DIETI UNINA UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

Corso di Laurea in Ingegneria Informatica

Corso di Ingegneria del Software

Introduzione alla Ingegneria del Software

Sommario

- Definizioni
- Breve storia dell'ingegneria del software
- Processo e prodotto
- Fattori di qualità del software

Definizioni

Ingegneria del Software (glossario dell'IEEE): "applicazione di un approccio sistematico, disciplinato e quantificabile allo sviluppo, funzionamento e manutenzione del software".

Software: "i programmi, le procedure, l'eventuale documentazione associata e i dati relativi all'operatività di un sistema di elaborazione".

L'ingegneria del software è la disciplina tecnica, tecnologica e gestionale che riguarda la produzione sistematica e la manutenzione dei prodotti software, entro **tempi** e **costi preventivati.**

Definizioni

- L'**ingegneria del software** si occupa di definire **metodologie, modelli, tecniche** e **strumenti** utili a governare il "**Ciclo_di_Vita**" (*Software Life Cycle*, SLC) di un **prodotto software**

- **Ciclo_di_Vita:** l'insieme delle attività da svolgere per la realizzazione, il rilascio e la manutenzione di un prodotto software
  - **dall'analisi dei requisiti**, al **progetto**, allo **sviluppo**, al collaudo, fino alla **manutenzione** dopo il **rilascio**.

Ingegneria del Sw: breve storia

- Nasce alla fine degli anni '60 dall'esigenza di riguardare la produzione del software come una attività ingegneristica (alla stregua di altri prodotti dell'ingegneria, ad es. civile e industriale), anziché come una attività artigianale (com'era agli albori)
- Inizialmente, nell'uso del calcolatore i problemi da risolvere erano ben compresi e noti (ad es. risoluzione di un'equazione):
  - il problema era circoscritto tra il computer e l'***utente-programmatore*** (ad es. un fisico)
- Successivamente la diminuzione dei prezzi fa crescere il numero di utilizzatori dei calcolatori e le loro applicazioni.

Ingegneria del Sw: breve storia

- A fine anni ’50 nascono i linguaggi di alto livello e nasce la professione del **programmatore**, *che si separa dal ruolo dell’utente*
- Anni ’60: sviluppo dei primi progetti **complessi**, ad es.:
  - sistema operativo CTS (dal MIT)
  - IBM OS 360
- Lo sviluppo di sistemi complessi fa sorgere la necessità di un **nuovo tipo di approccio**:
  - Grossi progetti spesso sforavano i budget previsti
  - Problemi nell’adattare tecniche di sviluppo di piccoli programmi su grossi software
  - …

Ingegneria del SW: breve storia

- Alto numero di persone coinvolte => problemi di comunicazione e cooperazione
- Lunga durata dei progetti, cambiamento del personale => necessità di "documentare" ogni attività
- Gestione dei cambiamenti dei requisiti

Esigenza di un approccio ingegneristico

- L'ingegneria del software nasce dunque come **maturazione dell'attività di programmazione**, e dall'esigenza di un processo produttivo sistematico, che vede la costruzione e gestione di un prodotto software come **una attività ingegneristica** (alla stessa stregua di altri campi dell'ingegneria).

Ingegneria del SW: fattori socio-economici

- La diminuzione del costo dell'hardware e l'aumento di quello del software ha accentuato **l'importanza economica** dell'IS
- L'impatto economico continua a crescere
- Dal punto di vista **sociale,** l'impatto del software è evidente, ed è accentuato dalla pervasività dei sistemi software nella nostra vita quotidiana

Processo (di produzione del) software

- Come ogni tipo di prodotto, il software è sviluppato con un **processo produttivo**, che influenza fortemente la qualità del prodotto finale
- Inizialmente, c'era scarsa attenzione all'organizzazione del processo (approccio ***code-and-fix***)
- Successivamente, sono nate proposte su come organizzare il processo produttivo, partendo da una riflessione su quale fosse il ***ciclo di vita di un prodotto***

Processo, fasi, artefatti

- All'interno del ciclo di vita del prodotto software, si distinguono delle fasi che producono dei "risultati parziali" (semilavorati o **artefatti**).
- Le **fasi** del ciclo di vita sono (ad alto livello):
  - **Analisi e specifica**: si occupa del "**cosa**".
    - Determinazione dei requisiti, informazioni da elaborare, funzioni e prestazioni attese, comportamento del sistema, interfacce, vincoli progettuali, criteri di validazione.

