Modelli di qualità del sw

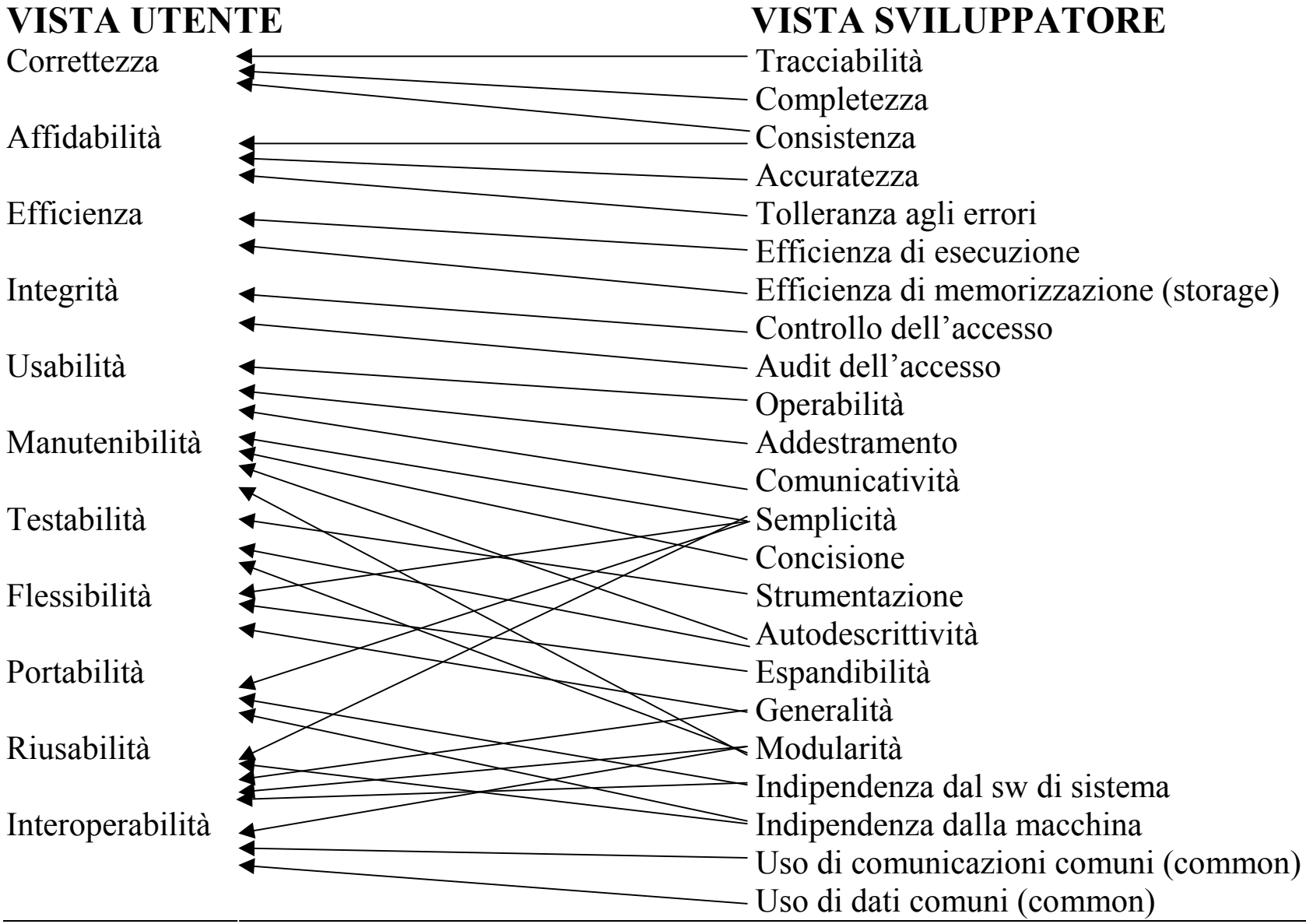
- Qualità del prodotto (per confrontare un prodotto con un altro)
  - ✓Modello di McCall
  - ✓Modello di Boehm
  - ✓ISO 9126
  - ✓Modello di Dromey

- Qualità del processo (per confrontare un processo con un altro)
  - ✓ CMM
  - ✓ ISO 9001

Modello di McCall, Richards e Walters (1977)

Collega i fattori di qualità esterna, secondo la visione dell'utente, con quelli di qualità interna, secondo la visione dello sviluppatore, e ritiene questi ultimi misurabili

Modello di McCall et al. (cont.)



