**TESTING AND CONFIGURATION**

**VERIFICA E VALIDAZIONE**

**Verifica**= Stiamo costruendo bene il prodotto? Il sistema rispetta i requisiti?

**Convalida**= Stiamo costruendo il prodotto giusto? Il cliente sarà soddisfatto del software che stiamo realizzando?

Questi due processi andrebbero **applicati ad ogni stage** nella realizzazione del progetto

**OBIETTIVI**

Hanno ***due principali obiettivi***:

* La scoperta di difetti nel nostro sistema
* Certificare il sistema funzioni o no

Questi due processi dovrebbero convalidare che il progetto stia facendo quello che deve fare (*non vuol dire senza difetti, ma utile per l’uso previsto*)

**FIDUCIA**

Essa dipende dallo scopo del sistema, dalle aspettative degli utenti e dell’ambiente di marketing

* Funzione software
  + Il livello di confidenza dipende da quanto il software è critico per un’organizzazione
* Aspettative degli utenti
  + Gli utenti possono avere basse aspettative su alcuni software (determinato da scarsa pubblicità, beta piena di bug)
* Ambiente di marketing
  + Portare un prodotto sul mercato, sponsorizzandolo come un ottimo software con pochi bug presenti, puo farci vendere prima che veniamo scoperti e spendere più soldi per la manutenzione dopo

**VERIFICA STATICA DINAMICA**

**Ispezione software (verifica statica)**= Analisi della rappresentazione del sistema statico per scoprire problemi, documenti basati su strumenti e analisi del codice

**Test del software (verifica dinamica)**= Si preoccupa del comportamento del prodotto, il sistema viene eseguito con i dati di test e viene osservato il suo comportamento operativo

Meglio spendere di più su programmatori bravi e veloci a scrivere, ed effettuare più test dinamici

L’ispezione è integrata nella programmazione di coppia nella programmazione estrema (Uno scrive ed uno controlla il codice)

TEST DEL PROGRAMMA

Puo rivelare presenza di errori ma NON la loro assenza

Dovrebbe essere usata alla verifica statica per una copertura completa

TIPI DI TEST

* Test dei difetti
  + Test progettatti per scoprire i difetti del sistema
  + Test di difetti di successo 🡪 se rivela presenza di difetti in un sistema
* Test di convalida
  + Destinato a dimostrare che il software soddifsa i requisti del cliente

TEST E DEBUGGING

Test= V&C si occupano di stabilire l’esistenza di difetti di un programma

Debug= si occupa di individuare e riparare gli errori

IL PIANO DI TEST DEL SFOTWARE

Immagine che contiene testo

Descrizione generata automaticamente

ISPEZIONI E TEST

* Ispezioni e prove di test sono complementari e non opposte

ISPEZIONI

* Approccio formalizzato alle revisioni dei documenti
* Destinato per rilevamento del difetto
* Difetti possono essere errori logici, anomalie nel codice che potrebbero indicare una condizione errata (Es. una var non inizializzata)
* specifica precisa

VERIFICA E MODALITA’ FORMALI

* I metodi formali possono essere utlizzati quando viene prodotta una specifica matematica del sistema
* Ultima tecnica di verifica statica
* Implicano un’analisi matematica dettagliata della specifica e possono sviluppare argometi formali, che un programma è conforme alla sua specifica matematica
* **Serve personale altamente specializzato e costa tanto**

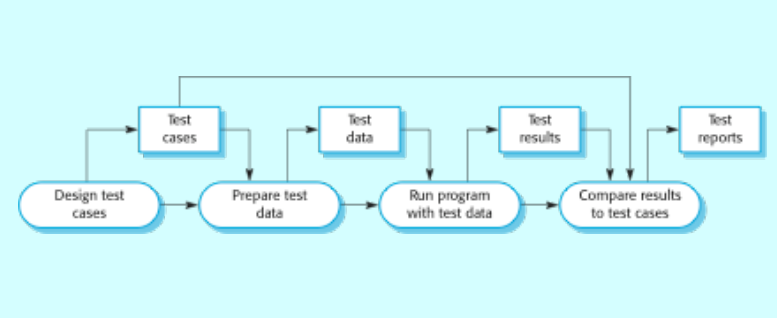
**CLEANROOM SOFTWARE DEVELOPMENT**= La filosofia è evitare i difetti piuttosto che rimuoverli

