Error Injection and Removal & Testing Methodology

Software Engineering

Docente: Dr. Anna Rita Laurenzi, PhD. – Docente aggiunto: Dr. Ercole Colonese, Consulente

Agenda

- *Immissione e rimozione degli errori e difettosità residua*
- *Propagazione degli errori*
- *Curve di Raleigh*
- *Rimozione degli errori: revisioni tecniche e testing*

Rieferimenti:

1. **Roger S. Pressman – Principi di ingegneria del software – Mc GrawHill - 2005**
2. **Stefano Nocentini – Il sistema di qualità del software – ETASLIBRI – 1993**
3. **G. Cicogni e P. Risi – Il test e la qualità del software – Il Sole 24 Ore - 1998**

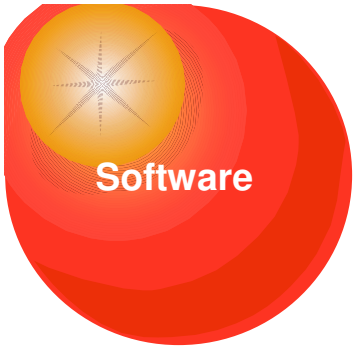
I difetti nel software

Il software

Il software è il prodotto di un'attività intellettuale creativa ad alto contenuto umano e quindi soggetta ad errori.

La probabilità di commettere errori (e quindi di "iniettare errori nel software") è alta quando non sono garantite le componenti organizzative quali:

- *controllo della complessità della soluzione attraverso una progettazione razionale;*
- *maturità del processo adoperato;*
- *competenza delle persone;*
- *adozione di metodi, tecniche e strumenti di produttività.*



Rimozione degli errori

Le due tecniche più efficaci per la rimozione degli errori nel software sono:

1. ***Revisione tecnica****: adoperata nelle fasi alte del ciclo di sviluppo, consiste nella revisione dei documenti prodotti con lo scopo di rimuovere gli errori nella stessa fase in cui sono "immessi" ed evitare che si propaghino nelle fasi successive;*
2. ***Testing****: utilizzata per collaudare il software prodotto con prove di tipo e profondità diverse: unitario, d'integrazione, di sistema, ecc.*

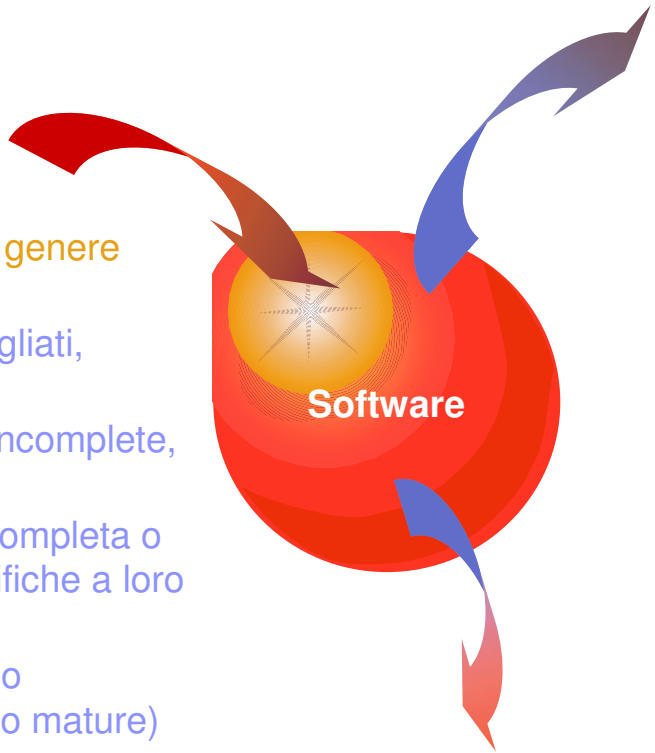
Errori residui

Gli errori rimasti nel software dopo il suo rilascio in esercizio causano problemi nell'operatività degli utenti finali.

La loro rimozione ha un costo elevato.

L'impatto è quindi negativo e fortemente penalizzante sia per il cliente che per il produttore del software.

I difetti nel software



Immissione di errori

Gli errori immessi sono in genere causati da:

- requisiti poco chiari, sbagliati, mancanti
- specifiche poco chiare, incomplete, errate
- progettazione errata, incompleta o basata su requisiti o specifiche a loro volta errate
- utilizzo di tecnologie poco conosciute o instabili (poco mature)
- errori di logica
- competenze inadeguate alla complessità del progetto

Rimozione di errori

Il numero e la tipologia di errori rimossi dipendente dalla competenza delle persone e dalle tecniche adottate:

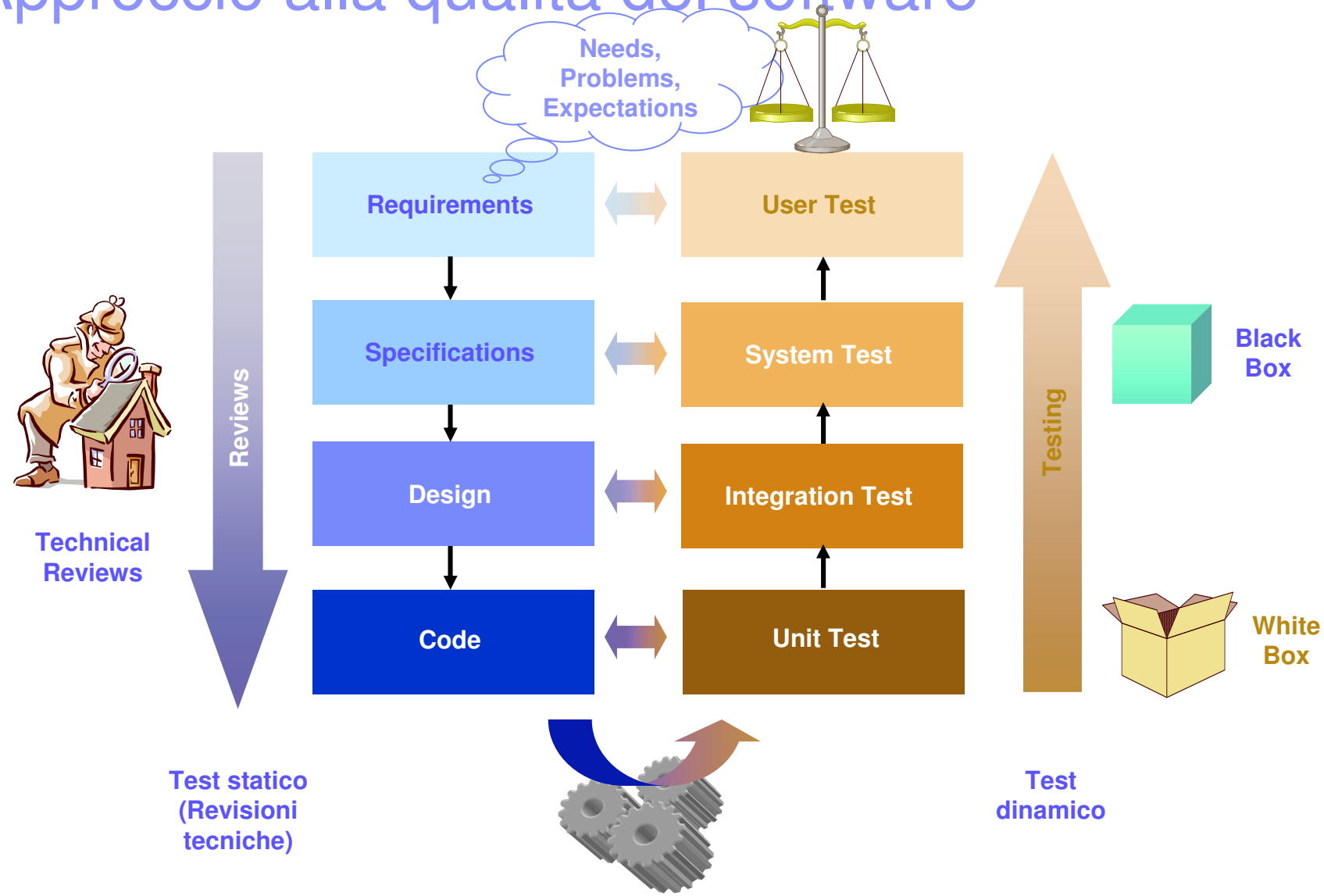
- persone competenti rimuovono più errori a parità di tempo impiegato
- le revisioni tecniche sui documenti delle fasi alte del ciclo di vita rimuovo errori dai prodotti ispezionati e riducono il propagarsi degli stessi nelle fasi successive
- il livello di copertura dei test e la correttezza della progettazione dei casi di prova massimizza il numero di errori rimossi

Errori residui

Gli errori residui gravano sull'operatività degli utenti finali e sui costi dell'organizzazione che deve correggerli:

- la rimozione controllata degli errori durante lo sviluppo, fatta in maniera efficace e metodica riduce la difettosità residua

Approccio alla qualità del software



Approccio alla qualità del software

Lo sviluppo del software:

Il software è realizzato tramite una trasformazione continua e progressiva di problemi e necessità in oggetti via via sempre più materiali.

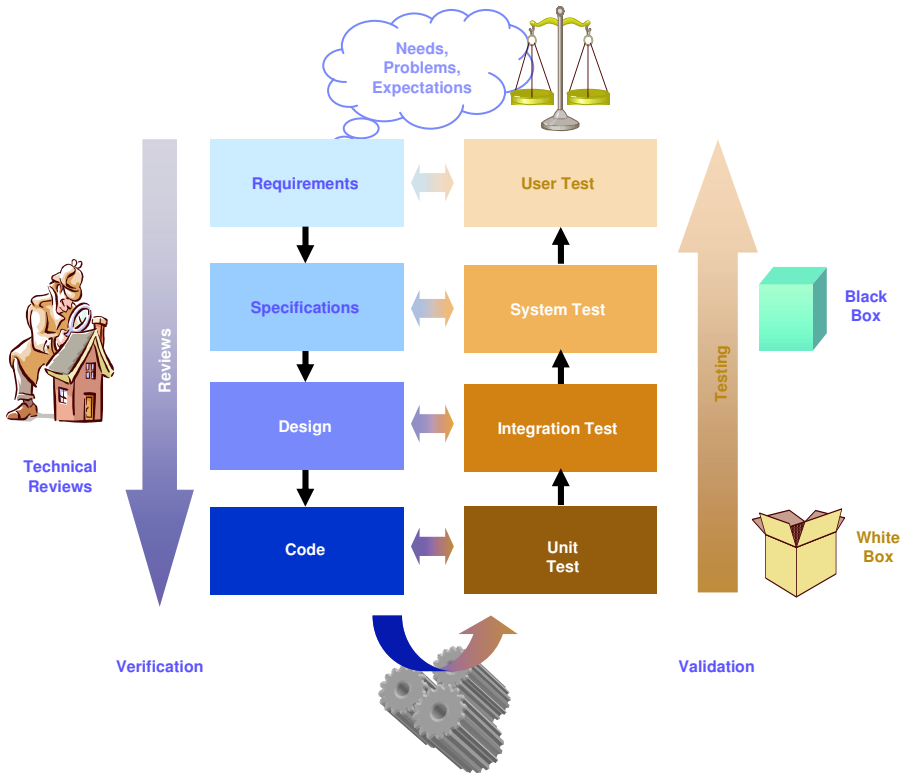
Le idee si traducono in requisiti; questi in specifiche e poi in progetto. Infine in codice e manuali.

L'attività di continua trasformazione di idee in qualcosa di più concreto è soggetta ad equivoci, malintesi, errori.

