mediazione (scaffolding) l'allievo viene sostenuto da tale attività,
- aiutando l'allievo a rendere il materiale di studio acquisibile. (liberamente tratto da Wikipedia)

## Strumenti

### Ambienti di Apprendimento basati su Visualizzazione e Animazione (AAVA)




**Visualizzazione** e **animazione** facilitano l'apprendimento di concetti astratti, fornendone rappresentazioni più concrete. La diffusione di computer in grado di gestire immagini e animazioni ha consentito lo sviluppo d'ambienti di apprendimento virtuali in cui oggetti grafici vengono visualizzati e animati interattivamente dagli input prodotti dall'utente.

In passato ho prodotto piccoli strumenti digitali, applicazioni o siti web, ad uso dei miei alunni che incarnano questo concetto. Ad esempio nel 2005 ho costruito una piccola applicazione (vedi figura accanto) che rappresenta un byte come una sequenza di otto interruttori, con cui l'alunno può agire sulle lampadine collegate. Per ogni configurazione di lampadine accese vengono mostrati tre significati: la rappresentazione binaria di un numero (e il corrispondente valore in base 10), il carattere ASCII corrispondente e una combinazione di colori RGB.

Non avevo mai utilizzato quelli che propriamente vengono definiti ***Ambienti di Apprendimento basati su Visualizzazione e Animazione*** (AAVA), strumenti di *active learning* che consentono al discente di costruire soluzioni algoritmiche di problemi.

Essenzialmente si tratta di linguaggi/ambienti di programmazione concepiti in modo specifico per la didattica ma perfettamente utilizzabili anche per implementare applicazioni complesse. Citerò alcune di esse di cui ho avuto breve ma proficua frequentazione nel corso del PAS: LOGO, Scratch e Blockly di code.org. In questa sede accennerò brevemente a LOGO.

### *LOGO*

Pur conoscendone l'esistenza non mi ero mai avvicinato a questo ambiente di sviluppo. Nato dal genio creativo di Seymuor Papert e concepito come artefatto cognitivo per concretizzare la pedagogia costruzionistica,

LOGO è un linguaggio di programmazione deriva dal LISP, formalmente molto rigoroso, orientato alla gestione delle liste e al controllo grafico della "tartaruga", comprensibile ed usabile anche da bambini delle scuole elementari, dimostrando tra l'altro l'utilità del computer come supporto per l'apprendimento anche per i più piccoli.

Dalle esperienze con LOGO avute nel corso del PAS vorrei sottolineare alcune caratteristiche positive, che depongono a favore di una sua possibile futura applicazione didattica:

- permette di gestire le liste (e stringhe) in modo nativo);
- permette la definizione di procedure, rendendolo così un linguaggio procedurale;
- ogni procedura può a sua volta essere interpretata come una funzione in senso matematico, prerogativa ereditata da LISP che fa di LOGO, un linguaggio funzionale;
- permette quindi la definizione ricorsiva di funzioni;
- utilizzabile per l'apprendimento della geometria;
- orientato alla grafica secondo il paradigma della "geometria della tartaruga" (*turtle geometry*), basata sul concetto di orientamento locale, diverso dal più comunemente utilizzato orientamento cartesiano, e funzionale all'uso per il controllo di robot.

### Python

Da circa due anni ho introdotto Python nel curriculo didattico delle mie classi. Mi ha colpito principalmente per due caratteristiche:

- essendo interpretato permette un utilizzo interattivo che consente di sperimentare immediatamente i passi elementari (le istruzioni) di programmi più complessi;
- ha una sintassi basata sull'indentazione che lo rende pulito, facilmente leggibile e adatto a esercitarsi al rigore formale della programmazione.

Da quest'anno ho iniziato a raccogliere in un sito web le esperienze scolastiche realizzate con Python (che prevedo in un prossimo futuro di riprogettare come portale dedicato alla didattica informatica - http://www.studiogregorinformatica.eu/didattica/python). In questo contenitore raccolgo il materiale da me prodotto per e con gli alunni e propongo i link a risorse interessanti per consentire loro di evolvere autonomamente sull'argomento. In particolare segnalo il portale di e-learning http://www.codecademy.com (solo in inglese, ma questo può costituire un valore aggiunto) dedicato all'apprendimento di linguaggi di programmazione (HTML/CSS, Javascript, jQuery, Python, Ruby, PHP).

Lascio la parola al più qualificato David Beazley docente all'Università di Chicago, autore di *Python Essential Reference*, per rispondere alla domanda: "Perché Python nella scuola?"

«*Python è un linguaggio di programmazione divertente e semplice da usare, la cui popolarità è andata via via crescendo nel corso degli ultimi anni. Python è stato sviluppato più di dieci anni fa da Guido van Rossum che ne ha derivato semplicità di sintassi e facilità d'uso in gran parte da ABC, un linguaggio dedicato all'insegnamento sviluppato negli anni '80. Oltre che per questo specifico contesto, Python è stato creato per risolvere problemi reali, dimostrando di possedere un'ampia varietà di caratteristiche tipiche di linguaggi di programmazione quali C++, Java, Modula-3 e Scheme. Questo giustifica una delle sue più rimarchevoli caratteristiche: l'ampio consenso nell'ambito degli sviluppatori professionisti di software, in ambiente scientifico e di ricerca, tra i creativi e gli educatori. Nonostante l'interesse riscosso da Python in ambienti così disparati, potresti ancora chiederti "Perché Python?" o "Perché insegnare la programmazione con Python?". Rispondere a queste domande non è cosa semplice, specialmente quando l'interesse generale è rivolto ad alternative più masochistiche quali C++ e Java.* ***Penso comunque che la risposta più diretta sia che la programmazione in Python è semplice, divertente e più produttiva.*** *Una delle ragioni per cui mi piace Python è che esso permette un ottimo equilibrio tra l'aspetto pratico e quello concettuale. Dato che Python è interpretato, gli studenti possono fare qualcosa*

*quasi subito senza perdersi in problemi di compilazione e link. Inoltre Python è fornito di un'ampia libreria di moduli che possono essere usati in ogni sorta di contesto, dalla programmazione web alla grafica. Questo aspetto pratico è un ottimo sistema per impegnare gli studenti e permette loro di portare a termine progetti non banali. Python può anche servire come eccellente punto di partenza per introdurre importanti concetti di informatica: dato che supporta procedure e classi, possono essere gradualmente introdotti argomenti quali l'astrazione procedurale, le strutture di dati e la programmazione ad oggetti, tutti solitamente relegati a corsi avanzati di Java o C++. Python prende a prestito un certo numero di caratteristiche da linguaggi di programmazione funzionali e può essere quindi usato per introdurre concetti che sarebbero normalmente trattati in dettaglio in corsi di Scheme o di Lisp.*» (Beazley) (Forlizzi, Il concetto di paradigma di programmazione)

Ritengo che tutti questi concetti, metodologie, strumenti e suggestioni appartengano almeno in parte o in embrione alla mia esperienza didattica pregressa. Grazie alle conferme e stimoli ricevuti nel Percorso Abilitante Speciale, mi sento spronato a rendere sistematica l'applicazione di questi strumenti, utilizzandoli con continuità e accorta programmazione degli interventi.

