UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

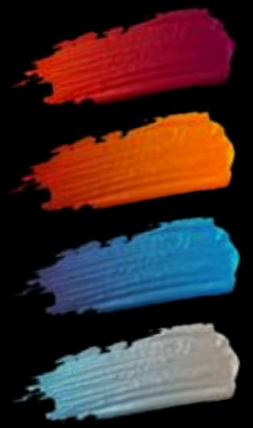
SERLAB
Software Engineering Research

Integrazione e Test di Sistemi Software

Test statici

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



Che cosa sono i test statici?

Il test statico è un tipo di software
Metodo di test eseguito per verificare
i difetti nel software senza effettivamente
eseguire il codice dell'applicazione (come in Dyn
(Test)

Test statici come «test umani»

I test statici (noti anche come revisioni del software) possono essere considerati **" test umani".**

Perché? • **La lettura del codice** fa parte di un test completo.

• Il test del software cerca di identificare i guasti causando guasti. • Alcuni tipi di revisioni del software (ad esempio, l'ispezione del codice) cercano di identificare gli errori che, una volta eseguiti, causano guasti.

Quando?

• Prima di iniziare il test basato sul computer.

Test nello sviluppo del software

Test statici:

Non richiede l'esecuzione del codice

Può essere eseguito manualmente o tramite un set di strumenti

contro

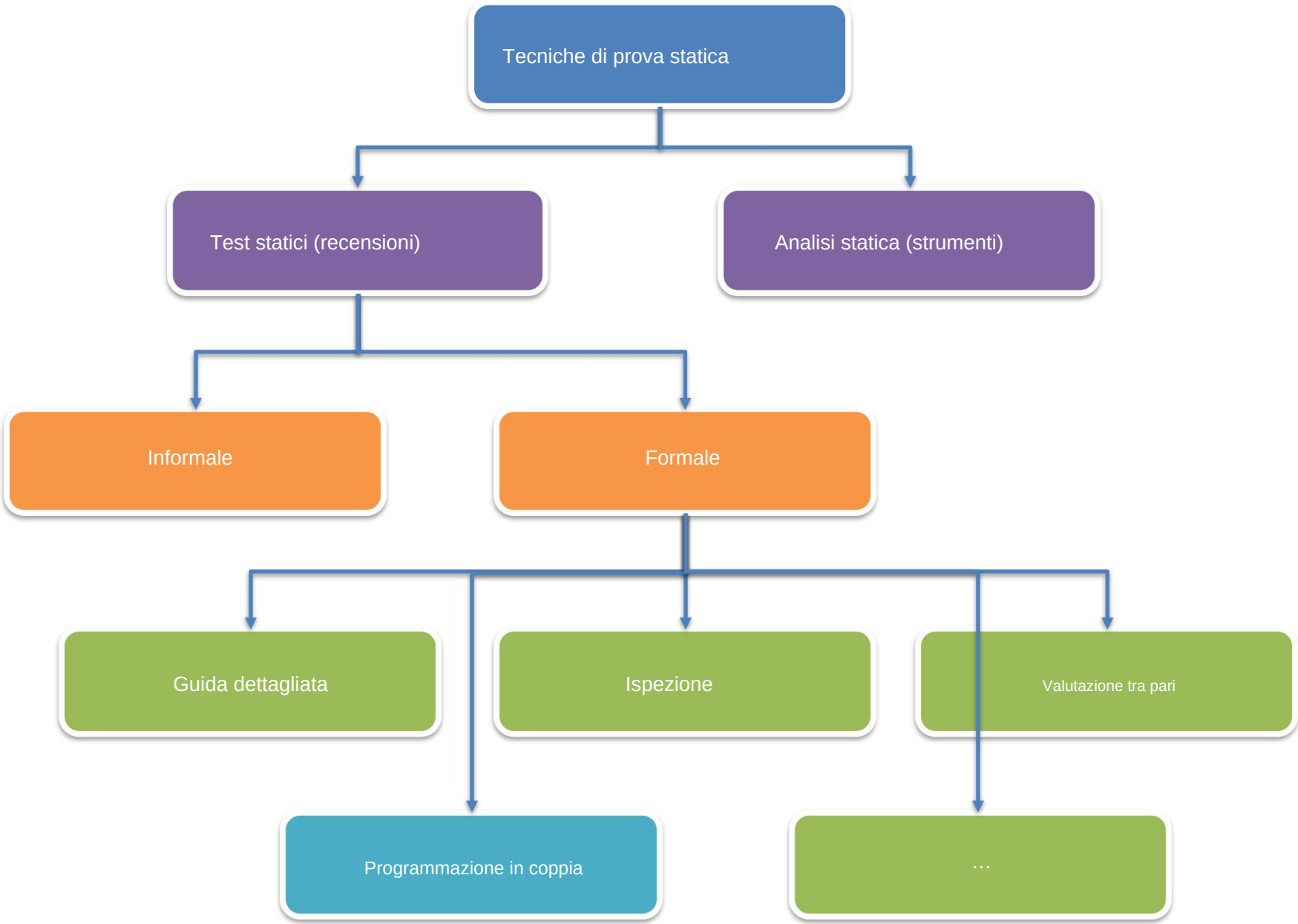
Test dinamici:

Comporta l'esecuzione del software e la valutazione del suo comportamento durante il runtime (ad esempio test unitari, di integrazione, di sistema e di accettazione)

Vantaggi della «sperimentazione umana»

Gli errori vengono individuati prima (riduzione dei costi)

All'inizio c'è meno pressione sui programmatori: i programmatori commettono più errori se un errore viene trovato in seguito



Analisi statica

L'analisi statica include la valutazione della qualità del codice. Diversi strumenti vengono utilizzati per analizzare il codice e confrontarlo con lo standard. Aiuta anche a identificare i seguenti difetti:

(a) Variabili non utilizzate (b) Codice morto (c) Cicli infiniti (d) Variabile con valore non definito (e) Sintassi errata

Perché i test statici (revisioni)?

K. Wiegers [1995] riferisce che, in un'azienda non nominata, correggere un guasto trovato tramite test costava **15 volte di più** rispetto a riparare un guasto trovato durante un'ispezione (un esempio di test statico), e questo costo aumentava fino a **68 volte** se il guasto veniva segnalato da un cliente.

K. Wiegers [1995] riferisce anche che, in un'altra azienda non nominata, il costo per riparare un guasto riscontrato tramite ispezione era di **$ 146** mentre il costo per riparare un guasto riscontrato dal cliente era **di $ 2900**

Tipi di recensioni di software

Ispezione del codice

Guida al codice

Controllo della scrivania

Valutazione tra pari

Revisione contabile

Programmazione in coppia

Diapositiva per gentile concessione di Simone Romano

Recensione del software

Un test durante il quale un prodotto di lavoro (ad esempio, un pezzo di **documentazione** o **codice)** viene valutato da una o più persone per rilevare **problemi** come code smell, errori, ecc. (ovviamente, siamo interessati a rilevare gli errori)

Due approcci principali alle revisioni del codice:

- **ispezione del codice**
- **Guida al codice**

Ispezioni e verifiche del codice

Possono essere utilizzati praticamente in qualsiasi fase dello sviluppo del software, dopo che un sistema o un componente/unità è considerato completo

Entrambi coinvolgono un team di persone che leggono o ispezionano visivamente il codice

Questo team partecipa ad una riunione con l' **obiettivo di trovare problemi** ma **non trovare soluzioni** a questi problemi (è un test, non un debug!)

Ispezioni e verifiche del codice

Solo un partecipante è l'autore del programma (è inefficace testare il nostro programma)

Questi metodi di test umani sono generalmente efficaci nel trovare dal 30 al 70 percento della progettazione logica e della codifica errori nei programmi tipici.