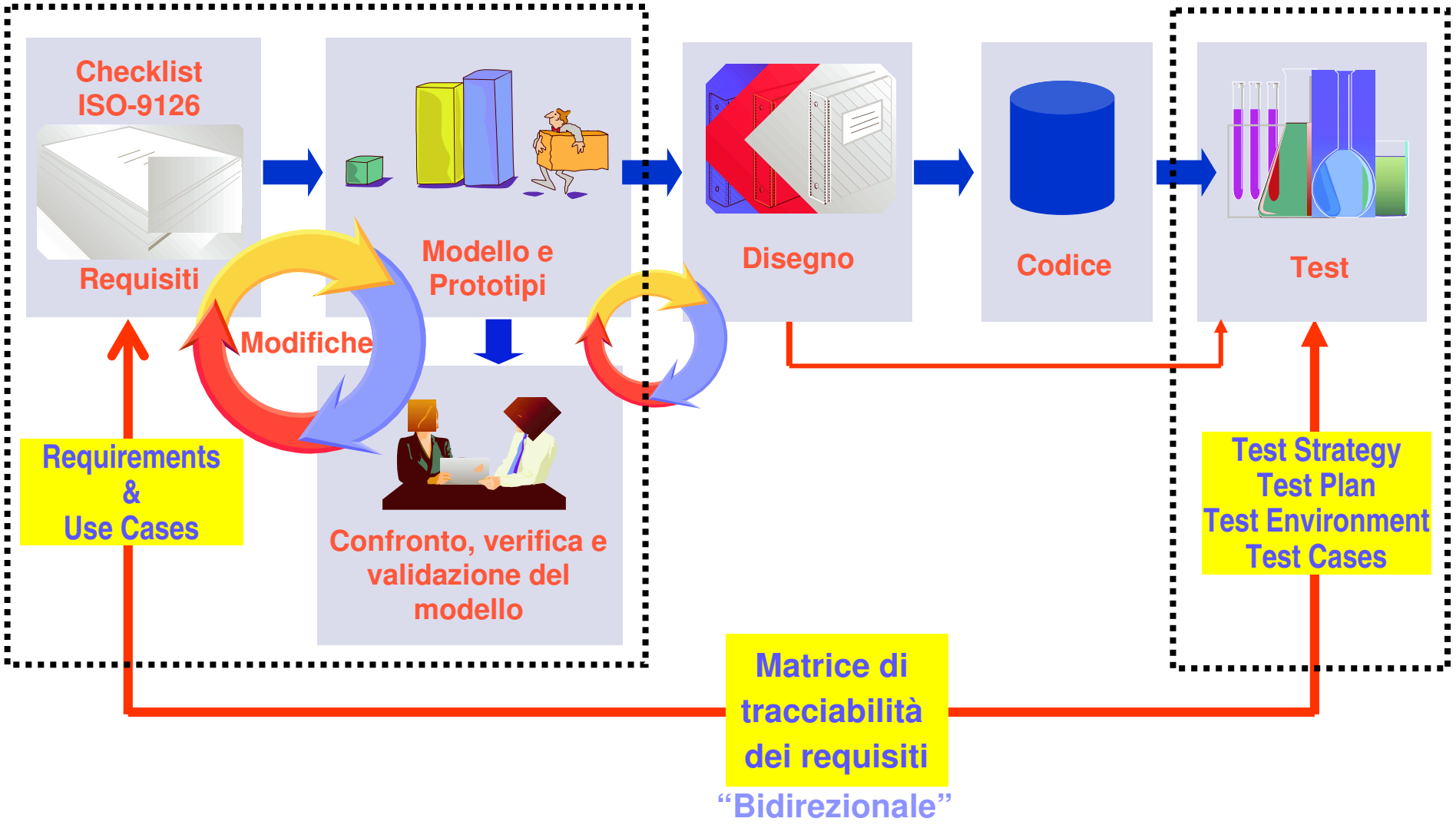
La fase di costruzione deve essere controllata con revisioni tecniche che "verifichino" la correttezza delle singole trasformazioni.

Una volta costruito il codice, occorre "validarlo" con un processo di collaudo a ritroso.

Dai singoli componenti unitari si passa al collaudo d'integrazione dei moduli, al sistema nel suo complesso ed infine all'accettazione da parte degli utenti finali.



In pratica …


Checklist
ISO-9126
Requisiti
Modello e
Prototipi
Disegno
Codice
Test
Modifiche
Requirements
&
Use Cases
Confronto, verifica e
validazione del
modello
Test Strategy
Test Plan
Test Environment
Test Cases
Matrice di
tracciabilità
dei requisiti
"Bidirezionale"

In pratica …

Lo sviluppo del software:

La prima fase dello sviluppo (Anali) consiste nel raccogliere, analizzare, valutare le richieste del cliente.

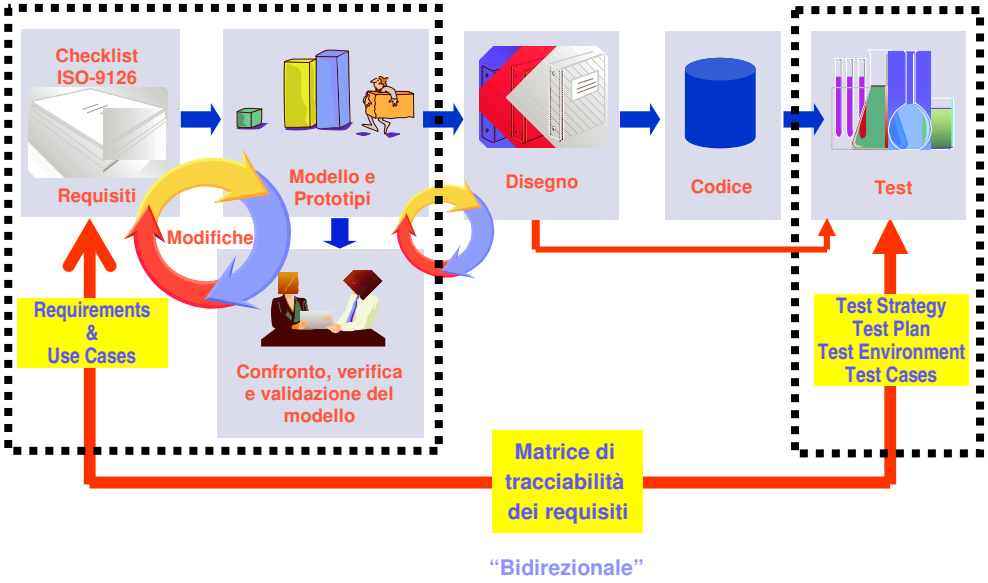
Le norme **ISO 9126** forniscono un elenco di caratteristiche del software che sono una valida guida (*checklist*) per verificare se si stiano dimenticando o sottovalutando requisiti.

La realizzazione di **prototipi** è di grande utilità per verificare la comprensione dei requisiti e la risposta che ad essi si vuole dare.

L'attività di analisi è quindi "*iterativa*" e la si può ritenere completa solo quando siano stati individuati tutti i requisiti più importanti e siano stati verificati con il cliente.

La formalizzazione dei requisiti in un **documento approvato dal cliente** chiude formalmente la fase di analisi.

L'attività complementare è la realizzazione dei **casi d'uso** (*use case*) che descrivono il comportamento delle interfacce e le interazioni con gli utenti.



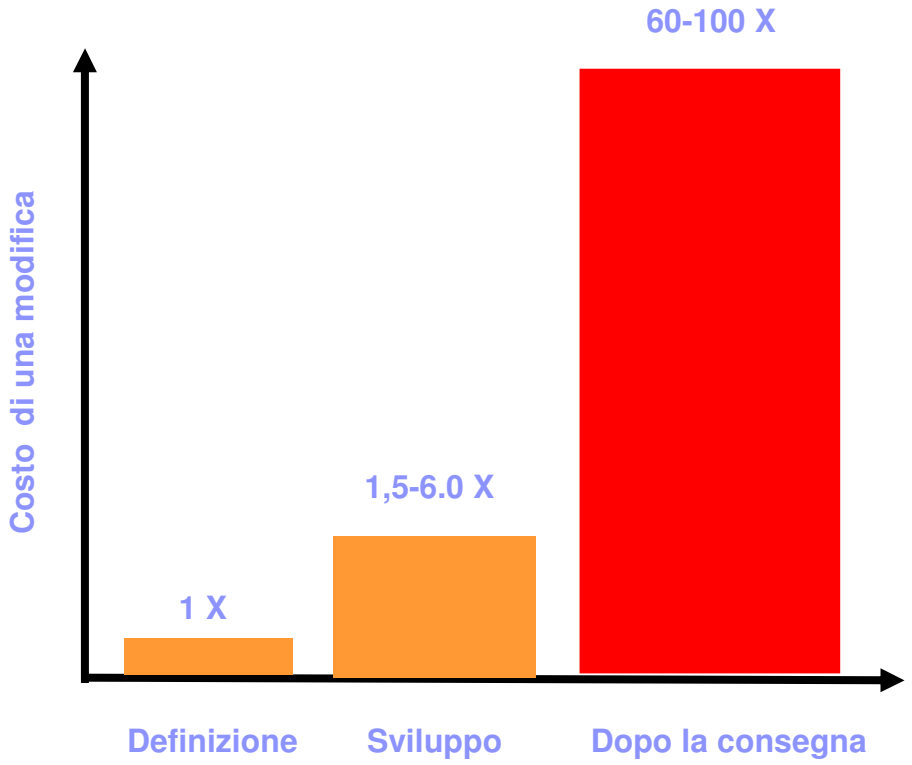
La fase successiva (Design) realizza il progetto di dettaglio partendo da quanto consolidato in precedenza (documento dei requisiti e casi d'uso).

Il "codice" è a sua volta realizzato sulla base del disegno.

Infine il collaudo valida il codice in base alla progettazione e ai requisiti, funzionali e qualitativi. La garanzia che tutti i requisiti, nessuno escluso, siano collaudati è data dalla "**matrice di tracciabilità**" biunivoca.

Impatto delle modifiche ai requisiti software

- I requisiti rappresentano il primo anello della catena nello sviluppo del software
- Essi si traducono in specifiche e poi in progettazione, quindi in codice
- La modifica ad un requisito può comportare variazioni in ogni fase successiva del ciclo di sviluppo a seconda del momento in cui è richiesta
- Preso come unitario l'impegno richiesto a modificare un requisito nella prima fase del ciclo di sviluppo, la stessa modifica richiede un impegno maggiore a sviluppo iniziato ed un impegno ancora maggiore dopo la consegna del software al cliente



(Fonte: R.S. Pressman, 2000)

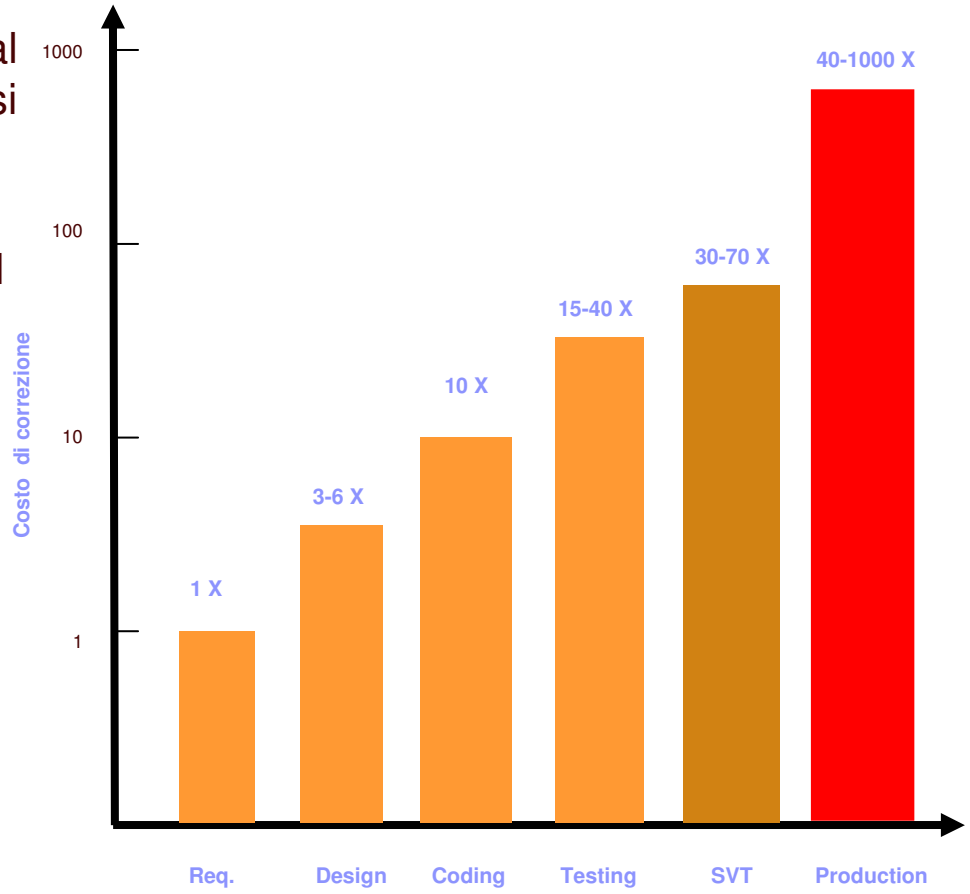
Costo di rimozione degli errori nel software

• Gli errori immessi nel codice sono rimossi con un costo che aumenta al passare del tempo e quindi delle fasi del ciclo di sviluppo

• Il costo diventa massimo quando il software è messo in produzione ed è utilizzato dai sui utenti finali

• Questo andamento suggerisce di rimuovere gli errori il più presto possibile, direttamente nella fase in cui ci si trova

• Ritardare la rimozione degli errori ha come diretta conseguenza l'aumento dei costi del progetto



(Fonte: Boehm, 1981)

