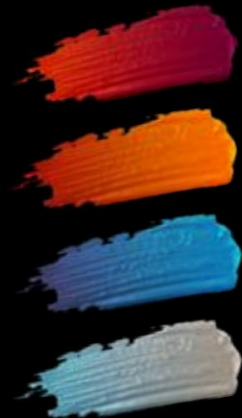


# Integrazione e Test di Sistemi Software

## **Test del software: definizioni preliminari**

Azzurra Ragone

Dipartimento di Informatica - Università degli Studi di Bari  
Via Orabona, 4 - 70125 - Bari  
Tel: +39.080.5443270 | Fax: +39.080.5442536  
[serlab.di.uniba.it](http://serlab.di.uniba.it)



# Test del software

Processo costituito da qualsiasi attività relativa alla pianificazione, preparazione e valutazione di prodotti software per: 1.

Determinare che soddisfano i requisiti specificati 2. Dimostrare che sono adatti allo scopo 3. Rilevare difetti



# Terminologia

- Nella letteratura sui test è presente una terminologia confusa
  - Ad esempio, gli errori vengono **erroneamente** utilizzati come sinonimi di bug
- L'International Software Testing Qualification Board (ISTQB) ha fornito un ampio glossario di termini di test
- <http://glossary.istqb.org/>
- La terminologia ISTQB è compatibile con quella dell'Institute of Electronics and Electrical Engineers (IEEE) Computer Society



# Terminologia

**Errore:** le persone commettono errori (cioè, sbagli)

Ad esempio, quando le persone commettono errori durante la codifica, chiamiamo il risultato di quegli errori bug

**Bug (detto anche difetto o guasto):** risultato di un errore

Più precisamente, un bug è la rappresentazione di un errore nel testo narrativo, nei diagrammi UML, nel codice sorgente, ecc.

**Errore:** si verifica quando viene eseguito il codice corrispondente a un bug

**Incidente:** sintomo che si è verificato un guasto. Qualcosa che merita un'indagine.



# Terminologia

**Test:** l'atto di esercitare un sistema software con un caso di test (o più casi di test)

Il suo obiettivo è quello di rompere il sistema o dimostrarne la corretta esecuzione

**Caso di prova:** un prodotto di lavoro riconosciuto

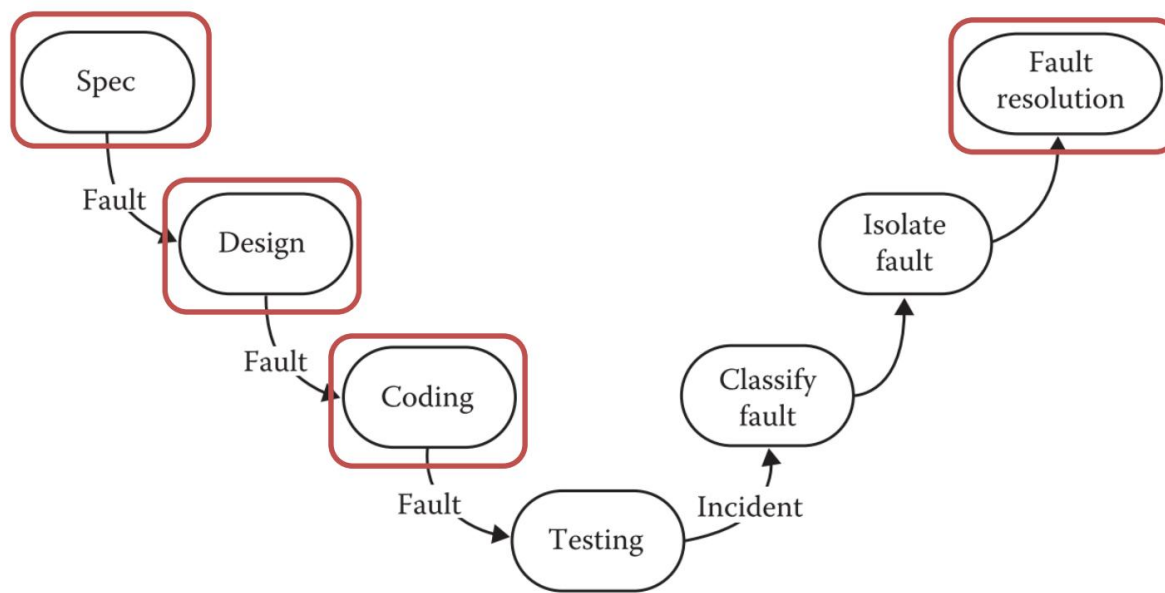
È associato a un comportamento del sistema.

Ha un insieme di input e output previsti (ad esempio, **oracolo**).

**Suite di test:** insieme di casi di test



# Modello del ciclo di vita per i test



Nelle fasi di sviluppo (lato sinistro), si presentano tre opportunità per introdurre bug  
La risoluzione dei guasti rappresenta un'altra opportunità per introdurre bug:  
modificare un programma esistente è un processo più soggetto a errori rispetto alla  
scrittura di un nuovo programma.