Per ripartire con rinnovato impegno nella mia attività didattica, nel seguito rimarcherò gli stimoli disciplinari e pedagogici ricevuti durante il PAS. Dividerò queste considerazioni in tre sottoinsiemi:

- gli ingredienti disciplinari di base della didattica dell'informatica nella scuola secondaria:
  - la programmazione
  - le basi di dati
  - le reti
- i "sapori" migliorativi di tale ricetta
  - la teoria degli algoritmi e la computabilità
  - l'ingegneria del software
- l'approccio psicopedagogico e l'attenzione alla persona.

# Gli "ingredienti" di base

## Programmazione

Ho programmato per lavoro, per passione e per didattica in vari linguaggi, da Prolog a Visual Basic, da SQL a C, da C++ a Python. Programmare è appassionante ed educativo perché come afferma Donald Knuth, autore di *The Art of Computer Programming*, «se è vero che un problema non si capisce a fondo finché non lo si deve insegnare a qualcuno 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.» Ma appunto *programmare è un'arte* perché scrivere un programma funzionante non è difficile, ma progettarlo in modo intelligente (con attenzione alla struttura dei dati e alla semplicità algoritmica), scriverlo ottimizzato (in tempi di esecuzione e occupazione di memoria), robusto (con un'oculata gestione delle eccezioni), leggibile e mantenibile (ben strutturato, ben commentato) è difficile. Il compito di un educatore informatico è coltivare queste virtù "artistiche". E questo richiede come in tutte le arti esercizio che un docente non dovrebbe mai abbandonare esattamente come un musicista il suo strumento. Durante il PAS abbiamo avuto modo di avere brevi ma significativi assaggi di tale esercizio.

### Programmazione tra scienza e arte

Il docente d'informatica deve innanzitutto avere ben chiari alcuni concetti fondanti ed essere in grado di trasferirli ai discenti in modo semplice ma non ambiguo:

- concetto di algoritmo, di procedimento effettivo, di istanza di un problema;
- concetto di esecutore o macchina astratta;
- concetto di programma e di linguaggio di programmazione;
- concetto di paradigma di programmazione.

Se questi concetti nel bagaglio cognitivo ed esperienziale di un docente possono essere ben definiti, egli ancora deve trovare il percorso formativo adatto per proporli ai discenti e soprattutto per renderli concretamente manipolabili.

"Simple is better" e la semplicità deriva dalla buona comprensione di un concetto. Perseguendo questa semplicità in passato ho adoperato i fogli di calcolo, con la definizione di formule e il successivo arricchimento tramite macro, come primo passo nel territorio della programmazione. Da un paio d'anni l'innamoramento per Python, e la sua adozione come strumento didattico, mi ha portato ad anticipare già al primo anno della scuola superiore l'approccio alla programmazione vera e propria.

Il PAS grazie alla proposta degli AAVA e di LOGO, da cui sono derivate interessanti frequentazioni di siti dedicati a Scratch e a Blockly, mi ha aperto nuove e stimolanti prospettive verso un apprendimento sempre più attivo, laboratoriale, costruttivista. E questa piccola grande rivoluzione personale mi sta facendo desiderare opportunità di lavoro con gli alunni più giovani del primo biennio delle superiori, per potermi sfidare nella responsabilità di dar loro l'*imprinting* e l'impulso motivazionale sulla via di quest'arte.

### Paradigmi

Anche negli indirizzi specifici dell'informatica (indirizzi tecnici) la questione dei paradigmi di programmazione è spesso affrontata frettolosamente, come un argomento appena sussurrato, un po' come si fa per gli argomenti scabrosi della vita, magari a sostegno di un cambio di linguaggio («Ora siete diventati grandi, passiamo da C a C++ ragazzi!»). In questa sede vorrei sostenere la necessità di una coraggiosa presentazione e discussione del concetto di paradigma di programmazione nel percorso di formazione di uno studente d'informatica.

Una possibile definizione di tale concetto è la seguente: «I paradigmi di programmazione sono euristiche utilizzate per risolvere un problema algoritmico. Un paradigma di programmazione analizza un problema specifico attraverso una specifica lente, e sulla base di questa analisi, formula una soluzione per il problema dato, suddividendola progressivamente fino a blocchi specifici e definendo la relazione tra loro.» (Forlizzi, Il concetto di paradigma di programmazione)

Ripropongo ora il mio tentativo di definizione, svolgimento di un esercizio dato nel modulo di "Programmazione" del PAS.

«Descriverei un paradigma di programmazione attraverso una metafora: come nella comunicazione umana esiste l'aspetto linguistico (sintassi e semantica) della comunicazione che riguarda il "cosa" si comunica, vi è un aspetto comportamentale (atteggiamento) che riguarda il "come" si comunica. Considerando che nell'attività di programmazione il programmatore tramite il linguaggio comunica con l'esecutore (potremmo dire che s'immedesima nell'automa), così l'atteggiamento con cui si guarda la realtà da modellare e si definiscono le modalità risolutive costituiscono il paradigma (modello, archetipo) di programmazione. Viene definito anche come stile di programmazione, in quanto i paradigmi intervengono prima dell'attività di codifica stessa definendone modi e concetti.» (Gregori)

Spesso in modo errato si tende a definire un linguaggio tramite un paradigma. In effetti esistono linguaggi che possono essere utilizzati applicando paradigmi diversi e che offrono costrutti sintattici adatti a diversi paradigmi: ad esempio C++ e Python possono essere utilizzati sia in modalità procedurale sia in modo strettamente orientato agli oggetti.

