

# Linguaggi di Programmazione

## Corso di Laurea in "Informatica"

Introduzione ai linguaggi di  
programmazione

*Valeria Carofiglio*

a.a. 2015-2016

(questo materiale è una rivisitazione del materiale  
prodotto da Nicola Fanizzi)

Cos'è un linguaggio di programmazione?

Una notazione per descrivere  
algoritmi e strutture dati  
può essere considerata un  
linguaggio di programmazione

Ma attenzione!!!!

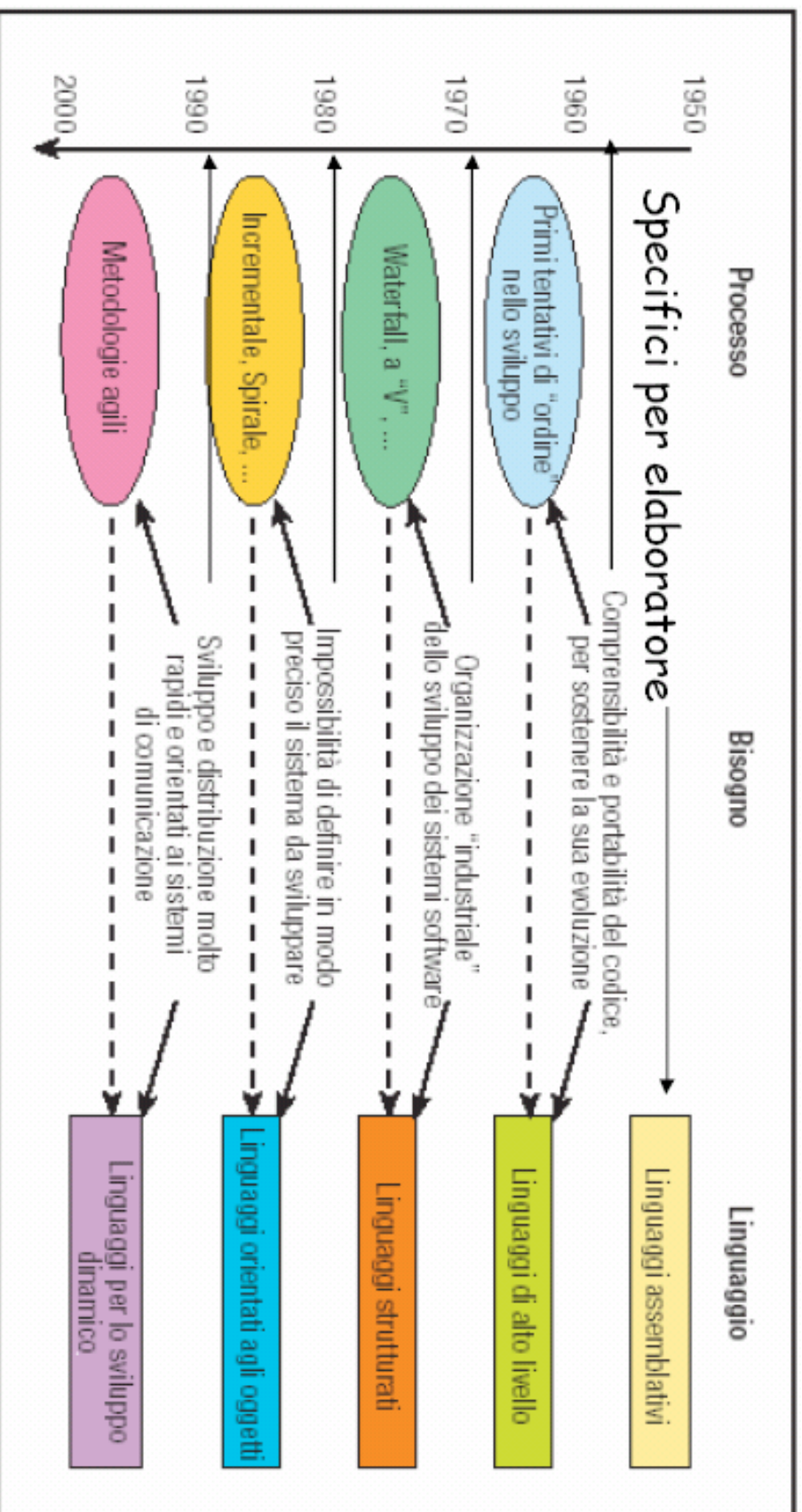
# Quanti sono i linguaggi di programmazione?