Modello di Boehm et al. (1978)

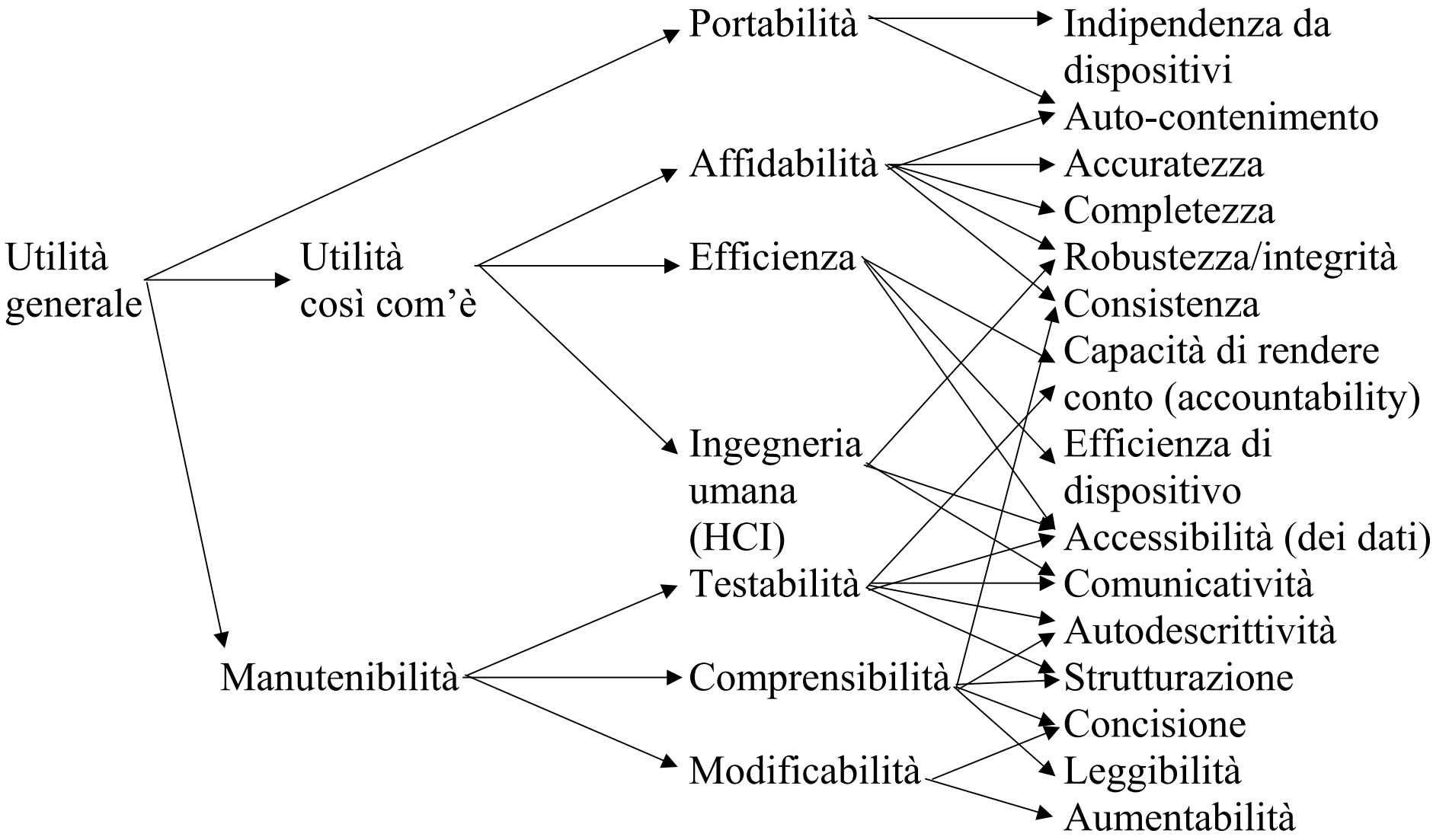
È uno dei più noti; include caratteristiche prestazionali dell'hw assenti nel modello di McCall

Fra i fattori di qualità (esterna) si annoverano:

Integrità = capacità del sistema di produrre il risultato corretto al livello corretto di accuratezza

Consistenza = capacità del sistema di produrre sempre lo stesso risultato a parità di dati di ingresso e di condizioni

Modello di Boehm et al. (cont.)