Al contrario linguaggi come il Prolog o il LISP derivano rigorosamente dal modello dichiarativo e nascono per essere utilizzati rispettivamente per la programmazione logica il primo e per quella funzionale il secondo.

A conclusione vorrei sottoporre alcune riflessioni e proposte derivate dalle esperienze didattiche e stimolate dalla discussione in aula su questi temi.

- **Affrontare la presentazione dei vari paradigmi in fase preliminare del corso di programmazione.** Premesso che il concetto di paradigma di programmazione è un argomento di non facile trattazione dal punto di vista teorico e sicuramente più adatto ai percorsi didattici specifici quali gli istituti tecnici industriali, commerciali (SIA) e come approfondimento culturale nei licei per

le scienze applicate, proporrei di non relegarlo ad una fase troppo avanzata della programmazione (magari solo dopo aver consolidato il solito ed unico modello di programmazione strutturata) ma di cercare un modo semplice e diretto di presentazione in una fase introduttiva.

- **Far scoprire i paradigmi di programmazione come diversi modi di porsi nei confronti della realtà.** Nella definizione di paradigma di programmazione ho utilizzato la metafora dell'atteggiamento psicologico (cognitivo) nelle relazioni umane e seguendo questa indicazione si potrebbe impostare un modulo in cui presentare i diversi paradigmi di programmazione come "angoli prospettici con cui guardare la realtà da modellare". E ciò prima che uno di essi diventi preponderante (di solito l'"uno-e-trino" paradigma imperativo-strutturale-procedurale!) con la proposta spesso un po' pedante di un unico linguaggio di programmazione (per quanto importante come il Pascal o il C). Si potrebbe partire da un'attività di discussione in cui proporre di definire delle modalità descrittive e operative della realtà e mostrare con esempi semplici ad hoc come queste modalità corrispondono ai vari paradigmi.
- **Attività di sperimentale in laboratorio.** Quindi immediatamente, per non rendere troppo teorica la trattazione in questa fase iniziale, proporre dei semplici programmi in Python per illustrare l'approccio sia procedurale che ad oggetti (ad es. la soluzione di semplici problemi di geometria). A seguire una breve sperimentazione con il Prolog, basata su semplici clausole e regole di derivazione (semplici sillogismi). Questo modulo deve essere chiaramente ben preparato e ben programmato con attività stimolanti le capacità creative e intuitive dei discenti (active learning e costruzionismo), ma comunque con il rigore necessario a definire correttamente i percorsi di approfondimento successivi.

### Ricorsione

«La ricorsione in informatica è un metodo in cui la soluzione di un problema dipende dalla soluzione di istanze più piccole del problema stesso (al contrario di iterazione). L'approccio può essere applicato a molti tipi di problemi, e la ricorsione è una delle idee centrali dell'informatica.» (Wikipedia)

Sono profondamente affascinato da questo concetto (come dalla sua stretta parente, l'induzione), come folgorato sin dalla prima volta che lo incontrai nel mio percorso universitario. Il fascino della ricorsione risiede nella sua semplice immediatezza e nella sua potenza espressiva!

Per Chomsky ciò che contraddistingue il linguaggio dell'uomo sono l'infinità discreta e le proprietà ricorsive della grammatica, che consentono modi potenzialmente infiniti di espressione linguistica (e anche infiniti modi di espressione linguistica del pensiero) sfruttando un limitato numero di regole.

Quindi la ricorsività appartiene al funzionamento della nostra mente e consente di esprimere soluzioni a problemi anche complessi con sinteticità, espressività ed eleganza sorprendenti (si pensi alla soluzione ricorsiva del gioco delle Torri di Hanoi).

Inoltre si consideri che alcuni paradigmi di programmazione (logico e funzionale) dispongono esclusivamente della ricorsività per raggiungere la soluzione di un problema che richieda l'iterazione della stessa sequenza di operazioni.

Pur se sintetica, espressiva ed elegante la soluzione ricorsiva spesso pecca in efficienza per l'*overhead* introdotto dalle chiamate ricorsive.

Nella esperienza didattica normalmente la soluzione ricorsiva può risultare non di immediata comprensione e il docente dovrebbe riservare particolare attenzione a:

- individuazione della condizione di terminazione
- comprensione dei meccanismi di passaggio dei parametri e dello stack delle chiamate
- necessità di comprendere bene lo stack delle chiamate
- abilità nel trasformare una soluzione ricorsiva in iterativa e viceversa.

Quando e dove inserire il concetto di ricorsione nella didattica: suggerirei di non relegarlo al bagaglio cognitivo del secondo biennio degli indirizzi tecnici specifici (ITIS e ITC-SIA), ma di utilizzarlo già dal terzo anno dei Licei Scienze Applicate come strumento logico-matematico, consequenzialmente alla nozione

d'induzione.

## Basi di dati

Le basi di dati sono un argomento fondamentale delle Scienze dell'Informazione e dovrebbero essere introdotte sin dai primi anni dell'istruzione superiore di secondo grado, in ogni indirizzo. Ogni campo sociale, tecnologico, economico, ricreativo richiede la gestione di dati strutturati persistenti, modificabili e soprattutto efficientemente reperibili. E fortunatamente è un ambito che appartiene anche alla mia esperienza professionale (oltre che universitaria).

Generalmente introduco i concetti relativi con l'utilizzo di semplici esempi presi dall'esperienza quotidiana degli alunni (rubrica telefonica, registro scolastico, le "amicizie" di Facebook) ed esemplifico la necessità di modelli e tecnologie adatti alla gestione di ordini di grandezza elevati di dati partendo da semplici tabelle in fogli di calcolo, introducendo successivamente l'associazione tra informazioni di natura diversa e mostrando l'inefficienza e l'inaffidabilità di soluzioni tabellari monolitiche.

Fortunatamente la disponibilità di applicazioni (DBMS) di basso o nullo costo eseguibili su PC di un laboratorio scolastico (Microsoft Access, Microsoft SQL Server Express Edition, MySQL) consentono di esperimentare il processo di costruzione di semplici basi di dati. Ritengo di particolare importanza queste sperimentazioni per diversi motivi che cercherò di illustrare nel seguito.