L'evoluzione della comunicazione uomo-elaboratore  
crea dialetti e gerghi  
(analogia con linguaggio naturale)



# Evoluzione dei linguaggi di programmazione

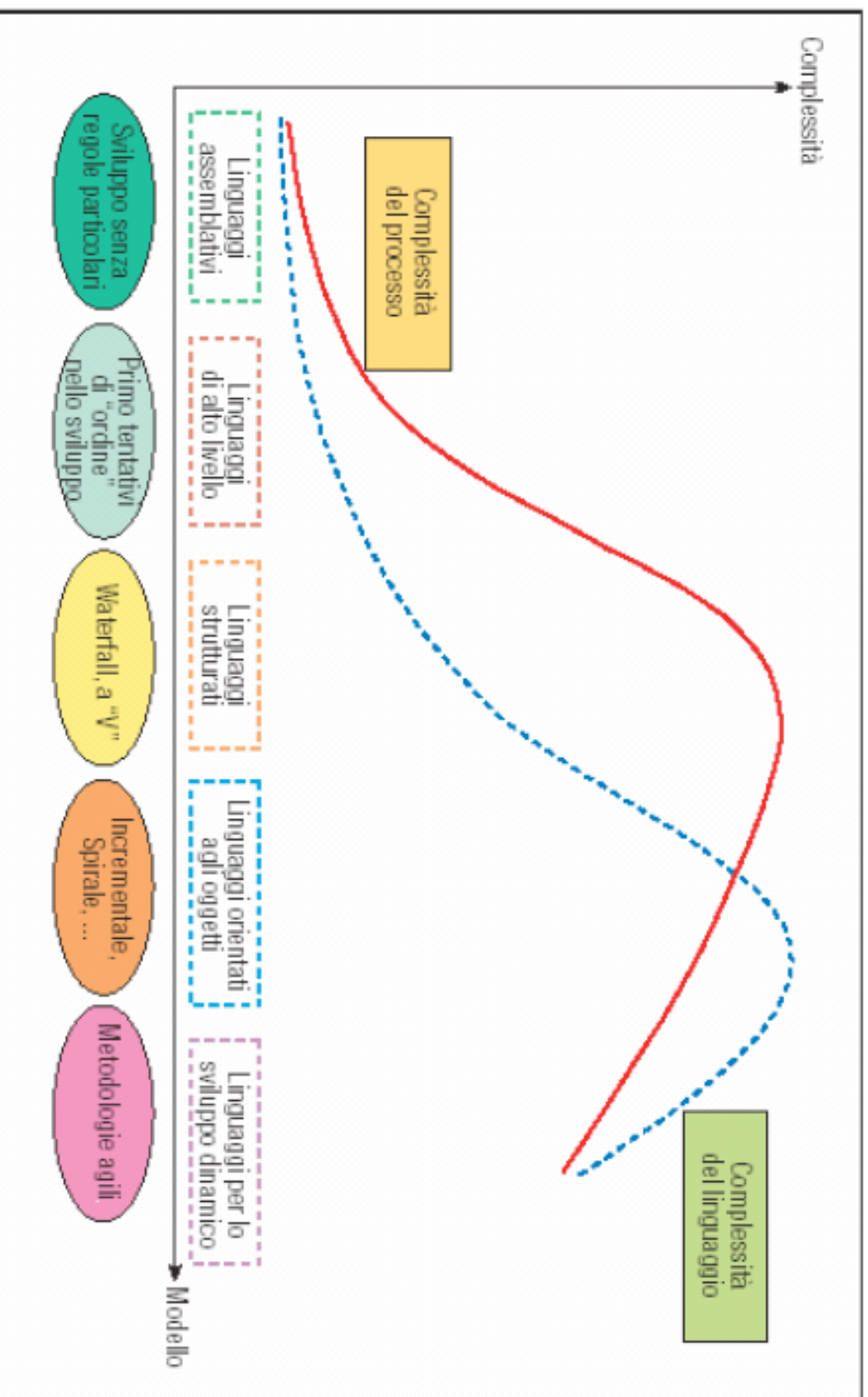
**Con riferimento ai processi di sviluppo**