Verifica e validazione della qualità del software

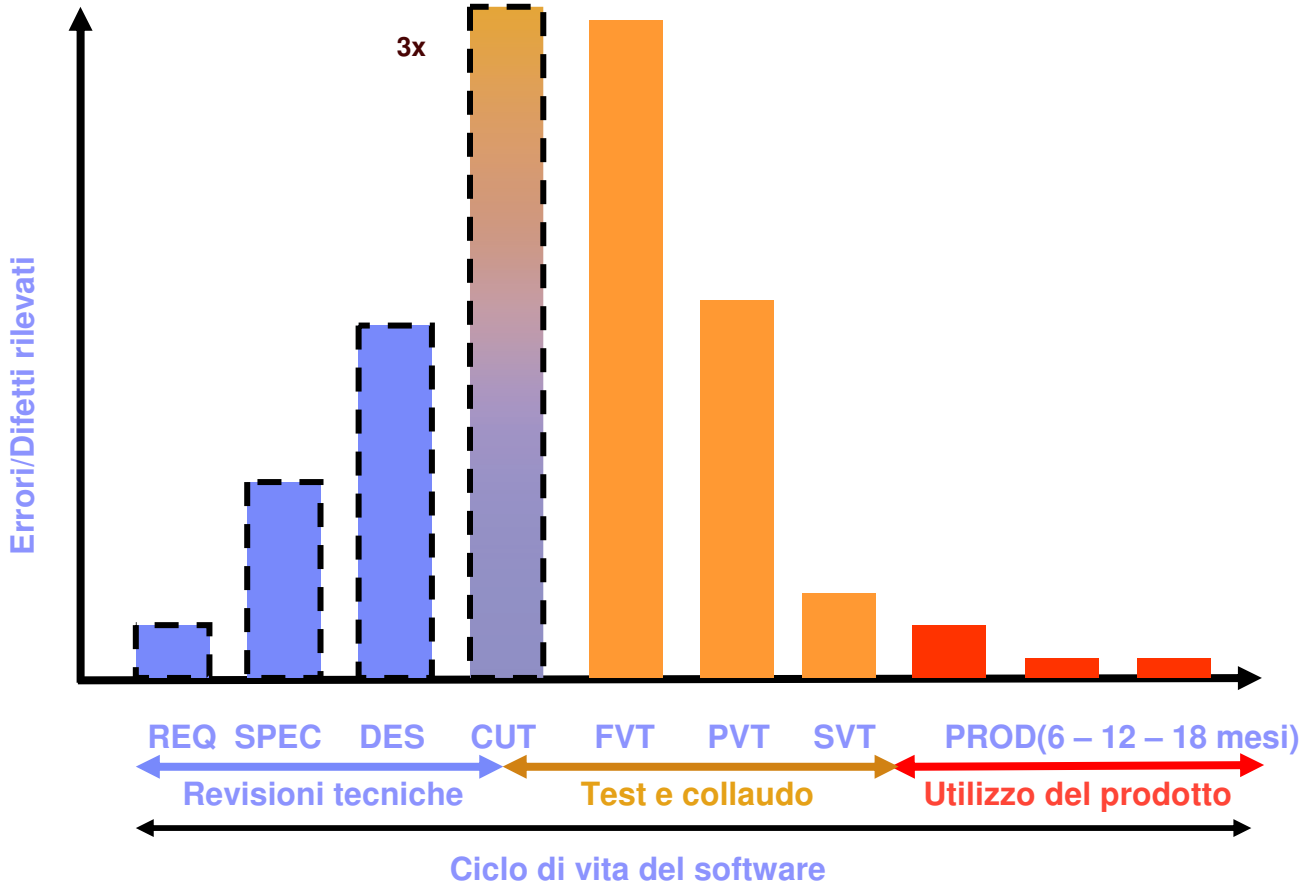
- *Profilo della difettosità del software*
- *Propagazione degli errori*
- *Revisioni tecniche*
- *Test*

"Il 20% del codice contiene l' 80% dei difetti. Troviamoli e correggiamoli."

Lowell Arthur

Docente: Dr. Anna Rita Laurenzi, PhD. – Docente aggiunto: Dr. Ercole Colonese, Consulente

Profilo della difettosità del software



Profilo della difettosità del software

Le revisioni tecniche del software:

La tecnica delle revisioni, concepita originariamente da IBM con il nome di ***Ispezione***, è stata successivamente raffinata e ed adottata esternamente dal mondo dello sviluppo software.

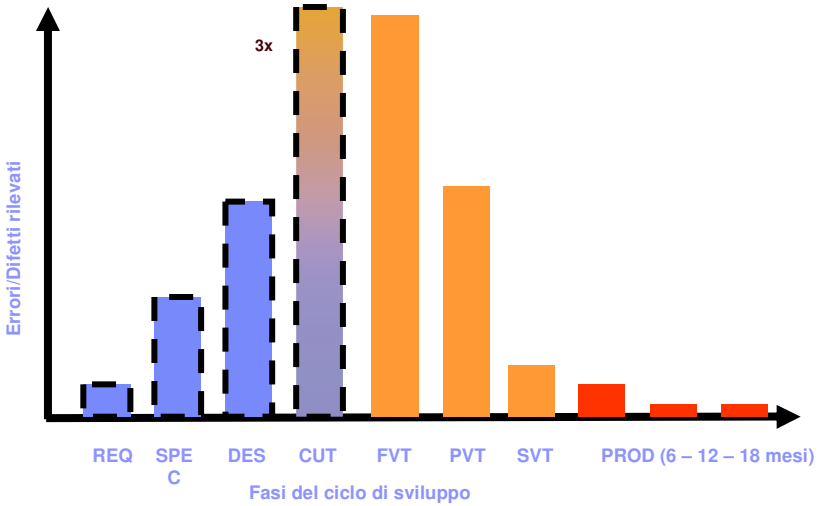
E' diventata una "*best practice*" ed è prevista anche dal modello CMM con il nome di ***Peer Review***.

E' una tecnica semplice ed efficace per rimuovere gli errori nelle fasi alte del ciclo di sviluppo.

Il ***Testing*** è l'altra tecnica per rimuovere gli errori dal codice già sviluppato.

Registrando il numero di errori rilevati nelle diverse fasi del ciclo di sviluppo si può costruire il profili mostrato in figura.

Man mano che si procede con le fasi iniziali cresce il numero di errori rilevati (barre azzurre). Il picco è raggiunto in corrispondenza della fase di codifica e di test unitario.

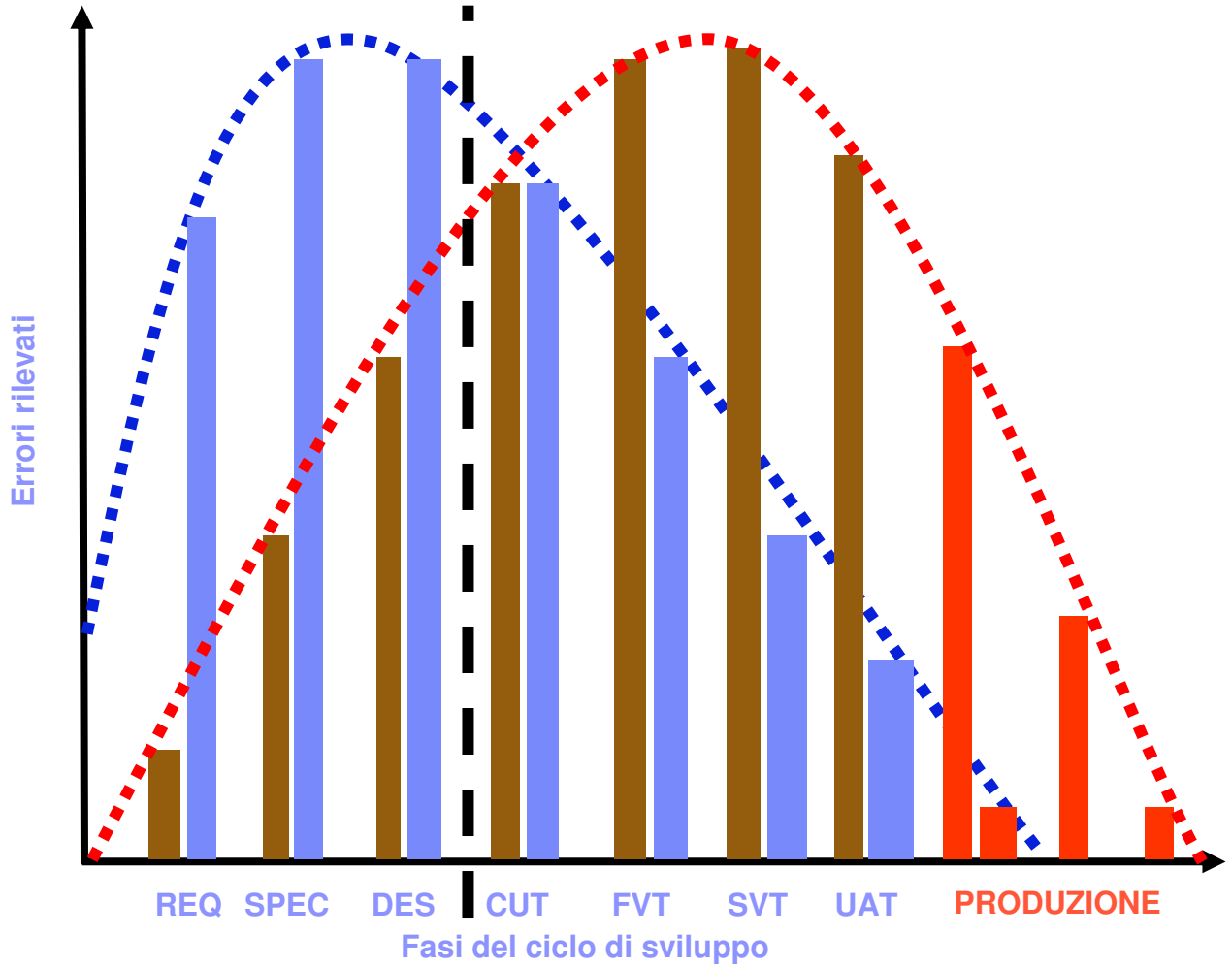


Tale fase ha un doppio colore in quanto si eseguono sia revisioni tecniche del codice (a campione) sia test unitari sui moduli. Nella figura il valore reale è ridotto di 3 volte per motivi di scala.

Le fasi di test (in giallo) vedono invece una diminuzione progressiva dei valori e quindi del numero di errori rilevati che si riducono man mano che si procede con le attività di collaudo.

Per il software vale il concetto statistico: "*Più errori si trovano e più errori rimangono da scoprire. Meno errori si trovano e meno errori rimangono da scoprire*".

Curve di rimozione degli errori



Curve di rimozione degli errori

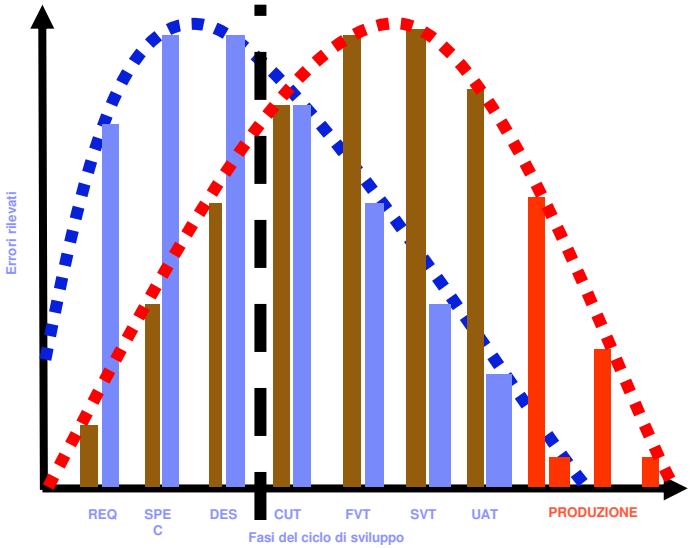
Numero di errori rimossi:

Le revisioni tecniche e le attività di test e collaudo permettono di individuare gli errori nei documenti prodotti nelle fasi alte del ciclo di sviluppo (Requisiti, Specifiche, Disegno) e successivamente nel codice sviluppato.

Riportando il numero degli errori rilevati in un istogramma si può costruire la curva mostrata nella figura. Essa rappresenta il profilo della rimozione degli errori da parte dell'organizzazione.

La curva tratteggiata in rosso è decentrata rispetto alla linea di demarcazione (linea nera) tra le fasi di "analisi, progettazione e codifica" e quelle di "test e collaudo".

Come conseguenza si rilevano più errori nelle fasi di test e collaudo (più costose) e diventa importante il numero di errori rilevati dagli utenti finali (barre rosse) con un costo notevole per la loro risoluzione.



Una curva migliore:

La curva tratteggiata in blu è invece decentrata a sinistra della linea di demarcazione.

Come conseguenza si rileva un numero maggiore di errori nelle fasi alte del ciclo di sviluppo ed un numero minore durante le fasi di test e collaudo.

Ancora più favorevole risulta essere il numero di errori rilevati dagli utenti finali durante l'utilizzo in esercizio del prodotto.

Cause più frequenti degli errori

Tipologia di errori rilevati:

Registrando sia il numero che la tipologia di errori rilevati è possibile costruire una tabella come quella mostrata.

Dall'analisi si evince quali errori siano comunemente introdotti dagli sviluppatori.

Gli errori più frequenti riguardano l'interpretazione dei requisiti, la scrittura delle specifiche, la progettazione delle basi dati, il collaudo del codice.

Da soli, queste tipologie di errore rappresentano la maggioranza dei difetti (63%).