- Innanzitutto la progettazione e l'implementazione di una base di dati "didattica", per quanto semplice quale la gestione della propria videoteca o della biblioteca scolastica ad esempio, implica un processo che ripercorre le fasi del ciclo di vita produttivo delle reali applicazioni informatiche e introduce tematiche proprie dell'ingegneria del software (di cui tratterò nel seguito).
- È un processo costruttivo completo, realistico e che conduce soprattutto a un prodotto finito, eseguibile, quindi di grande attrattività dal punto di mista motivazionale, se portato a termine.
- Utilizza strumenti potenti di rappresentazione concettuale e consente naturalmente di utilizzare didattica basata sull'*active* e cooperative learning, sessioni di brainstorming, con la realizzazione a valle di mappe concettuali della realtà da modellare (ad es. diagrammi ER, tabelle esemplificative delle relazioni, ecc.)
- Consente di sviluppare a raggiera altre tematiche importanti quali le applicazioni web, l'architettura client-server, la sicurezza dei dati, ecc., ma soprattutto di poterlo fare con un approccio laboratoriale, costruttivistico.
- Infine consente di trattare di argomenti fini delle scienze dell'informazione quali i concetti di dato e informazione, di dato astratto, di formalismo algebrico, di modelli e paradigmi, di ingegneria del software, sempre però potendo contare sulla possibilità di manipolare concretamente gli oggetti del discorso (si pensi ad esempio alla possibilità di trasformare immediatamente un'operazione di algebra relazionale nella corrispondente istruzione SQL e verificare operazionalmente la sua correttezza)

Il modulo del PAS dedicato a questa vasta tematica, stretto in una tempistica sicuramente insufficiente, ha comunque percorso a volo molti concetti fondamentali relativi alla realizzazione e gestione delle basi di dati relazionali, argomenti assai vasti e che preludono campi applicativi di grande interesse quale il *data mining*, il *data warehouse*, o gli sviluppi quali le kwnoledge base o le basi di dati *object-oriented*. Tutte tematiche su cui è stato destato il mio spirito di ricerca e che non mancherò di introdurre nel curricolo di indirizzi tecnici.

## Reti e Sistemi Informativi

A scuola abbiamo a che fare con *nativi digitali* (o *net generation*). Ecco come li descrive Bizzarri: «I nativi digitali sono abituati a cercare e trovare le informazioni velocemente nella rete (ad esempio con Wikipedia), amano i processi paralleli e il multitasking (ad esempio può accadere che allo stesso momento mandino un sms, guardino lo schermo del pc e ti rispondano coerentemente se poni loro delle domande), preferiscono la grafica al testo, alla lettura sequenziale di un libro preferiscono l'accesso random (tramite link come negli ipertesti), sono sempre collegati alla rete e con gli amici, dormono con il telefono acceso, aspirano alla gratificazione istantanea e a riconoscimenti frequenti.» (Bizzarri)

Io li ho definisco criticamente avidi *consumatori digitali*, e anche *consumatori dipendenti* ma con scarsa o nulla conoscenza della Rete. A oltre quarant'anni dalla sua nascita possiamo affermare che Internet (e in particolare il Web) è diventata mezzo insostituibile di relazione col mondo. Ritengo quindi che compito inalienabile di un educatore sia non solo di fornire conoscenze scientifico-tecnologiche, ma di orientarne l'uso a quella che viene definita "saggezza digitale" ("Digital Wisdom") da Marc Prensky in un articolo del 2009 [Prensky, 2009] proprio a proposito dei *nativi digitali*: «possiamo definire saggezza la capacità di trovare soluzioni a complessi problemi umani che siano pratiche, creative, appropriate al contesto e emozionalmente soddisfacenti. Sembra quindi una complessa attività di *problem solving*, ovvero l'essenza del *computational thinking*. Come la tecnologia diviene più sofisticata, sviluppare la capacità di aiutarci a prendere decisioni etiche e pragmatiche, ciò che chiamiamo "saggezza umana" raggiungerà nuovi livelli.» (Bizzarri)

Ritengo quindi che superando il limitato ruolo di formatori professionali, come educatori dovremmo diventare esperti nel curare quegli aspetti della teoria delle reti che consentano ai nostri discenti questa "saggezza digitale".

Due argomenti hanno particolarmente colpito la mia attenzione nello svolgimento del modulo di Reti e Sistemi Informativi: la **sicurezza** e la **virtualizzazione**. Quest'ultima sta acquistando celermente centralità nel processo di dematerializzazione delle risorse informative personali e d'impresa (*Storage Area Network* e *Cloud Computing*)

In particolare vorrei soffermarmi sul tema della sicurezza. Nell'ambito del modulo dedicato alle reti ho avuto modo di approfondire alcuni aspetti inerenti alla sicurezza: **crittografia**, **firma digitale** e **posta elettronica certificata**. Al di là delle conoscenze tecnologiche ritengo che l'argomento della sicurezza e della riservatezza delle informazioni dovrebbe trovare ampio spazio nella trattazione scolastica, oltre che per le prospettive occupazionali che può offrire, anche per le implicazioni etiche, giuridiche, di responsabilità sociale, politiche, culturali e, perché no, ludiche!

I nostri studenti dovrebbero essere da noi guidati sulla strada di quella "saggezza digitale" a cui si accennava più sopra, proprio partendo dai loro interessi e difficoltà verso un mondo digitalmente connesso, che ossessivamente utilizzano, ma di cui spesso conoscono gli aspetti più appariscenti, ignorandone i meccanismi, i rischi e le opportunità.

Per ciò che concerne la didattica mi permetto di fare una provocazione: i nostri laboratori d'informatica soffrono di pesanti carenze, sia per quanto riguarda l'aggiornamento e manutenzione delle risorse, sia per la disponibilità di connessioni sufficienti e affidabili. In entrambi i casi il docente deve essere in grado di sopperire con la sua competenza almeno alle deficienze più gravi, anzi quando possibile dovrebbe sfruttare i disservizi in situazioni didattiche, per insegnare a individuare i problemi, diagnosticare le cause ed eventualmente risolvere le situazioni bloccanti insieme agli alunni. Spesso le problematiche riguardano proprio la configurazione di rete: bisogna imparare a trasformare i vizi in virtù!.

Infine consiglio vivamente di sfruttare tecniche di *role play* per spiegare concetti di comunicazione in rete: i protocolli digitali, il dialogo client-server, la propagazione di messaggi si offrono naturalmente per l'attuazione di queste tecniche di apprendimento ludico.