**ATTENZIONE:**

**molto spesso il linguaggio anticipa il processo cui fa riferimento!!**

# Evoluzione della complessità di linguaggi e processi



# Evoluzione della complessità di linguaggi e processi

*Con riferimento all'architettura degli elaboratori*

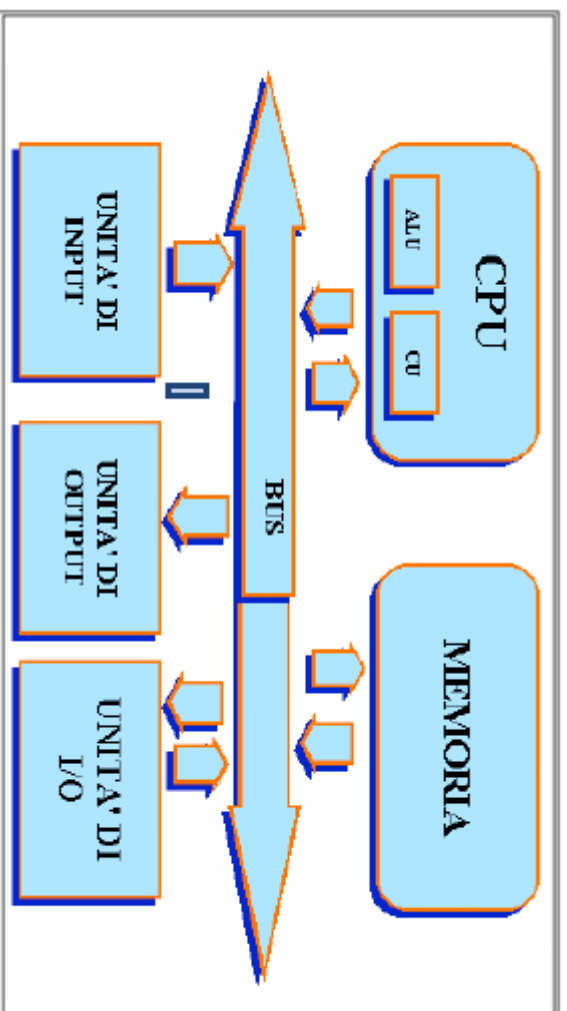
**I Ldp convenzionali sono visti come**

## **ASTRAZIONI**

**di una architettura di VonNeumann:**

# L'architettura di Von Neumann

Alla base delle macchine attuali



Cicli di:

- CPU: preleva una istruzione per volta dalla memoria
- Esecuzione di una istruzione: prelievo dati dalla memoria + Manipolazione dati + Copiatura risultati

Cambiamento  
di stato

# Evoluzione dei linguaggi di programmazione

Con riferimento all'architettura degli elaboratori

I LdP convenzionali sono visti come ASTRAZIONI di una architettura di VonNeumann:

## Architettura di Von Neumann

Prelievo sequenziale della CPU + esecuzione

Esecuzione sequenziale delle istruzioni

Cella di memoria (indirizzo, contenuto)

Variabili (nome, valore)