Processo, fasi, artefatti

Progettazione e sviluppo: si occupa del ***come***

- Definizione del progetto, dell'architettura software, della strutturazione dei dati e delle interfacce e dei dettagli procedurali; traduzione del progetto nel linguaggio di programmazione; collaudi

Manutenzione: si occupa delle ***modifiche***

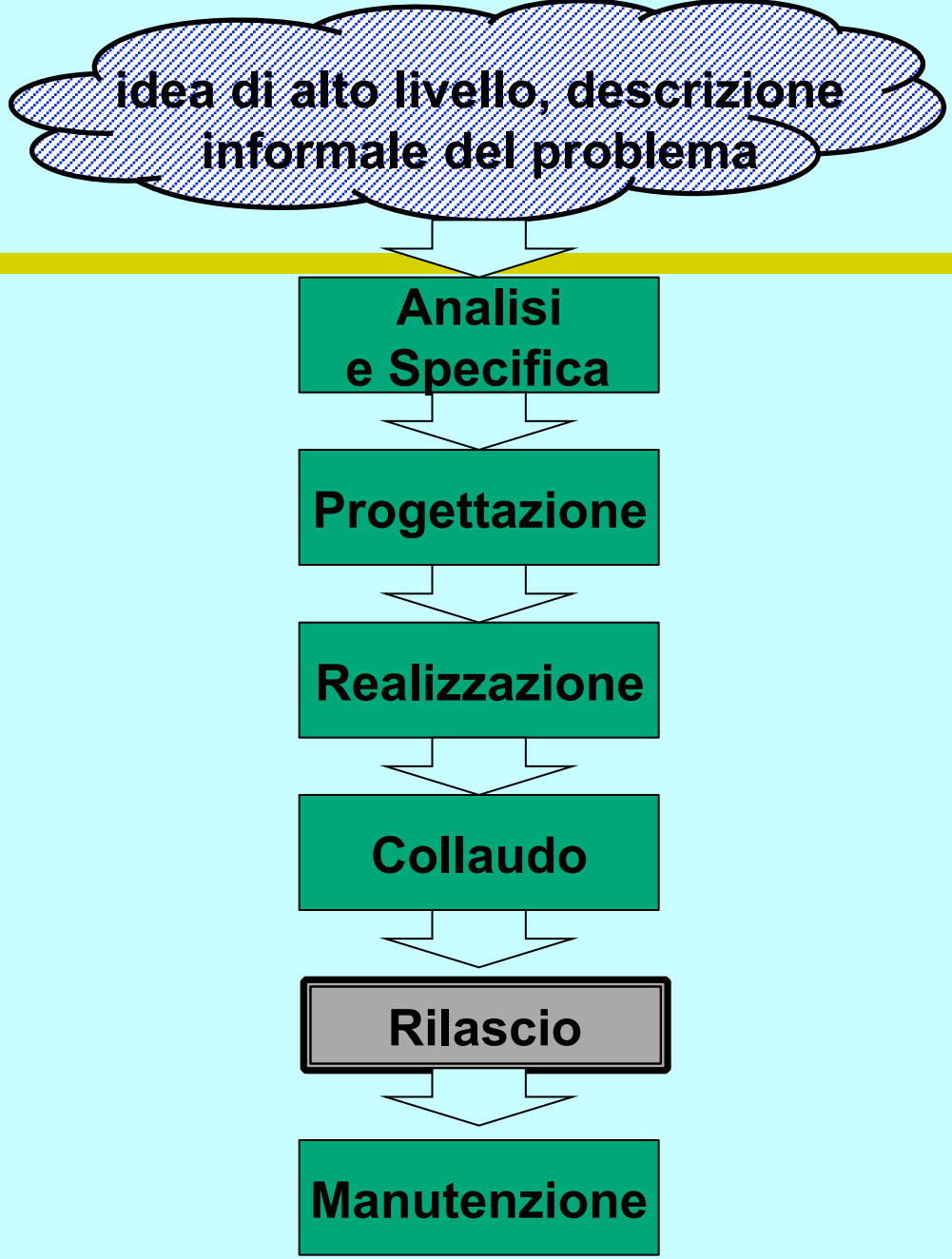
- Correzioni, adattamenti, miglioramenti, prevenzione

- A seconda di come sono organizzate, gestite ed implementate queste fasi si hanno diversi modelli di **ciclo di vita** o **processi software**

::.. Processo software

- ✓ quali sono le attività da svolgere per la realizzazione del **prodotto**?
- ✓ quali sono i **ruoli** professionali coinvolti nel processo di realizzazione?
- ✓ quali sono i manufatti **intermedi**?
- ✓ come condurre progettazione, realizzazione, collaudo, etc