# Caso di prova

- Una descrizione completa del caso di test contiene:
  - Un identificatore
  - Una descrizione del caso di test (dichiarazione di intenti)
  - Una descrizione delle precondizioni
  - Input
  - Output previsti
  - Una descrizione delle postcondizioni previste
  - Cronologia di esecuzione
  - Data della sua esecuzione
  - Persone che lo hanno eseguito
  - Versione del sistema
  - Risultato superato/fallito







# Caso di prova

La sua esecuzione

comporta: 1. Stabilire le premesse necessarie (se presenti)

2. Fornire input per i casi di test

3. Osservare gli output

4. Confrontare gli output osservati con quelli previsti

5. Assicurarsi che esistano le postcondizioni previste (se presenti)



## Compiti 0

- Trova un buon modello per i casi di test
- Esistono diversi modelli a seconda del tipo di test che stiamo eseguendo (Unità, Integrazione, Sistema)



# Identificazione dei casi di prova

Il **dominio di input** di un sistema è costituito da tutti i possibili input di quel sistema

Anche per un sistema di piccole dimensioni, il dominio di input è così grande che è praticamente **impossibile** testare tutti gli input

Per identificare i casi di prova (ovvero per selezionare un insieme finito di valori dal dominio di input con cui esercitare il sistema), si possono seguire due strategie principali:

Test **funzionali/basati sulle specifiche**

Test **strutturali/basati sul codice**

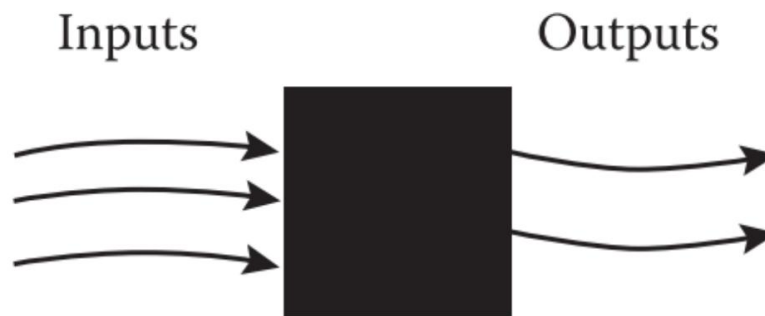


# Test basati sulle specifiche (1/2)

I sistemi sono considerati scatole nere

Il contenuto (implementazione) della scatola nera non è noto ma la funzione della scatola nera è completamente compresa in termini di input e output

È anche noto come **test della scatola nera** (ad esempio, alla guida di un'auto)



# Test basati sulle specifiche (2/2)

L'unica informazione utilizzata è la specifica del software

PIÙ

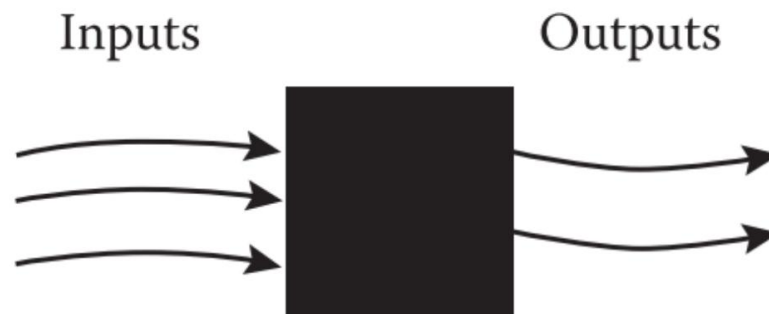
- I test sono indipendenti da come viene implementato il software (se cambia l'implementazione, non dobbiamo cambiare i test)

- Lo sviluppo e l'implementazione dei casi di test possono avvenire in parallelo

MENO

- Lacune del software non testato

- Licenziamenti



# Test strutturali (basati su codice)

In questo caso, l'implementazione della casella è nota e utilizzata per identificare i casi di test

Vedere all'interno della casella consente al tester di identificare i casi di test in base al modo in cui la funzione è effettivamente implementata

È anche noto come **test white-box** o **test basato sul codice**

Le metriche **di copertura dei test** indicano esplicitamente in che misura un elemento software è stato testato



# Test basati sulle specifiche vs test strutturali

Nessun approccio, di per sé, è sufficiente (ricordiamo *il paradosso dei pesticidi*):

- 1) Se alcuni requisiti specifici non sono stati implementati  
i test strutturali (basati sul codice) non saranno mai in grado di riconoscerlo
- 2) Se il programma implementa comportamenti che non sono stati specificati,  
questo non verrà mai rivelato dai casi di test basati sulle specifiche

Hai bisogno di una combinazione dei due approcci



# Errore e guasto

Errori e difetti sono legati alla distinzione tra processo e prodotto: -  
**il processo**

si riferisce al modo in cui facciamo qualcosa

- **il prodotto** è il risultato finale di un processo

La garanzia della qualità del software (SQA) cerca di migliorare il prodotto migliorando il processo, riducendo gli errori endemici nel processo di sviluppo

