Il Giuoco delle Perle di Vetro

Claudio Mirolo

Dipartimento di Matematica e Informatica,
Università di Udine, via delle Scienze 206 – Udine
claudio.mirolo@uniud.it

INFOCULT
Gargnano (BS)
9-11 Ottobre 2011

Sommario

Il Giuoco delle Perle di Vetro

Nulla si sottrae tanto alla rappresentazione mediante la parola [. . .] quanto certe cose [. . .], le quali però appunto perché uomini pii e coscienziosi le trattano quasi fossero cose esistenti, si avvicinano un poco all'essere e alla possibilità di nascere.

[. . .] Sotto l'alterna egemonia di questa o di quell'altra scienza o arte, il Giuoco dei giuochi era diventato una specie di linguaggio universale col quale i giuocatori erano in grado di esprimere valori mediante simboli e di metterli in vicendevole rapporto.

Hermann Hesse, 1943, "Das Glasperlenspiel"

Spunti di riflessione

- Linguaggio universale per rappresentare la conoscenza scientifica e artistica
- Informatica come nuovo "giuoco delle perle di vetro" per interpretare e creare realtà?
- Oggetto della ricerca è anche la struttura della spiegazione, del modello, astratta dalla realtà
- Interpretazione della realtà vs. contemplazione (estetica) delle strutture del gioco
- Da *"Magister ludi"* a *"Ludi magister"*: giocare, creare, interpretare la realtà e apprendere confluiscono . . .

"Neve" per gli Inuit di Labrador e Groenlandia

1.	aput	neve in generale (in terra)
2.	apirlaat	neve fresca
3.	massak, akkilokipok	neve soffice
4.	mauja	neve soffice e profonda
5.	mangokpok, mangiggal, mangikaajaaq	neve pesante
6.	pukak	neve incrostata
7.	massalerauvok, qinuq	neve fradicia
8.	tipvigut, apusiniq	banco, mucchio di neve
9.	aputitaq	chiazza di neve
10.	putsinniq, puvvinniq	neve bagnata sopra il ghiaccio
11.	qaniit	neve in aria che cade
12.	nittaalaq	neve densa in aria
13.	imalik	neve bagnata che cade

Quali accezioni?

1. *Strumentale*
   Focalizzazione sui prodotti, mentre "le competenze [...] sono formulate in modo molto generale, [...] senza coerenza didattica e suddivisione rigorosa dei compiti" (Baudé, 2007).
2. *Tecnologica*
   Dimensioni tecnologica e sociale (Paoletti, 1993): schemi generali di interazione nell'uso di "artefatti software", invarianti cognitivi, . . . (Baudé, 2007; Vandeput, 2009).
3. *Disciplinare*
   Identità, concetti e metodologie proprie pertinenti la dimensione scientifica (Paoletti, 1993): principi generali per capire i processi di calcolo, problem-solving.

Schematizzando un po'

accezione strumentale	accezione tecnologica	accezione disciplinare
competenze contingenti	. . .	competenze stabili
saper fare	saper generalizzare	saper creare
abilità manipolatorie	intuizione analogica	ragionamento critico
accento sui prodotti	invarianti cognitivi	accento sui *processi*
addestramento	. . .	formazione

Contenuti vs. Processi

Didattica della disciplina . . .

- Impostazione tradizionale a partire dall'individuazione dei *contenuti* e delle competenze operative (conoscenze)
- Diverso approccio a partire dall'individuazione dei *processi* messi in atto (analisi, scoperta, progetto) 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?

Natura dell'informatica?

Per affrontare la didattica dell'informatica può essere utile riflettere sulla natura della disciplina e quindi sui *processi* (analisi, scoperta, progetto, . . .) che, più delle nozioni, stanno alla base dell'educazione.

