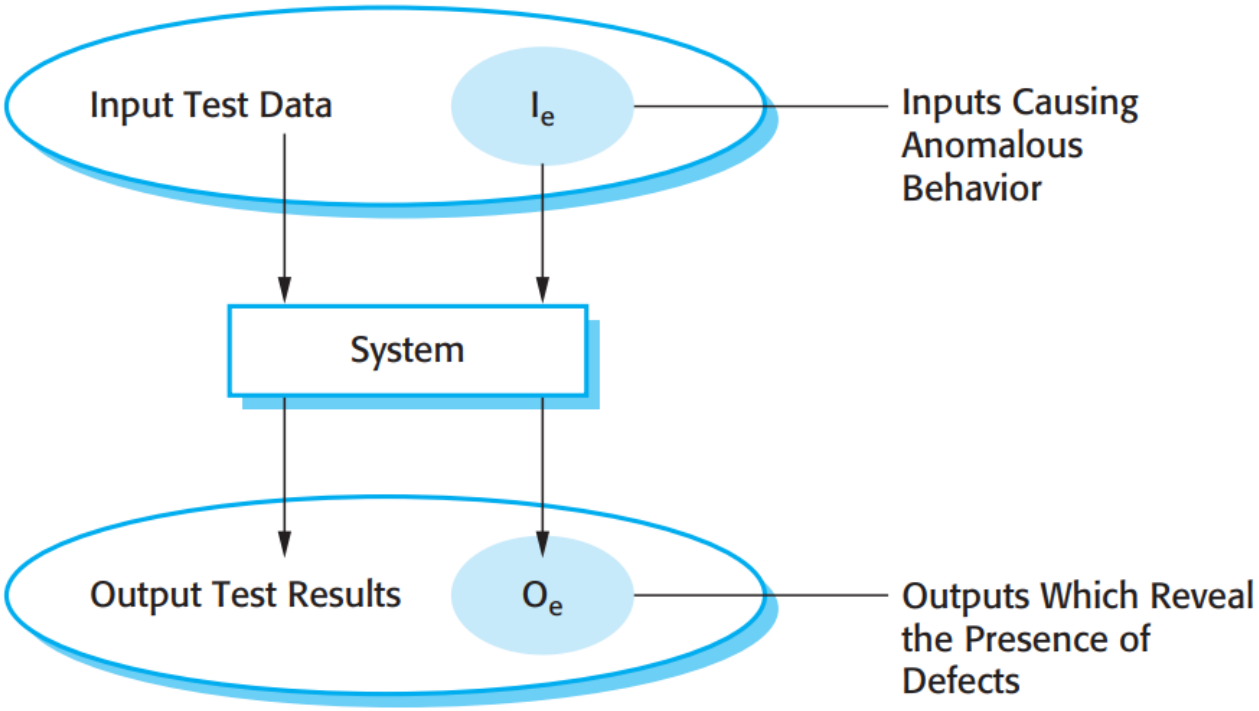
Verifica e Validazione

Testing

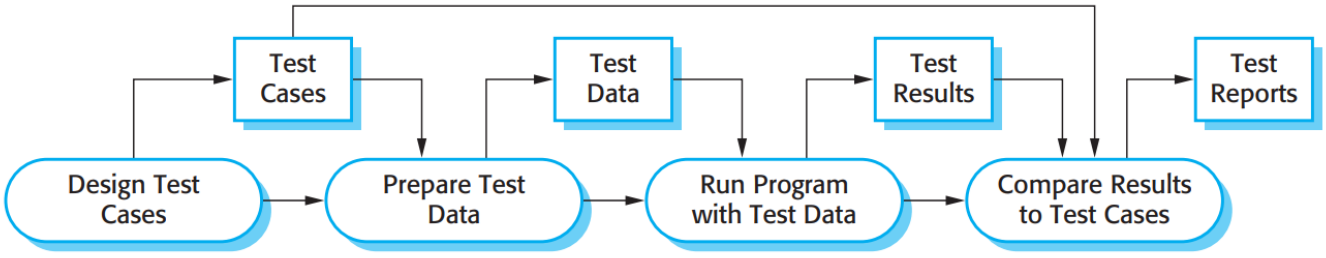
- Serve a trovare dei comportamenti errati ragionando nel modo corretto (cercando la presenza di errori e non la loro assenza)