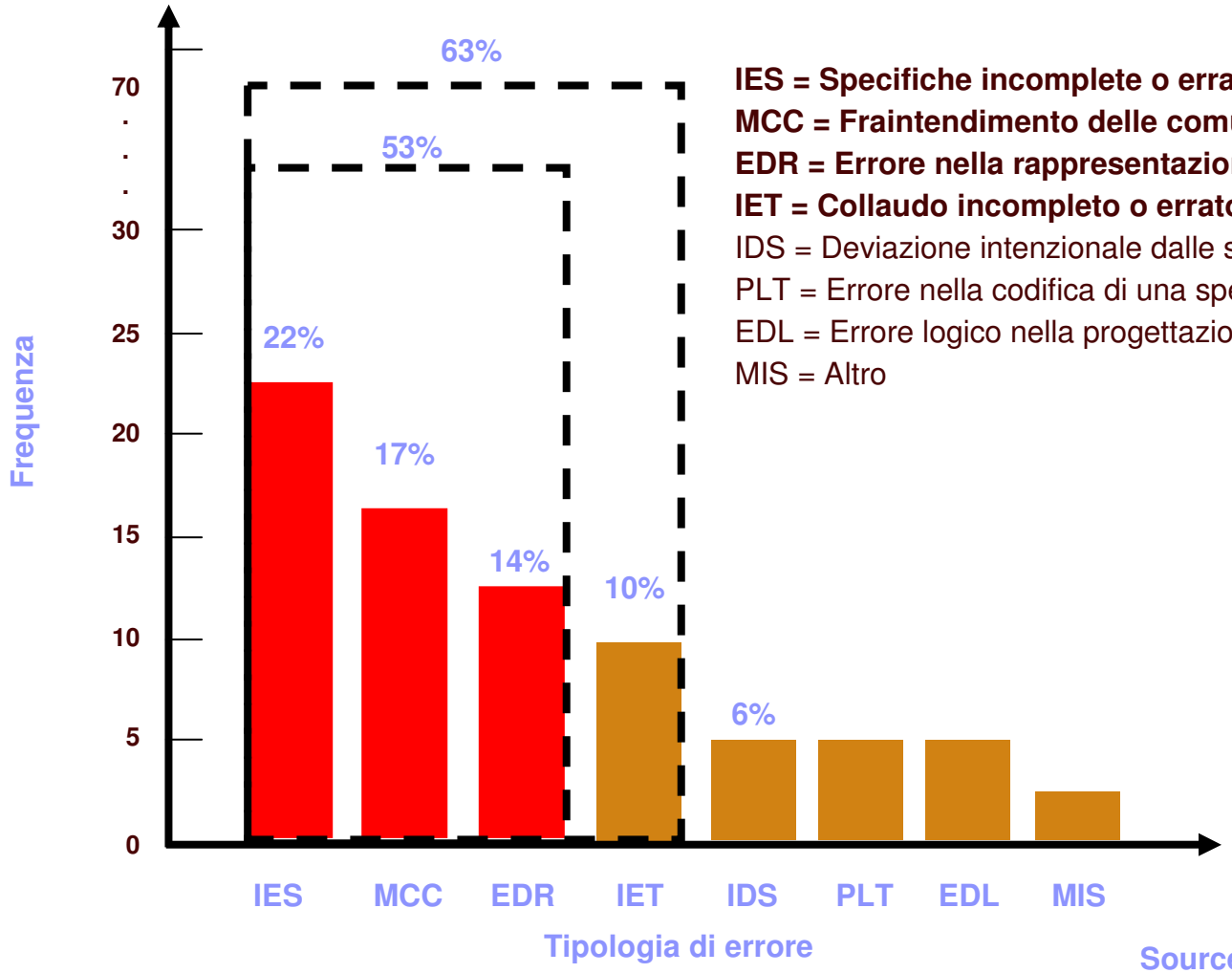
Agire per prevenirli è la cosa migliore che possa fare un'organizzazione di sviluppo software.

DESCRIZIONE (TIPOLOGIA)	%
Specifiche incomplete o errate (IES)	22
Fraintendimento delle comunicazioni del cliente (MCC)	17
Errore nella rappresentazione dei dati (EDR)	14
Collaudo incompleto o errato (IET)	10
Errore nella traduzione della progettazione nella codifica (PLT)	6
Inconsistenza dell'interfaccia tra i componenti (ICI)	6
Errore logico nella progettazione (EDL)	5
Deviazioni intenzionali dalle specifiche (IDS)	5
Documentazione imprecisa o incompleta (IID)	4
Violazione di standard di programmazione (VPS)	3
Interfaccia uomo-macchina ambigua o inconsistente (HCI)	3
Altro (MIS)	5
Totale	100

63%

Source: Roger S. Pressman, 2000

Cause più frequenti degli errori



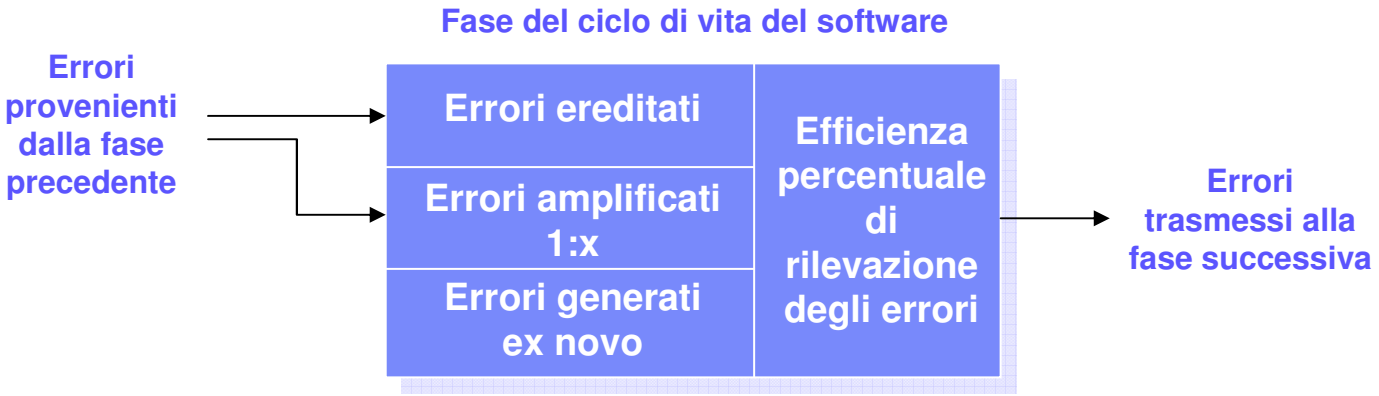
IES = Specifiche incomplete o errate
MCC = Fraintendimento delle comunicazioni con il cliente
EDR = Errore nella rappresentazione dei dati
IET = Collaudo incompleto o errato
IDS = Deviazione intenzionale dalle specifiche
PLT = Errore nella codifica di una specifica di disegno
EDL = Errore logico nella progettazione
MIS = Altro

Source: Roger S. Pressman, 2000

Propagazione degli errori

"Alcune malattie, dicono i medici, all'inizio sono facili da curare ma difficili da riconoscere, ma con il passare del tempo, se non vengono riconosciute e trattate adeguatamente, divengono facili da riconoscere ma difficili da curare"

Nicolò Macchiavelli

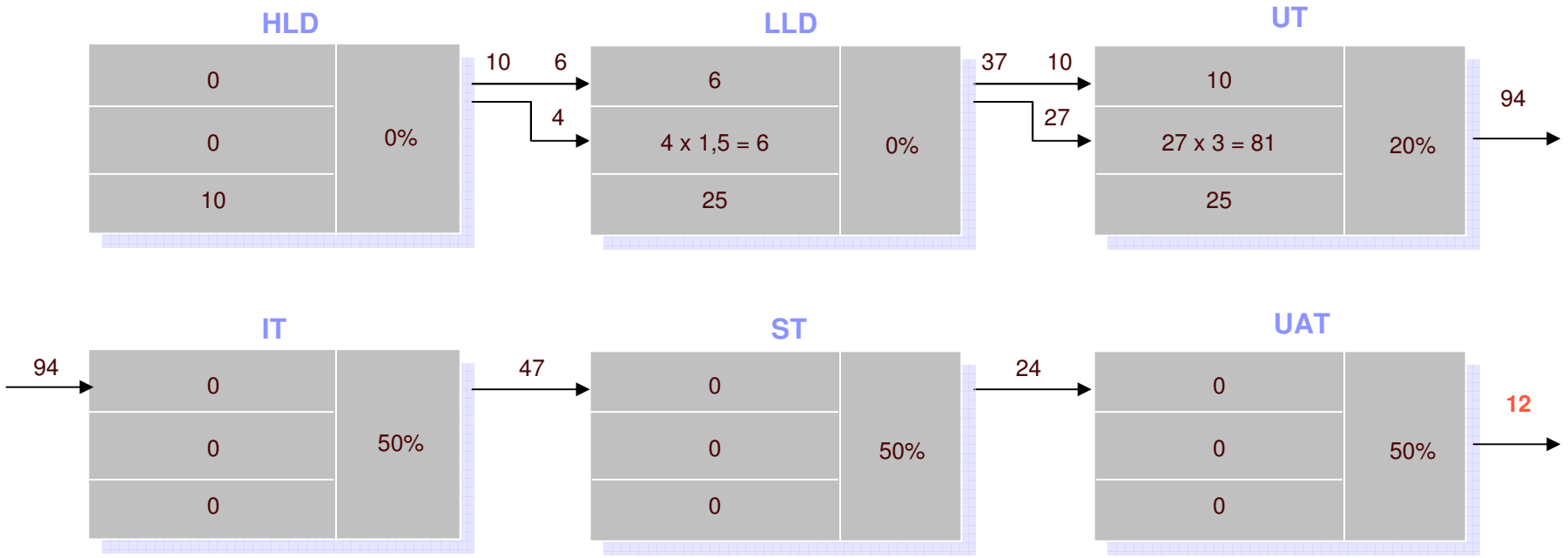


Source: IBM System Scientific Institute, 1981

Costo della rimozione degli errori "senza" revisioni tecniche = 2177 unità di lavoro (errori residui = 12)
Costo della rimozione degli errori "con" revisioni tecniche = 780 unità di lavoro (errori residui = 3)

Propagazione degli errori (senza revisioni)

Fase	Analisi e Disegno	Prima del collaudo	Durante il collaudo	Dopo il collaudo
Costo di rimozione	1	6,5	15	67

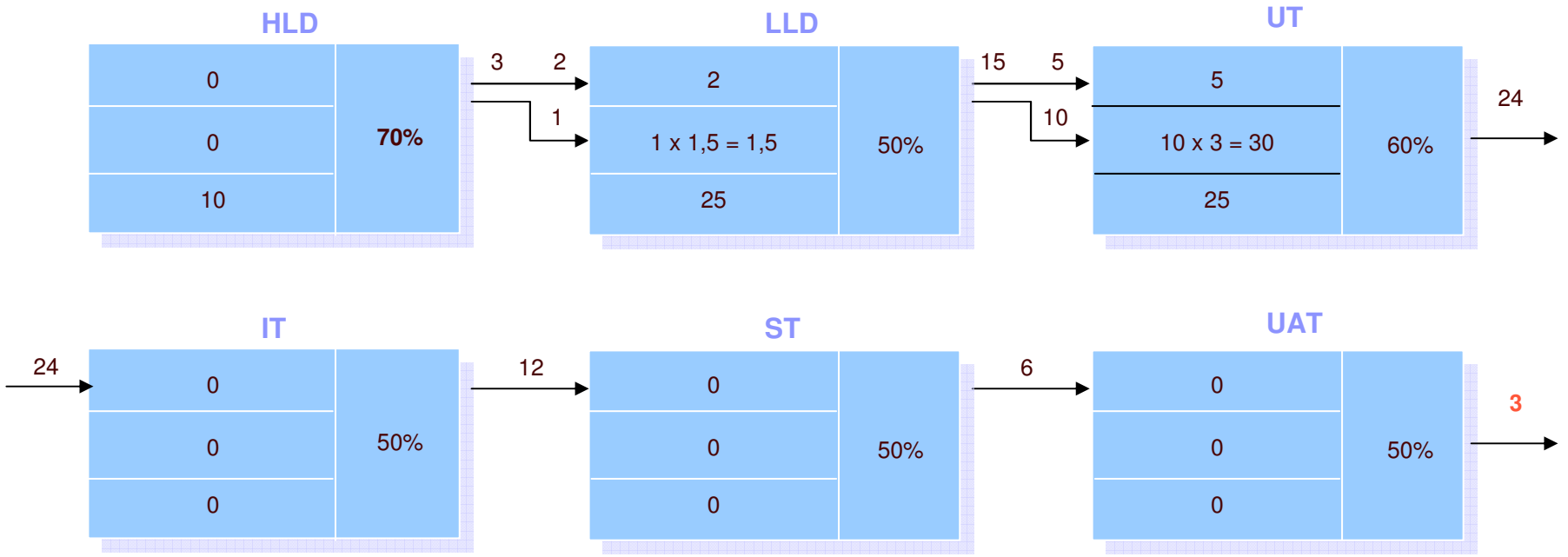


Source: IBM System Scientific Institute, 1981

Costo totale rimozione errori = 2177 unità di costo

Propagazione degli errori (con revisioni)

Fase	Analisi e Disegno	Prima del collaudo	Durante il collaudo	Dopo il collaudo
Costo di rimozione	1	6,5	15	67



Source: IBM System Scientific Institute, 1981

Costo totale rimozione errori = 780 unità di costo

Le revisioni tecniche del software

Effettuare le revisioni tecniche è necessario oltre che utile, ma ha anche un costo.