PRODOTTO


idea di alto livello, descrizione informale del problema
Analisi e Specifica
Progettazione
Realizzazione
Collaudo
Rilascio
Manutenzione

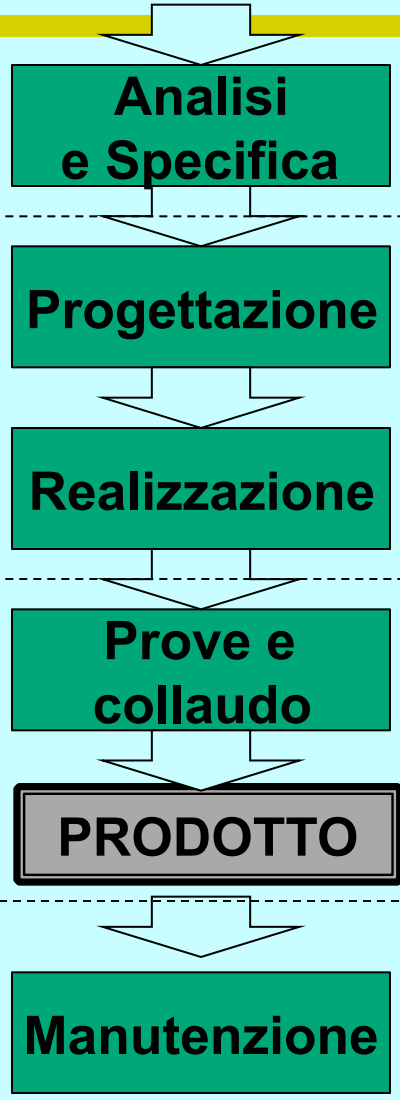
idea di alto livello,
descrizione del problema

- Comprensione del problema (analisi)
- Descrizione di "cosa" si vuole produrre (specifica)

- Costruzione della soluzione ("come" è realizzato il prodotto)

- Verifica e validazione del prodotto
- Rilascio del prodotto

- Correzione di difetti
- Adattamenti, miglioramenti
- Evoluzione (aggiunte, aggiornamenti, …)



Rilascio

- comprensione del problema;
- descrizione del problema (specifica)