## Affrontare la complessità

L'informatica è una scienza complessa, i suoi prodotti possono esserlo altrettanto e per motivi diversi. Esiste una difficoltà intrinseca, computazionale, e studiarla è compito di quella disciplina chiamata Teoria degli Algoritmi e della Computabilità. Per quanto concerne la complessità dei suoi prodotti tecnologici la complessità viene affrontata dalla prospettiva dell'ingegneria dei sistemi, con la sua declinazione informatica, l'ingegneria del software. Affrontare questi argomenti nella scuola secondaria comporta sicuramente una sfida per il docente d'informatica. Una sfida che dovrebbe essere affrontata non solo negli indirizzi tecnologici, ma anche là dove si coltivano gli scienziati e i tecnici del futuro, cioè nel liceo delle scienze applicate. Questa consapevolezza dovrebbe essere bagaglio di ogni figura professionale, come abilità valutativa di strumenti del proprio lavoro.

## Teoria degli algoritmi e computabilità

Materia difficile da cui mi sono proposto di distillare alcuni argomenti di lavoro, sondando anche proposte didattiche esistenti per un graduale avvicinamento fino alle problematiche più astratte.

- Comprendere la natura dei problemi e il concetto di computabilità:

  $$|\{\text{Algoritmi}\}| < |\{\text{Problemi}\}|$$

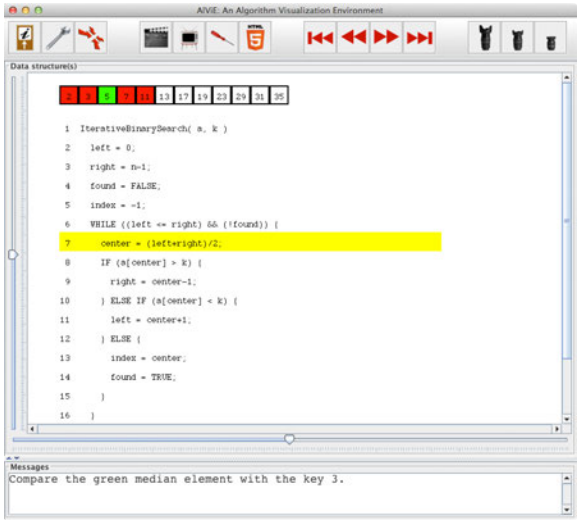
  cioè dimostrare che esistono molti più problemi che soluzioni e che quindi devono esistere problemi per cui non esiste un algoritmo di calcolo, cioè problemi non calcolabili.

- Definire il concetto di algoritmo e di modello computazionale: Macchina di Turing e Random Access Machine, conseguentemente proporre la tesi di Church-Turing.
- Di fronte alle soluzioni algoritmiche di un problema computazionale, e ai relativi programmi, porsi e provare a rispondere a queste domande:
  - Come possiamo quantificare la "bontà" di un algoritmo?
  - Quanto è performante ogni algoritmo risolutivo del problema?
  - Quante risorse impegna?
  - Possiamo migliorare le soluzioni adottate? E fin dove possiamo spingere questo miglioramento?
- L'ultima domanda introduce il "problema dei problemi": "Quanto è difficile il problema, ovvero, qual è la complessità temporale/spaziale del miglior algoritmo risolutivo che posso sperare di progettare?" (Proietti, 2014)
- Misurare e confrontare il costo computazionale delle soluzioni algoritmiche: misura di costo uniforme e logaritmica.
- Valutare la complessità intrinseca dei problemi: la notazione asintotica O, Ω e Θ
- Affrontare il concetto di struttura dei dati al fine di adottare l'opzione più efficiente.
- Introdurre la classificazione dei problemi P, ExpTime e NP ed NP-completi con esempi (SAT, il problema dell'arresto in al più K passi, del commesso viaggiatore, ecc.) e le inclusioni e congetture. Soprattutto la sfida in questo campo irto di formalismi non proprio immediati è di portare questi argomenti nel loro territorio esperienziale operando un accorto scaffolding!

È mia convinzione che debba cercare di proporre percorsi tematici su ristretti e coerenti sottoinsiemi dei temi in precedenza elencati, purché adeguati a età e indirizzo dei discenti e affrontati con approccio costruttivo e cooperativo. Condizione necessaria è che il docente affronti questa sfida con il quadro teorico generale ben chiaro!

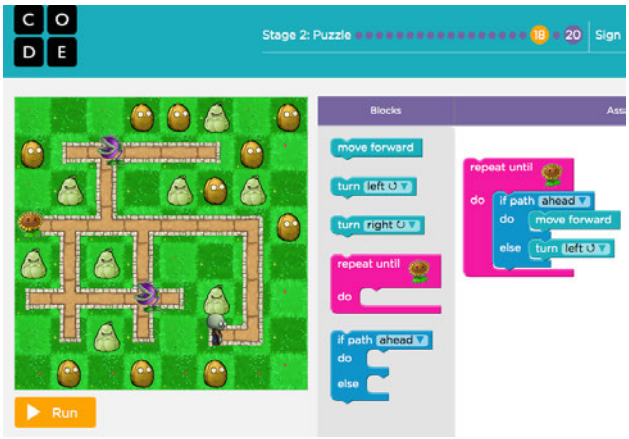
Alcuni esempi di obiettivi semplici su cui lavorare:

- calcolo della successione di Fibonacci con diverse soluzioni algoritmiche;
- confronto tra ricerca sequenziale e binaria (il divario tra i costi computazionali dei due algoritmi è un esempio che già ora propongo per indurre un positivo e stimolante stupore verso il pensiero computazionale!);
- soluzioni varie del problema dell'ordinamento: selectionSort, insertSort e mergeSort.




Ritengo che esista uno spazio prossimale sfruttabile in questo vasto e difficile territorio anche per i più giovani, per stimolare in loro il pensiero computazionale attraverso il gioco e la programmazione visuale. Una scoperta interessante fatta durante il modulo di TAC è stata proprio quella di strumenti visuali per la didattica sul pensiero computazionale.

ALVIE (http://www.algoritmica.org/software) è un ambiente di visualizzazione di algoritmi sviluppato in Java. La distribuzione qui disponibile include la




visualizzazione di circa settanta algoritmi e strutture di dati. ALVIE include anche un editore grafico di strutture di dati per creare ulteriori input per l'esecuzione e la visualizzazione degli algoritmi.