Stato = Contenuto della memoria

Stato = Valore delle variabili

**Modello computazionale di un linguaggio imperativo:**

Esecuzione sequenziale di istruzioni

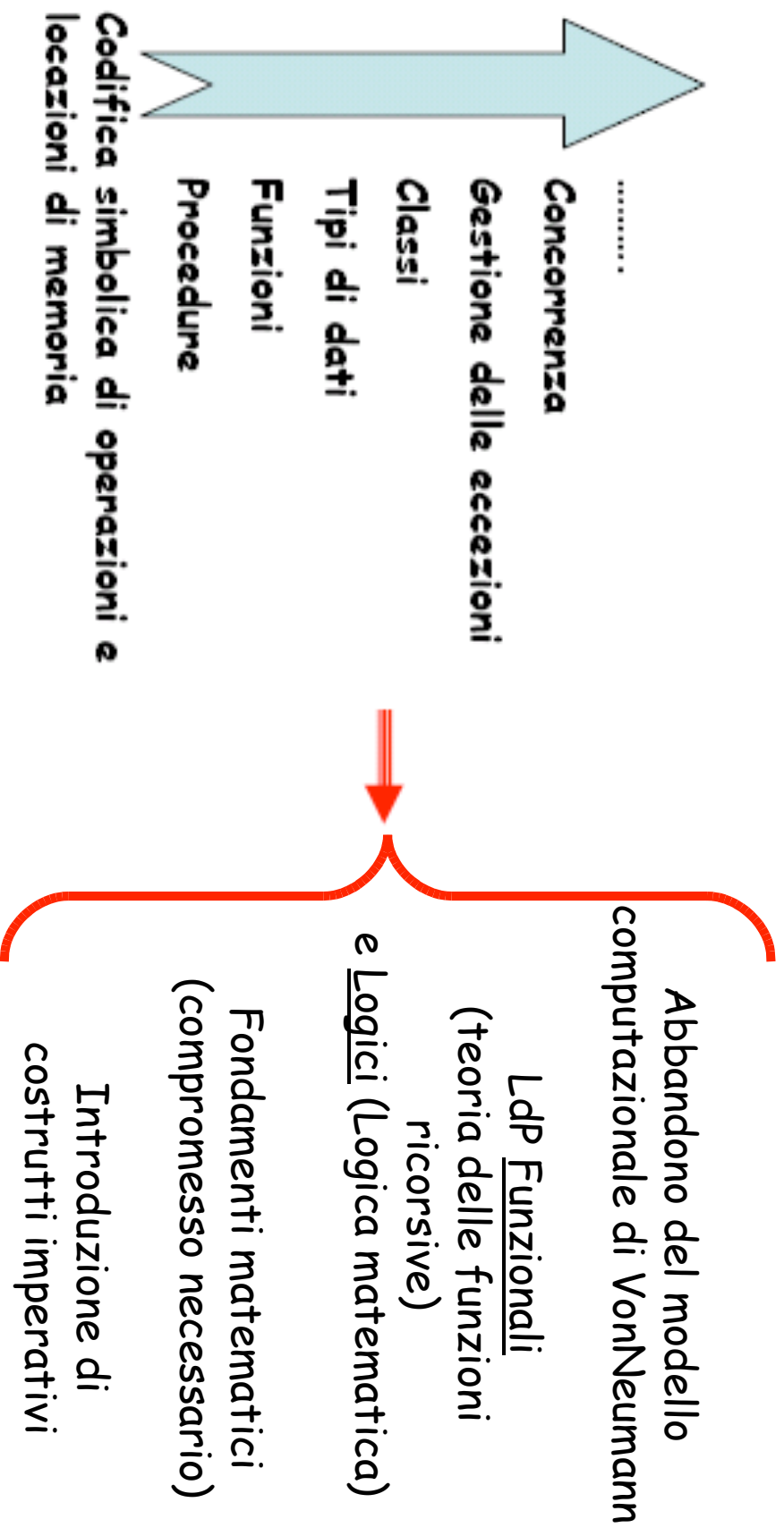
che cambiano lo stato della computazione  
mediante la modifica dei valori di un insieme di variabili



# Evoluzione dei linguaggi di programmazione

Con riferimento all'architettura degli elaboratori

I Ldp si sono evoluti verso livelli di astrazione crescenti



# Quanti sono i linguaggi di programmazione?

L'evoluzione della comunicazione uomo-elaboratore  
crea dialetti e gerghi  
(analogia con linguaggio naturale)



**Un progettista di LdP deve saper bilanciare:**

- 1) Computazione espressa convenientemente per la persona
- 2) Uso efficiente degli elaboratori (in funzione anche del dominio applicativo)

# Quanti sono i linguaggi di programmazione?

L'evoluzione della comunicazione uomo-elaboratore  
crea dialetti e gerghi  
(analogia con linguaggio naturale)