- Fare un software "fit per purpose" = soddisfi sia i requisiti utente che i requisiti azienda/proponente

Verifica

- Si agisce su singoli pezzi utilizzando il metodo di verifica già collaudato e funzionante (in modo oggettivo = qualità) nei passi precedenti



- *Did I build the right system?*

Quanto fatto deve corrispondere alle aspettative temporali e pratiche:

- Milestone: Date di calendario/Eventi che sono rispettati in modo atteso
  - Definire una serie di milestone utili in base al contesto
  - Introdurne "ad alto livello"
- Baseline = Approvazione di un insieme di passi evidenti (insieme di milestone) visibili in un prodotto finito

Analisi statica

- Non richiede esecuzione dell'oggetto di verifica
- Se definisco bene delle metriche, sono in grado di confermare che una cosa funzioni bene durante tutte le fasi
- Impiega dei metodi formali o dei metodi semplici
- Voglio ottenere dei comportamenti predicibili che sono in grado di tracciare (efficacia ed efficienza)

Esempi di analisi statica:

- Flusso di controllo
- Flusso dei dati
- Flusso delle informazioni
- Limite del programma
- Uso dello stack

Metodi di lettura

- Possono essere sia umani/automatizzati
- Coinvolgono i verificatori e i programmatori

Walkthrough

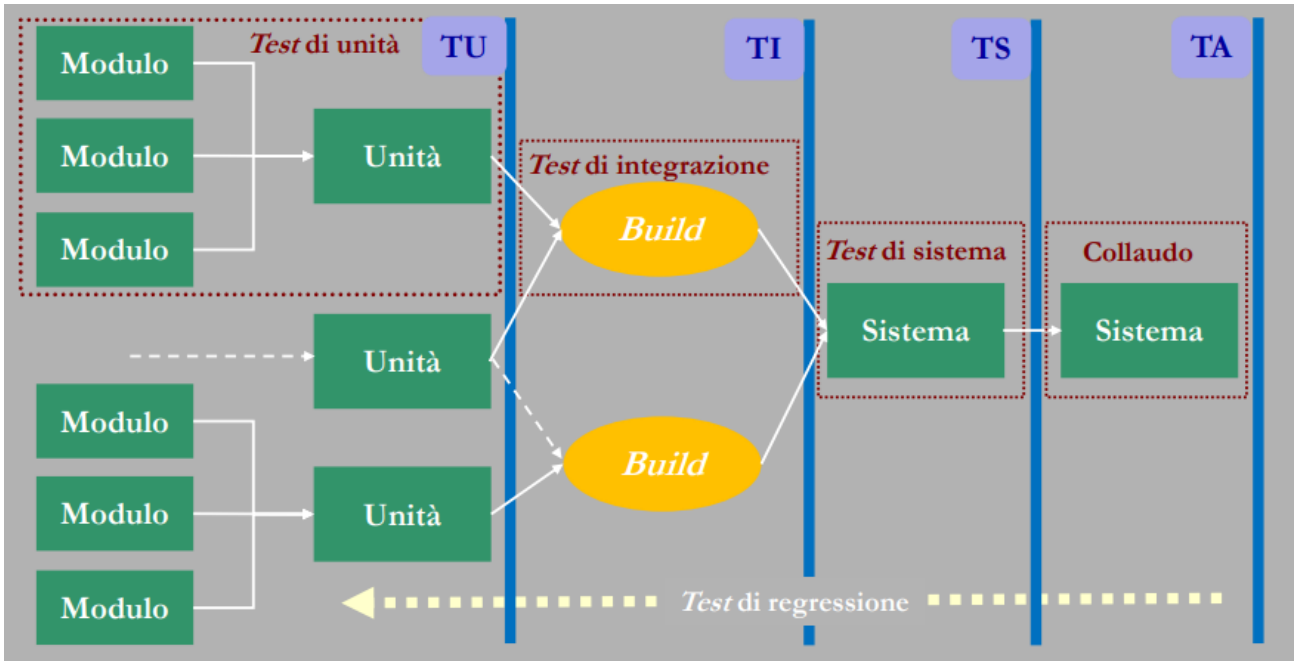
- Lettura ad ampio spettro da parte di tutti in modo simultaneo
- Visione di insieme
- Ci vuole una pianificazione adeguata per simulare questa cosa e poi corregge i difetti nell'insieme