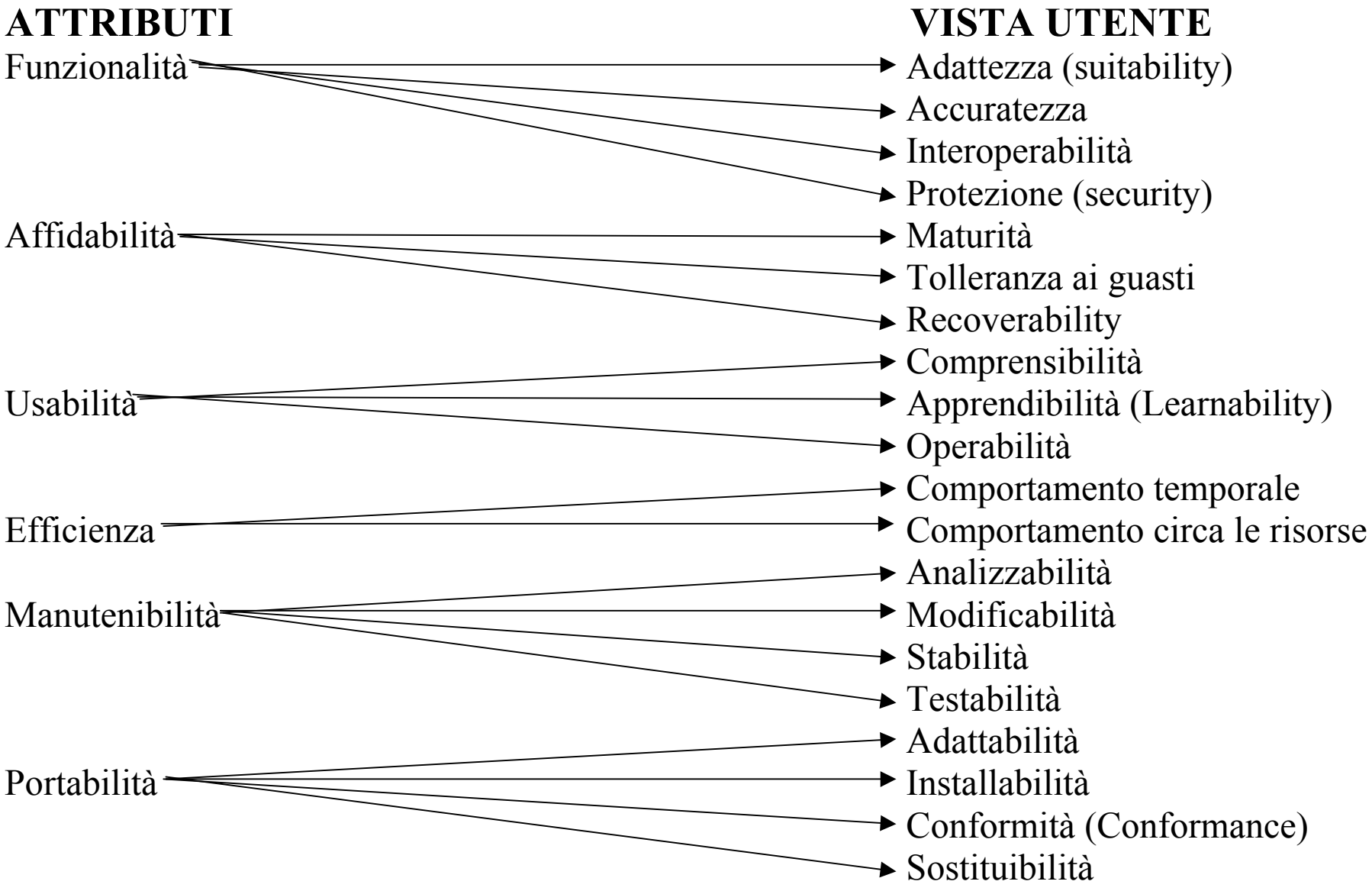


ISO 9126 (1991)

Confronto coi modelli di McCall e Boehm

- Differenze:
  - ✓ ogni caratteristica sulla destra è posta in relazione con un solo attributo sulla sinistra
  - ✓ tutte le caratteristiche a destra si riferiscono alla visione dell'utente, non a quella dello sviluppatore
- Analogie:
  - ✓ si raccomanda di misurare direttamente le caratteristiche sulla destra ma non si dice come
  - ✓ non si spiega perché alcune caratteristiche e attributi sono inclusi e altri no (ad es. safety)
  - ✓ non si spiega come comporre la valutazione delle caratteristiche di livello più basso per determinare il valore di ogni attributo

ISO 9126 (cont.)



Modello di Dromey (1996)

- Classifica gli attributi di qualità tangibili in 4 categorie non disgiunte:
  - ✓ proprietà di correttezza
  - ✓ proprietà interne
  - ✓ proprietà contestuali
  - ✓ proprietà descrittive
- Propone che gli attributi di qualità siano quelli di elevata priorità per il progetto in corso (quindi, non fissi come nei modelli precedenti ma diversi da progetto a progetto)
- Propone un metodo per la creazione di modelli, ciascuno relativo a un prodotto del processo di sviluppo (requisiti, design, codice, ecc.)