Tuttavia, non sono efficaci nel rilevare errori di progettazione di alto livello, come errori commessi nei requisiti processo di analisi.

Ispezioni del codice

Un'ispezione del codice è un insieme di procedure e tecniche di rilevamento dei problemi per la lettura del codice di gruppo

Peculiare delle ispezioni del codice è l'uso di **liste di controllo dei problemi**

Consiste in una riunione interrotta, che dovrebbe durare dai 90 ai 120 minuti

La velocità del codice valutato è solitamente di 150 istruzioni all'ora

Se un sistema software è troppo grande per una riunione di ispezione del codice di 90-120 minuti, è possibile pianificare più riunioni di ispezione del codice

Squadra di ispezione del codice

Di solito è composto da quattro persone:

Il **moderatore:**

Ci si aspetta che sia un programmatore competente, ma non è l'autore del codice da valutare e non è necessario che sia a conoscenza di tutti i dettagli del sistema

I compiti del moderatore includono:

- Distribuzione dei materiali per la riunione di ispezione e programmazione
- Guidare la riunione
- Registrazione di tutti i problemi riscontrati
- Garantire che i problemi vengano successivamente risolti

Il **produttore, ovvero l'** autore del codice da valutare

Le altre due persone sono solitamente il **progettista** del sistema e uno **specialista dei test: lui/lei** dovrebbe avere familiarità con i problemi di programmazione più comuni (non solo con gli errori)

Ordine del giorno dell'ispezione del codice

Alcuni giorni prima della riunione di ispezione, il moderatore distribuisce del materiale (ad esempio, l'elenco dei programmi e le specifiche di progettazione) agli altri partecipanti, che dovrebbero familiarizzare con tale materiale prima della riunione.

Durante la riunione di ispezione si svolgono due attività:

1. Il **produttore** narra, affermazione per affermazione, la logica del codice

   Durante il discorso, gli altri partecipanti dovrebbero porre domande per determinare se esistono problemi

   È probabile che il produttore, piuttosto che gli altri membri del team, trovi la maggior parte dei problemi solo perché sta leggendo ad alta voce il codice a un pubblico

2. Il codice viene analizzato rispetto ad una checklist di elementi comuni problemi di programmazione

Ordine del giorno dell'ispezione del codice

Il moderatore guida la discussione durante la riunione e si assicura che i partecipanti concentrino la loro attenzione sulla **ricerca problemi, non correggerli**

- L'obiettivo è trovare i problemi!

Alla fine dell'incontro, al produttore viene consegnato un **elenco dei problemi emersi,** che dovrà risolvere dopo l'incontro.

Se vengono scoperti altri problemi o potrebbero richiedere correzioni sostanziali, il moderatore potrebbe pianificare un'altra riunione di ispezione per rivalutare il codice

L'elenco dei problemi viene utilizzato per **aggiornare la checklist dei problemi** per migliorare l'efficacia delle ispezioni future

Lista di controllo dei problemi

Ogni azienda di software ha la sua lista di controllo dei problemi

Alcuni problemi riguardano errori comuni, altri possono riguardare problemi di leggibilità, ecc.

Le checklist dei problemi solitamente hanno delle categorie

Esempio di checklist:

Il ciclo tutti i file sono stati chiusi dopo l'uso?

Will alla fine terminerà? Ogni variabile assegnata è la corretta lunghezza e tipo di dati?

I commenti sono valori accurati e significativo?

Fa un loop, la variabile di riferimento non inizializzati?

In ci sono troppo profondi o poche iterazioni?

...

Aspetti umani nelle ispezioni del codice

Se il produttore considera l'ispezione come un attacco nei suoi confronti e adotta un atteggiamento difensivo, il processo sarà inefficace.

I risultati di un'ispezione dovrebbero essere riservati e condivisi solo tra i partecipanti

Per esempio, È quei risultati che non si ottengono presuppongono un programmatore insufficiente/incompetente

Vantaggi collaterali delle ispezioni del codice

Il produttore riceve **un feedback** prezioso riguardo programmazione stile, scelta degli algoritmi e tecniche di

Ogni partecipante può acquisire conoscenze dalle ispezioni del codice

Ad esempio, stili di programmazione, problemi di programmazione comuni, ecc.

