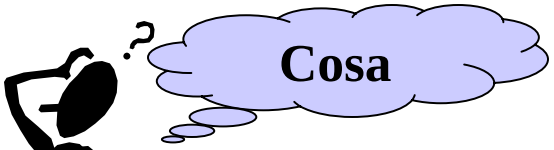
Introduzione

“A computer will do what you tell it to do, but that may be much different from what you had in mind”

Joseph Weizenbaum

Time

Dal “cosa” al “come” …



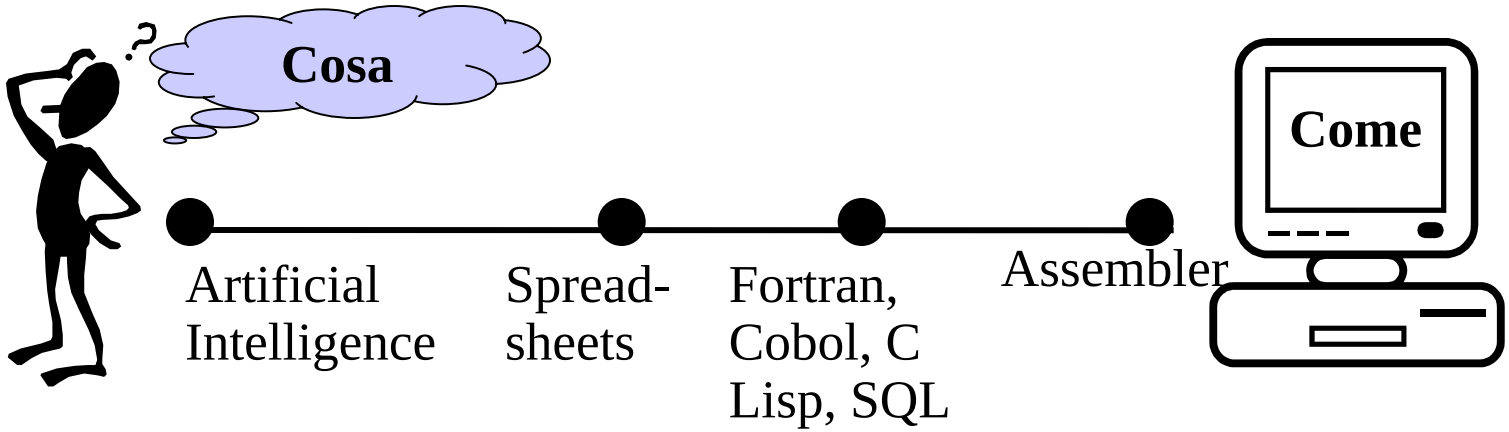
- Problem domain
- Knowledge
- Human language
- Goals, needs, desires

«La scienza del software studia il passaggio dal “cosa” al “come” »

E.A. Feigenbaum

Communications of the ACM, Maggio 1996

Dal “cosa” al “come” …



L’estremo “cosa” (what) raccoglie gli obiettivi, i desideri e i bisogni dell’utente, la sua conoscenza del dominio del problema, tutti espressi in linguaggio naturale.

L’estremo “come” (how) è l’hardware che può raggiungere quegli obiettivi e soddisfare i bisogni e i desideri in modo estremamente procedurale e rigorosamente sequenziale.

... Dal “cosa” al “come”

La storia dell’informatica è stato un lento ma costante attraversamento dello spettro what-how.

La scienza della programmazione ha esplorato molti punti dello spettro, sempre combattuta fra il “cosa” e il “come”.