Per i più giovani (ma non solo) vorrei citare il portale di e-learning code.org (http://learn.code.org), creato per insegnare in modo divertente e intelligente a programmare e risolvere problemi in modo algoritmico utilizzando linguaggi di programmazione visuale e animazioni.

Tra gli obiettivi disciplinari e transdisciplinari perseguibili infine ne proporrei alcuni che mi stanno particolarmente a cuore:

- Insegnare il pensiero computazionale indipendentemente dai linguaggi; è un argomento da affrontare perché tra i discenti spesso i piani si confondono e si attribuisce erroneamente ai linguaggi potenza computazionale. Insegnare che i criteri di scelta sono altri!
- Sfruttare le connessioni tra informatica e matematica (concetto di infinito, di limite, di funzione) per potenziare le conoscenze in entrambe le discipline.

## Software Engineering

Un sistema software può avere complessità di ordine molto elevato. Senza arrivare alle dimensioni dei sistemi informativi di grandi aziende o istituzioni, o di sistemi di controllo industriali sofisticati, la “semplice complessità” di un’applicazione con interfaccia grafica adattiva a dispositivi diversi (PC, tablet, smartphone), con requisiti di mobilità, architettura distribuita o semplice interfaccia verso risorse di cloud computing, richiede la messa in gioco di discipline, teorie e competenze diversificate e di alto livello.

I concetti, le nozioni e suggestioni presentati dal modulo di Progetto e Sviluppo di Applicazioni di Rete (PSAR) sono stati veramente copiosi, forse troppi per essere apprezzati meritevolmente. La responsabilità non è certo del docente, ma del tempo tiranno e della complessità e varietà delle discipline in gioco.

Prima fra tutte l’Ingegneria del Software, su cui appunterò alcune note sugli argomenti che hanno particolarmente suscitato il mio interesse e che mi riprometto di inserire nella mia didattica:

- Approccio ingegneristico alla produzione del software: tra scienza, arte, artigianato e industria. Questo è un ottimo argomento culturale che può essere inserito in qualsiasi secondo biennio dei nostri indirizzi di attività, con classi che abbiano già sperimentato la programmazione.
- Il processo software e sue fasi: analisi e specifica dei requisiti, sviluppo, validazione ed evoluzione.
- Pattern di architettura.
- Concetto di riuso e la progettazione per componenti.
- Modelli di sistemi, ULM e utilizzo dei diagrammi per la modellazione delle diverse prospettive con cui si analizza e descrive il sistema (*Activity* diagram, *Use case* diagram, *Sequence* diagram, *State* diagram).
- Utilizzare i diagrammi del punto precedente per sviluppare abilità di rappresentazione concettuale.

Non svilupperò oltre queste note perché ognuna di esse richiederebbe approfondimenti specifici, ma tra gli stimoli ricevuti dal corso di PSAR, in particolare vorrei riuscire a concretizzare nella mia didattica gli ultimi due dell’elenco precedente. ULM è un insieme di formalismi grafici utilizzati per descrivere specifiche, progetto, funzionalità, processi di business, documentazione, ecc. dei sistemi software, che vorrei tentare di proporre come strumenti descrittivi delle idee degli studenti si fanno di sistemi informatici più o meno complessi (ad es. un motore di ricerca, un social network o anche semplicemente il gestionale della loro scuola). Quindi utilizzare i formalismi grafici di ULM per rappresentare mappe concettuali di casi di studio. La domanda a cui vorrei provare a dare una risposta sperimentale: potrebbe

essere inserito anche nel primo biennio?
Per il secondo biennio (indirizzi tecnici e LSA) proverei a proporre anche tool specifici come *Visual Paradigm*, potente strumento di progettazione visuale gratuito nella sua versione *Community Edition* (http://www.visual-paradigm.com)

Infine al termine del quarto e nel quinto anno degli istituti tecnici e licei scienze applicate si potrebbero proporre progetti per lo sviluppo di semplici applicazioni web comprendenti:

- sviluppo pagine e form HTML,
- competenze sistemistiche (LAMP),
- competenze di programmazione (PHP, Javascript),

utilizzando il metodo di apprendimento per progetto. È un'esperienza che ho fatto realmente presso l'Istituto Tecnico Industriale di L'Aquila nel 2011, con una quinta classe con cui abbiamo realizzato un piccolo social network cittadino (progetto AQLive).

# Curare le persone, cambiare la società

*«Chi lavora con le sue mani è un lavoratore. Chi lavora con le sue mani e la sua testa è un artigiano. Chi lavora con le sue mani e la sua testa ed il suo cuore è un artista.» Francesco d'Assisi*

Desidero dedicare la parte conclusiva di questa tesi più che a un'elencazione dei contenuti delle discipline pedagogiche, a riflessioni personali su ciò che questi fondamentali contributi hanno determinato nella mio percorso evolutivo di docente. Ritengo che questa sia il miglior modo per esprimere gratitudine nei confronti di chi fa ricerca e formazione in questo campo e soprattutto verso le nostre docenti. I prossimi titoli evocano proprio gli elementi di crescita che ho avvertito e su cui intendo lavorare da ora in poi.

## Empatia

Assumere il ruolo di discente nel PAS e affrontarne le difficoltà logistiche, degli orari estenuanti a volte in continuità con gli impegni scolastici e le incombenze di fine anno scolastico, nonché quel senso di inadeguatezza rispetto a problematiche disciplinari ormai dimenticate, mi ha permesso di rivivere quel processo che viene definito nella saggezza orientale di "trasformazione del veleno in medicina", cioè utilizzare gli aspetti critici (a volte drammatici) della vita come stimoli ad un cambiamento positivo.

Ma soprattutto mi ha consentito di specchiarmi nella realtà della scuola dalla prospettiva degli alunni, di sperimentare quale distanza possano creare simili elementi di disagio, quante incomprensioni, quanta poca chiarezza possano seminare e soprattutto, fatto negativo preoccupante, è che rischino di essere sopportati solo o maggiormente in virtù di motivazioni esterne ( "devo affrontare tutto questo per rimanere in graduatoria!").

All'inizio del PAS mi sentivo spesso demotivato, schiacciato dalla stanchezza e da un'imposizione di ritmi e regole di cui non comprendevo le ragioni. Mi sono chiesto spesso in questo periodo se i miei alunni vivano le stesse sensazioni rispetto all'ambiente scolastico.