Inspection

- Rilevo la presenza di difetti "singoli" sulla base di una checklist
- Correzione dei difetti più a grana fine (fine-grained)

Analisi dinamica

- Esegui l'oggetto di verifica
- Si usano dei test (a tutti i livelli)
  - Unit
  - Component
  - System
- Oggetti ripetibili e visibili da un punto di vista pratico
  - Driver = L'oggetto che inizia i test (top-down/bottom-up a tutte le componenti)
  - Stub = Singolo oggetto che viene testato
  - Logger = Ne salvo i risultati
- Agisce a vari livelli di dettaglio:



Analizziamo i vari livelli di dettaglio:

- modulo = pezzi piccoli di codice riuniti all'interno di unità
- unità = raggruppano un insieme di moduli
- componente = insieme di unità e di moduli

```
class Pippo{ # unit to test
```

```
        fun A{} # module 1
        fun B{} # module 2

}

package Disney; # component
```

Tipi di test

Unità

- Pezzi atomicamente indipendenti
- Parti singole autonome testabili

Regressione

- Tornare indietro verso le unità, sapendo che il sistema (macroscopicamente = ad alto livello) è stato fatto benbe
- Architettura

Integrazione

- Mette insieme più unità

Sistema

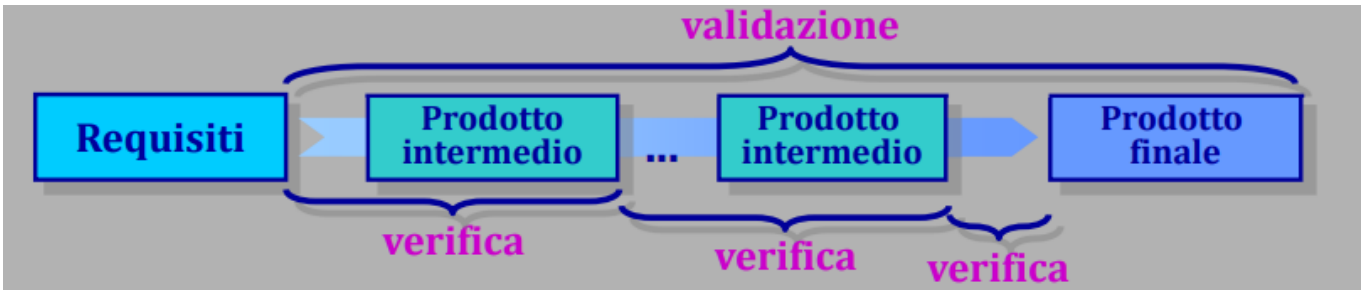
- Test a livello di tutto ciò che riguarda una certa architettura/progetto

Accettazione

- Test che verifica che il collaudo funzioni bene per il cliente

Validazione

- Controlla che quanto prodotto incrementalmente soddisfi le aspettative



- *Did I build the system right?*

Riferimenti

- Mantenimento qualità = Quality Assurance (ISO 12207 - Software Lifecycle)
- V Model