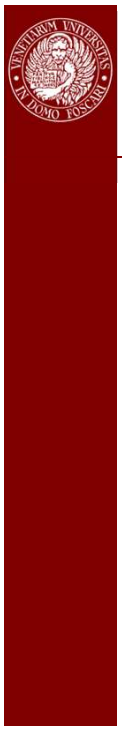
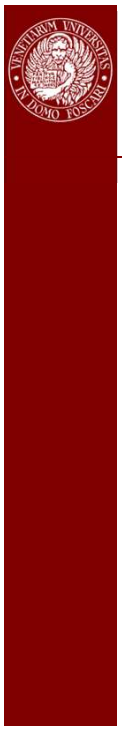


Corso di Project Management

Il Test del software

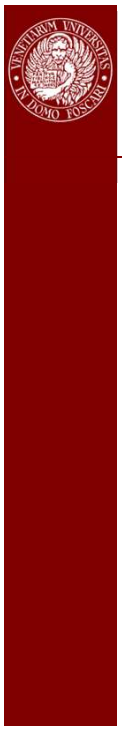
Roberto D'Orsi **Anno Accademico 2013/2014**



Il Test del software

Alcune osservazioni sul testing (segue)

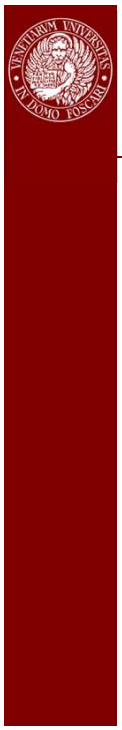
- Nei modelli dei cicli di sviluppo del SW, il *testing* viene spesso rappresentato come una fase posta tra lo sviluppo e il rilascio in produzione: in realtà il *testing* è un'attività complessa che si trova distribuita lungo tutto il ciclo di vita (il *test* del codice o la validazione del prodotto SW rispetto ai requisiti rappresentano solo due delle varie attività di *testing*)
- Sull'attività di *testing* non si deve cercare di risparmiare: in un progetto di sviluppo SW può anche superare il 30% dei costi complessivi, se si vuole un ragionevole livello di Qualità
- La "regola d'oro" per un progetto SW suggerisce: Analisi 25%, Disegno 25%, Codifica 10%, Ispezioni, Test e Collaudo 40%
- Tesi di Dijkstra: il test può rilevare la presenza di malfunzioni, ma non può dimostrare che il prodotto è privo di errori
- Non esiste un software con zero difetti: quindi il *testing* è un'attività che non si può mai considerare definitivamente conclusa. Dal punto di vista pratico, a seconda della delicatezza del prodotto SW, va trovato un giusto compromesso tra completezza dei test e costo
- I casi di *test* vanno definiti già durante fase di progettazione concettuale, completati e arricchiti durante quella tecnica



Il Test del software

Alcune osservazioni sul testing

- La pianificazione dei *test*, in termini di tempo e di risorse da dedicare, è sempre fortemente condizionata dai tempi di rilascio, per cui molto spesso il tempo dedicato al *testing* è quello che va dalla conclusione delle attività di sviluppo alla data di consegna al Cliente, indipendentemente dai risultati raggiunti durante le prove
- La progettazione dell'attività di *testing* è spesso informale e basata sull'esperienza, il "fiuto" e la memoria dello sviluppatore, manca la predisposizione di un piano
- La registrazione delle anomalie riscontrate è carente o del tutto assente, con l'immediata conseguenza che non vi sono quasi mai dati storici da cui partire le volte successive in cui si devono eseguire nuovamente i *test* su quella applicazione, oppure quando si vuole pianificare un piano di *test* per un altro progetto con caratteristiche simili
- Il processo di *test* è un tipico esempio di approccio *bottom-up*, in quanto si parte quasi sempre dai singoli oggetti (*unit test*) per arrivare all'intero sistema (*system test* e collaudo)
- In realtà esiste anche un secondo tipo di approccio al *testing*: quello *top-down*, ma in questo caso, per poter procedere, è necessario costruire degli *stub*, cioè dei tronconi fittizi usa e getta che servono a simulare la presenza dei moduli ancora mancanti



Il Test del software

Riesami, Verifiche, Validazioni in un Progetto ICT (segue)