Molte delle attività laboratoriali svolte nelle ore di pedagogia, mi hanno indotto a una profonda autoanalisi dei miei atteggiamenti in classe, soprattutto riguardo alla mia capacità di ascolto, di desiderio reale di comprendere anche i comportamenti apparentemente più ostili degli alunni.

Anche le metodologie proposte mi facevano scaturire le domande interiori: "faccio questo?", "sto attento a quest'altro? "

## Autoanalisi

Molti dei concetti, dei metodi e degli strumenti pedagogici proposti già mi appartenevano in sostanza, grazie al mio percorso di emancipazione personale e alle mie esperienze didattiche formali e non formali

(scoutismo, animazione educativa con ARCI-Ragazzi, servizio civile presso l'Unione Italiana Ciechi di Pisa, esperienze educative/sportive di volontariato con disabili, educazione ambientale, responsabilità in ambito buddista). In particolare quelli indirizzati a:

- includere, integrare, (io preferisco "abbracciare") tutte le peculiarità individuali e sociali;
- coltivare/utilizzare le relazioni positive (costruttive) e trasformare quelle negative attraverso il dialogo, la cooperazione e il sostegno reciproco;
- coltivare l'autostima;
- combattere l'appiattimento e il conformismo;
- ricercare soprattutto le motivazioni interne, e in senso ampio, la propria strada forti dell'assunto che ogni caratteristica, se usata in senso evolutivo (anche quella meno "gradita") è funzionale allo sviluppo personale e collettivo dell'umanità ("la mia vita con tutte le sue caratteristiche ed esperienze ha sempre un senso nell'economia universale!")

Tutto ciò però come docente e adulto responsabile di giovani in evoluzione richiede

- di essere maturato a livello metacognitivo,
- potenziato con lo studio e la ricerca,
- reso sistema di lavoro, programmato.

## Scaffolding

Ho rimesso al centro il ruolo di sostegno che devo assumere nello sviluppo cognitivo dei ragazzi: individuare le competenze attuali, le potenzialità da sviluppare e la "zona di sviluppo prossimale".

Inoltre timori e stress dell'alunno devono essere sempre affrontati e trasformati; il docente dovrebbe essere sempre disponibile a sostenere questa lotta nell'alunno, standogli a fianco e ascoltandolo prima di tutto (la "compassione " come miglior sostegno).

## Obiettivi primari: felicità e creazione di valore

Secondo il pedagogo giapponese e leader buddista Tsunesaburo Makiguchi, che da oltre trent'anni ho eletto mio maestro, **felicità** e **creazione di valore** sono fini e mezzi dell'educazione. Per sostenerlo arrivò a testimoniare la sua determinazione verso questi nobili obiettivi pagando con la vita il suo impegno a favore dell'educazione e della pace: imprigionato per essersi opposto alle politiche del regime militarista giapponese, morì in carcere di stenti all'età di settantatre anni. Recentemente le sue teorie pedagogiche sono diventate oggetto di crescente interesse internazionale. Ecco come Makiguchi definisce queste due categorie.

### Felicità e creazione di valore

La felicità individuale è un processo di tutta una vita, non fine a se stesso ma teso al miglioramento e alla trasformazione dei limiti e dei punti morti della persona e della società.

«L'idea di felicità [in Makiguchi] non è in alcun modo egoistica. Il processo di creazione di valore non è mai un processo individuale compiuto da un singolo avulso dal suo contesto sociale. Anzi nella creazione di valore, e nella felicità che è un processo parallelo a essa, vi è implicita una forte responsabilità etica e sociale (Gius & Zamperini, 1995¸ Gius & Coin, 1999). Scrive Makiguchi: "Dove si trova, allora, la nostra felicità? Essa proviene unicamente dalla condivisione di gioie e dolori con gli altri e con la collettività intera. È essenziale, dunque, che ogni vera idea di felicità racchiuda un serio impegno a partecipare alla vita sociale.» (Tarozzi, 2002)

La felicità per Makiguchi è quindi la competenza esistenziale a risolvere efficacemente i problemi della vita e a utilizzarli per creare valore per se stessi e per gli altri.

Ecco cosa afferma in proposito il ricercatore pedagogista Massimiliano Tarozzi: «Felicità. La parola stessa fa paura se applicata all'educazione e ai processi di istruzione e apprendimento. Più facile è parlare di "obiettivi cognitivi", di "competenze professionali" o anche di "pieno sviluppo della personalità", perché paiono mete più prossime o talmente astratte e generiche da risultare sicuramente condivisibili. Invece

la felicità spaventa. Perché da una parte appare un obiettivo troppo grande, irraggiungibile se non addirittura illusorio, dall'altra mette in crisi la sua solare semplicità e concretezza. Non è forse questa la finalità di ogni esistenza, indipendentemente dal senso che ognuno attribuisce a questa parola? Perché mai allora i processi di formazione dell'individuo, formalizzati come la scuola o informali come la famiglia, il territorio, il gruppo dei pari eccetera dovrebbero ignorare questa finalità ultima dell'azione umana? Eppure nella scuola italiana appare inattuale ogni richiamo alla felicità come finalità fondamentale dei processi educativi.» (Tarozzi, 2002)

A volte dimentico questo imperativo categorico. Il PAS, obbligandomi a reagire a mie insoddisfazioni e difficoltà, mi ha stimolato spesso a rimettere al centro questi pilastri del mio sviluppo personale e delle mie relazioni con gli altri.

Mi sono posto questa domanda "Riesco a perseguire questi obiettivi sempre? O almeno mi ridetermino ogni volta che non ci riesco?" Se non lo facessi come potrei sostenere i miei alunni

- nel credere in queste finalità della vita,
- nel perseguirle,
- nel valutare costruttivamente successi e insuccessi?

### Informatica, la Cenerentola della scuola.

Spesso rinchiusa nello scantinato ideale dei laboratori, a lustrare le pentole (ECDL!) l'informatica, oltre a essere scienza foriera di potenti concetti e modelli di conoscenza, può realmente offrire strumenti didattici che vanno oltre la mera applicazione strumentale, perché:

- è il campo predefinito per intervenire sui fattori informali (socioculturali) dell'educazione dei *nativi digitali*;
- permetterebbe all'alunno di riappropriarsi della sua centralità, consentendogli di costruire strumenti di conoscenza, di lavoro, di gioco;
- contiene già "misure compensative", il passo successivo è farle costruire all'alunno!