Poiché il codice di un produttore è oggetto di valutazione, egli è **incoraggiato a scrivere codice pulito** (ad esempio, codice leggibile)

L'ispezione favorisce **il comportamento cooperativo** e l'identificazione delle **sezioni del programma più soggette a errori** (maggiore attenzione a queste sezioni sarà dedicata nella fase di test di automazione)

Guide dettagliate al codice

Come un'ispezione del codice, un'analisi dettagliata del codice è un insieme di procedure e tecniche di rilevamento dei problemi per la lettura del codice di gruppo

Tuttavia, **peculiare** delle procedure dettagliate del codice è l'uso di **casi di test cartacei**

Consiste in una riunione interrotta, che dovrebbe durare da 1 a 2 ore

Team di analisi del codice

Di solito è composto da tre o cinque persone:

- Il **moderatore**
- Il **segretario**
  Lui/lei registra tutti i problemi riscontrati
- Il **tester**

Le altre due persone possono essere:

- Il **produttore**
- Un programmatore altamente esperto
- La persona che alla fine manterrà il sistema
- Un nuovo programmatore da un progetto diverso (per offrire una prospettiva nuova e imparziale)
- Uno sviluppatore alle prime armi
- …

Programma della guida al codice

Come in un'ispezione, il moderatore fornisce agli altri partecipanti il materiale con diversi giorni di anticipo, per consentire loro di familiarizzare con esso

Durante la riunione di ispezione:

1. I partecipanti “giocano al computer”
2. Il tester porta con sé un piccolo set di casi di test cartacei--- insiemi rappresentativi di input (e output previsti) per il sistema (o componente) da valutare
3. Ogni caso di prova viene eseguito mentalmente
   Cioè, i dati di prova vengono "guidati" attraverso la logica del sistema
4. Lo stato del sistema (ovvero i valori delle variabili) viene monitorato su carta o su una lavagna

Programma della guida al codice

I casi di test cartacei devono essere semplici e pochi in numero

- Le persone eseguono i sistemi a una velocità molto più lenta di una macchina
- I casi di test in sé non svolgono un ruolo critico
  - Sono un mezzo per iniziare e per interrogare il produttore sulla sua logica e sulle sue ipotesi

Nella maggior parte delle procedure dettagliate, la maggior parte dei problemi vengono rilevati durante il processo di interrogazione del produttore, piuttosto che eseguendo i casi di test

Aspetti umani nelle esercitazioni di codice e vantaggi collat

Aspetti umani:

Come in un'ispezione del codice, l'atteggiamento dei partecipanti è fondamentale

Ad esempio, se il produttore vede la procedura dettagliata come un attacco a lui/lei e adotta una posizione difensiva, il processo sarà inefficace.

Benefici collaterali:

Lo stesso dell'ispezione del codice

Ad esempio, i partecipanti possono acquisire conoscenze tramite la guida dettagliata al codice (identificazione e istruzione di sezioni soggette a errori in errori, stile e tecniche)

Controllo della scrivania

Può essere visto come un'ispezione da parte di una sola persona

- Una persona legge il codice da valutare, lo controlla rispetto a un elenco di problemi e/o lo esamina attraverso i dati di test

Per essere efficace, il controllo della scrivania dovrebbe essere eseguito da una persona diversa dal produttore

- Le persone sono generalmente inefficaci nel testare i propri programmi

Meno efficace dell'ispezione del codice e della procedura dettagliata

- Nell'ispezione e nella verifica del codice si promuove un **ambiente sano e competitivo** in cui i partecipanti cercano di dimostrare la loro **capacità di identificare i problemi**
- Un ambiente del genere non è presente nel desk checking

Valutazione tra pari

Si tratta di un tipo di revisione il cui obiettivo non è quello di trovare errori