- **Riesami:**

  Il riesame è un'attività che ha lo scopo di valutare, in un determinato momento, lo stato di un progetto o, in generale, di un processo e i risultati che sta producendo, alla ricerca di difetti e di possibili soluzioni

  Esempi di riesami:

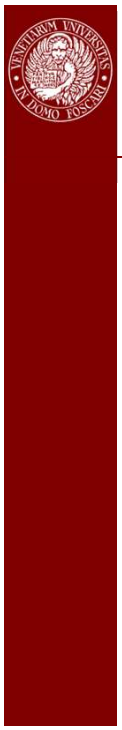
✓ Controlli dello **Stato Avanzamento Progetto** da parte del Project Manager

✓ **Joint-Review**: incontri tra *project team* e committente per riesaminare requisiti, stato del progetto ed eventuali rischi

✓ **Peer-Review**: si tratta di revisioni tecniche, di incontri con esperti esterni al progetto per valutare completezza dei requisiti, analisi, disegno, architettura, codice, ecc.

✓ Sessioni di **Project Assurance**, per verificare conformità ai requisiti, agli standard, adeguatezza del processo e dei *deliverable*, identificare eventuali aree di rischio non considerate, ecc.

✓ Ispezioni da parte di colleghi di altri gruppi di lavoro (***inspection*** e ***walkthrough***) su disegno, codice (*code inspection*), casi di *test*, ecc, con l'obiettivo di trovare gli errori, non di giudicare il lavoro fatto o la persona, ma nemmeno di trovare le soluzioni



Il Test del software

Riesami, Verifiche, Validazioni in un Progetto ICT (segue)

- **Verifiche:**

  La verifica è un'attività che ha lo scopo di valutare se l'output di una determinata fase di un progetto, o in generale di un processo, è conforme agli *standard* adottati e ai requisiti imposti dall'output della fase precedente. Si effettua tramite un'ispezione o una revisione.

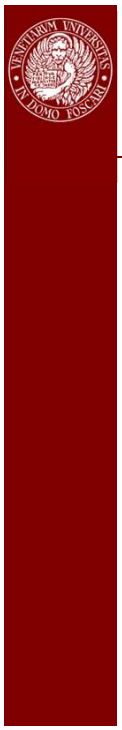
  "Stiamo costruendo il prodotto nel modo giusto?" (Barry Boehm 1979)

  Il focus è quindi sulla conformità dei processi del ciclo di vita rispetto agli standard e ai requisiti interni

  Esempi di verifiche:

✓ Verifica (revisione) ed approvazione dei documenti di Progetto, che descrivono la conclusione di una fase ed i risultati raggiunti

✓ Verifiche (*test*) del software, in vari punti del suo ciclo di sviluppo



Il Test del software

Riesami, Verifiche, Validazioni in un Progetto ICT (segue)

La verifica può essere basata su un'analisi **statica**, cioè effettuata "a tavolino", senza eseguire il codice (verifica delle regole di codifica, dei diagrammi di flusso, della struttura del SW, del contenuto, delle metriche, della tracciabilità, dell'adeguatezza dei piani di test, della documentazione), oppure **dinamica**, cioè effettuata eseguendo il codice e osservandone il comportamento (esecuzione del piano di *test*, verifica di copertura del piano di *test*)

La verifica statica consente di rimuovere gli errori prima che si propaghino nelle fasi successive, consentendo così di ridurre i costi di rimozione

- **Validazioni:**

✓ Test dei deliverable realizzati rispetto ai requisiti iniziali dell'utente

✓ Validare un prodotto significa assicurare che è adatto ad essere utilizzato nell'ambiente per cui è previsto

"Stiamo costruendo il prodotto giusto?" (Boehm 1979)

Il focus è quindi sulla conformità del prodotto rispetto ai requisiti utente

Il Test del software

Riesami, Verifiche, Validazioni in un Progetto ICT (segue)

Esempi di verifiche statiche:

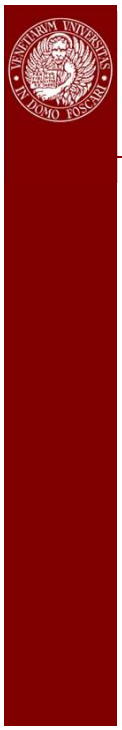
- ✓ I requisiti del software soddisfano i requisiti della fase precedente?
- ✓ I requisiti del software sono tra loro consistenti?
- ✓ Le specifiche di disegno del software sono complete e consistenti rispetto ai requisiti del software?
- ✓ Le specifiche dell'architettura e quelle relative al disegno del software sono tra loro consistenti?
- ✓ La documentazione associata al software è consistente, coerente, completa, rispetto a quanto il software è in grado di fare?
- ✓ Il piano di *test* è adeguato per provare sia l'architettura che il disegno complessivo del software?
- ✓ Il disegno del software e dell'architettura su cui si appoggia sono conformi rispetto al Piano di Qualità?
- ✓ Gli *standard* di progettazione e sviluppo sono stati applicati in modo corretto?
- ✓ In definitiva, la documentazione tecnica che verrà utilizzata per sviluppare il codice è completa, corretta, non ambigua, consistente?

Il Test del software

Riesami, Verifiche, Validazioni in un Progetto ICT

Esempi di verifiche dinamiche:

- ✓ Unit Test
- ✓ Integration Test
- ✓ System Test
- ✓ Validation Test
- ✓ User Acceptance Test
- ✓ Regression Test
- ✓ Smoke Test
- ✓ Performance Test
- ✓ Stress Test
- ✓ Recovery Test
- ✓ Security Test
- ✓ Reliability Test
- ✓ Usability Test
- ✓ Quality Test
- ✓ Test di copertura funzionale
- ✓ Test di ripristino (backup & recovery)
- ✓ Test di installazione



Il Test del software

Modalità di test white-box (testing strutturale)

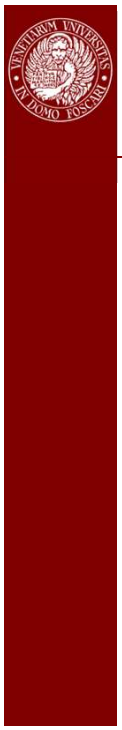
Il verificatore conosce la logica e la struttura interna del software che deve testare e quindi prepara i casi di prova in modo da poter coprire il più possibile:

- ✓ Le istruzioni: in modo da eseguire almeno una volta tutte le parti del codice che è stato scritto
- ✓ Le decisioni: in modo da eseguire almeno una volta tutti i rami logici (cammini linearmente indipendenti) che derivano dai nodi decisionali contenuti nel codice, comprese tutte le possibili scelte multiple. Questa verifica consente anche di rilevare l'eventuale presenza di cammini non eseguibili dovuti a condizioni contraddittorie che non potranno mai presentarsi

La complessità di un piano di prova *white-box* aumenta esponenzialmente con il grado di copertura che si vuole garantire

Copertura: grado con cui strutture di controllo, archi, nodi, predicati,….. sono attivati almeno una volta nella fase di *test*

Il piano di test va quindi predisposto valutando attentamente anche il rapporto costi/benefici



Il Test del software

Modalità di test black-box (testing funzionale)

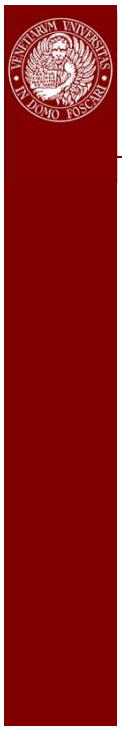
Il verificatore conosce la logica del software che deve testare, ma ne ignora del tutto la struttura interna.

Il *testing* é fondato sull'analisi degli *output* generati dal sistema o da suoi componenti, confrontati con i risultati attesi (oracolo), in risposta ad *input* definiti sulla base della sola conoscenza dei requisiti in ingresso.

Quindi i casi di prova vanno costruiti prendendo in considerazione tutte le possibili combinazioni tra dati di ingresso e risultati in uscita, scegliendo in modo significativo i valori di prova.

Ogni funzionalità (oppure ogni caso d'uso) devono essere eseguiti almeno una volta.