Analisi e Specifica ↔ ANALISTA

Progettazione ↔ PROGETTISTA

- costruzione della soluzione

Realizzazione ↔ PROGRAMMATORE

- verifica e validazione
- rilascio del prodotto

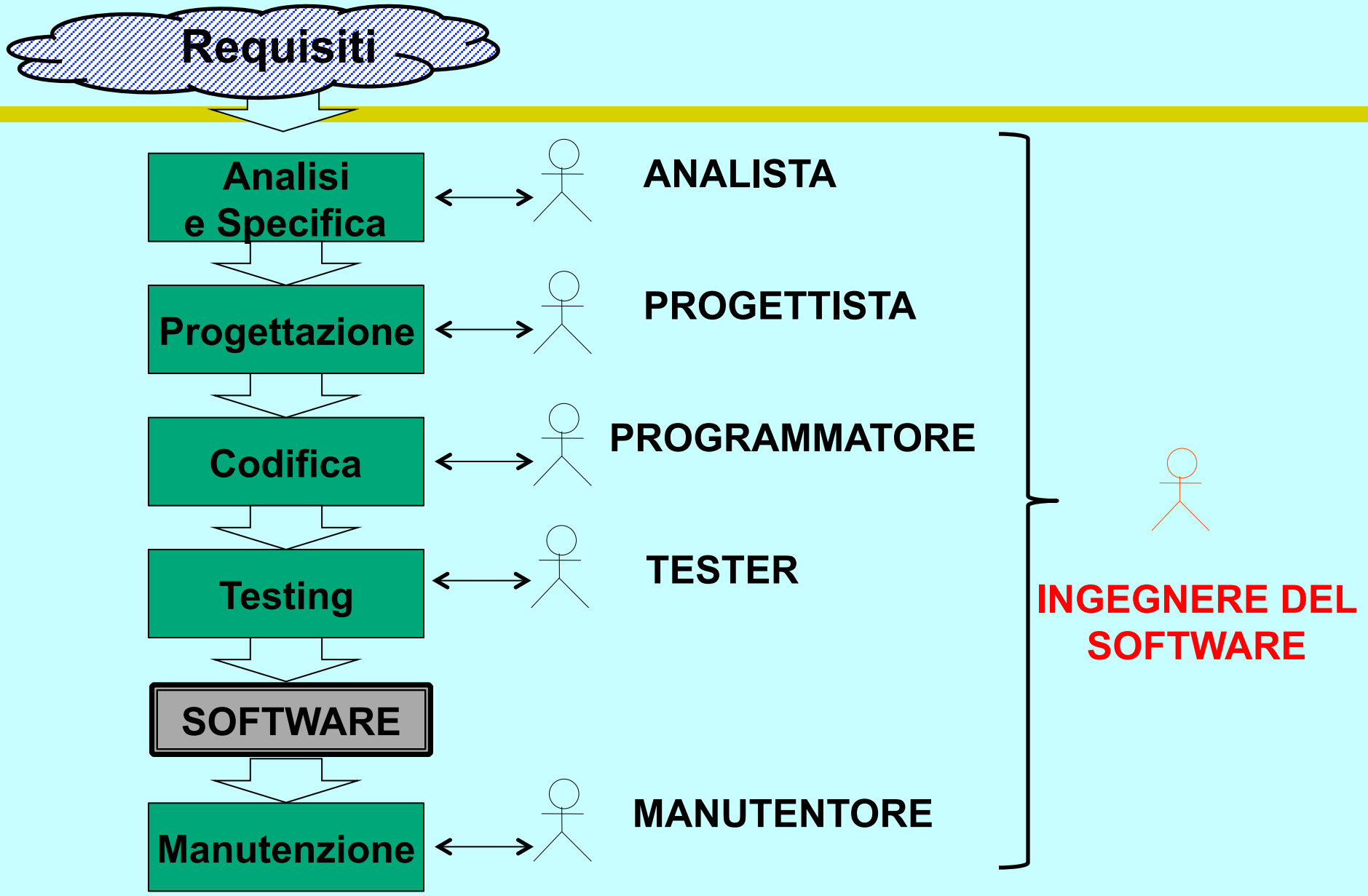
Prove e collaudo ↔ COLLAUDATORE

Rilascio

PRODOTTO

- Correzione di difetti
- Adattamenti, miglioramenti
- Evoluzione

Manutenzione ↔ MANUTENTORE


Requisiti
Analisi e Specifica
ANALISTA
Progettazione
PROGETTISTA
Codifica
PROGRAMMATORE
Testing
TESTER
SOFTWARE
Manutenzione
MANUTENTORE
INGEGNERE DEL SOFTWARE

FATTORI DI QUALITA' DEL SOFTWARE

Qualità esterne e interne
Qualità di processo e di prodotto

Due classificazioni:

Qualità esterne → **black-box:** viste dall'utente
Qualità interne → **white-box:** viste dallo sviluppatore

- Le qualità interne tipicamente determinano le qualità esterne
- Non è sempre possibile operare una distinzione marcata

Qualità del processo
Qualità del prodotto (anche i prodotto "intermedi", cioè gli artefatti, sono soggetti a requisiti di qualità)

N.B. I fattori di qualità sono spesso correlati tra di loro. Si influenzano a vicenda

Correttezza

- **Correttezza**: un software è corretto se soddisfa i *requisiti funzionali*
- Proprietà matematica che stabilisce l'**equivalenza tra il software e la sua specifica**
- Assume che le specifiche siano disponibili e non ambigue. È raro che sia così
- Tuttavia la definizione esprime un obiettivo "desiderabile"
- Metodi utilizzati per valutare e migliorare la correttezza:
  - *Testing,*
  - Verifica formale,
  - Ispezione
  - Utilizzo di processi e metodologie di provata efficacia

Affidabilità

- **Affidabilità**: probabilità che il software **operi come atteso** in un intervallo di tempo determinato
- La correttezza è una qualità assoluta (sì/no)

  → non tollera scostamenti dal servizio atteso;
- La nozione di affidabilità è invece relativa (scostamenti tollerabili)

Affidabilità vs correttezza

- **La correttezza "rientra" nell'affidabilità nell'ipotesi che la specifica dei requisiti**
  - colga tutte le proprietà desiderate, e
  - non contenga erroneamente proprietà indesiderate.

- *La figura assume che i requisiti siano corretti. Ci possono essere applicazioni corrette sviluppate sulla base di requisiti non corretti*
- Molto spesso il software soddisfa quanto specificato, ma non soddisfa pienamente esigenze/desideri dell'utilizzatore e/o del committente.