Informatica nella scuola del riordino

DM 22/08/2007 n. 139
Asse scientifico-tecnologico

Competenze di base a conclusione dell'obbligo di istruzione:

- Osservare, descrivere ed analizzare fenomeni appartenenti alla realtà naturale e artificiale e riconoscere [. . .] i concetti di *sistema* e di *complessità*
- Essere consapevole delle *potenzialità* e dei *limiti* delle tecnologie nel contesto culturale e sociale in cui vengono applicate

Informatica nella scuola del riordino

DM 7/10/2010 n. 211
Liceo Scientifico – Opzione delle scienze applicate

Competenze di base a conclusione dell'obbligo di istruzione:

[. . .] Il collegamento con le *discipline scientifiche*, ma anche con la *filosofia* e l'italiano, deve permettere di riflettere sui fondamenti teorici dell'informatica e delle sue connessioni con la logica, sul modo in cui l'informatica influisce sui metodi delle scienze e delle tecnologie, e su come permette la nascita di nuove scienze.

A.M. Eden, 2007

- Monografia di Minds and Machines dedicata alla "filosofia dell'informatica"
- Aspetti analizzati:
  - Metodi applicati per acquisire nuove conoscenze
  - Natura degli oggetti studiati (*ontologia*)
  - Natura della conoscenza su di essi (*epistemologia*)

Domande

- 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?

Tre "anime" dell'informatica

All'interno della disciplina convivono tre concezioni principali:

- Anima matematica (paradigma razionalista), tipica dell'informatica teorica
- Anima ingegneristica (paradigma tecnocratico), tipica dell'ambito dell'ingegneria del software
- Anima scientifica (paradigma scientifico), tipica dell'ambito dell'intelligenza artificiale

L'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, ...

L'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

L'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, . . .)

L'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

L'anima scientifica

- In accordo con con il punto di vista *scientifico*
- La conoscenza *a priori* (deduttiva) deve essere corroborata/refutata da evidenza *empirica* (sperimentale)
- L'informatica è una disciplina scientifica e le proprietà dei programmi sono oggetto di indagine scientifica
- Esempi:
  debugging (!), intelligenza artificiale, reti neurali artificiali, modelli e simulazione, programmazione evolutiva, . . .

L'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

In sintesi...

anima	**metodo**
matematica	analisi teorica e deduzione (razionalismo)
ingegneristica	controllo del livello di qualità/affidabilità (empirismo)
scientifica	verifica sperimentale di ipotesi (scienze della natura)

In sintesi...

anima	**ontologia**
matematica	testo programma = espressione matematica programma = oggetto matematico
ingegneristica	testo programma = aggregato di dati programma (immateriale): non esiste
scientifica	testo programma assimilabile a DNA, . . . programma assimilabile a processo mentale

In sintesi...

anima	**epistemologia**
matematica	caratterizzazione formale ben definita conoscenza a priori (certezza)
ingegneristica	caratterizzazione formale impraticabile conoscenza a posteriori (affidabilità)
scientifica	caratterizzazione formale incompleta conoscenza a priori e a posteriori

T. Colburn & G. Shute, 2006

In European Conference on Computing and Philosophy

[Analogamente alla matematica,] l'informatica, per quanto riguarda l'ambito del software, si distingue dalle scienze empiriche in quanto i soli modelli che costruisce e studia sono astrazioni. [Tuttavia] si tratta solo di una somiglianza superficiale, e la natura sostanziale dell'astrazione in informatica è molto diversa da quella pertinente la matematica.

Le astrazioni primarie della matematica sono *schemi di deduzione*, mentre le astrazioni primarie dell'informatica sono *schemi di interazione*. Si tratta di una distinzione cruciale, che caratterizza le forme dell'astrazione tipiche delle due discipline.

Abstraction in Computer Science

Il successo delle scienze è in parte dovuto alla costruzione di modelli matematici [...] che eliminano dettagli inessenziali [*information neglef*]. L'informatica non può trattare l'informazione in questo modo. Deve usare l'astrazione per gestire la complessità [delle interazioni].