Modello di Dromey: metodo

1) Identificare un insieme di attributi di qualità (ad es. i sei dello standard ISO 9126)
2) Identificare i componenti del prodotto (ad es. variabili, espressioni, ecc.)
3) Identificare e classificare le proprietà più significative che siano indicatori di qualità per ciascun componente
4) Proporre un insieme di assiomi per collegare le qualità del prodotto con gli attributi di qualità
5) Valutare il modello, identificarne le debolezze, rifinirlo o ricrearlo

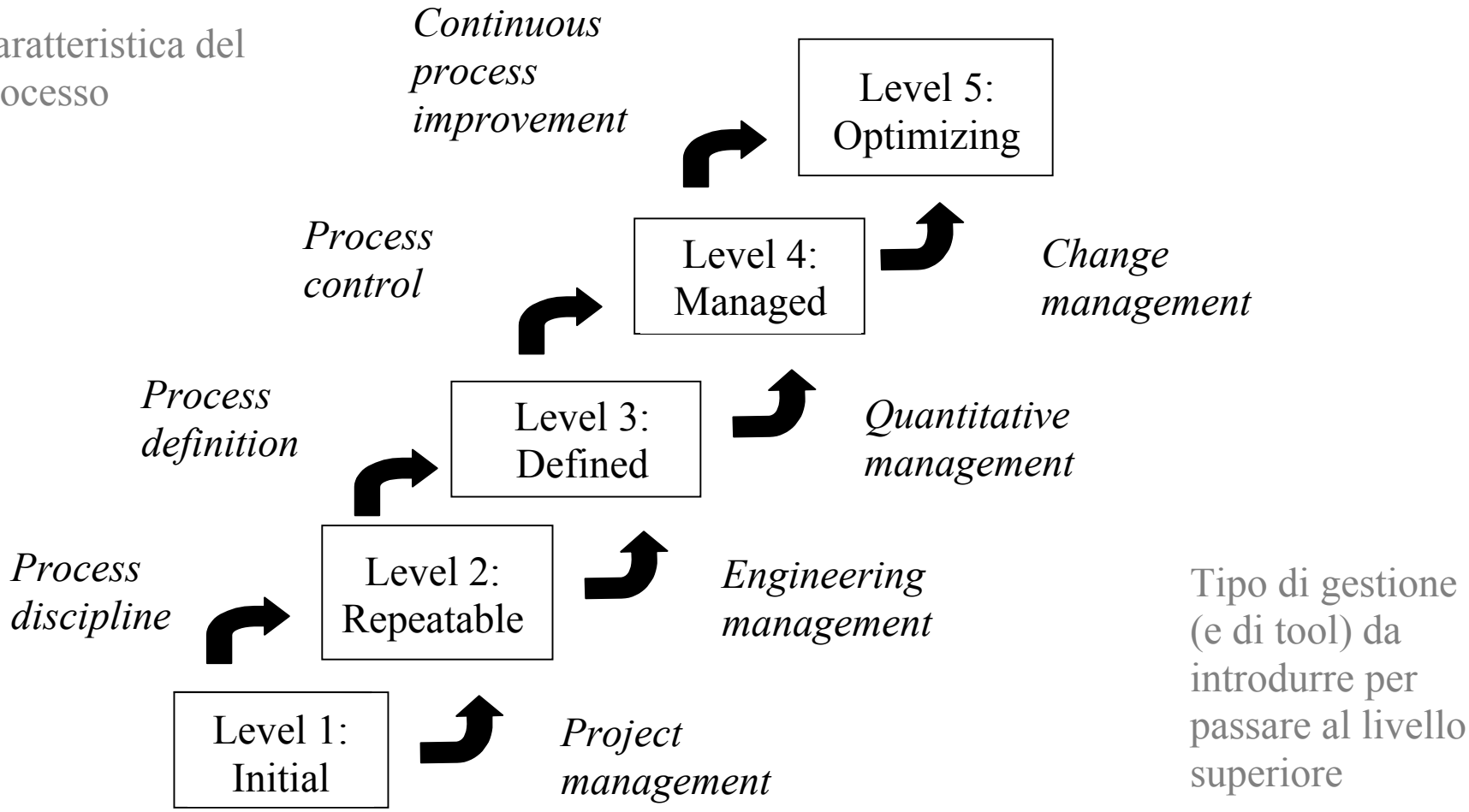
Capability Maturity Model (CMM): primi passi

- Sviluppato da Software Engineering Institute (SEI) per assistere il Dipartimento della Difesa statunitense nel valutare la qualità dei suoi contraenti
- Inizialmente si trattava di un *process maturity model* che collocava l'organizzazione da valutare su un livello di una scala a 5 valori in base alle risposte a un questionario (per passare da un livello al successivo, tutte le risposte al gruppo di domande corrispondenti a tale passaggio dovevano essere positive)

CMM

- Aggiunge al questionario delle richieste di dimostrare con fatti concreti la veridicità delle risposte
- Descrive, organizzandoli sempre in 5 livelli, principi e modalità pratiche per migliorare il processo produttivo
- Associa a ogni livello di capacità un insieme di aree chiave (Paulk et al. 1993) del processo su cui quel livello si focalizza
- Associa a ogni area chiave un insieme di pratiche chiave relative a implementazione e istituzionalizzazione (ripetibilità e durata) dell'area stessa; le domande del questionario sono inerenti a queste pratiche
- L'impiego del modello diviene così duplice:
  - ✓ consente ai clienti potenziali di identificare punti di forza e debolezza dei loro fornitori
  - ✓ consente agli sviluppatori di verificare e migliorare le loro capacità