Ottimizzare tali attività è dunque fondamentale.

L'approccio più produttivo che si possa tenere nell'adoperare tale tecnica è di

- Concentrarsi sulla revisione della documentazione più "critica" (requisiti, specifiche, disegno, parti critiche di codice, casi di test, manuali etc.)
- Revisionare solo documenti "completati" o "parti rilevanti" di documenti completati
- Usare liste di controllo (*checklist*) costruite sull'esperienza
- Non spendere più di 2 ore per ciascuna revisione e altre 2 ore per la preparazione (tot. 4 ore per ciascuna revisione)
- Registrare i risultati ed aggiornare le liste di controllo (checklist)
- Non far partecipare i manager dalle revisioni tecniche (potrebbero formulare giudizi sulla persona invece che adoperarsi per trovare gli errori!)
- Usare un approccio "Peer activity" (colleghi che si aiutano a vicenda)
- Scegliere solo i documenti (o parti di essi) più importanti e critici per il progetto in modo da dedicare lo sforzo necessario agli argomenti veramente utili, quelli che fanno la differenza in termini di qualità. Ricordarsi di Pareto:

"***L'80% degli errori è concentrato nel 20% delle componenti!***"

La metodologia della revisione tecnica

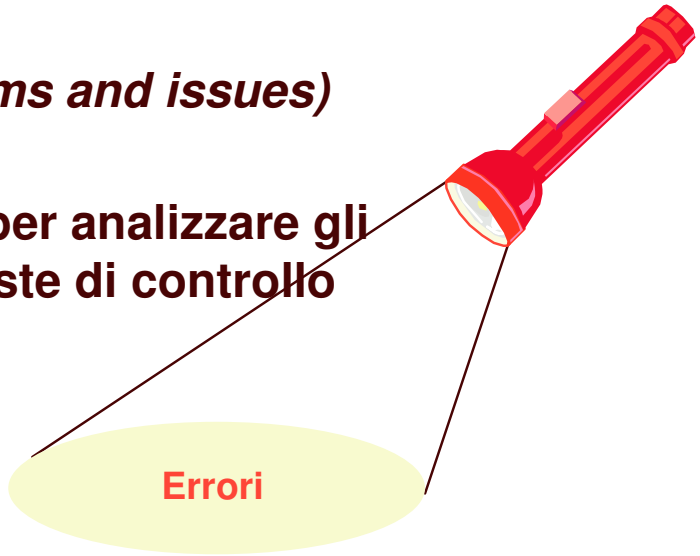
E' necessario adottare una metodologia consolidata per evitare che le revisioni abbiano un effetto meno positivo di quello che ci aspetta abbiano.

Le regole definite per una revisione efficace sono:

- Il responsabile del progetto seleziona quali documenti sia utile revisionare, identifica le persone più adatte a condurre le revisioni, quantifica il tempo necessario e pianifica le attività;
- La responsabilità del buon esito di ciascuna revisione è assegnata ad una persona (*Moderatore*);
- Le persone identificate per effettuare le revisioni sono dette *Revisori*;
- L'autore del documento da sottoporre a revisione lo fornisce appena completato ai revisori illustrandone il contenuto;
- I revisori leggono con cura il documento e annottano gli errori rilevati ed i commenti;
- Si organizza un incontro in cui si discutono gli errori rilevati ed i commenti. Non si parla mai di possibili soluzioni. Il moderatore ha la responsabilità garantire l'efficacia della revisione e di registrare gli errori rilevati;
- L'autore provvede a correggere gli errori rilevati a sua discrezione;
- Il moderatore chiude la revisione dal punto di vista formale quando l'autore comunica di aver effettuato tutte le modifiche.

Le revisioni tecniche sono efficaci quando …

- **sono pianificate *(esse costano e devono quindi essere previste nel piano)***
- **sono preparate *(“No preparation no results”)***
- **sono proattive *(No judge, help)***
- **i risultati sono controllati *(Solve problems and issues)***
- **la registrazione dei risultati è utilizzata per analizzare gli errori e per migliorare le statistiche e le liste di controllo (*checklists*)**



Report di revisione tecnica

REPORT DI REVISIONE TECNICA			
Progetto: <nome del progetto> **Documento**: <nome del documento revisionato> **Data**: <data della revisione> **Autore del documento**: <nome> **Revisiori**: <nomi (il primo nome è quello del moderatore)>		**Stato:** <stato della revisione (vedi "Note per la stesura del Report")> **Data di effettuazione:** <data in cui è stata eseguita la revisione> **Data di chiusura:** <data di chiusura della revisione>	
N.	**Errore**	**Gravità**	**Tipologia**
1	Descrizione dell'errore		
2			
3			
…	…	…	…
Note: <inserire qui eventuali informazioni aggiuntive che posano meglio chiarire qualche aspetto della revisione eseguita>			

Note per la stesura del "Report":

1) La GRAVITA' è "GRAVE" o "LIEVE" (gli errori "gravi" dovranno essere corretti, quelli "lievi" sono da considerarsi come suggerimenti;
2) La TIPOLOGIA di errore è quella definita nella tabella mostrata di seguito;
3) Lo STATO della revisione può essere: PIANIFICATA, EFFETTUATA o RIMANDATA, CHIUSA;
4) La revisione è chiusa quando l'autore ed il moderatore concordano che tutti gli errori segnalati siano stati "corretti" oppure questi (l'autore) decida che uno o più errori non debbano essere corretti (è una sua prerogativa).

Tipologia di errore

DESCRIZIONE (TIPOLOGIA)	Codice
Specifiche incomplete o errate	IES
Fraintendimento delle comunicazioni del cliente	MCC
Errore nella rappresentazione dei dati	EDR
Collaudo incompleto o errato	IET
Errore nella traduzione della progettazione nella codifica	PLT
Inconsistenza dell'interfaccia tra i componenti	ICI
Errore logico nella progettazione	EDL
Deviazioni intenzionali dalle specifiche	IDS
Documentazione imprecisa o incompleta	IID
Violazione di standard di programmazione	VPS
Interfaccia uomo-macchina ambigua o inconsistente	HCI
Altro (Miscellanea)	MIS

Test del software

"Lo sviluppo del software è un'attività produttiva ad alto contenuto umano dove le occasioni di commettere errori abbondano…

… A causa dell'incapacità umana di comunicare in modo perfetto, lo sviluppo del software è affiancato da un'attività di garanzia di qualità."

Deutsch, 1979

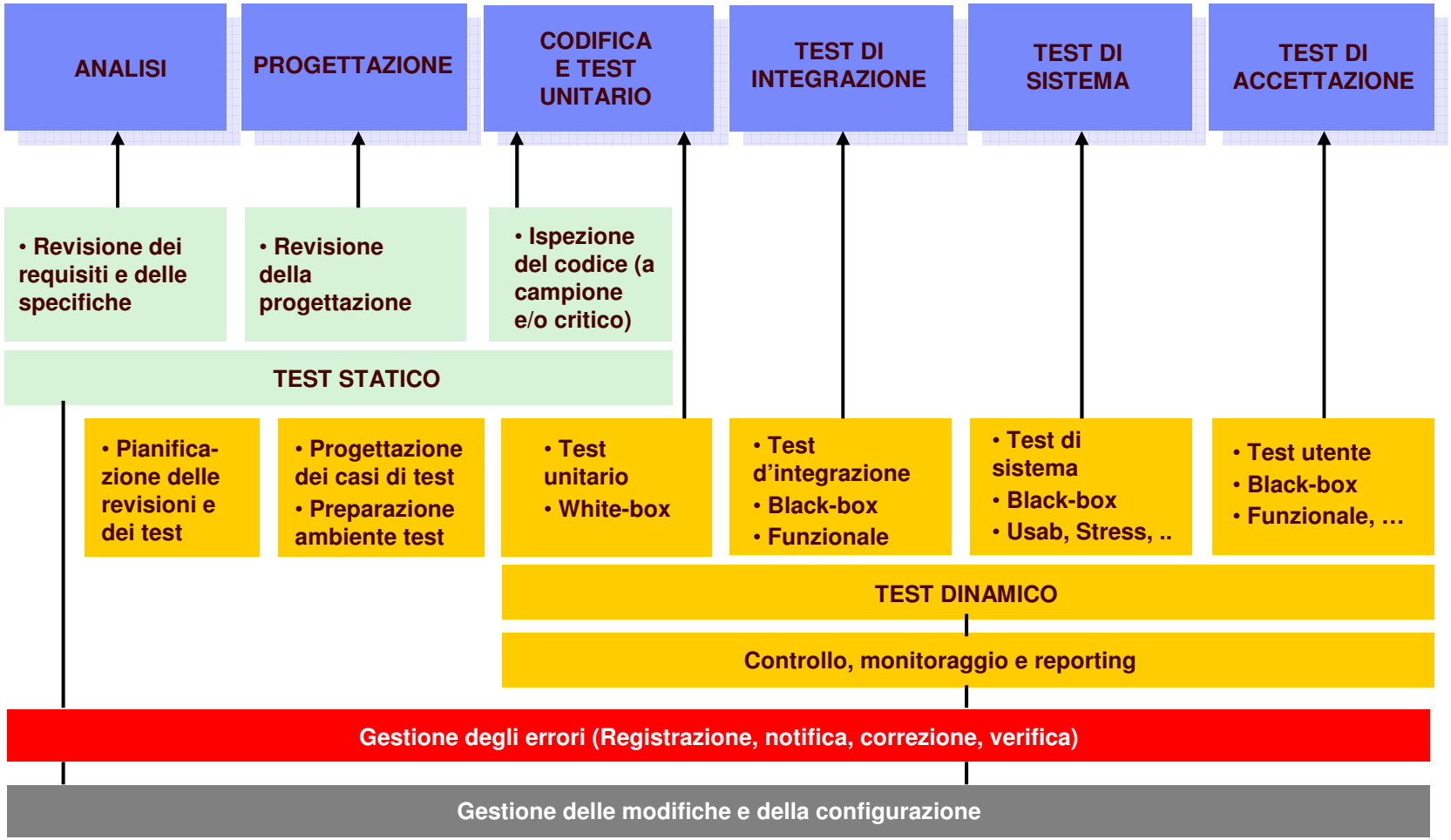
Elementi di Testing

- L'attività di **Testing** è molto importante e critica ai fini della qualità del prodotto software

- **Obiettivo** principale delle attività di Testing è quello di verificare che il prodotto software realizzato:
  - implementi i requisiti specificati (sia cioè il prodotto "**giusto**")
  - implementi i requisiti nel modo corretto (sia cioè un prodotto "**corretto**")

- Il Testing è un'attività continua durante l'intero ciclo di sviluppo del software:
  - **Testing statico**: è rappresentato dalle attività di "*revisione tecnica*" eseguite sui documenti tecnici realizzati (non viene eseguito nessun codice)
  - **Testing dinamico**: è rappresentato dalle attività di testing vere e proprie tramite l'esecuzione del codice sviluppato in appositi ambienti di test

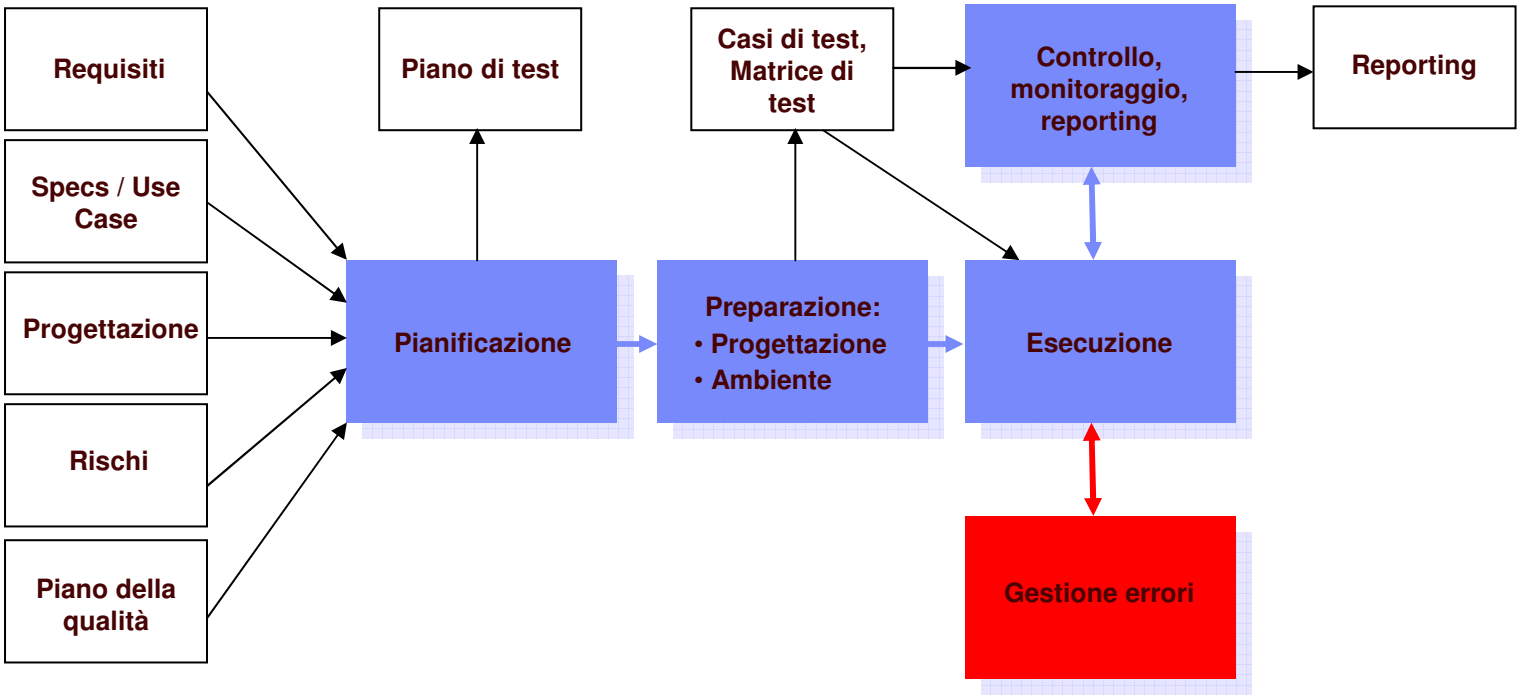
Modello di Testing (ciclo completo)