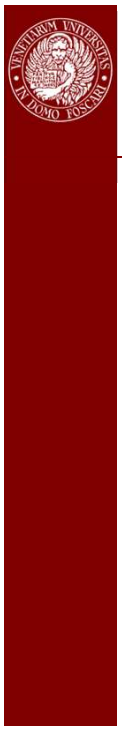
Vi sono varie tecniche, da usare congiuntamente, per la predisposizione dei casi di prova: *test* positivi, *test* negativi (valori non ammessi o fuori dominio), valori di confine (ai limiti del dominio), valori speciali, combinazioni causa/effetto (gruppi di valori in *input* per ottenere determinati risultati), *error guessing* (metodo intuitivo basato sull'esperienza del verificatore),….



Il Test del software

Cosa contiene un Piano di Test (segue)

- Quali sono i componenti del sistema da sottoporre a verifica
- Quali sono, per ogni componente, gli obiettivi del *test*
- Come tracciare i *test* rispetto ai requisiti
- Quale processo di *testing* verrà adottato
- A che tipologia di *test* verrà sottoposto ogni componente e quali sono i livelli di profondità da raggiungere
- Quante e quali Risorse Umane verranno impiegate per effettuare i *test* e quali livelli di professionalità e quali *skill* sono richiesti
- Chi ha la responsabilità di certificare il processo di *test*
- Quali strumenti saranno necessari per effettuare il *test*
- Quali sono i requisiti HW e SW per l'ambiente di *test*
- Qual'è la procedura di registrazione dei risultati dei *test*
- Quali sono gli eventuali vincoli e rischi di cui tener conto
- Quali sono i casi di *test* sia per i requisiti funzionali che non funzionali



La Gestione della Documentazione

Cosa contiene un Piano di Test

- Definizione di tutti i momenti di revisione, verifica, *audit*, *testing*, necessari per abbattere drasticamente gli errori del prodotto SW
- Definizione della gerarchia di *testing*: dal singolo modulo fino alla prova dell'intero sistema in un ambiente il più possibile simile a quello *target*
- Definizione di scopo, obiettivi, regole, metodi, risorse necessarie (*skill* e quantità) necessarie per ogni attività
- Definizione dei criteri di ingresso e di uscita
- Definizione delle modalità di gestione e risoluzione dei problemi che si dovessero incontrare
- Definizione dei criteri di classificazione della severità dei problemi
- Predisposizione dei casi di prova sia per i requisiti funzionali (cosa deve fare il sistema) che per quelli non funzionali (come il sistema lo deve fare, requisiti qualitativi/prestazionali compresi)
- Formulazione del risultato atteso a fronte di ogni caso prova
- Qual'è il criterio per decidere che i *test* possono considerarsi conclusi

Il Test del software

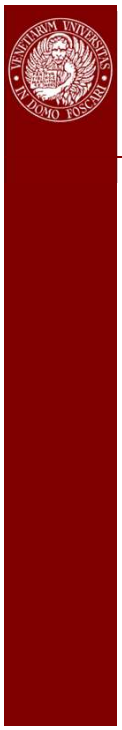
La registrazione dei Test (esempio)

Codice Requisito	Predecess.	Funzionale/ Non Funz.	Caso prova	Risultato atteso	Esito OK - NO OK	Eseguito da	Data test

Il Test del software

I vari momenti della fase di Test (segue)

- **Unit Test (Test di Unità o di Modulo):**
  - ✓ Ha l'obiettivo di individuare gli errori nel singolo modulo software (a seconda dei casi è un Riesame o una Verifica)
  - ✓ Verifica che il SW faccia tutto quello che deve fare
  - ✓ Verifica che il SW non faccia quello che non deve fare
  - ✓ Si effettua in ambiente di sviluppo
  - ✓ Viene effettuato dal programmatore che ha sviluppato il modulo con un approccio di tipo *white-box*
  - ✓ Viene documentato mediante una *checklist* che riporta i requisiti funzionali da soddisfare, i casi prova e l'esito dei singoli *test*
- **Integration Test (Test di Integrazione):**
  - ✓ Ha l'obiettivo di individuare gli errori nel software quando tutti i moduli che compongono un sottosistema o l'intero sistema vengono fatti lavorare insieme (è un esempio di Verifica)
  - ✓ Si effettua, nella maggior parte dei casi, in ambiente di sviluppo con un approccio quasi sempre di tipo *white-box*
  - ✓ Viene effettuato congiuntamente dal gruppo dei programmatori che hanno sviluppato i vari moduli
  - ✓ Viene documentato mediante una *checklist*