CMM: livelli


Caratteristica del processo
Continuous process improvement
Level 5: Optimizing
Process control
Level 4: Managed
Change management
Process definition
Level 3: Defined
Quantitative management
Process discipline
Level 2: Repeatable
Engineering management
Level 1: Initial
Project management
Tipo di gestione (e di tool) da introdurre per passare al livello superiore

CMM Level	Key Process Areas	Processo
Initial	None	Processo ad hoc o caotico; ingressi del processo mal definiti, trasformazione dagli ingressi alle uscite né definita, né controllata; visibilità nulla e difficoltà di misurazione; il successo dello sviluppo dipende dagli sforzi individuali, non dalla forza di squadra Suggerimenti: strutturare e controllare di più il processo

CMM Level	Key Process Areas	Processo
Repeatable	• Requirements management • Sw project planning • Sw project tracking and oversight • Sw subcontract management • Sw quality assurance • Sw configuration management	Identificazione (→ visibilità → possibilità di misurare) di ingressi, uscite, vincoli (come budget, schedule, direttive del management, standard, ecc.) e risorse (tool e staff) del processo (visto come un tutto unico); il successo di progetti precedenti può essere ripetuto con progetti simili; nessuna visibilità circa come sono prodotte le uscite

CMM Level	Key Process Areas	Processo
Defined	• Organization process focus • Organization process definition • Training program • Integrated sw management • Sw product engineering • Intergroup coordination • Peer reviews	Processo standardizzato a livello di organizzazione (ogni adattamento contingente dello standard deve essere approvato dal management) → il processo è suddiviso in attività gestionali e ingegneristiche, che sono documentate, standardizzate e integrate, c'è visibilità (→ analizzabilità e possibilità di misurare) su ingressi e uscite delle attività intermedie → le misure relative ai prodotti delle fasi più arretrate sono utili indicatori per le misure dei prodotti successivi → maggior controllo sullo sviluppo e riduzione dei rischi, grazie al trattamento tempestivo dei problemi

CMM Level	Key Process Areas	Processo
Managed	• Quantitative process management • Sw quality management	Introduzione di misure dettagliate della qualità di processo e prodotto al fine di stabilizzare il processo e, quindi, ottenere produttività e qualità desiderate
Optimizing	• Fault prevention • Technology change management • Process change management	Introduzione nel processo di una retroazione quantitativa per produrre miglioramenti continui dello stesso: test e monitoraggio di nuovi strumenti e tecniche per valutare come possano influenzare processo e prodotti, possibile aggiunta di nuove attività o cambiamento dinamico (nell'ambito del progetto corrente) della struttura del processo in risposta alla retroazione

ISO 9001 (1987)

Lo standard (che fa parte della serie ISO 9000) spiega cosa deve fare un acquirente per assicurarsi che il fornitore si conformi ai requisiti di progettazione, sviluppo, produzione, installazione e manutenzione (certificazione di un processo generico)

Documento ISO 9000-3 (1990)
Fornisce l'interpretazione di ISO 9001 nel contesto del sw

COnstructive COst MOdel: COCOMO II

- È un modello predittivo dello sforzo (espresso in mesi-persona) di un progetto, sviluppato come aggiornamento del modello sperimentale di successo COCOMO
- COCOMO (Boehm et al. 1981) usava le linee di codice sorgente come ingresso primario … ma queste non si possono conoscere all'inizio del processo
- COCOMO II (Boehm et al. 1995) allows increasingly detailed estimates to be prepared as development progresses (e la comprensione dei parametri del progetto aumenta):
  - ♦ Early prototyping level (detto anche Application composition): estimates based on object points
  - ♦ Early design level: estimates based on function points that are then translated to LOC
  - ♦ Post-architecture level: estimates based on lines of source code

N.B. COCOMO II assume implicitamente che il codice sia scritto in 4GL

COCOMO II: early prototyping level

- Supports prototyping projects and projects where there is extensive reuse
- Based on standard estimates of developer productivity in (object points/month)
- Takes CASE tool use into account
- Formula is
  - $PM = (NOP \times (1 - \%reuse/100)) / PROD$
  - PM is the effort in person-months, NOP is the number of object points and PROD is the productivity

Il mese-persona di COCOMO non comprende ferie, vacanze o weekend

COCOMO II: early prototyping level (cont.)