Processo di Testing

- La **Pianificazione** comporta:
  - Definizione della **Strategia di test** (quali, quanti test prevedere, in quali contesti/scenari, con quali risorse) in base ai requisiti, ai rischi, ecc.
  - Definizione del **Piano di test** che implementa la strategia, con definizione delle attività di dettaglio, risorse necessarie (umane e tecnologiche)
- La **Preparazione** che comporta:
  - Progettazione dei test casi di test, scenari di test, matrici di test e basi di dati
  - Predisposizione dell'ambiente di test (apparati, software, base dati, strumenti, …)
- L'**Esecuzione** comporta:
  - Esecuzione dei casi di prova progettati secondo i tempi, le modalità e la sequenza stabiliti
  - Gestione degli errori (Registrazione, notifica, correzione, verifica dell'efficacia delle soluzione e verifica dell'assenza di introduzione di nuovi errori)
- Il **Controllo, monitoraggio e reporting** comporta:
  - Controllo dello stato di avanzamento delle attività (numero di casi di prova eseguiti, completati rispetto al piano)
  - Controllo dello stato di risoluzione degli errori riscontrati
  - Valutazione del livello di copertura dei test e di difettosità del software
  - Reporting sullo stato dei test

Elementi del Processo di Testing



Livelli e tipi di Testing

- Il *livello* di Testing è:
  - **Unitario** (*Unit Test*): verifica dei singoli moduli
  - d'**Integrazione** (*Integration Test*): verifica delle componenti integrate nel prodotto
  - di **Sistema** (*System Test*): verifica del prodotto nell'ambiente target
  - di **Acettazione** (*User Acceptance Test*): verifica del prodotto da parte degli utenti

- I *tipi* di test indirizzano le diverse caratteristiche del prodotto software:
  - **Funzionale**: verifica delle funzioni realizzate a fronte dei requisiti
  - **Usabilità**: verifica della facilità d'uso, di comprensione e di memorizzazione
  - **Stress**: verifica della robustezza a fronte di carichi diversi e maggiori
  - **Performance**: verifica dell'efficienza del prodotto (tempi di risposta, memoria, …)
  - **Installabilità**: verifica della corettezza dell'installazione e della personalizzazione
  - **Operatività**: verifica delle funzionai operative a carico del gestore

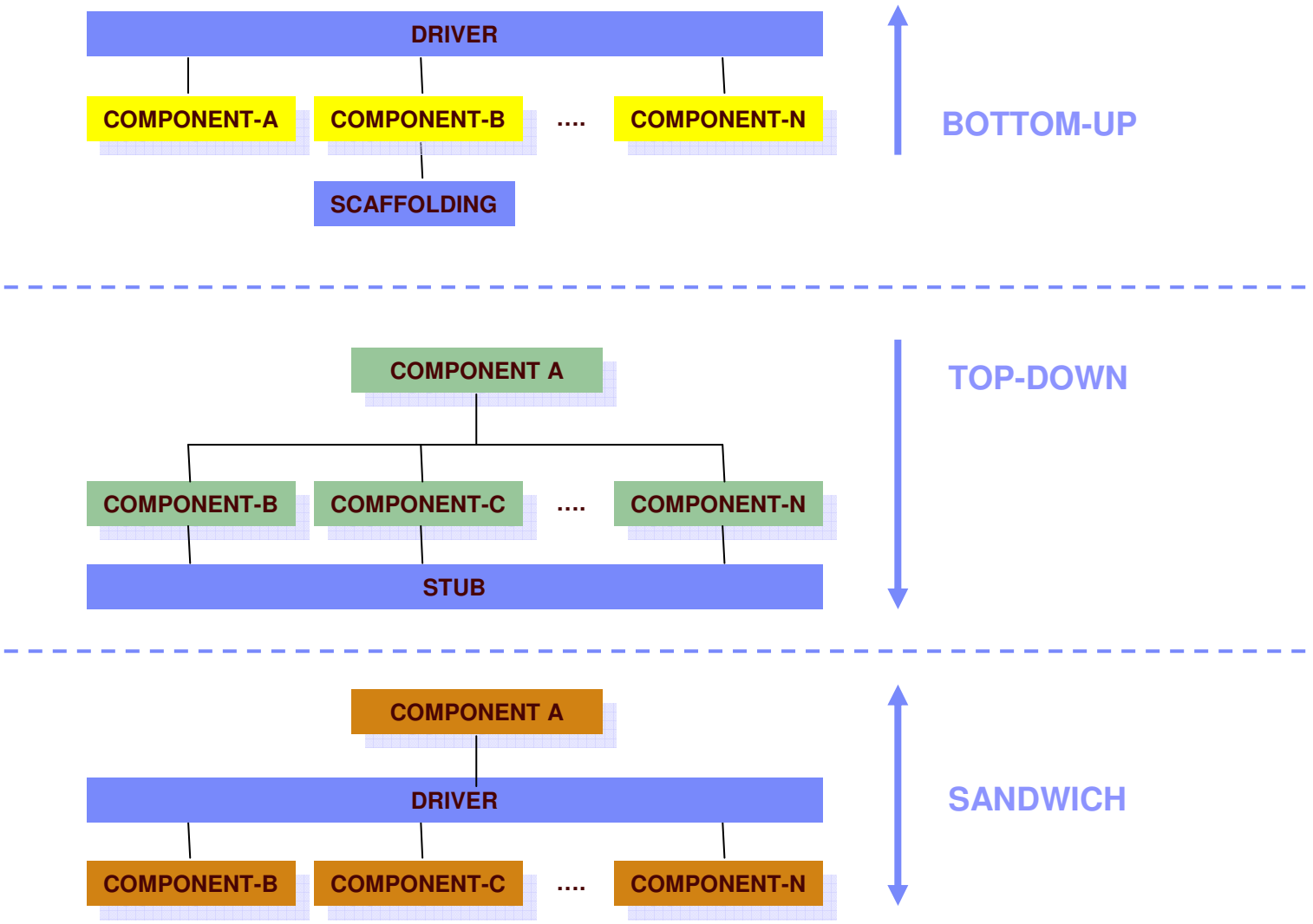
Livelli e tipi di Testing

- Le *tecniche* di Testing sono:
  - **White-box**: verifica del codice (cammini, scelte, algoritmi, dati, condizioni, …)
  - **Black-box**: verifica delle caratteristiche esterne tramite input e verifica dell'output

  - **Top-down**: verifica delle componenti di alto livello simulando le componenti di basso livello (*stub* e *scaffolding*)
  - **Bottom-up**: verifica delle componenti di basso livello simulando quelle di alto livello (*driver*)

- Si può utilizzare codice scritto ad hoc
  - **Driver**: sono programmi che simulano componenti non ancora pronti e che invocano le funzionalità di componenti sottostanti da collaudare
  - **Stub**: sono programmi che simulano componenti non ancora pronte e invocate da componenti di livello più alto da collaudare
  - **Scaffolding**: sono programmi che forniscono dati simulando componenti non disponibili

Driver, Stub, Scaffolding



Metriche del Testing

- ***Efficacia*** *dei test:*
  - *Livello di copertura dei test*
  - *Rimozione degli errori da parte dei test*
- ***Produttività*** *dei test*
  - *Casi di prova eseguiti nell'unità di tempo (g/p)*
- ***Stato di completamento*** *dei test:*
  - *Casi di prova eseguiti e completati con successo sul totale*
  - *Numero di errori rilevati, corretti e verificati*

Copertura	CT1	CT2	CT3	CT4	CTn-1	CTn
Req. 1	■	■		■		
Req. 2	■		■	■	■	■
Req. 3		■	■	■		
Req. N	■	■	■		■	■

Stato CT	**IT**	**ST**	**UAT**
Totale	127	43	21
Eseguiti	122	41	12
Completati	121	23	7
Delta	**- 6**	**- 20**	**- 14**

Rimozione	**REQ**	**DES**	**CUT**	**IT**	**ST**	**UAT**
Review	7	18	23			
Testing			84	46	27	3
Total	**7**	**18**	**107**	**46**	**27**	**3**

Produttività	**IT**	**ST**	**UAT**
CT eseguiti	127	43	21
G/P spesi	43	27	24
Produttività	**2,95**	**1,59**	**0,87**

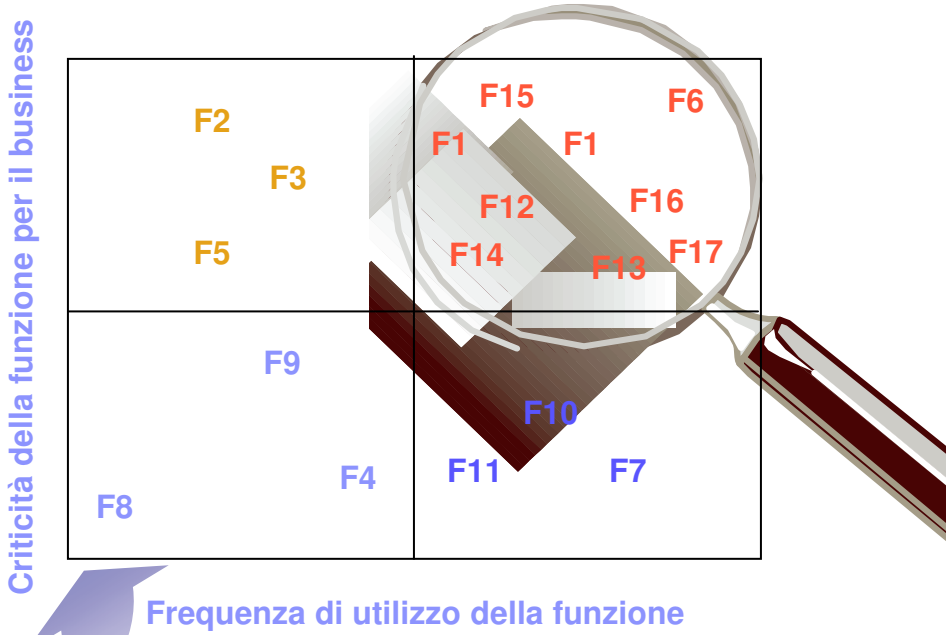
Correzione	**IT**	**ST**	**UAT**
Rilevati	46	27	3
Corretti	42	21	3
Verificati	41	21	3
Delta	**- 5**	**- 6**	**0**

Strategia di test

Quali e quanti test eseguire, quale livello di copertura occorre raggiungere, a quale livello di profondità occorre arrivare?

I test devono:

- *tener conto dei rischi identificati*
- *essere costruiti partendo dai requisiti, dai casi d'uso e dalle specifiche tecniche*
- *indirizzare sia le funzionalità che le caratteristiche di qualità (performance, usabilità, robustezza, operatività, integrazione, portabilità ecc.)*
- *verificare le "condizioni d'errore" previste e gestite*
- *verificare le "condizioni limite" e le condizioni "al contorno"*
- *avere una base dati progettata opportunamente (oracolo)*



In una rappresentazione delle funzionalità e delle caratteristiche del software basata sulla frequenza di utilizzo e sulla loro criticità, il quadrante in alto a destra costituisce l'obiettivo principale dei collaudi (vedi elementi in rosso).