DETTAGLI SUL TEST DEL SOFTWARE

IL PROCESSO DI TEST

* Test dei componenti e test del sistema
* Test dei singoli componenti del programma e collaudo di gruppi di componenti integrati per realizzare un sistema o sottosistema

IL PROCESSO DI TEST DEL SOFTWARE



**POLICY DEI TEST**

* Solo test approfonditi possono dimostrare che un programma è privo di difetti
* Le **politiche di test definiscono l’approccio** da utilizzare nella selezione dei test di sistema:
  + Tutte le **funzioni** accessibili da **menu** devono essere **testate**
  + **Testare** le **combinazioni di funzioni** a cui si accede **tramite** lo stesso **menu**
  + Dove è richiesto **input dell’utente**, tutte le **funzioni** devono essere **testate** con **input corretto e errato**

**TEST DEL SISTEMA**

* Test d’integrazione = Test effettuato quando i componenti sono integrati
* Test di rilascio = Test completo del sistema, assicurandoci che funzioni per lo scopo preventivato

CASI D’USO

* Base per derivare i test per un sistema
* Da un diagramma di sequenza associato si possono identificare gli ingressi e le uscite da creare per le prove

TEST DELLE PRESTAZIONI

* Parte del test di rilascio
* I test delle prestazioni di solito comportano la pianificazione di una serie di test in cui il carico viene costantemente aumentato fin quando le prestazioni del sistema diventano inacettabili

TEST DA SFORZO

* Esercita il sistema oltre il suo carico massimo
  + Stressarlo fa spesso emergere dei difetti
* I sistemi non dovrebbe fallire in modo catastrofico
* Lo stess trest verifica la perdita inaccettabile di servizi o dati (Molto utile per i sistemi distrbuiti)

TEST CASE DESIGN

* **Obiettivo** = creare una serie di test efficaci nella convalida e nel test dei difetti
* **Approcci:**
  + **Test basati sui requisiti**
  + **Test delle partizioni**
  + **Prove strutturali**

TEST BASATI SUI REQUISITI

* Un principio dell’ingegneria dei requisiti è che i requisiti dovrebbero essere testabili
* Test basato sui requisiti è una tecnica di test di convalida in cui, si considera ogni requisto e si ricava una serie di test per quel requisito

TEST DELLE PARTIZIONI

LINEE GUIDA PER I TEST (Sequenze)

TEST DEL PERCORSO

* Obiettivo = garantire che l’insieme dei test case sia tale che ogni percorso attraverso il programma venga eseguito almeno una volta
* Il punto di partenza è un grafico di flusso del programma, che mostra i nodi che rappresentano le decisioni del programma e gli archi che rappresentano il flusso di controllo
* Istruzioni con condizioni 🡪 nodi del grado di flusso

AUTOMAZIONE DEI TEST

* Il test è una fase di processo costosa, infatti abbiamo una gamma di strumenti per ridurre il tempo richiesto e i costi totali
* Sistemi come JUNIT supportano esecuzione automatica dei test, in pratica scriveremo i test da mettere in input una sola volta, e verranno eseguiti in automatico ogni volta che c’è ne sia bisogno

**GESTIONE DELLA CONFIGURAZIONE(CM)**

* Essa riguarda la gestione dei sistemi software in evoluzione, coinvolgendo lo sviluppo e l’applicazione di procedure e standard
* Mirando a controllare i costi e gli sforzi necessari per apportare modifiche a un sistema

PIANIFICAZIONE:

* Questo lavoro agisce su tutto (Design,Specifiche,Dati di test,Programmi)
* Possono essere creati migliagli di documenti separati per un softw complesso

IDENTIFICAZIONE DELL’ELEMENTO DI CONFIGURAZIONE

CAMBIO GESTIONE

* La gestione del cambiamento si occupa di tenere traccia di questi cambiamenti e di garantire che vengano implementati nel modo più conveniente
* Le modifiche dovrebbero essere riviste da un gruppo esterno che decide se sono convenienti o meno da un punto di vista strategico e organizzativo piuttosto che da un punto di vista tecnico