Il Test del software

I vari momenti della fase di Test (segue)

- **System Test (Test di Sistema)**

✓ Ha l'obiettivo di garantire che il prodotto SW nel suo complesso soddisfi completamente i requisiti iniziali (collaudo interno):
  - Funzionali (Test Funzionale)
  - Non Funzionali (affidabilità, stabilità, usabilità, manutenibilità, portabilità, interoperabilità, sicurezza, modificabilità, scalabilità, tempi di risposta, apprendibilità, installabilità, sostituibilità, ecc.)
  - Di documentazione (help online, manuali, corsi programmati,….)

✓ Viene effettuato in un ambiente di *test* in grado di riprodurre nel modo più simile possibile le condizioni dell'ambiente di produzione

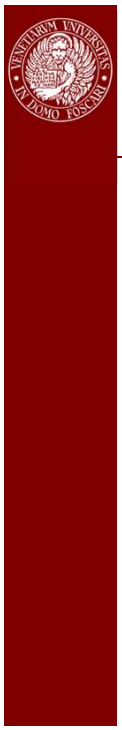
✓ Viene effettuato simulando le condizioni di esercizio (approccio sempre di tipo *black-box*): dovrebbero essere attive nell'ambiente di *test* anche le altre applicazioni che normalmente girano nell'ambiente *targe*t

✓ E' di responsabilità del Project Manager, coadiuvato dal *team* di *test* e dall'analista che ha sviluppato la progettazione concettuale

✓ Richiede a priori la stesura di un Piano di *System Test* che indica tutte le Funzioni e Sottofunzioni da sottoporre a *test*

✓ Viene documentato mediante una *checklist* che riporta i requisiti funzionali e non funzionali casi prova e l'esito dei singoli *test*

✓ E' un esempio di Validazione



Il Test del software

I vari momenti della fase di Test (segue)

- **User Acceptance Test (Collaudo, Test di Accettazione, Test di Validazione)**

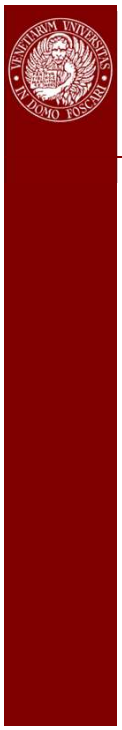
✓ Ha l'obiettivo di valutare insieme al Cliente la rispondenza dell'applicazione SW rispetto ai requisiti espressi inizialmente nel contratto (il SW è pronto per entrare in produzione?) e di ottenere la sua accettazione formale di quanto realizzato (= OK al pagamento)

✓ Viene effettuato dal Cliente (che ne è responsabile) nel suo ambiente di produzione oppure, ancora meglio, se possibile, in un apposito ambiente di test gemello di quello di produzione in termini di HW, SW installato e ambiente operativo

✓ Viene effettuato da un gruppo di utenti del Cliente, assistiti dal Project Manager del fornitore, sulla base di un insieme di casi prova che essi hanno predisposto (approccio sempre di tipo *black-box*)

✓ Viene documentato mediante un verbale di collaudo che viene sottoscritto dal Cliente, nel quale devono essere riportati: grado di soddisfazione, suggerimenti, eventuali errori riscontrati, modifiche da effettuare, tempi richiesti per la loro risoluzione

✓ E' un esempio di Validazione



Il Test del software

I vari momenti della fase di Test (segue)

- **Alpha Test e Beta Test**

✓ Un **Alpha Test** è un *test* preliminare di un'applicazione SW, anche non ancora completa, eseguito da alcuni potenziali utenti rappresentati da un *team* all'interno dell'organizzazione che ha sviluppato il SW

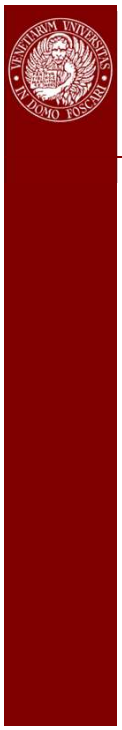
✓ Un **Beta Test** viene eseguito presso l'organizzazione cliente e ha l'obiettivo di far valutare al Cliente, prima del *roll-out* ufficiale del sistema, la reale funzionalità, completezza ed operatività dell'applicazione SW rispetto alle esigenze espresse, in qualche caso confrontate con i risultati di un'applicazione obsoleta che verrà sostituita (**Parallelo o Test di parallelo**)