PARAMETRI						
	Developer's experience and capability	Very low	Low	Nominal	High	Very high
	CASE maturity and capability	Very low	Low	Nominal	High	Very high
	PROD (espressa in NOP/month)	4	7	13	25	50

La PROD da usare nella formula è la media dei due valori di PROD in tabella corrispondenti ai due parametri

Per calcolare il valore di NOP:

1) contare quante schermate (screen), report, e componenti scritti in linguaggi di 3° generazione saranno presenti nell'applicazione
2) classificare il livello di complessità di ciascuno degli elementi di cui al punto precedente come semplice, medio, o difficile, secondo la tabella che segue

COCOMO II: early prototyping level (cont.)

Per gli screen				Per i report			
	# e sorgente delle tabelle dei dati				# e sorgente delle tabelle dei dati		
# viste contenute	Totale <4 (<2 server, <3 client)	Totale <8 (2-3 server, 3-5 client)	Totale 8+ (>3 server, >5 client)	# sezioni contenute	Totale <4 (<2 server, <3 client)	Totale <8 (2-3 server, 3-5 client)	Totale 8+ (>3 server, >5 client)
<3	semplice	semplice	medio	0 o 1	semplice	semplice	medio
3-7	semplice	medio	difficile	2 o 3	semplice	medio	difficile
8+	medio	difficile	difficile	4+	medio	difficile	difficile

3) Dato ciascuno degli elementi di cui al punto 2 e il suo livello di complessità, come determinato al punto precedente, trovare nella tabella che segue il peso di complessità corrispondente

Tipo di elemento	Semplice	Medio	Difficile
Screen	1	2	3
Report	2	5	8
Componente 3GL	-	-	10

NOP = somma di tutti i pesi di complessità così calcolati

COCOMO II: early design level

- Estimates can be made after the requirements have been agreed
- Based on standard formula for algorithmic models
  - $PM = A \times Size^{B} \times M + PM_m$ where
  - $M = PERS \times RCPX \times RUSE \times PDIF \times PREX \times FCIL \times SCED$
  - $PM_m = (ASLOC \times (AT/100)) / ATPROD$
  - A = 2.5 in initial calibration, Size in KLOC (per l'early design level è calcolata in base ai function point e al linguaggio), B varies from 1.01 to 1.26 depending on novelty of the project, development flexibility, risk management approaches and the process maturity (viene calcolato come sarà descritto per il Post-architecture level)

LOC = linea di codice = ciascuna linea non vuota di programma che non sia un commento, indipendentemente dal # di istruzioni o porzioni di istruzioni che contiene

COCOMO II: early design level (cont.)

- Multipliers reflect the capability of the developers, the non-functional requirements, the familiarity with the development platform, etc.
  - PERS - personnel capability
  - RCPX - product reliability and complexity
  - RUSE - the reuse required
  - PDIF - platform difficulty
  - PREX - personnel experience
  - FCIL - the team support facilities
  - SCED - required schedule
- PM_m reflects the amount of automatically generated code

COCOMO II: post-architecture level

- Uses same formula as early design estimates
- Estimate of size is adjusted to take into account:
  - ✓ Requirements volatility. Rework required to support change
  - ✓ Extent of possible reuse. Reuse is non-linear and has associated costs so this is not a simple reduction in LOC
  - ✓ $ESLOC = ASLOC \times (AA + SU + 0.4DM + 0.3CM + 0.3IM)/100$
    - ESLOC is equivalent number of lines of new code (è la variabile Size della formula). ASLOC is the number of lines of reusable code which must be modified, DM is the percentage of design modified, CM is the percentage of the code that is modified , IM is the percentage of external code (such as reused code) to be integrated
    - SU quantifies the cost of required sw understanding (questa quantificazione è una linea guida per lo sviluppo di sw manutenibile), AA reflects the initial assessment costs of deciding if sw may be reused

COCOMO II: post-architecture level

	Molto bassa	**Bassa**	**Nominale**	**Alta**	**Molto alta**
Struttura	Coesione molto bassa, elevato accoppiamento, codice a spaghetti	Coesione moderatamente bassa, elevato accoppiamento	Alcune aree deboli	Elevata coesione, basso accoppiamento	Forte modularità, info hiding nelle strutture dati e di controllo
Chiarezza dell'applicazione	Nessuna corrispondenza fra le visioni del mondo del programma e dell'applicazione	Qualche correlazione fra programma e applicazione	Modesta correlazione fra programma e applicazione		Chiara corrispondenza fra le visioni del mondo del programma e dell'applicazione
Auto-descrittività	Codice oscuro; documentazione mancante, oscura od obsoleta	Alcuni commentary header nel codice; un po' di documentazione utile	Modesto livello di commentary header e documenta-zione		Codice auto-descrittivo; documentazione aggiornata, ben organizzata con giustificazione del design
SU	50	40	30	20	10

COCOMO II: post-architecture level

- The exponent term B depends on 6-scale factors (see next slide). Their sum/100 is added to 1.01

Extra high	Very high	High	Nominal	Low	Very low
0	1	2	3	4	5

- Example
  - Precedenteness - new project - 4
  - Development flexibility - no client involvement - Very high - 1
  - Architecture/risk resolution - No risk analysis - Very Low - 5
  - Team cohesion - new team - Nominal - 3
  - Process maturity - some control - Nominal - 3