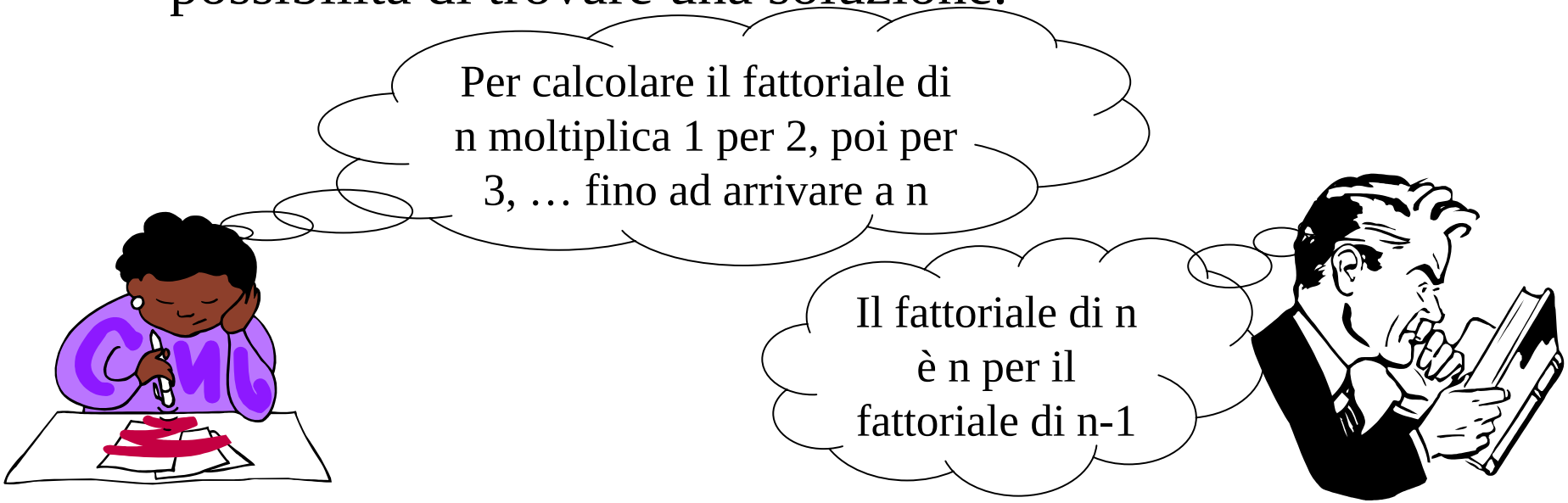
Ne sono nati diversi **paradigmi di programmazione**, ovvero *collezioni di modelli concettuali che insieme plasmano il processo di analisi, progettazione e sviluppo di un programma*.

Paradigma: dal greco παραδειγμα ‘modello, esempio’, da παρα ‘presso, accanto’ (indica similarità) + δεικνυαι ‘mostrare,indicare’

I paradigmi di programmazione

Questi modelli concettuali “strutturano” il pensiero in quanto determinano la forma di programmi validi.

Essi influenzano il modo in cui pensiamo e formuliamo le soluzioni, arrivando a condizionare perfino la possibilità di trovare una soluzione.



I paradigmi di programmazione

Un paradigma cambia fondamentalmente il modo in cui guardiamo ai problemi incontrati nel passato.

Esso ci deve dare un nuovo schema per pensare ai problemi futuri.

Esso cambia le nostre priorità, le nostre idee su quanto merita attenzione e su cosa sia importante.

Come nasce un paradigma?

Un nuovo paradigma è spesso introdotto per risolvere un particolare problema, ma si rivela poi adatto per altri.

L'idea di oggetto che incorpora dati e metodi per lavorare sui dati venne introdotta per la prima volta in SIMULA (K. Nygaard & O.J. Dahl), un linguaggio pensato per la simulazione.

È stata subito ripresa in altri linguaggi di programmazione, come Smalltalk, e approfondita tanto da divenire il cardine del paradigma object-oriented, oggi adottato in molti altri campi diversi da quello della simulazione.

Come nasce un paradigma?

Spesso la sinergia fra discipline apparentemente differenti è un terreno fertile per lo sviluppo di nuovi paradigmi.

Questo è il caso della programmazione logica, sintesi degli studi compiuti in:

- **Logica**: studio della deduzione, cioè del passaggio da assunzioni a conclusioni, indipendentemente dalla verità o falsità delle assunzioni e dal particolare argomento trattato.

Come nasce un paradigma?

- **Intelligenza artificiale**: ha offerto i principi generali per automatizzare la dimostrazione di teoremi in particolari logiche, come quella clausale.
- **Linguaggi formali**: studio di sintassi, semantica e delle procedure di interpretazione di particolari linguaggi di programmazione logica, come il Prolog.