✓ Viene effettuato da un gruppo di utenti presso uno o più Clienti pilota in ambiente di produzione, che effettuano una serie di prove, spesso con "dati a perdere" per verificare la rispondenza del prodotto SW rispetto alle esigenze operative (approccio *black-box*)

✓ La responsabilità di pianificare e coordinare le attività di Beta Test, di preparare i casi prova, di rilevare e valutare i risultati è interamente del Cliente

✓ Il responsabile del Progetto (o il Responsabile del Prodotto) forniscono supporto esterno al Cliente e redigono un verbale finale contenente grado di soddisfazione, eventuali anomalie, suggerimenti, ecc.)

✓ E' un esempio di Validazione



Il Test del software

I vari momenti della fase di Test (segue)

- **Regression Test (Test di non Regressione)**

✓ Ha l'obiettivo di verificare, a valle di una manutenzione correttiva, se il *bug* è stato individuato e corretto: pertanto vanno eseguiti esattamente gli stessi *test* che erano stati effettuati quando era stato individuato il problema

✓ Serve inoltre a verificare che non vi siano altri *bug* correlati con quello che è stato risolto, per i quali è sufficiente apportare le stesse modifiche

✓ Comunque il *regression test* va eseguito anche a seguito di un intervento di manutenzione evolutiva, per verificare che le modifiche effettuate non abbiano fatto degradare il sistema

✓ Inoltre le modifiche apportate al codice potrebbero aver generato nuovi problemi su altre parti del prodotto non modificate : quindi occorre controllare il corretto funzionamento delle parti di programma che potrebbero essere state compromesse dalle modifiche

✓ Soprattutto quando l'applicazione è piuttosto complessa, per eseguire un *test* di regressione sono particolarmente utili gli strumenti di *test* automatico

✓ A seconda del grado di copertura del *testing*, è una Verifica o una Validazione

Il Test del software

I vari momenti della fase di Test (segue)

- **Smoke Test**

✓ Il termine *smoke test* nasce nell'ambiente elettrico/elettronico e consiste in una prima prova, molto grossolana, di un'apparecchiatura che, una volta assemblata o riparata non deve generare fumo, scintille o altri effetti macroscopici, quando viene alimentata

✓ In modo del tutto simile, per *smoke test* in campo informatico si intende un insieme di prove, effettuate ad esempio dopo un intervento di manutenzione correttiva, atte a garantire che l'integrità funzionale del sistema informatico (il sistema "sta in piedi")

✓ Si tratta quindi di un primo *test* di massima dell'applicazione, non particolarmente approfondito, nel quale si verifica che tutte le funzionalità principali rispondano ai requisiti in base ai quali sono state progettate, in particolare se sono intervenute delle modifiche di manutenzione correttiva o evolutiva

✓ Uno *smoke test* eseguito con successo costituisce il punto di partenza per qualsiasi altro tipo di *test*

✓ E' un esempio di Verifica

Il Test del software

I vari momenti della fase di Test

- **Stress Test o Test di Carico**

✓ Lo *stress test* fa parte, insieme a quelli di *usability*, di *performance*, di *security*, di *reliability*, dei *test* di sistema e ha l'obiettivo di determinare il punto di rottura di un sistema software (*breakpoint),* oltre il quale si verificano perdite di dati e/o un'interruzione del servizio

✓ Il *test* viene eseguito utilizzando dei *tool* in grado di simulare il possibile carico previsto per il sistema sotto prova, di misurare le prestazioni e di verificare in che condizioni di carico il sistema inizia a degradare