  Scale factor is therefore 1.17

COCOMO II: post-architecture level

Scale factor	Explanation
Preceden-tedness	Reflects the previous experience of the organisation with this type of project. *Very low* means no previous experience; *Extra high* means that the organisation is completely familiar with this application domain
Development flexibility	Reflects the degree of flexibility in the development process. *Very low* means a prescribed process is used; *Extra high* means that the client only sets general goals
Architecture/ risk resolution	Reflects the extent of risk analysis carried out. *Very low* means little analysis; *Extra high* means a complete a thorough risk analysis
Team cohesion	Reflects how well the development team know each other and work together. *Very low* means very difficult interactions; *Extra high* means an integrated and effective team with no communication problems
Process maturity	Reflects the process maturity of the organisation. The computation of this value depends on the CMM Maturity Questionnaire but an estimate can be achieved by subtracting the CMM process maturity level from 5

COCOMO II: multipliers (cost drivers)

Il loro valore di default è 1. I valori degli indicatori di costo chiave del sistema considerato si ottengono consultando il manuale di riferimento di COCOMO II (Boehm 1997)

- Product attributes
  - ♦ concerned with required characteristics of the sw product being developed
- Computer attributes
  - ♦ constraints imposed on the sw by the hw platform
- Personnel attributes
  - ♦ take the experience and capabilities of the people working on the project into account
- Project attributes
  - ♦ concerned with the particular characteristics of the sw development project

COCOMO II: multipliers (cont.)

Product attributes			
RELY	Required system reliability	DATA	Size of database used
CPLX	Complexity of system modules	RUSE	Required percentage of reusable components
DOCU	Extent of required documentation		

Computer attributes			
TIME	Execution time constraints	STOR	Memory constraints
PVOL	Volatility of development platform		

COCOMO II: multipliers (cont.)

Personnel attributes			
ACAP	Capability of project analysts	PCAP	Programmer capability
PCON	Personnel continuity	AEXP	Analyst experience in project domain
PEXP	Programmer experience in project domain	LTEX	Language and tool experience

Project attributes			
TOOL	Use of sw tools	SITE	Extent of multi-site working and quality of site communications
SCED	Development schedule compression		

COCOMO II: effects of cost drivers

Exponent value System size (including factors for reuse and requirements volatility) **Initial COCOMO II estimate without cost drivers**	1.17 128,000 DSI (Delivered Source Instructions) Variabile Size della formula **730 person-months (=2.5*128$^{1.17}$)**
Reliability (RELY) Complexity (CPLX) Memory constraint (STOR) Tool use (TOOL) Schedule (SCE) **Adjusted COCOMO estimate**	Very high, multiplier = 1.39 Very high, multiplier = 1.3 High, multiplier = 1.21 Low, multiplier = 1.12 Accelerated, multiplier = 1.29 **2306 person-months (= 730*1.39*1.3*1.21*1.12*1.29)**
Reliability (RELY) Complexity (CPLX) Memory constraint (STOR) Tool use (TOOL) Schedule (SCE) **Adjusted COCOMO estimate**	Very low, multiplier = 0.75 Very low, multiplier = 0.75 None, multiplier = 1 Very high, multiplier = 0.72 Normal, multiplier = 1 **295 person-months (= 730*0.75*0.75*1*0.72*1)**

Project planning

- Algorithmic cost models provide a basis for project planning as they allow alternative strategies to be compared
- Cost components
  - Target hardware
  - Development platform
  - Effort required

Management options

Es.

A: Usare hw, sistema di sviluppo e squadra di sviluppo preesistenti

B: Upgrade di processore e memoria (il costo dell'hw aumenta, l'esperienza diminuisce)

C: Upgrade della sola memoria (il costo dell'hw aumenta)

D: Personale più esperto

E: Nuovo sistema di sviluppo (i costi aumentano, l'esperienza diminuisce)

F: Personale con più esperienza hw

Option	RELY	STOR	TIME	TOOL	LTEX	Total effort	Sw cost	Hw cost	Total cost
A	1.39	1.06	1.11	0.86	1	63	949393	100000	1049393
B	1.39	1	1	1.12	1.22	88	1313550	120000	1402025
C	1.39	1	1.11	0.86	1	60	895653	105000	1000653
D	*1.39*	*1.06*	*1.11*	*0.86*	*0.84*	*51*	*769008*	*100000*	*897490*
E	1.39	1	1	0.72	1.22	56	844425	220000	1044159
F	1.39	1	1	1.12	0.84	57	851180	120000	1002706

Option choice

- Option D (use more experienced staff) appears to be the best alternative
- However, it has a high associated risk as experienced staff may be difficult to find
- Option C (upgrade memory) has a lower cost saving but very low risk
- Overall, the model reveals the importance of staff experience in sw development

Produttività

Definizione semplicistica di produttività di un soggetto *P* mentre produce l'oggetto *A*:

$$prod(P) = \frac{size(A)}{time\_to\_produce(A)}$$

Ad es., produttività di un programmatore: LOC / person_month

Produttività del programmatore

La definizione precedente è inaccettabile perché influenzata dai seguenti fattori:

a) definizione precisa di LOC
b) linguaggio di programmazione
c) riuso
d) qualità (sia interna, sia esterna)

Seguono alcune esemplificazioni a supporto della tesi

Produttività del programmatore (cont.)