Rapporto Paradigma-Linguaggi

Nel senso della macchina di Turing, tutti i linguaggi di programmazione più comuni sono universali.

Tuttavia ogni linguaggio di programmazione si basa, o meglio *supporta*, un particolare paradigma, fornendo:

a) Le *primitive* di quel paradigma.

b) I *metodi di composizione* di quel paradigma.

c) Un *linguaggio utente appropriato* che rende chiari i programmi scritti secondo quel paradigma

d) Una *esecuzione efficiente* di programmi scritti in quello stile.

Rapporto Paradigma-Linguaggi

I linguaggi di programmazione sono, dunque, dotati di opportuni *costrutti linguistici* che riflettono i modelli concettuali di un paradigma, al fine di facilitare l'espressione di una soluzione definita attraverso i modelli concettuali del paradigma.

In realtà, i linguaggi di programmazione possono supportare più di un paradigma.

Categorie di paradigmi

1. Paradigmi che supportano tecniche di programmazione di basso livello.
2. Paradigmi che supportano metodi di progettazione di algoritmi (e.g., divide-et-impera, programmazione dinamica)
3. Paradigmi che supportano gli approcci di alto livello alla programmazione (e.g., paradigma funzionale e quello basato su regole).

Floyd, R.W. (1978). Turing Award Lecture.

FOCUS del corso: paradigmi del terzo tipo

Approccio dei paradigmi al problem solving

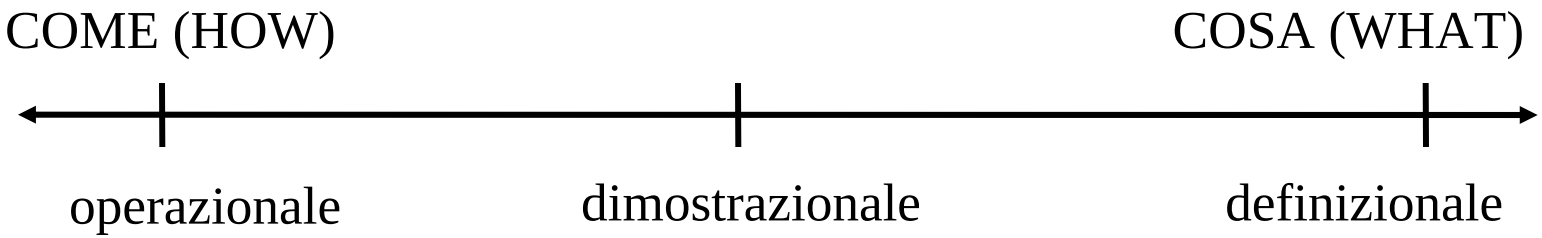
I paradigmi che supportano la programmazione ad alto livello possono essere raggruppati in base al loro approccio alla soluzione dei problemi.

a) *Approccio operazionale*: descrive per passi COME costruire una soluzione.

b) *Approccio dimostrazionale*: è una variante del precedente che illustra la soluzione in modo operativo per esempi specifici e lascia al sistema il compito di generalizzare queste soluzione di esempi ad altri casi.

c) *Approccio definizionale*: esso stabilisce delle proprietà della soluzione in modo da vincolarla senza per questo descrivere come viene calcolata.

Approccio dei paradigmi al problem solving



Cosa deve indagare la scienza della programmazione?

- I *metodi di analisi dei problemi*, che consentono di passare da formulazioni spesso imprecise ed ambigue di un problema, a specifiche espresse in un qualche linguaggio formale.
- Le *tecniche di trasformazione dei programmi*, o tecniche di programmazione, che trasformano la specifica di un problema in un programma che lo risolve.
- I *metodi di verifica della correttezza*, che affrontano il problema di verificare la terminazione dei programmi e la rispondenza alle specifiche date.