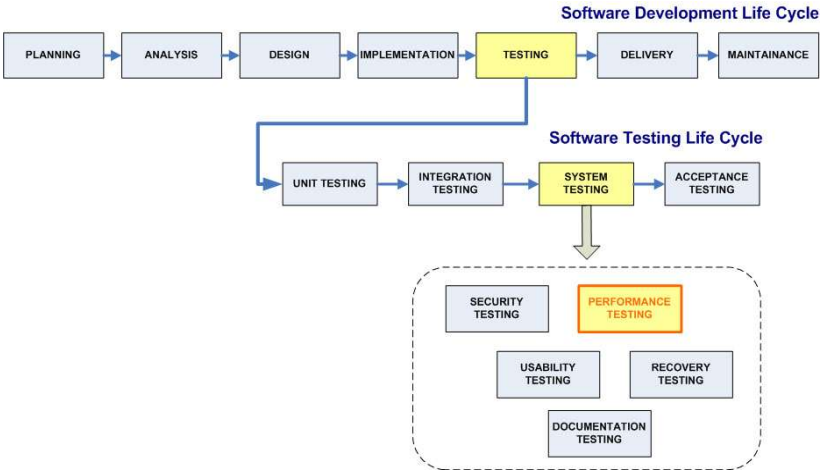
✓ Per effettuare uno *stress test* viene simulata una situazione di carico costante pari ai picchi di utilizzo misurati in ambiente di produzione (massimo numero di utenti concorrenti)

✓ Consente di valutare il comportamento del sistema a fronte di un aumento non previsto del carico e di verificare gli effetti dovuti all'instabilità che si viene a creare

✓ Lo *stress test* serve anche per quantificare i tempi di ritorno ad una situazione di regime, quando il carico ritorna a livelli normali

Il Test del software

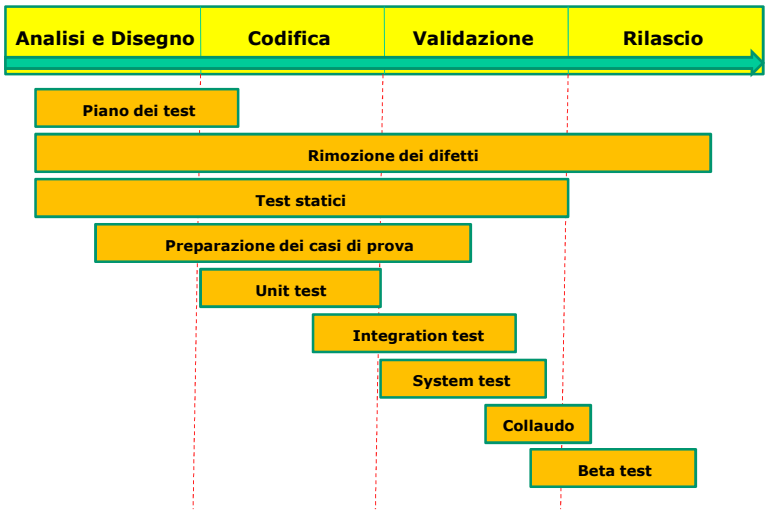
Posizionamento dei test prestazionali nel ciclo di vita del SW



Fonte: G. Zanetti

Il Test del software

I processi di testing lungo il ciclo di vita del prodotto



Il Test del software

Criteri generali per la fase di test (segue)

- E' quasi impossibile sviluppare un'applicazione software priva di errori: è quindi indispensabile avere il tempo e la capacità di trovarli e correggerli attraverso un adeguato processo di *test*
- Il costo del *test* è in realtà un investimento perché consente di ridurre i costi nelle successive fasi di rilascio in produzione e di esercizio
- L'obiettivo del *test* dovrebbe essere quello di trovare il maggior numero di errori con il minimo numero di casi prova
- I *test* devono essere effettuati sulla base di un Piano di Test che individui le prove da effettuare, i risultati attesi, i prerequisiti di ingresso, i criteri di accettazione
- Il Piano di Test va progettato in modo coerente con gli obiettivi qualitativi e con quanto previsto nel Piano della Qualità
- Nelle parti del prodotto dove è stato trovato il maggior numero relativo di errori è bene concentrarsi maggiormente, alla ricerca di possibili errori residui

Il Test del software

Criteri generali per la fase di test (segue)