Robustezza

- **Robustezza**: un software è robusto se si comporta in maniera accettabile anche in corrispondenza di situazioni non specificate nei requisiti
  - Ad esempio, in presenza di input non corretti o inattesi
- Se si potesse definire esattamente cosa fare per rendere un'applicazione robusta, si potrebbe **specificare** tale comportamento

  => *la robustezza diventerebbe equivalente alla correttezza.*
- La linea di demarcazione tra robustezza e correttezza è pertanto **la specifica**

Prestazioni

- **Prestazioni:** qualità esterna basata sui requisiti dell'utente e sull'utilizzo efficiente delle risorse (tempo di esecuzione, memoria occupata)
- È diversa dall'**efficienza**:
  - **l'efficienza è una qualità interna** che si riferisce all'adeguato uso delle risorse da parte del software
  - le prestazioni sono legate ai requisiti utente
- L'*efficienza* influenza e spesso determina le *prestazioni*

Prestazioni

- Per valutare le prestazioni di un algoritmo si può usare la ***teoria della complessità computazionale***
- Per valutazioni più specifiche (efficienza e/o prestazioni su uno specifico sistema di elaborazione), vi sono tre approcci:
  - ***Misuristico***: misurazioni su sistemi reali
  - ***Analitico (model-based)***: costruzione di un modello analitico (ad es. basato su teoria delle code). Utile in fase di progettazione (sistema non ancora implementato)
  - ***Simulativo***: costruzione ed esecuzione di un modello simulativo
- Le varie tecniche sono complementari

Usabilità

- **Usabilità**: un sistema è usabile se i suoi utenti lo reputano facile da utilizzare
- È una **qualità soggettiva**
- Le interfacce utente influiscono molto sulla "amichevolezza" (*user frendliness*) e dunque sull'usabilità
- Studiata molto dalla disciplina *Human-Computer Interaction* (HCI) che si occupa dei "fattori umani" (*human factors*)

Manutenibilità

- **Manutenibilità**: facilità con cui le attività di manutenzione vengono eseguite ed economicità dei relativi processi
- *Caratteristica importante:* la manutenzione supera il 60% dei costi totali del software

***Tipi di manutenazione**:*

Correttiva: per eliminare errori residui (circa il 20% dei costi di manutenzione)

Adattativa: modifiche da effettuare per adattare l'applicazione a cambiamenti dell'ambiente (circa il 20% dei costi)

Perfettiva: per eliminare, aggiungere e modificare alcune caratteristiche o funzionalità (circa il 50% dei costi)

Manutenibilità

- La manutenibilità può essere vista come l'insieme di:
  - **Riparabilità**: facilita la correzione dei difetti
    - Una corretta strutturazione in moduli favorisce la riparabilità
  - **Evolvibilità**: facilita cambiamenti a fronte di nuovi requisiti
    - Richiede capacità di **anticipare I cambiamenti** in fase di progettazione
    - Tende a diminuire con i rilasci successivi del prodotto (diventa più complicato apportare modifiche => aumenta il rischio di introdurre nuovi errori)

Manutenibilità

- **Riusabilità**: la possibilità di re-impiegare moduli esistenti nella realizzazione di nuovi prodotti
- **Portabilità:** se può essere eseguito in ambienti diversi
- **Comprensibilità**: gli artefatti devono essere chiari e comprensibili al fine di verificarne la correttezza, di apportare modifiche, e ai fini del riuso

Interoperabilità

- **Interoperabilità**: capacità di coesistere e cooperare con altri sistemi.
  - Un concetto correlato è quello di "**sistema aperto**" (*open system*)
    - Tipicamente basato su standard
    - Il termine aperto è in contrapposizione a chiuso, cioè di tipo proprietario

Produttività

- **Produttività**: fattore di qualità del processo di produzione, che ne indica le prestazioni e l'efficienza

Dependability

- Per i **sistemi critici** si enfatizzano altri attributi di qualità
- Si parla di ***dependability***, macroattributo costituito da:
  - **Affidabilità**
  - **Manutenibilità**
  - **Disponibilità**: probabilità che il software operi come atteso in un dato *istante* di tempo
  - ***Safety***: assenza di conseguenze catastrofiche su utenti e ambiente
  - **Integrità**: assenza di alterazioni improprie

Ulteriori punti di vista

- **Confidenzialità:** *assenza di diffusione non autorizzata di informazioni*
  - Insieme alla disponibilità e alla integrità compone l'attributo di **sicurezza**:

    Sicurezza = Disponibilità + Integrità + Confidenzialità

- Non sempre c'è completo accordo sulle definizioni degli attributi di qualità del software
- Ad esempio, il **modello di qualità dello standard ISO 9126** adotta definizioni e punti di vista parzialmente diversi.