L'informatica si distingue dalla matematica nell'uso di un tipo di astrazione che gli informatici denominano *information hiding*. La complessità del comportamento dei moderni sistemi di calcolo renderebbe il compito del programmatore inattuabile senza strumenti di astrazione che nascondano, ma non trascurino, i dettagli essenziali a un livello più basso di elaborazione [...].

— anima matematica?

D. Goldin & P. Wegner, 2008

In Minds and Machines

Tradizionalmente l'elaborazione viene vista come una trasformazione a 'scatola chiusa' da dati di ingresso [...] a dati di uscita.

Secondo il punto di vista *interattivo*, l'elaborazione è un processo continuo piuttosto che una trasformazione funzionale dall'input all'output.

The Interactive Nature of Computing

Più specificamente, la comunicazione con il mondo esterno avviene durante l'elaborazione, non prima o dopo di essa. Questo approccio cambia radicalmente la nostra comprensione di cosa è la computazione e di come può essere modellata.

I modelli interattivi superano i sistemi formali 'razionalisti', che limitano l'espressività, per approdare a sistemi 'empirici' semi-formali, che sono più espressivi.

— anima ingegneristica?

G. Dodig-Crnkovic, 2002

In Computer Science in a Theory of Science Discourse

L'informatica è un campo nuovissimo e una fra le più giovani scienze. Di conseguenza, ha le tipiche caratteristiche moderne: è interdisciplinare ed ha rapporti stretti con la tecnologia.

Soffre la mancanza di una propria tradizione scientifica nello stesso tempo in cui è soggetta a uno sviluppo tremendamente dinamico.

Computer Science in a Theory of Science Discourse

Malgrado tutti gli aspetti che rendono la giovane disciplina informatica diversa dalle scienze di tradizione millenaria come la matematica e la logica, si può concludere che l'informatica comprende una massa critica di caratteristiche scientifiche tali da permetterci di qualificarla come scienza.

La differenza importante è che il computer [...] non è un oggetto di indagine [...] ma è piuttosto *teoria materializzata*, uno strumento capace di trasformarsi al fine di "ospitare" concetti teorici sempre più potenti.

— anima scientifica?

G.J. Sussman, 2004; Abelson et al., 1996

In Computer Science: Reflections on the Field

L'informatica non è una scienza, e il suo significato profondo ha poco a che vedere con i computer. La rivoluzione informatica è una rivoluzione nel modo di pensare e di esprimere quello che si pensa. L'essenza di questo cambiamento è l'emergere [di un'] *epistemologia procedurale* — lo studio della struttura della conoscenza da un punto di vista procedurale, in contrasto con il punto di vista più dichiarativo degli ambiti matematici classici.
La matematica tradizionale fornisce una cornice per trattare precisamente con la nozione di "cosa?".
La computazione fornisce una cornice per trattare precisamente con la nozione di "come?".

The Legacy of Computer Science

Un linguaggio di programmazione non è semplicemente un modo per fare eseguire delle operazioni a un computer, ma è piuttosto un nuovo strumento formale per esprimere idee sulla metodologia.

Perciò i programmi devono essere scritti affinché li possano leggere le persone, e solo incidentalmente per farli eseguire dalle macchine.

— quale anima?

Per fissare le idee... Algoritmi di ordinamento

- Problema abbastanza semplice, ma non del tutto artificioso (si pone nella realtà e gli allievi ne sperimentano esempi)
- Osservazioni e risultati non scontati
- Attività interessanti
  che mettono in luce le diverse anime dell'informatica

Insertion Sort

Insertion Sort

```
public void insertionSort( int[] seq ) {

  int n = seq.length;

  for ( int i=1; i<n; i++ ) {

    int  x = seq[i],  j = i - 1;

    while ( (j >= 0) && (seq[j] > x) ) {
      seq[j+1] = seq[j];  j = j - 1;
    }
    seq[j+1] = x;
  }
}
```