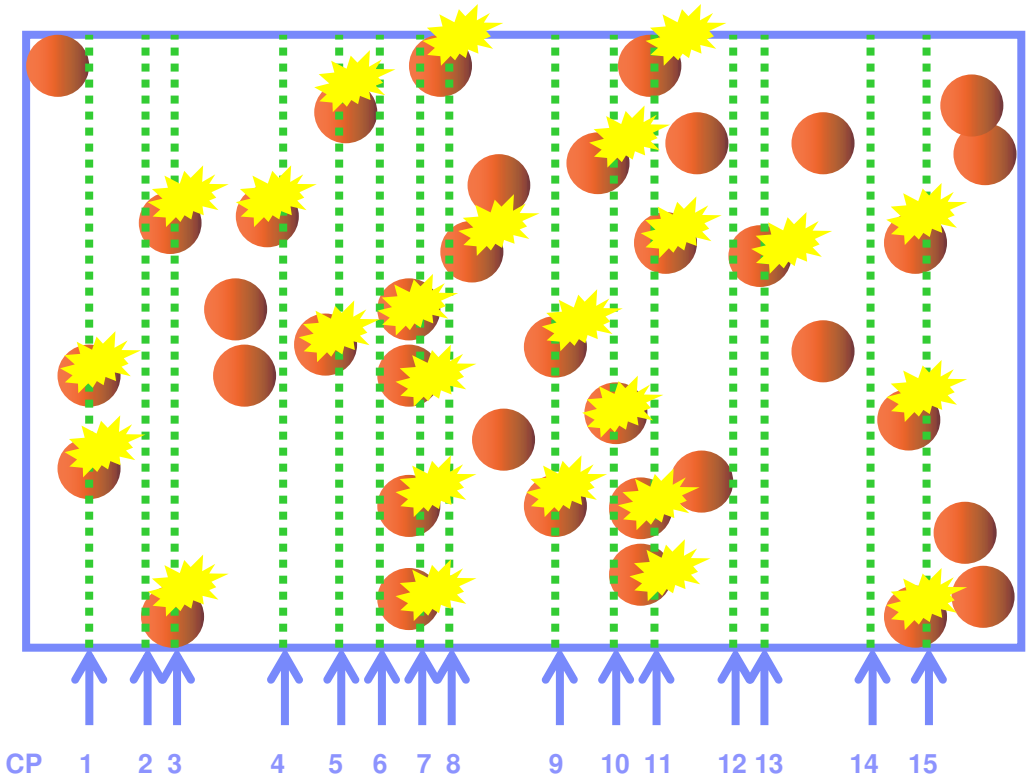
"Quanto testare?"

Se non si conosce quanti e quali tipi di errori siano presenti in un software è molto difficile pensare che si possano rimuoverli tutti con l'esecuzione dei test.

I 15 casi di prova (CP) progettati e mostrati nella figura con delle frecce e relative linee tratteggiate sono in grado di "scoprire" solo parte degli errori presenti.

Gli errori non scoperti rimangono "latenti" nel software.

Se invece si conosce il "profilo" di rimozione degli errori (vedi "Curva di rimozione degli errori") allora sarà possibile valutare se il numero di errori rilevati con i 15 casi di prova progettati sia in linea con la previsione statistica oppure no!



Nota: nella figura le palle rosse rappresentano gli errori nel software, le frecce i casi di prova, le linee tratteggiate il percorso dei casi di prova, le "esplosioni" la scoperta degli errori da parte dei casi di prova.

"Quanti errori rimangono dopo la fase di test?"

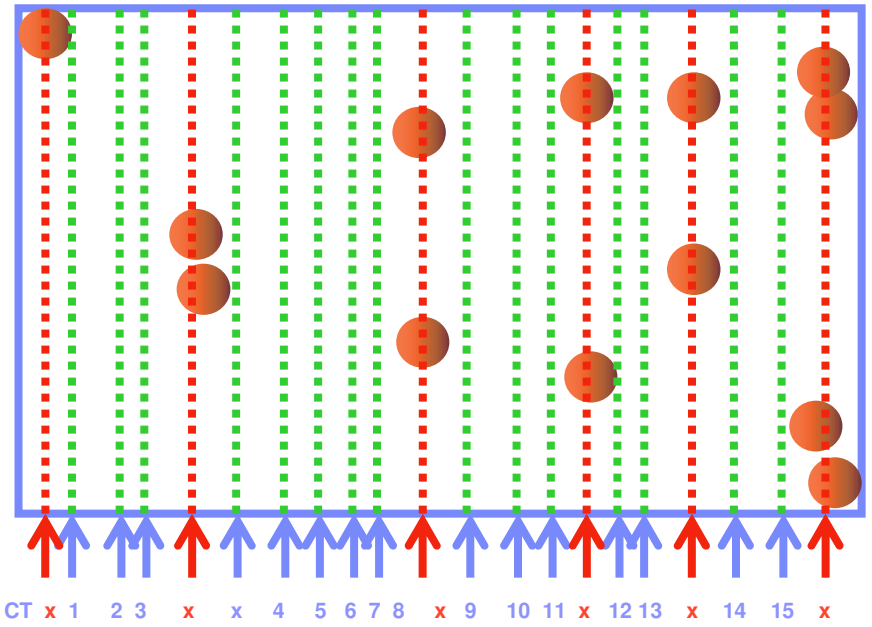
Il caso mostrato in figura rappresenta una situazione in cui i 15 casi di prova non hanno rimosso tutti gli errori "attesi".

Ovviamente gli errori residui non possono essere visti come in figura, ma la previsione statistica ci dirà che ci sono!

L'analisi dei risultati evidenzia che il numero di errori rilevati è inferiore a quello previsto dal modello statistico.

Occorre quindi progettare nuovi casi di prova per rimuovere gli errori residui (vedi frecce in rosso).

Una buona progettazione dei casi di prova consentirà di "scoprire" gli errori latenti!



Nota: nella figura i percorsi dei casi di prova sono lineari solo per motivo di semplicità grafica. In realtà il percorso di un caso di prova all'interno del software è un camino molto complesso e tortuoso ed è rappresentato dalle istruzioni di codice eseguite!

Curva di rimozione degli errori durante la fase di test

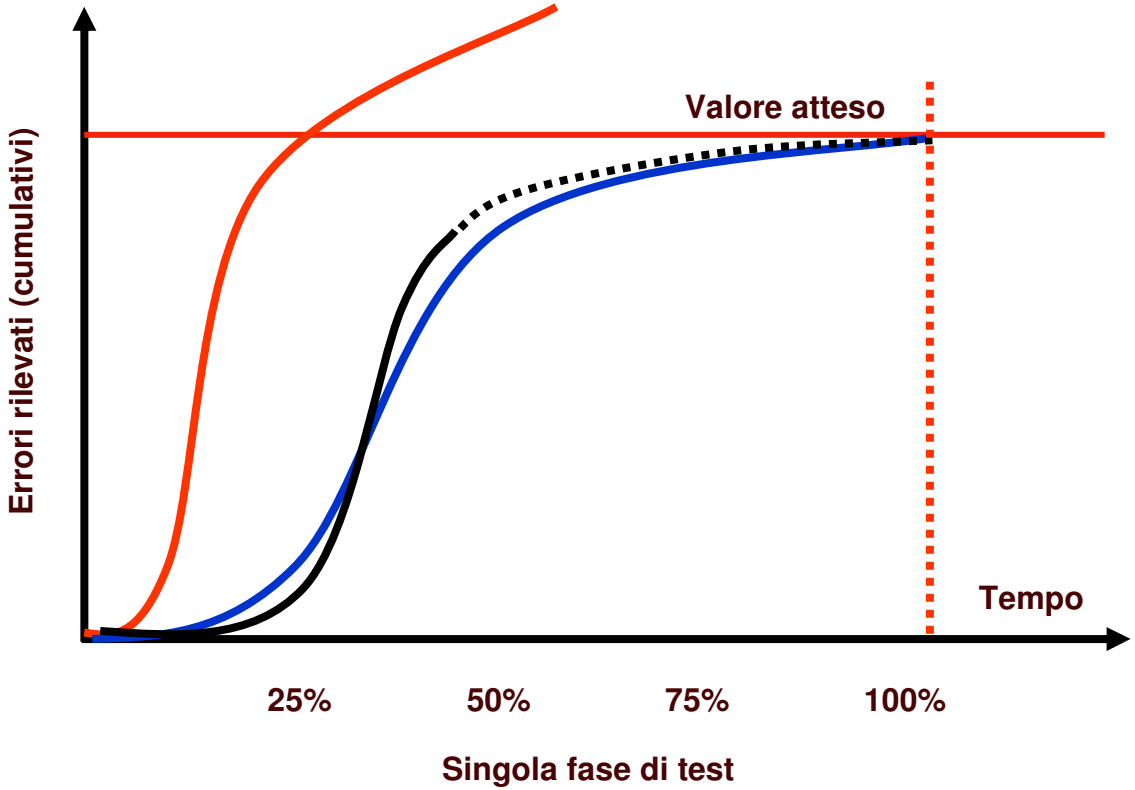
La rappresentazione del numero cumulativo di errori rilevati nel tempo durante lo svolgimento di una singola fase di test acquista la forma mostrata in figura.

Al completamento del 50% dei casi di prova eseguiti la curva tende ancora a salire, mentre al completamento del 75% dei casi di prova eseguiti la curva tende ad appiattirsi.

L'asintoto verso cui tende la curva rappresenta il numero massimo di errori che la fase di test è in grado di rilevare.

L'utilizzo del modello statistico ("profilo") consente di prevedere il valore atteso (l'asintoto).

L'utilizzo di entrambe le tecniche ("profilo" e "curva di rimozione dei test") permettono di controllare statisticamente l'efficacia dei test e la difettosità residua.



Nota: la curva rossa indica che il software ha una difettosità superiore a quella prevista. Occorrerà quindi prolungare la fase di test fino a vedere la curva appiattirsi verso un nuovo asintoto (di valore superiore a quello atteso).

In questo caso il modello statistico risulta essere impreciso perché non sono state eseguite con cura le fasi precedenti e/o non sono stati registrati tutti valori rilevati.

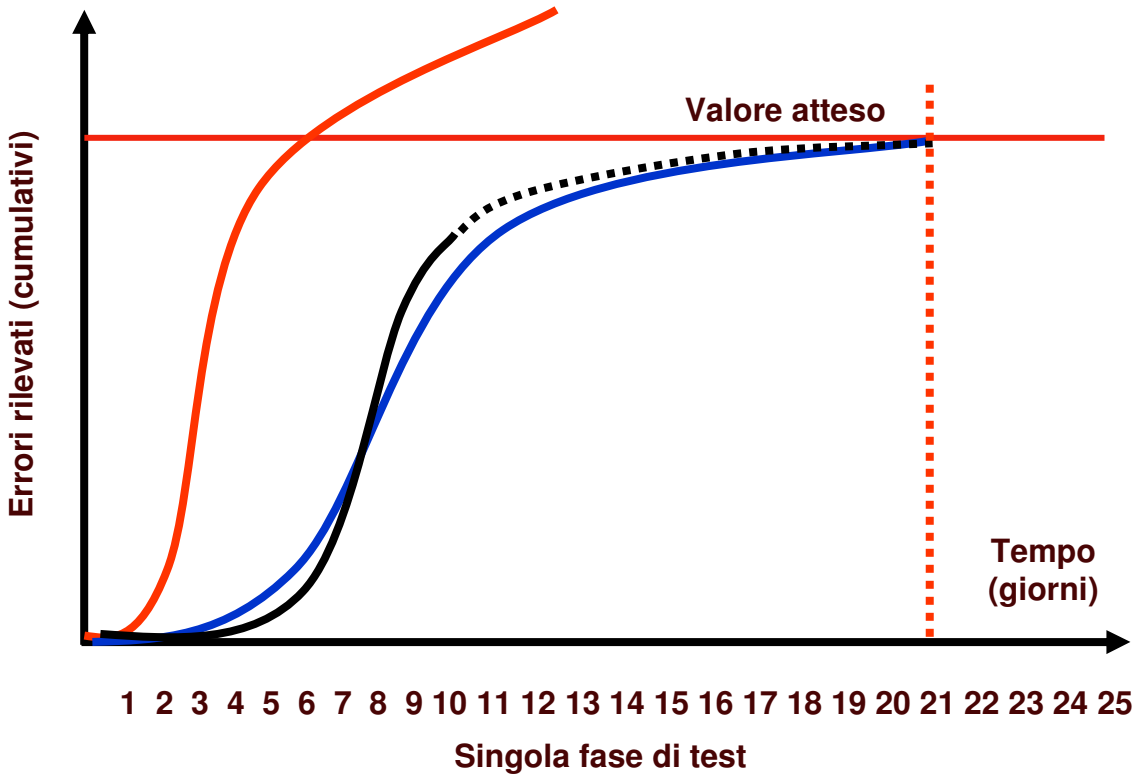
In pratica …

Un modo pratico di costruire la curva di rimozione degli errori durante una fase di test consiste nel registrare ogni giorno il numero di errori rilevati e riportare il valore cumulativo su di un grafico come quello mostrato in figura.

La tecnica è quella di eseguire il numero massimo di casi di prova possibili.