VERSIONI/VARIANTI/RELEASE

* + **Versione**: Istanza del sistema funzionalmente distinta (abbiamo cambiato/aggiunto qualche funzionalità per migliorarla) da altre istanze di sistema
  + **Variante**: Istanza del sistema, funzionalmente identica(Mi fa fare le stesse cose, per esempio guidare un auto elettrica) ma non funzionalmente distinta (cambia da un punto di vista non funzionale, performance ecc..) da altre istanze di un sistema (Versione Free e Versione Pro, per esempio)
  + **Release**: Istanza di sistema, distribuita agli utenti al di fuori del team di sviluppo

IDENTIFICAZIONE DELLA VERSIONE

* Le procedure per l'identificazione della versione dovrebbero definire un modo univoco per identificare le versioni dei componenti
* Esistono tre tecniche di base per l’identificazione:
  + Numerazione versione (2.0, 2.1 , 2.1.2 e cosi via)
  + Identificazione basata sugli attributi
  + Identificazione orientata al cambiamento

GESTIONE DEI RILASCI

* Le versioni devono incorporare le modifiche imposte al sistema da errori scoperti dagli utenti e da modifiche hardware
* La pianificazione del rilascio riguarda il momento in cui rilasciare una **versione** di sistema come **release**

RILASCI DI SISTEMA

* Può includere anche:
  + File di configurazione che definiscono come è configurata la versione per una particolare installazione;
  + File di dati necessari per il funzionamento del sistema;
  + Un programma di installazione o uno script di shell per installare il sistema sull'hardware di destinazione;
  + Documentazione elettronica e cartacea;
  + Packaging e pubblicità associata

PROBLEMI DI RILASCIO

* Il cliente potrebbe non volere una nuova versione del sistema, essendo soddisfatti del loro sistema attuale
* Tutti i file necessari per una versione devono essere ricreati quando viene installata una nuova versione

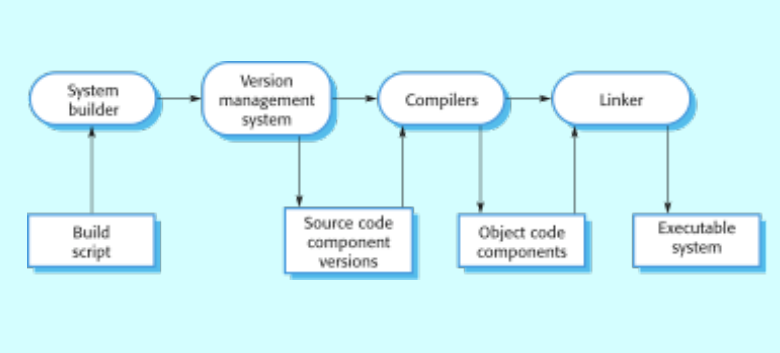
STRATEGIA DI RILASCIO DEL SISTEMA

Immagine che contiene testo

Descrizione generata automaticamente

COSTRUZIONE DEL SISTEMA

* E’ il processo di compilazione e collegamento di componenti software in un sistema eseguibile
* Questo processo è ora sempre supportato da strumenti automatizzati



STRUMENTI CASE PER LA GESTIONE DELLA CONFIGURAZIONE

* Il supporto dello strumento CASE per la CM è essenziale, essendo necessario gestire grandi quantità di dati

STRUMENTI CASE

* Strumenti di gestione del cambiamento
  + Editor di moduli, sistema di flusso di lavoro (Chi fa cosa)
  + Modifica banca di dati, sistema di segnalazione delle modifiche (report di gestione)
* Strumenti di gestione delle versioni
  + Identficiazione della versione e della release
  + Gestione del magazzino (memorizzare diifferenze tra le versioni, registra motivi per la creazione della versione, modifica registrazione cronologia)
  + Sviluppo al progetto (gestire gruppi di file piuttosto che singoli file)
* Strumenti per la creazione di sistemi possono fornire
  + Linguaggio e interprete per la specifica delle dipendenze
  + Compilazione distribuita
  + Gestione degli oggetti derivati