Quick Sort

Quick Sort

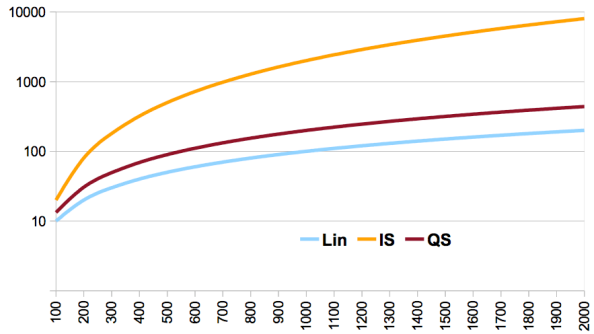
```
private void quickSort( int l, int u, int[] seq ) {

  if ( l < u ) {
    int  m = seq[l],  i = l,  j = u;
    do {
      while ( seq[i] < m ) {  i = i + 1;  }
      while ( seq[j] > m ) {  j = j - 1;  }
      if ( i < j ) {
        int x = seq[i];  seq[i] = seq[j];  seq[j] = x;
        i = i + 1;  j = j - 1;
      }
    } while ( i < j );
    if ( seq[j] > m ) {  j = j - 1;  }
    quickSort( l, j, seq );
    quickSort( j+1, u, seq );
  }
}
```

Approccio matematico: Domande

- 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?

Approccio matematico: Analisi



Approccio matematico: Suggestioni

Quanti confronti in media?

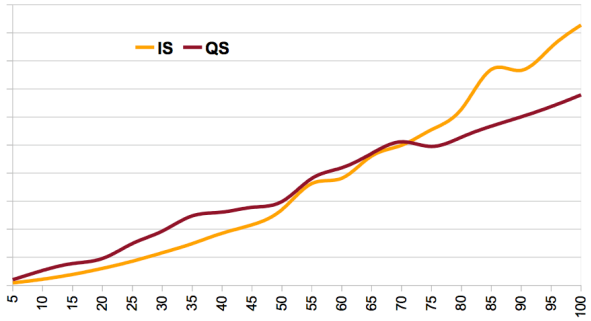
- *InsertionSort*: $T(n) \approx k \cdot n^2$
- *QuickSort*: $T(n) \approx k \cdot n \log n$

Quanti "rimescolamenti" diversi sono possibili?

Approccio scientifico: Domande

- 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?
- *QuickSort* è sistematicamente migliore di *InsertionSort*? si osserva qualcosa di nuovo rispetto all'analisi teorica?

Approccio scientifico: Osservazioni



Approccio scientifico: Suggestioni

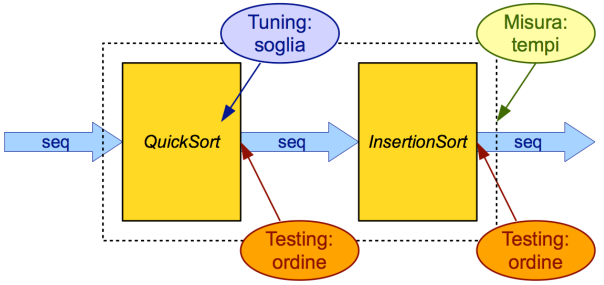
Problematiche della sperimentazione:
messa a punto di un adeguato "strumento di misura"

- *Ordine di grandezza* e unità di misura dei tempi di calcolo
- *Risoluzione*, ripetibilità, interferenza di fattori estranei
- Misura dei tempi *medi* di ordinamento, generazione dei campioni, criteri
- Compensazione di *errori sistematici* della misurazione

Approccio ingegneristico: Domande

- 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?