Un progettista di LdP deve saper bilanciare:

- 1) Computazione espressa convenientemente per la persona
- 2) Uso efficiente degli elaboratori (in funzione anche del dominio applicativo)

E' necessario un criterio di catalogazione  
(I paradigmi di programmazione)

# Criterio di catalogazione dei LdP

**In base alle loro caratteristiche principali**

## •Imperativi

- PROGRAMMA = ALGORITMO +DATI
- Le caratteristiche centrali sono le variabili, l'istruzione di assegnazione e l'iterazione
- Esempi: C, Pascal

## •Funzionali

- PROGRAMMA = COLLEZIONE DI FUNZIONI MATEMATICHE
- Il principale meccanismo di calcolo è l'applicazione di funzioni (con dominio e codominio) ai parametri dati
- Esempi: LISP, Scheme

# Criterio di catalogazione dei LdP

**In base alle loro caratteristiche principali**

## •Logici

•PROGRAMMA = COLLEZIONE DI DICHIARAZIONI LOGICHE SU COSA UNA CERTA FUNZIONE DEBBA COMPUTARE

•Basati su regole (in un ordine non particolare). Il meccanismo di calcolo è l'applicazione di dichiarazioni per trovare soluzioni (non uniche al problema)

•Esempi: Prolog

## •Orientati agli oggetti

•PROGRAMMA= collezione di oggetti che interagiscono passandosi messaggi che trasformano il loro stato

•Astrazione dati, ereditarietà, late binding (associazione a run time di valori e identificatori)

•Esempi: Java, C++

# Criterio di catalogazione dei LdP

**In base alle loro caratteristiche principali**

- **Markup**
  - Nuovi; non sono in senso stretto linguaggi di programmazione, ma specificano il layout dell'informazione nei documenti per il Web
- Esempi: XHTML, XML

# Perché studiare concetti dei LdP?

- **Accrescimento della capacità di esprimere idee**
  - Profondità del nostro pensiero  $\leftrightarrow$  potere espressivo del linguaggio
  - LdP vincola il modo in cui esprimiamo la risoluzione di un problema (algoritmo)

# Perché studiare concetti dei Ldp?


- **Accrescimento della capacità di esprimere idee**
  - Profondità del nostro pensiero  $\leftrightarrow$  potere espressivo del linguaggio
  - Ldp vincola il modo in cui esprimiamo la risoluzione di un problema (algoritmo)
- **Miglioramento della capacità di scelta di Ldp appropriati**



# Perché studiare concetti dei LdP?

- **Accrescimento della capacità di esprimere idee**
  - Profondità del nostro pensiero  $\leftrightarrow$  potere espressivo del linguaggio
  - LdP vincola il modo in cui esprimiamo la risoluzione di un problema (algoritmo)
- **Miglioramento della capacità di scelta di LdP appropriati**
- **Accrescimento della capacità di imparare nuovi linguaggi**

Tecnologia sw legata a


- 
- Metodologia di progetto
  - Strumenti di sviluppo
  - **Ling.di Programmazione**

Apprendimento di un nuovo LdP è un processo **lungo e faticoso** ed è agevolato dalla conoscenza di concetti dei LdP, indipendentemente dal particolare LdP

# Perché studiare concetti dei LdP?

- **Accrescimento della capacità di esprimere idee**
  - Profondità del nostro pensiero  $\leftrightarrow$  potere espressivo del linguaggio
  - LdP vincola il modo in cui esprimiamo la risoluzione di un problema (algoritmo)
- **Miglioramento della capacità di scelta di LdP appropriati**
- **Accrescimento della capacità di imparare nuovi linguaggi**

Tecnologia sw legata a

- 
- Metodologia di progetto
  - Strumenti di sviluppo
  - **Ling.di Programmazione**

Apprendimento di un nuovo LdP è un processo **lungo** e **faticoso** ed è agevolato dalla conoscenza di concetti dei LdP, indipendentemente dal particolare LdP

- **Accrescimento della capacità di progettare nuovi LdP**

# Domini Applicativi

- **Applicazioni Scientifiche**
  - Gran numero di calcoli in virgola mobile
  - Fortran (1957)
- **Applicazioni Commerciali / Bancarie**
  - Produzione di report, uso di numeri decimali e caratteri
  - COBOL (1959)
- **Intelligenza Artificiale**
  - Manipolazioni su simboli piuttosto che su numeri
  - LISP (1958), Prolog (1970), ...