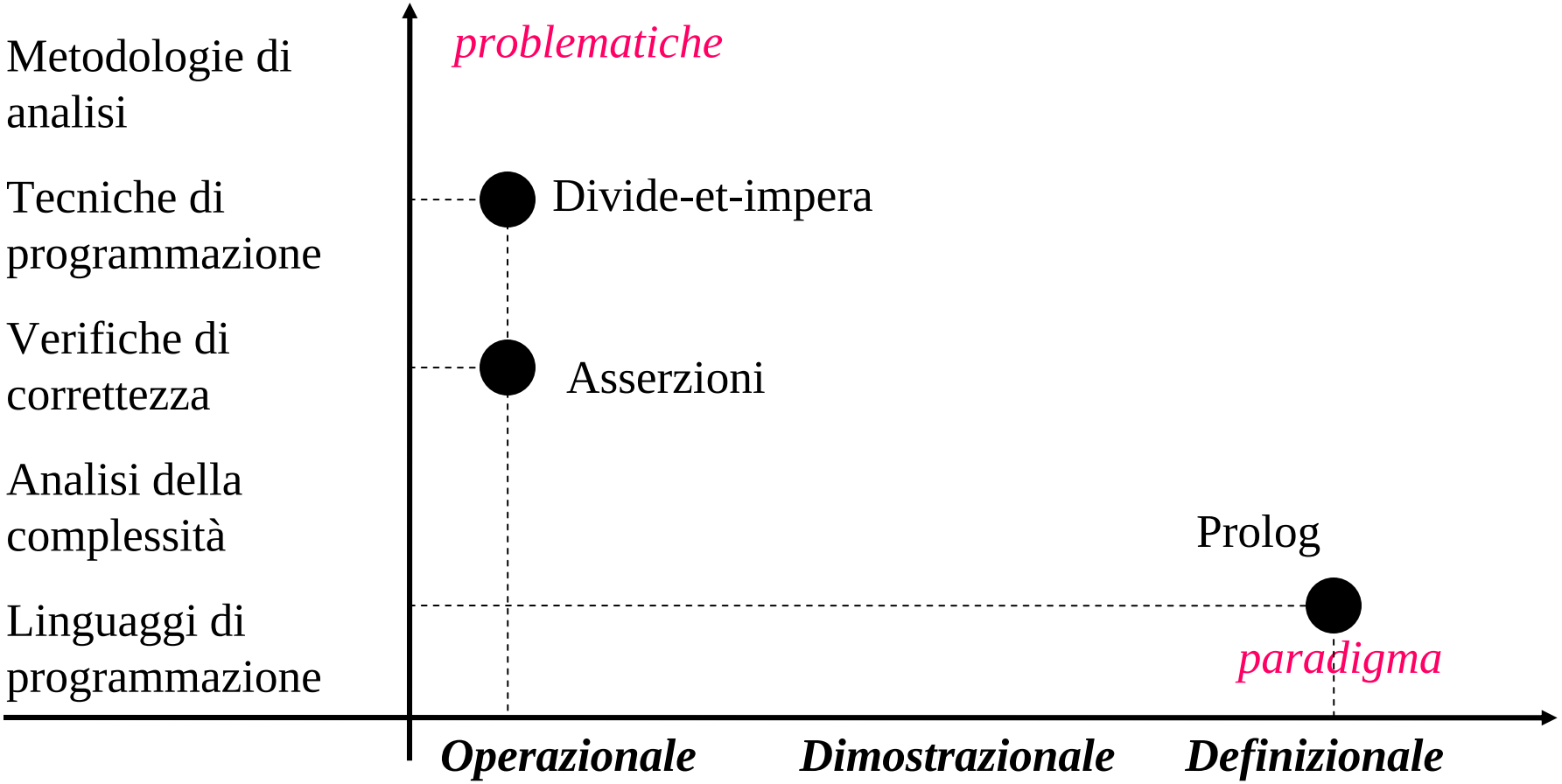
Cosa deve indagare la scienza della programmazione?

- La *teoria della complessità computazionale*, che studia l'efficienza delle soluzioni trovate.
- La *teoria dei linguaggi formali e traduttori*, che studia gli strumenti utilizzabili per comunicare un algoritmo ad un elaboratore.

Quindi le *problematiche* suddette rappresentano una dimensione rispetto alla quale classificare gli studi nel campo della programmazione.

La dimensione paradigmatica è ortogonale a quella per problematiche.

Dimensione della programmazione



Perché studiare diversi paradigmi?