Approccio ingegneristico: Pianificazione



Approccio ingegneristico: Suggestioni

```
private void quickSort( int l, int u, int[] seq ) {
  if ( l + delta < u ) {  // soglia
    int  m = seq[l],  i = l,  j = u;
    do {
      while ( seq[i] < m ) {  i = i + 1;  }
      while ( seq[j] > m ) {  j = j - 1;  }
      if ( i < j ) {
        int x = seq[i];  seq[i] = seq[j];  seq[j] = x;
        i = i + 1;  j = j - 1;
      }
    } while ( i < j );
    if ( seq[j] > m ) {  j = j - 1;  }
    quickSort( l, j, seq );
    quickSort( j+1, u, seq );
  }
}
```


In conclusione . . .

Nessuna ricetta

È difficile, persino tra informatici,
intenderci sui concetti essenziali,
fondanti (Duchâteau, 1992)

Solo qualche spunto
per affrontare il problema (anche)
secondo prospettive meno scontate . . .

. . . e l'interesse a discuterne assieme

claudio.mirolo@uniud.it

The End

Grazie
per la vostra
pazienza

Annotazioni evocative

L'informatica è una ricerca incessante per "stanare" il significato dietro la forma e per costringere il significato nella "camicia di forza" della forma: nessuno dovrebbe uscire dalla scuola senza averlo percepito. (Duchâteau, 1992)

Il primo miracolo è che combinare un gran numero di volte un piccolo insieme di operazioni elementari consente una potenza d'azione considerevole. [. . .] Il secondo miracolo è che una grande varietà di insiemi di operazioni elementari (ragionevoli) conduce alla stessa potenza [di calcolo]. Il terzo miracolo è che i limiti di questa potenza sono esprimibili formalmente.
(Mazoyer, 2005)

Riferimenti

- A.H. Eden, 2007
  Three Paradigms of Computer Science
  Minds and Machines
- G.J. Sussman, 2004
  The Legacy of Computer Science
  Computer Science: Reflections on the Field
- T. Colburn & G. Shute, 2006
  Abstraction in Computer Science
  European Conference on Computing and Philosophy
- D. Goldin & P. Wegner, 2008
  The interactive nature of computing
  Minds and Machines

Riferimenti

- F. Paoletti, 1993
  Épistémologie et technologie de l'informatique
  Le bulletin de l'EPI
- G. Dodig-Crnkovic, 2002
  Computer Science in a Theory of Science Discourse
  Master Thesis in CS, Mälardalen University
- C. Duchâteau, 1992
  Peut-on définir une "culture informatique"?
  Journal de Réflexion sur l'Informatique
- J. Mazoyer, 2005
  L'enseignement de l'informatique ...
  Académie des sciences

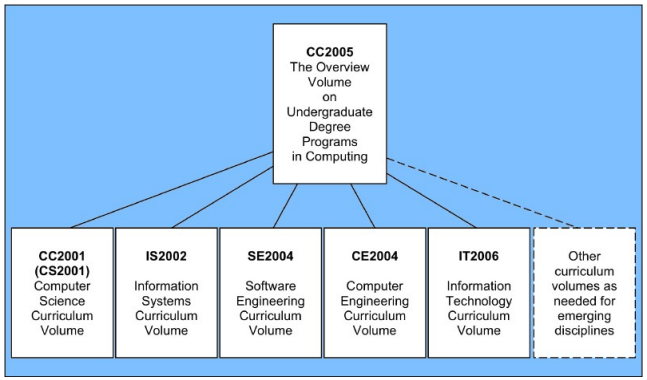
Riferimenti

- A. Tucker, Chair, 2003
  A Model Curriculum for K-12 Computer Science
  ACM & CSTA
- CC Joint Task Force, 2005
  Computing Curricula 2005: The Overview Report
  ACM & IEEE-CS
- E. Vandeput, 2009
  Milestones for teaching the spreadsheet program
  European Spreadsheet Risks Interest Group
- J. Baudé, 2007
  Le développement de l'informatique et des TIC dans l'enseignement — et si la voie suivie n'était pas la bonne?
  Association EPI