# Domini Applicativi

- **Programmazione di sistema**
  - Importanza dell'efficienza per via dell'uso continuo
  - C
- **Sistemi per il Web**
  - Insieme di vari linguaggi: markup (es., XHTML), scripting (es., PHP, Javascript), general-purpose (es., Java)

# Requisiti di qualità del software

e...

- Affidabilità
- Manutenibilità
- Efficienza

## ...Metodi per soddisfarli

- Metodi di sviluppo del Software
- Tools di supporto allo sviluppo
- *Caratteristiche dei LdP*

# Affidabilità del Software e LdP

Il software deve rispettare i requisiti  
in ogni circostanza

## Leggibilità e Scrivibilità

Un linguaggio che non supporta modalità "naturali" per esprimere un algoritmo userà necessariamente approcci "non naturali", e quindi una ridotta affidabilità

# Affidabilità del Software e LdP (cont.)

## Semplicità:

- Pochi costrutti,
- Un ridotto numero di primitive,
- Poche regole per la loro combinazione.
- Supporto all' astrazione: L'abilità di definire e usare strutture complesse o operazioni in modalità che consentano di trascurare i dettagli

## Sicurezza

- Evitare costrutti che consentano la scrittura di programmi pericolosi (Goto, Puntatori)



# Affidabilità del Software e LdP (cont.)

## Robustezza

- Grado di capacità di reazione ad eventi indesiderati (esaurimento memoria, input errato, overflow aritmetico)
  - Es: Gestione delle eccezioni

# Manutenibilità del Software e LdP

I programmi devono essere  
facilmente modificabili

## Fattorizzazione

La possibilità di modellare una caratteristica in un unico segmento di codice (sottoprogrammi, costanti simboliche)

## Localizzazione

La possibilità di restringere l'effetto di un costrutto ad una piccola porzione di programma

# Efficienza del Software e LdP

- **In termini di Prestazioni (spazio tempo)**
- **In termini di Produttività:**
  - Sforzo per produrre/manutenere
  - Possibilità di riuso
  - portabilità

**Esempio di influenza negativa sull'efficacia in spazio:**

**Il non riuso della memoria**

# Ulteriori caratteristiche di un LdP: Costo

- **Addestramento** dei programmatori all'uso di un linguaggio
- **Scrittura** dei programmi (vicinanza a particolari applicazioni)
- **Compilazione** dei programmi
- **Esecuzione** dei programmi
- Sistema di **implementazione** dei linguaggi:
  - Disponibilità di compilatori (gratuità)
- **Affidabilità**: una scarsa affidabilità porta all'aumento dei costi
- **Manutenzione** dei programmi

# Ulteriori caratteristiche di un LdP: Altro

- **Portabilità**
  - La facilità con cui i programmi possono essere spostati da una implementazione del linguaggio ad un'altra
- **Generalità**
  - L'applicabilità ad un ampio spettro di applicazioni
- **Precisione nella definizione**
  - La completezza e precisione nella definizione ufficiale del linguaggio

# Compromessi nella progettazione dei linguaggi

- **Affidabilità vs. Costo di esecuzione**
  - Esempio: Java richiede che tutti i riferimenti a componenti di un array elements siano controllati per quanto riguarda la corretta indicizzazione ma ciò comporta maggiori costi per l'esecuzione
- **Leggibilità vs. Scrivibilità**
  - Esempio: APL fornisce numerosi operatori potenti (e un gran numero di nuovi simboli), che consentono la scrittura sintetica di calcoli complessi e programmi compatti, al costo, però, di una scarsa leggibilità
- **Scrivibilità (flessibilità) vs. Affidabilità**
  - Esempio: i puntatori C++ sono potenti e molto flessibili ma di uso spesso poco affidabile