Il test è orientato al prodotto in quanto si occupa di scoprire i difetti in un prodotto



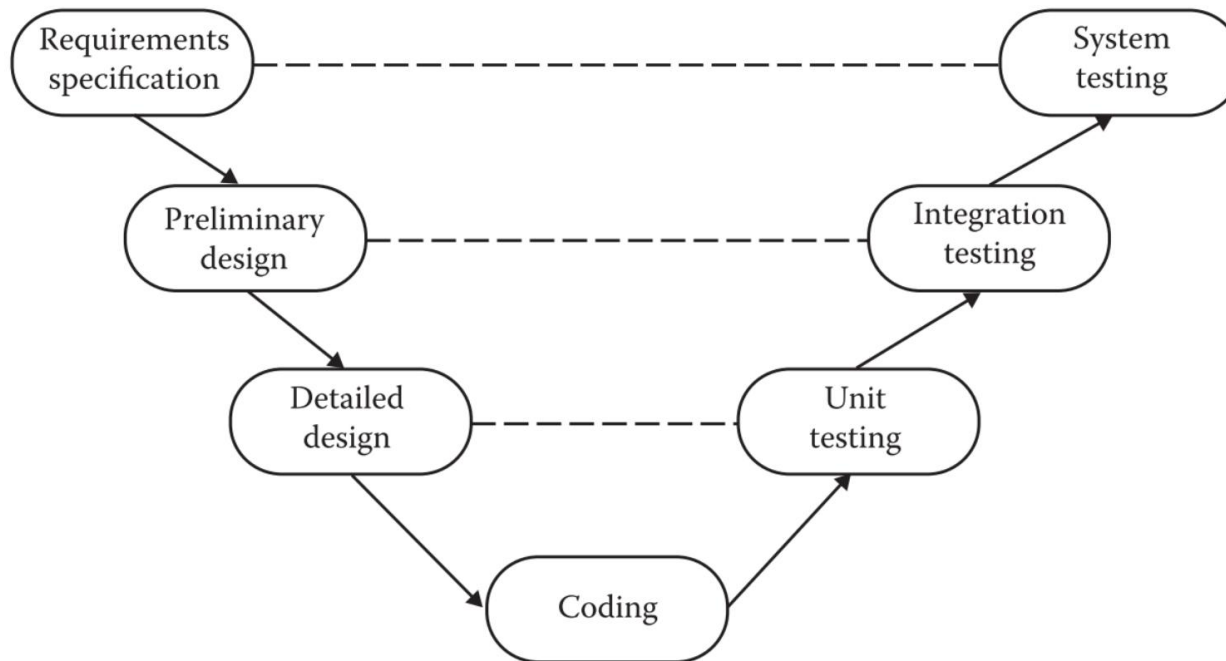


# Livelli di test (modello V)

Test di unità, integrazione e sistema

Il test di **accettazione** è considerato il quarto livello di test

Riecheggiano le fasi del modello a cascata

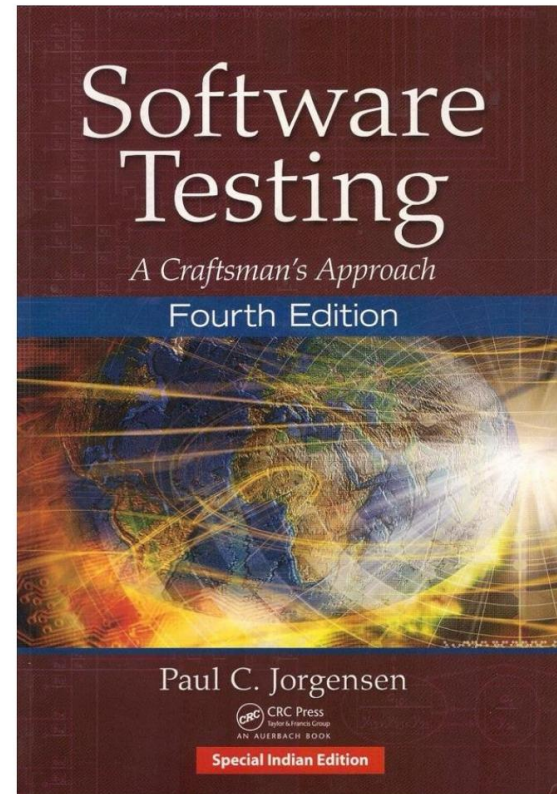


Diapositiva per gentile concessione di Simone Romano



# Libro di riferimento:

Test del software (un approccio artigianale). Paul C. Jorgensen. Ed. CRC Press (pdf disponibile online) – **Capitolo 1 – pdf disponibile online**



L





Azzurra Ragone

Dipartimento di Informatica - Piano VI - Stanza 616 Email:  
azzurra.ragone@uniba.it