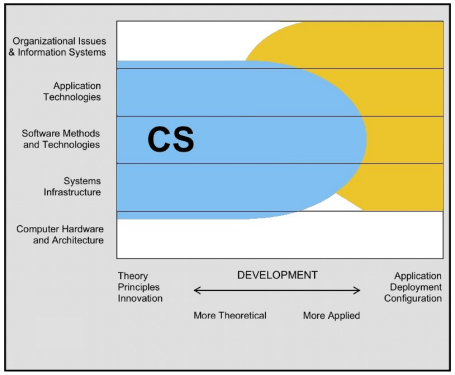
Modelli di riferimento

- Modelli curriculari ACM K-12 per la scuola
  (p. es.: A Model Curriculum for K-12 CS)
- Modelli curriculari ACM / IEEE-CS per l'università
  (p. es.: CS Curricula 1991; Computing Curricula 2001; CS Curricula 2008)
- Modelli specifici ACM / IEEE-CS per l'università
  (p. es.: Computing Curricula 2005)

ACM / IEEE-CS Computing Curricula 2005



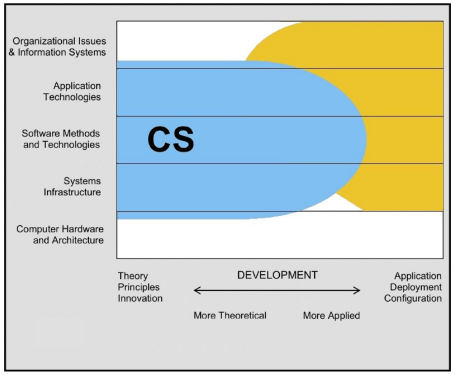
ACM / IEEE-CS Computing Curricula 2005



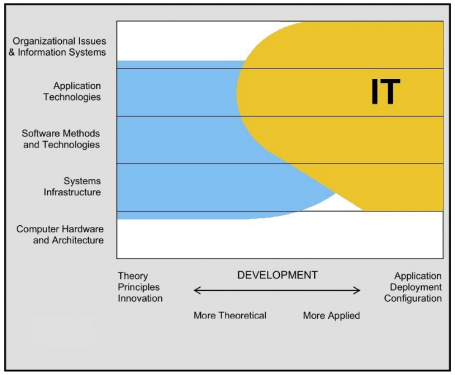
ACM / IEEE-CS Computing Curricula 2005

- Asse verticale: Identificazione di contenuti
- Asse orizzontale: . . . ?

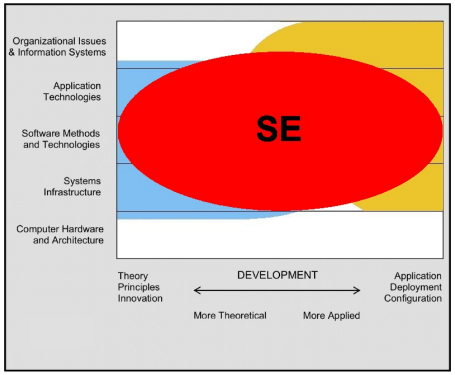
ACM / IEEE-CS Computing Curricula 2005



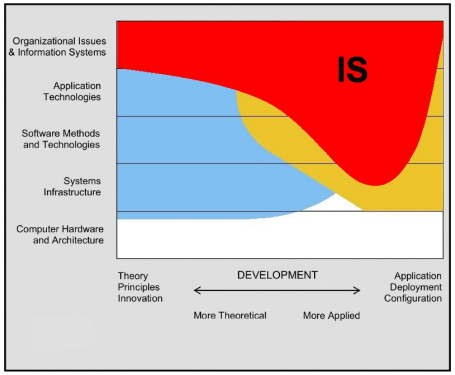
ACM / IEEE-CS Computing Curricula 2005



ACM / IEEE-CS Computing Curricula 2005



ACM / IEEE-CS Computing Curricula 2005



ACM / IEEE-CS Computing Curricula 2005