### Cosa farò (i buoni propositi!)

Sarò sicuramente più esigente verso me stesso nella programmazione curricolare e più ricettivo dei bisogni e delle richieste implicite degli alunni.

- ✓ Dedicare più tempo a valutare le condizioni iniziali del percorso didattico.
- ✓ Rimodulare più spesso gli obiettivi.
- ✓ Personalizzare il percorso didattico con più attenzione ad ogni singolo alunno.
- ✓ Per ogni unità didattica mi ripropongo di utilizzare questa *check list*:
  - introdurre l'argomento con una *mappa contestuale*;
  - far seguire discussione e *brainstorming* su argomento e sulle problematiche relative (*apprendimento cooperativo*);
  - costruire insieme *mappe concettuali*;
  - potenziare l'attività laboratoriale, nel senso di curare di più la valutazione delle abilità e competenze sviluppate dai singoli alunni.
- ✓ Essere più esigente nei confronti della scuola in cui esercito sulla disponibilità e qualità degli strumenti di laboratorio offerti
- ✓ Curare il materiale didattico da me sviluppato e prodotto tenendo conto di tutti i bisogni educativi del gruppo classe.

## Concretamente (conclusioni)

Come ho annunciato nell'introduzione tra i miei principali obiettivi in questo PAS vi era quello di "imparare qualcosa su come s'insegna" e nonostante mi aspettassi di lavorare più sul pratico, non ritorno a casa con le mani vuote.

Tra le cose che mi sono appuntato di realizzare concretamente e che desidero potenziare nella mia azione didattica, vi sono alcuni aspetti di metodo (programmazione, verifiche formative, cooperative learning), un atteggiamento su cui prioritariamente lavorerò (l'ascolto attivo) e un'attenzione per Bisogni Educativi Speciali e in particolare per i Disturbi Specifici dell'Apprendimento.

All'inizio di quest'anno scolastico, nel prendere servizio nelle scuole in cui sono stato destinato (tre classi del Liceo Scienze Applicate, una prima di un Istituto Professionale per l'Industria e l'Artigianato e quattro classi del primo biennio di un Istituto Tecnico Commerciale), mi è stato chiesto di redigere *una programmazione per curricoli verticali*, tenendo conto dei collegamenti tra assi culturali e strutturandolo per conoscenze, abilità e competenze. Tutto ciò mi ha disorientato perché da "precario storico" mi ero abituato negli anni a lavorare localmente sugli obiettivi di classe e per anno scolastico. La lettura di "Progettare, programmare e valutare l'attività formativa" di Emilio Lastrucci, testo adottato dalla professoressa Cavalieri per le lezioni di "Teorie e metodi di progettazione, programmazione e valutazione scolastica", mi ha chiarito i termini e indicato gli aspetti da migliorare.

- Dedicare tempo e attenzione all'importantissima valutazione *ex-ante*: analisi del contesto, delle risorse e valutazione diagnostica dei destinatari. Oltre alla raccolta d'informazioni necessarie alla programmazione personalizzata e individualizzata, curare questa fase predispone a un atteggiamento empatico verso l'alunno-persona che permane per tutto l'arco dell'anno scolastico.
- Nella programmazione curricolare curare meglio la tempistica degli interventi e i criteri e gli strumenti di valutazione formativa ricorrente, per poter velocemente rimodulare la programmazione unità per unità.

Riguardo alla valutazione *ex-ante*, per quanto riguarda la conoscenza dei bisogni personali degli alunni, avrò particolare cura dei soggetti con bisogni educativi speciali (BES). Inoltre predisporrò semplici prove di lettura e test preventivi per mettere in luce eventuali Disturbi Specifici dell'Apprendimento non ancora diagnosticati. Per questi ultimi sono state illuminanti le lezioni delle professoresse De Angelis e Prosperi, sia nel fornirci le conoscenze di base indispensabili per comprendere i DSA, sia nell'incoraggiarci ad attuare tutti i metodi dispensativi e compensativi necessari. In particolare come docente d'informatica sono consapevole di giocare un ruolo fondamentale all'interno della scuola, poiché l'informatica offre un ampio bagaglio di competenze e strumenti per una didattica efficace: attiva, multimediale, predisposta alla rappresentazione grafica dei concetti, all'elaborazione progettuale e interdisciplinare dei contenuti.

## Ringraziamenti

*Ai miei genitori Carla e Rolando,*

*al mio maestro Daisaku Ikeda,*

*a tutti i Docenti del corso*

*con sincera gratitudine*

# Bibliografia e sitografia

Beazley, David. *Pensare da informatico.* http://www.python.it/doc/Howtothink/Howtothink-html-it/introduzione.htm.

Bizzarri, Giuseppe. *Discenti, Astrazione e Competenze in Informatica.*

Demetrescu, Finocchi, e Italiano. *Algoritmi e strutture dati.* seconda edizione. McGraw-Hill, 2008.

Forlizzi, Luca. *Il concetto di paradigma di programmazione.* http://education.di.univaq.it/moodle2013/mod/page/view.php?id=354.

Forlizzi, Luca. *Metodi e Strumenti per la Didattica dell'Informatica .* http://education.di.univaq.it/moodle2013/course/view.php?id=7.

Forlizzi, Luca. «Teorie e Strumenti per la Didattica dell'Informatica.» *Doc. PAS 2014*, 2014.

Gregori, Mauro. «Paradigma di programmazione.» *Attività PAS 2014.*

Lastrucci, Emilio. *Progettare, programmare e valutare l'attività formativa.* ANICIA, 2006.

Proietti, Guido. *Corso di Teoria degli Algoritmi e Computabilità - PAS 2014.* 2014.

Tarozzi, Massimiliano. «Valore della felicità e felicità del valore.» *Buddismo e Società*, n. 90 (2002).

Wikipedia. *Active learning.* http://en.wikipedia.org/w/index.php?oldid=600567245.

Wikipedia. *Algoritmo ricorsivo.* http://it.wikipedia.org/wiki/Algoritmo_ricorsivo.

Wikipedia. *Apprendimento cooperativo.* http://it.wikipedia.org/wiki/Apprendimento_cooperativo.

Wikipedia. *Metafora.* http://it.wikipedia.org/wiki/Metafora.

Wikipedia. *Scaffolding.* http://it.wikipedia.org/wiki/Scaffolding.