- In particolare il Piano di Test stabilisce la gerarchia con cui i *test* vanno eseguiti, le regole ed i metodi di esecuzione, le risorse necessarie, i criteri di accettazione, le modalità di gestione dei problemi riscontrati, un criterio di classificazione della *severity* dell'errore
- Una volta progettato, il Piano di Test va assoggettato a verifica della progettazione dei casi di test
- Devono essere preventivamente definiti gli indicatori ed i criteri per misurare le caratteristiche di accettabilità dei prodotti SW, in particolare per quanto riguarda i requisiti non funzionali
- Durante l'esecuzione dei *test*, devono essere registrate le prove effettuate, i risultati ottenuti, le anomalie riscontrate, le correzioni da apportare
- Viene quindi emesso un rapporto (che costituisce una Registrazione della Qualità), che contiene gli esiti delle prove, l'analisi dei risultati e le azioni da intraprendere

Il Test del software

Criteri generali per la fase di test

- Deve esistere un ambiente di *test* che si appoggia su archivi di prova stabili, in modo da garantire che le prove siano ripetibili. Inoltre i dati devono essere sufficientemente significativi e il più possibile coerenti con i dati reali con cui girerà l'applicazione
- Deve esistere un sistema in grado di garantire la tracciabilità dei *test* eseguiti rispetto ai requisiti iniziali e alle funzioni da realizzare
- Deve anche esistere la tracciabilità tra esito del *test*, modifiche effettuate, moduli software coinvolti dalla modifica, ripetizione del *test*
- A fronte di modifiche apportate al SW a causa di errori rilevati in fase di *test*, oltre a ripetere il *test* per verificare il buon esito dell'operazione va anche effettuato il *Regression Test* dell'intera applicazione

Il Test del software

Considerazione finale: dove si annidano gli errori?

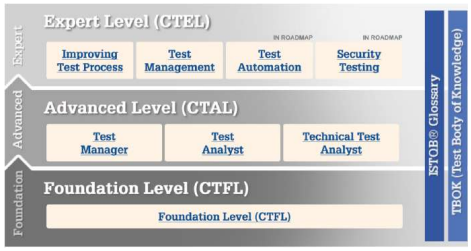
Specifiche incomplete o errate	22%	Analisi 53%
Fraintendimento delle comunicazioni del Cliente	17%	
Errori nella rappresentazione dei dati	14%	
Collaudo incompleto o errato	10%	
Errore nella codifica di una specifica tecnica	6%	
Inconsistenza nell'interfaccia tra componenti	6%	
Errore logico nella progettazione	5%	
Deviazione intenzionale rispetto alle specifiche	5%	
Documentazione imprecisa o incompleta	4%	
Violazione degli standard di programmazione	3%	
Interfaccia uomo/macchina ambigua o inconsistente	3%	
Altro	5%	
	100%	

Fonte: E.Colonese

Il Test del software

La certificazione professionale in Software & System Testing

- **Foundation**: comprende gli elementi di base del software testing ed è propedeutico ai livelli successivi
- **Advanced Test Manager**: rilascia un attestato di specializzazione per chi deve gestire attività di testing
- **Advanced Test Analyst**: è la specializzazione che deve avere chi progetta test funzionali in ambito applicativo
- **Advanced Technical Test Analyst**: per chi opera in prevalenza in ambito tecnico e in particolare ha necessità di avere la padronanza di tecniche di copertura del codice sorgente



ITA-STQB
ITAlian Software Testing Qualifications Board
www.ita-stqb.org
e-mail: info@ita-stqb.org

Il Test del software

Esempio: gli argomenti dell'esame Foundation

Fondamenti del SW Testing	IlTesting all'interno del ciclo di vita del software	Tecniche di test statico	Tecniche di progettazione dei test	Test management	Strumenti di test
Elementi di base	Modelli di sviluppo software	Le ispezioni e il processo di testing	Condizioni di test e progettazione dei casi di prova	Organizzazione delle attività di testing	Tipologie di strumenti
Definizione di testing	Livelli di test	Il processo di ispezione (review)	Tecniche di progettazione dei test	Pianificazione	Uso degli strumenti: benefici e rischi
Principi generali	Tipologie di testing	Analisi statica e strumenti associati	Tecniche basate su specifiche (black-box)	Controllo avanzamento	Introduzione degli strumenti di test
Processo di testing	Test di regressione		Tecniche basate su aspetti strutturali (white-box)	Configuration Management	
Aspetti psicologici del testing			Tecniche basate su esperienza	Risk Management	
			Selezione della tecnica di testing più appropriata	Gestione degli errori	