- Nessun singolo paradigma è appropriato per tutti i problemi. Le applicazioni complesse necessitano l'adozione di più paradigmi di programmazione.
- Ogni singolo paradigma di programmazione è caratterizzato da:

  a)Un diverso *potere di elisione*, cioè capacità di esprimere qualcosa in modo conciso.

  b)Una diversa *invarianza rispetto ai cambiamenti* apportati nella strategia di soluzione di un problema.

Approccio multi-paradimatico: quali difficoltà?

Occorre sempre bilanciare i costi dovuti all'uniformità di paradigma con i costi determinati dall'uso di diversi paradigmi per un medesimo problema.

I ***costi*** sono:

- Costi iniziali di apprendimento
- Costi continuati di debugging
- Costi di variazione dovuti all'evoluzione del programma
- Costi di esecuzione dell'applicazione.

Quale classe di applicazioni software?

Le applicazioni software possono essere classificate rispetto alla natura degli elementi che sono di interesse primario nell'applicazione:

- ***Applicazioni orientate alla realizzazione di funzioni***: la complessità prevalente riguarda le funzioni da realizzare.
- ***Applicazioni orientate alla gestione dei dati***: l'aspetto prevalente è rappresentato dai dati che vengono memorizzati, ricercati e modificati, e che costituiscono il patrimonio informativo di una organizzazione.
- ***Applicazioni orientate al controllo***: la complessità prevalente del sistema riguarda il controllo delle attività che si sincronizzano e cooperano durante l'evoluzione del sistema.

Quale classe di applicazioni software?

Rispetto al flusso di esecuzione delle attività:

- ***Applicazioni sequenziali***: sono caratterizzate da un unico flusso di controllo che governa l'evoluzione dell'applicazione. Queste sono le applicazioni più tradizionali, e vengono spesso adottate come riferimento per i metodi e le tecniche di base per la progettazione.
- ***Applicazioni concorrenti***: sono caratterizzate dal fatto che le varie attività che compongono il sistema non hanno una natura inerentemente sequenziale, ma necessitano di meccanismi di sincronizzazione e comunicazione.
- ***Applicazioni dipendenti dal tempo***: sono influenzate da vincoli temporali riguardanti sia la velocità di esecuzione delle attività sia la necessità di sincronizzare le attività stesse.

Di quali applicazioni software ci occupiamo nel corso?

	Sequenziali	Concorrenti	Dipendenti dal tempo
Appl. orientate alla realizzaz. di funzioni	Editor di testo	Elaborazione veloce di immagini	
Appl. orientate alla gestione dei dati	Sistema informativo biblioteca	Sistema informativo bancario	
Applicazioni orientate al controllo			Sistema di emergenza sanitaria

Paradigmi operazionali

Sono caratterizzati da sequenze computazionali passo-passo.

Problema: è difficile stabilire se l'insieme dei valori calcolati operativamente è proprio l'insieme dei valori soluzione.

Le tecniche di *verifica* e di *debugging* cercano di superare questo problema di programmazione.

Spesso ci si accontenta di stabilire che i due insiemi siano "sufficientemente vicini", ovvero indistinguibili per la sottoclasse attesa di problemi effettivi.

Side-effecting

I paradigmi operazionali si distinguono in:

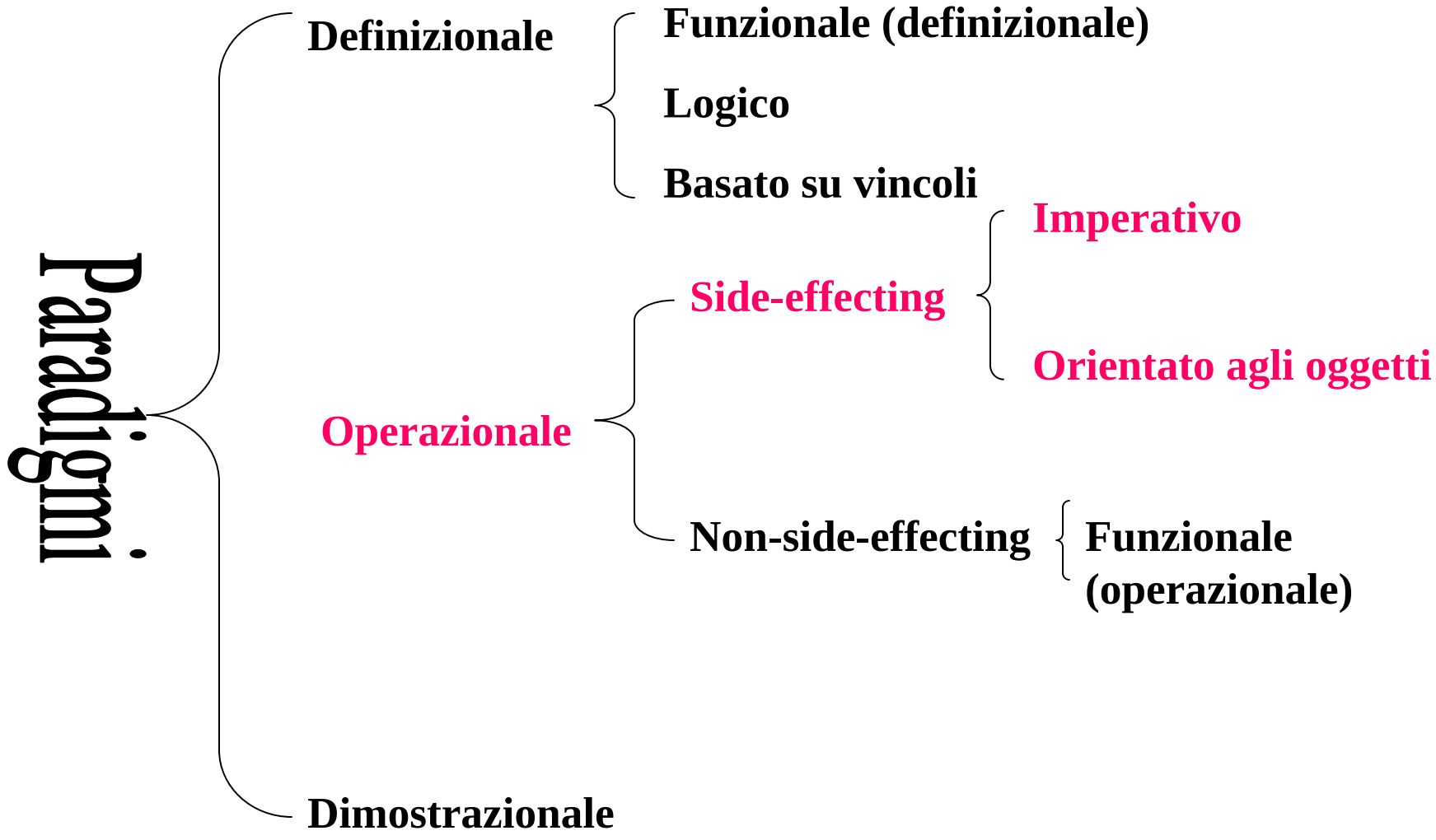
- ***Side-effecting***: procedono modificando ripetutamente la loro rappresentazione dei dati (le variabili sono legate a locazioni di memoria)
- ***Non-side-effecting***: procedono creando continuamente nuovi dati. Questi paradigmi includono quelli che tradizionalmente sono detti *funzionali*.

I paradigmi operazionali con effetti collaterali si distinguono in:

a) Imperativi

b) Orientati agli oggetti

Riepilogando …



Sequenziale vs. Concorrente

Per ognuno dei paradigmi visti precedentemente possiamo distinguere due versioni:

- *Sequenziale*: il flusso di controllo è unico
- *Concorrente* o *parallelo*: più flussi di controllo sono ammessi.

Nel corso verranno spiegati prevalentemente concetti legati alla versione *sequenziale*.

Riferimenti bibliografici

Per approfondimenti su una categorizzazione dei paradigmi di programmazione si raccomanda la lettura del seguente articolo:

- A.L. Ambler, M.H. Burnett, & B.A. Zimmerman. Operational Versus Definitional: A Perspective on Programming Paradigms. *IEEE Computer*, 25(9): 28-43, September 1992.