È una tecnica di **valutazione dei programmi anonimi** in termini di loro efficacia complessiva qualità, manutenibilità, estensibilità, usabilità e chiarezza

Un programmatore viene selezionato per svolgere il ruolo di amministratore del processo

L'amministratore, a sua volta, seleziona circa 6-20 partecipanti (ovvero programmatori con background simili, ad esempio, non raggruppare i programmatori Java con i programmatori C)

Valutazione tra pari

- A ciascun partecipante viene chiesto di selezionare due dei suoi programmi da recensire
  - Un programma dovrebbe essere rappresentativo di ciò che il partecipante considera il suo miglior lavoro
  - L'altro dovrebbe essere un programma che il programmatore considera di qualità inferiore
- Una volta raccolti i programmi, vengono distribuiti in modo casuale ai partecipanti
  - A ciascun partecipante vengono dati quattro programmi da rivedere, ma non riceve i suoi programmi.
  - Due dei programmi sono i programmi "migliori" e due sono i programmi "peggiori"---al partecipante non viene detto quale sia l'uno e quale l'altro

Valutazione tra pari

Ogni partecipante dedica 30 minuti alla revisione di ogni programma e poi compila un modulo di valutazione

Il modulo di valutazione chiede al revisore di rispondere, su una scala da 1 a 10, a domande come:

La programmazione era semplice,

Il progetto di basso livello era visibile e ragionevole?

Sarebbe per te

Al revisore vengono anche richiesti commenti generali e suggerimenti di miglioramento

Valutazione tra pari

Dopo la revisione, ai partecipanti vengono fornite le valutazioni anonime per i due programmi da loro forniti

- Viene inoltre fornito loro un riepilogo statistico che mostra la classifica generale e dettagliata dei loro programmi originali nell'intero insieme di programmi.
- E un'analisi di come le loro valutazioni di altri programmi si confrontano con quelle di altri recensori dello stesso programma

Lo scopo del processo è consentire ai programmatori di autovalutare le proprie competenze di programmazione

- Pertanto, il processo sembra essere utile sia in ambienti industriali che scolastici.

Revisione contabile

Un esame indipendente di un prodotto di lavoro eseguito da una terza parte per valutare la conformità a specifiche, standard, accordi contrattuali o altri criteri

Programmazione in coppia

Una pratica associata a processi agili come Extreme Programmazione (XP)

Due programmatori scrivono il codice insieme

Questi programmatori si alternano alla tastiera

- Il programmatore che scrive il codice è chiamato **driver**
- L'altro programmatore si chiama **navigatore**

Programmazione in coppia

L'autista:

- scrive nuovo codice
- concentrarsi sulla codifica dettagliata: sintassi e struttura

Il navigatore: -
dirige il lavoro del conducente e rivede il codice scritto

- evidenzia possibili problemi
- si concentra sul flusso del lavoro e lo rivede in tempo reale

e poi discutono insieme soluzioni alternative

Programmazione in coppia

PP migliora la qualità e la leggibilità del codice e può accelerare il processo di revisione

PP unisce le attività di codifica e di ispezione eliminando così il divario tra le fasi classiche di codifica e di ispezione

I programmatori alternano spesso i ruoli in modo che il codice scritto non sia "di proprietà" dei singoli programmatori, ma piuttosto una responsabilità collettiva di entrambi i programmatori.

