Perche’ e’ importante pensarci

Impostazione tradizionale a partire dall’individuazione dei contenuti e delle competenze operative (conoscenze)

Diverso approccio a partire dall’individuazione dei processi messi in atto che ne caratterizzano identità e “prospettiva culturale”

l’informatica è solo un ventaglio di competenze tecniche e professionali o è anche un modo di guardare il mondo?

credits prof. Claudio Mirolo UNIUD

L’informatica è un ramo della matematica,

una disciplina ingegneristica

o una scienza della natura?

La conoscenza sul comportamento dei programmi

si sviluppa deduttivamente o empiricamente?

I programmi sono assimilabili a oggetti matematici,

a meri dati, oppure a processi mentali?

anima matematica

Rimanda al razionalismo in filosofia

La ragione pura (conoscenza a priori) è più affidabile

dell’esperienza sensoriale (conoscenza a posteriori)

Programmare è assimilabile a un’attività matematica

Esempi:

teoria della calcolabilità, complessità computazionale,

verifica formale della correttezza dei programmi,

semantica dei linguaggi di programmazione, . . .

anima matematica

Metodologia:

Il ragionamento deduttivo è l'unico metodo accettato

per investigare le proprietà dei programmi

Ontologia:

Il testo di un programma è un'espressione matematica

e in quanto tale rappresenta un oggetto matematico

Epistemologia:

Conoscenza certa, a priori, sui programmi deriva

da puro ragionamento, attraverso deduzioni formali

anima ingegneristica

Rimanda all’empirismo in filosofia

L’esperienza è alla radice di ogni conoscenza

Da un punto di vista ingegneristico l’informatica

mira a produrre sistemi affidabili e i metodi dell’informatica teorica sono considerati speculativi

È impraticabile, se non impossibile, acquisire (dedurre) conoscenza a priori sui programmi reali

Esempi:

ingegneria del software (requisiti, progetto, design patterns,

architettura, manutenzione ed evoluzione, testing, . . .)

anima ingegneristica

Metodologia:

La qualità dei programmi è determinata su base statistica

dalle caratteristiche del processo di sviluppo e testing

Ontologia:

Il testo di un programma è un mero aggregato di dati e non

è necessario ipotizzare l’esistenza di un’entità immateriale

ad esso correlata (principio di parsimonia ontologica)

Epistemologia:

conoscenza a posteriori sull’affidabilità dei programmi

deriva da misure probabilistiche tramite batterie di test

anima scientifica

Metodologia:

Il comportamento potenzialmente caotico dei programmi
viene indagato applicando il metodo ipotetico-deduttivo

Ontologia:

Il testo di un programma è assimilabile a entità biologiche
(DNA, reti di neuroni) e i processi a cui dà luogo, contingenti
al supporto fisico in cui si manifestano, a processi mentali

Epistemologia:

La conoscenza è il risultato di deduzioni a priori
e osservazioni a posteriori, ma deve comunque
passare al vaglio di esperimenti scientifici

Ricapitolando:

Prospettiva matematica (razionalismo filosofico),

programma = mezzo per condividere conoscenza procedurale

Prospettiva ingegneristica (empirismo filosofico),

programma = risultato di processi di progetto e sviluppo

Prospettiva scientifica (scienze),

programma = oggetto di studio sperimentale

Processi di indagine scientifica

“la didattica scientifica nella scuola sviluppa le competenze base di indagine, come saper osservare, inferire, prevedere, misurare e sperimentare.” (Bell et al. 2013)

- porre domande che si possono testare
- progettare e realizzare esperimenti (debugging!)
- analizzare e interpretare i dati raccolti

domande matematiche

- L’algoritmo riesce a ordinare qualsiasi sequenza di dati?

  a prescindere da situazione iniziale, eventuali ripetizioni?
- È indifferente usare questo o quell’algoritmo per ordinare

una sequenza lunga? o ce n’è uno di prestazioni migliori?

- Si potrebbero concepire altri algoritmi ancora più veloci?

oppure c’è un limite che non si può sperare di superare?

domande scientifiche

- Come si può misurare sperimentalmente tempi di calcolo?
- con quale attendibilità? quanto sono generalizzabili?
- I tempi rilevati sperimentalmente confermano i risultati teorici?

- quali aspetti sono generali e quali contingenti?
- Quick Sort è sistematicamente migliore di Insertion Sort ?

si osserva qualcosa di nuovo rispetto all’analisi teorica?

domande ingegneristiche

- È possibile migliorare le prestazioni (combinando assieme i “blocchi funzionali” QuickSort e InsertionSort)?
- Come si può assicurare l’affidabilità del sistema che ne risulta?
- come organizzare il processo di sviluppo?
- Quali sono le condizioni “ottimali” di integrazione delle due componenti? come regolarle su base empirica?