1) Paradosso del riuso

Si supponga che, a fronte delle stesse specifiche, sia *P*, sia *P'* producano *n* LOC in t mesi-persona; poi *P'*, in un tempo ε, riusa *m* LOC (di altri programmi) che aggiunge alle precedenti.

In che relazione stanno *prod (P)* e *prod(P')*?

Secondo la definizione semplicistica di produttività data

prod(P') > prod(P) perché $\frac{n+m}{t+\varepsilon} > \frac{n}{t}$

(inoltre *t* può essere arbitrariamente grande ed ε arbitrariamente piccolo) ma, intuitivamente, ciò è falso

Produttività del programmatore (cont.)

2) Influenza della qualità

A puro titolo di esempio, si considerino le sole caratteristiche strutturali relative a coesione e accoppiamento (trascurando altre caratteristiche relative sia alla qualità interna, sia a quella esterna)

Programmatore	**LOC/mese-persona**	**Riuso**	**Livello di accoppiamento**	**Rapporto di coesione**
P	30000	20%	3.1	0.2
P’	22500	10%	1.4	0.9

Per valutare e comparare *prod (P)* e *prod(P’)* le info circa la qualità del sw prodotto sono cruciali: il primo programmatore, sebbene abbia prodotto più LOC nell’unità di tempo e sia stato abile nel riuso, ha generato sw strutturalmente povero

Metriche di prodotto

Modulo = sequenza contigua di istruzioni del programma, confinata entro opportuni delimitatori e dotata di un identificatore aggregato

La modularità di un programma si stima attraverso misure (medie) di accoppiamento e coesione dei suoi moduli

Misura di accoppiamento

Accoppiamento = grado di interdipendenza tra moduli

Misura	Tipo di accoppiamento fra due moduli
5	Sul contenuto
4	Su dati condivisi
3	Di controllo
2	Su tipi strutturati
1	Su parametri
0	Nessuno

Altre proposte di misure oggettive per l'accoppiamento tra i moduli sono:
- Il numero massimo (o medio o totale) di interconnessioni per modulo
- Il numero di moduli che accedono a interconnessioni di controllo
- Il numero di interconnessioni su strutture dati del modulo top-level

Misura di coesione

Coesione = affinità fra le diverse parti (componenti) che costituiscono un modulo, grado di necessità e utilità del fatto che esse stiano assieme, ciascuna esercitando una forza funzionale relativa, per concorrere alla realizzazione di un obiettivo comune

Misura	Tipo di coesione di un modulo	Relazione fra le diverse parti del modulo	Esempi
6	Funzionale	concorrono alla realizzazione di un'unica funzione	realizzazione di un calcolo complesso
5	Sequenziale	svolgono funzioni tali che l'uscita di una funzione costituisce l'ingresso di quella successiva	acquisizione di una misura da un impianto, sua interpretazione, elaborazione del dato interpretato e generazione di una indicazione, visualizzazione della indicazione

Misura di coesione (cont.)

Misura	Tipo di coesione di un modulo	Relazione fra le diverse parti del modulo	Esempi
4	Comunicazionale	svolgono funzioni tutte operanti su un medesimo insieme di dati	lettura, rappresentazione grafica, stampa del medesimo insieme di dati
3	Procedurale	svolgono funzioni da eseguire in un ordine prestabilito	lettura di una chiave, ricerca in un tabella, stampa del risultato
2	Temporale	svolgono funzioni tutte da attivare in un preciso intervallo temporale	funzioni di inizializzazione di un dato processo
1	Logica	svolgono funzioni simili	funzioni di comunicazione con l'esterno
0	Accidentale	nessuna	

Complessità strutturale del codice

Numero ciclomatico (McCabe 1976) = numero di cammini linearmente indipendenti nel grafo di flusso di controllo del programma (= numero dei blocchi decisionali + 1)

Per calcolarlo:
1) convertire il codice nel suo flusso di controllo
2) trasformare ogni blocco in un nodo di un grafo
3) dato il grafo così ottenuto, dalla teoria dei grafi discende che
   numero ciclomatico = numero archi – numero nodi + 2

Attenzione: la gerarchia di ereditarietà della programmazione OO ha ramificazioni che sfuggono allo studio del singolo componente isolato dal contesto