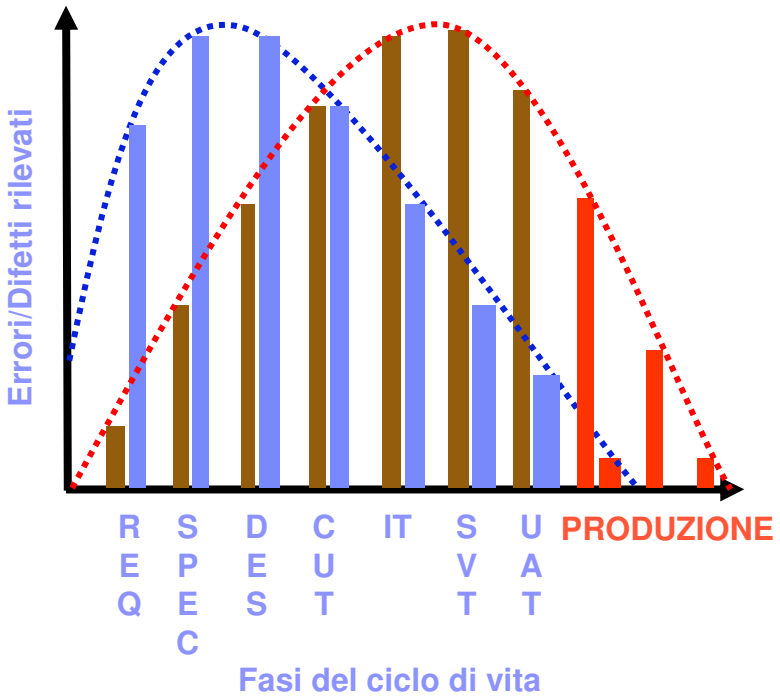
Molti di questi si bloccheranno per via di errori gravi che non permettono il loro completamento ma, appena risolti gli errori rilevati, i casi di prova proseguiranno la loro esecuzione rilevando un numero sempre maggiore di errori.

L'appiattimento della curva indicherà il momento in cui si è esaurita la capacità dei casi di prova di rilevare altri errori (nell'esempio dopo 21 giorni si può considerare completata la fase di test in oggetto).



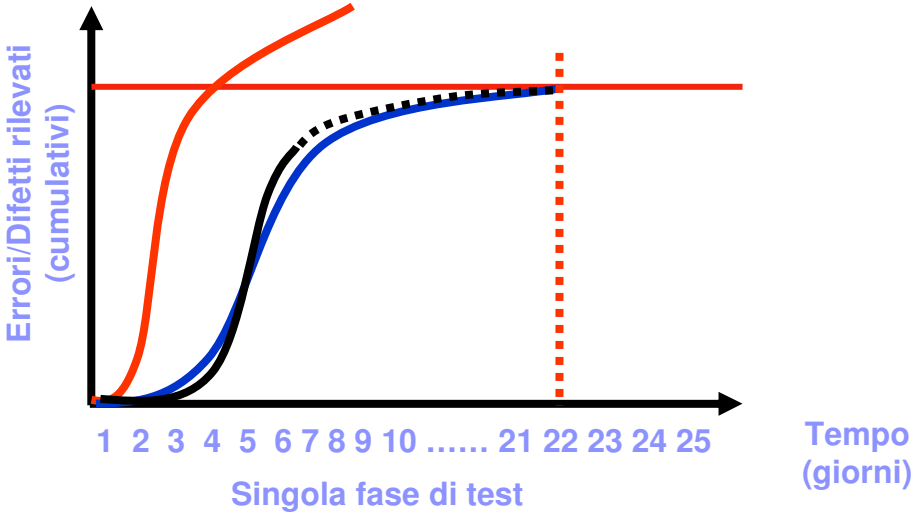
Nota: l'appiattimento della curva prima del valore "atteso" è dovuto ad una di queste possibili cause: 1) il software è meno difettoso di quanto previsto (poco probabile!); 2) i casi di prova non sono adeguati (in numero ed efficacia) a rilevare tutti gli errori; 3) il valore atteso non è corretto (il modello non è adeguato).

Concludendo …

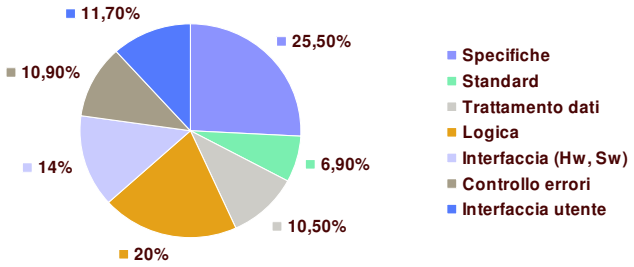


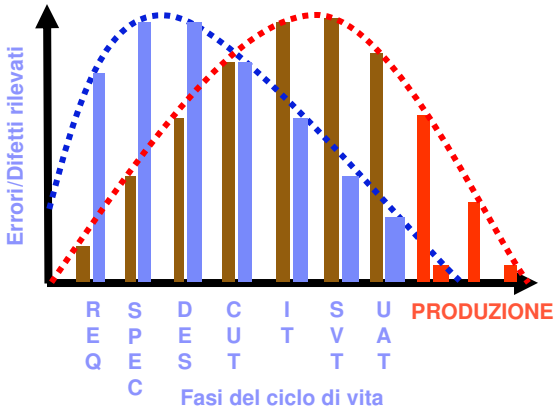
"Occorre assolutamente conoscere il profilo dell'organizzazione che sviluppa il software in oggetto e ...

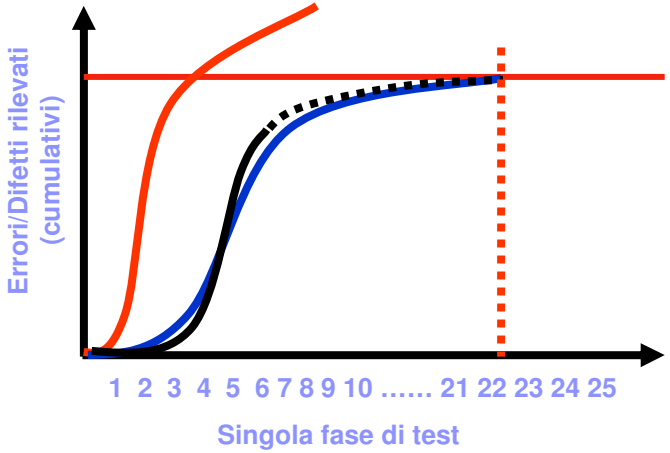
... occorre verificare la curva di rimozione degli errori nelle fasi di test e collaudo!"

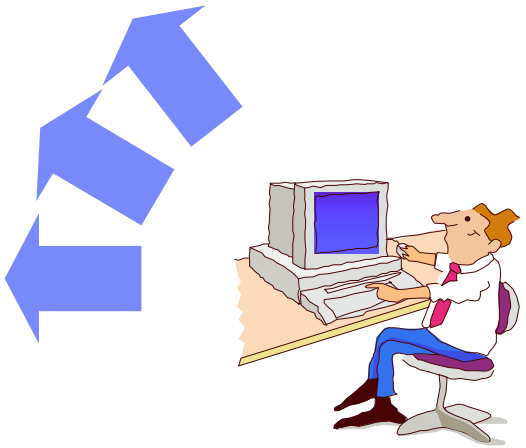


Rappresentazioni statistiche

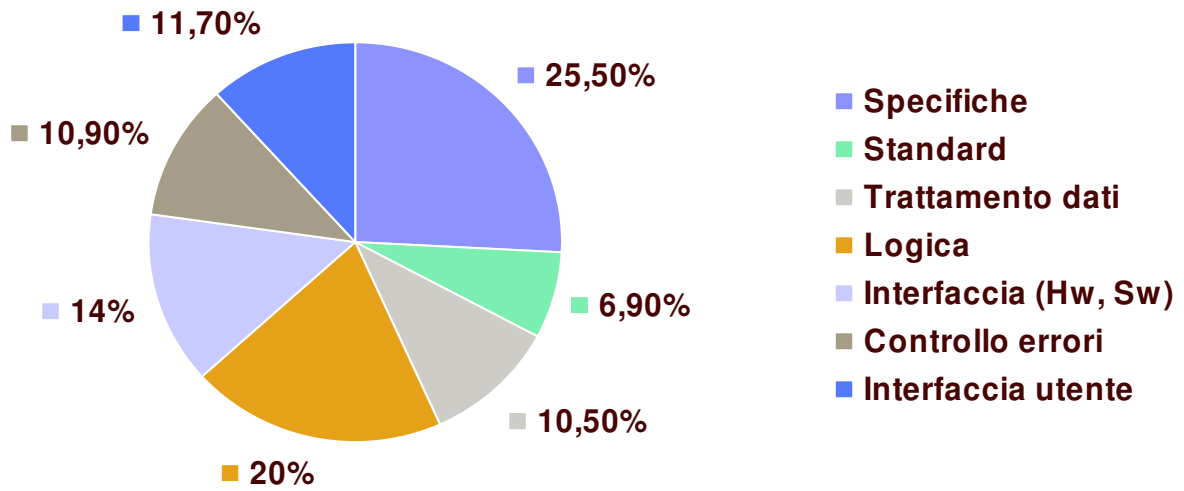






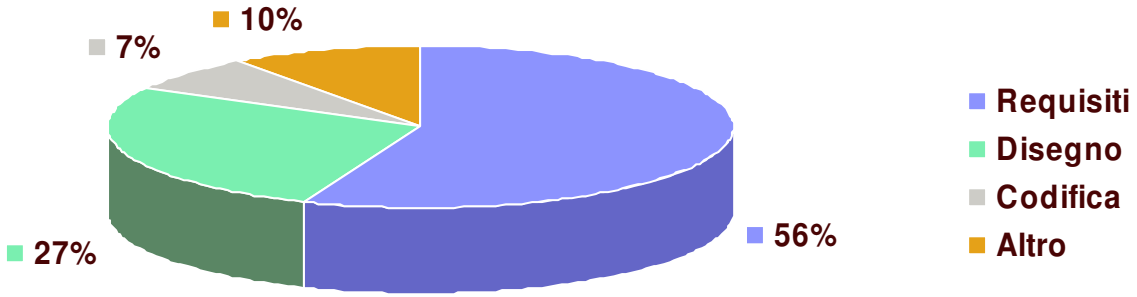


Difettosità rilevata nelle diverse fasi di sviluppo del software



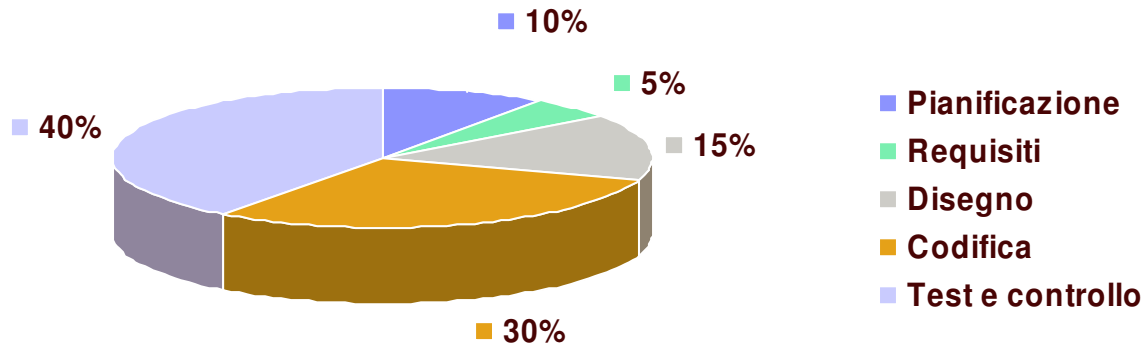
Cause e origine dei difetti di quattro progetti software, Grady, 1994

Inserimento di difetti nel software per fase



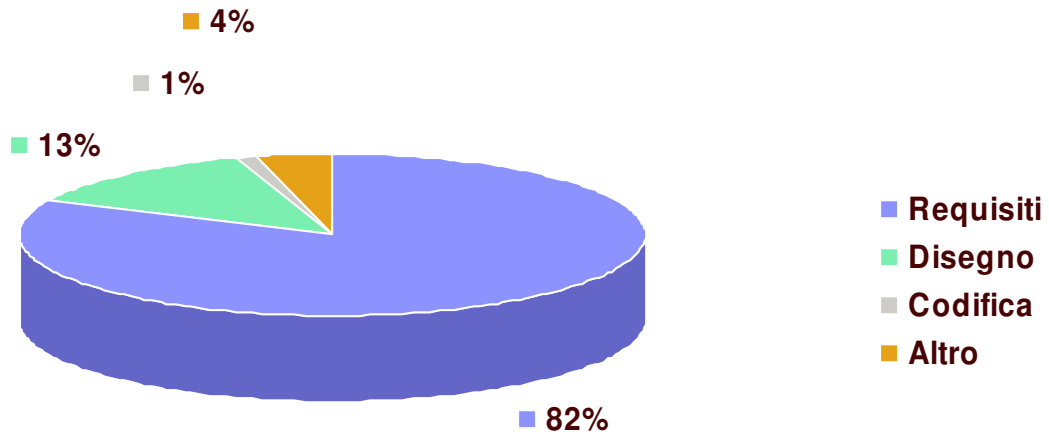
IBM, 1994

Costo delle fasi dello sviluppo software



IBM, 1994

Costo di rimozione degli errori per tipologia (fase di inserimento)



IBM, 1994