In XP, la programmazione in coppia viene solitamente eseguita in modalità normale (8-giorni lavorativi (ore)

Per essere efficaci è necessario un atteggiamento costruttivo dei programmatori e un'armonia tra loro

Richiedere a due programmatori di lavorare insieme in modo sincrono

Copilota di GitHub

Copilot di GitHub è uno strumento di sviluppo basato sull'intelligenza artificiale che suggerisce codice e funzioni complete in tempo reale, direttamente dal tuo editor

La programmazione in coppia basata sull'intelligenza artificiale cambia l'esperienza di codifica e rivela l'importanza di competenze diverse: l'importanza emergente per gli sviluppatori di sapere come rivedere il codice tanto quanto saperlo scrivere

Meno codifica, più revisione

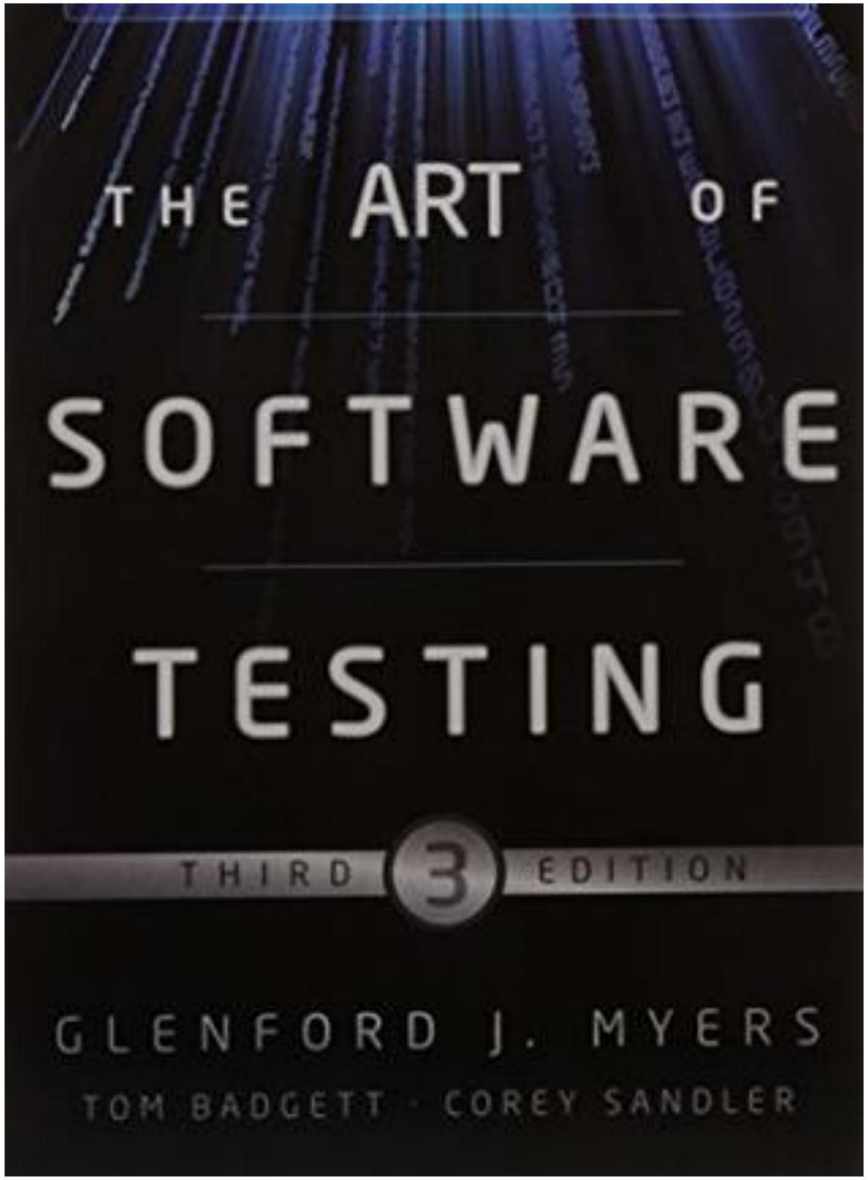
Il rischio è di capire meno come/perché funziona il codice

Libro di riferimento :

L'arte del software Test

Capitolo 3

Pdf disponibile online



Riferimenti interessanti

- **Prendere il volo con Copilot.** Prime intuizioni e opportunità dell'intelligenza artificiale strumenti di programmazione in coppia potenziati. https://queue.acm.org/detail.cfm?id=3582083

Azzurra Ragone

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