KER (ING SOFTWARE MOD 1)

- Caratteristiche prodotto software:
  - **Intangibile**: difficile da descrivere e valutare
  - **Malleabile**: puo essere trasformato e dotato di nuove funzionalità
  - **Human intensive**: non comporta nessun processo manifatturiero tradizionale

- Qualità processo software:
  - **trasparenza:** capacità di capire lo stato in cui si trova il processo
  - **produttività:** misura di efficienza in termini di velocità di consegna del processo
  - **tempestività:** capacità di rispettare i tempi di consegna

- qualità esterne black box view:
  - **correttezza**: se rispetta tutte le specifiche funzionali di progetto
  - **affidabilità**: probabilità di eassenza di fallimenti in un dato intervallo di tempo
  - **efficienza**: se usa intelligentemente le risorse di calcolo
  - **usabilità**: facilità di uso da parte dell'utente
  - **portabilità**: se funziona su piu piattaforme
  - **interoperabilità**: abilità di un sistema di coesistere e cooperare con altri sistemi
  - **robustezza**: se si comporta in modo ragionevole anche in circostanze non previste nelle specifiche di progetto

- qualita interne white box view:
  - **riusabilità**: se può essere utilizzato in tutto o in parte per costruire nuovi sistemi
  - **verifcabilità**: possibilità di dimostrare a posteriori la correttezza del software
  - **facilità di manutenzione**: facilità nel realizzare adattamenti o evoluzioni, agio nel correggere errori
- la sicurezza è un concetto legato al contesto ovvero significa che è qualcosa che può essere definito rispondendo alla domanda "sicurezza da chi e da cosa?" ciò implica stabilire delle policy di sicurezza
- un SW sicuro deve avere le seguenti proprietà:
  - **prevenzione**: anticipare possibili attacchi
  - **monitoraggio**: auditing in tempo reale, spesso a piu livelli e pieno di falsi positivi
  - **tracciabilità**: meccanismo per il quale è possibile in modo inequivocabile le relazioni di causa effetto tra elementi eventi o processi
  - **auditing**: processo di controllo di un sistema effettuato sulla base del confronto tra le attività svolte sul sistema con le policy di sicurezza
  - **privatezza**: diritto di un individuo di stabilire se come quando e a chi informazione che lo riguarda può essere rilasciata
  - **confidenzialità**: assicura che certi servizi e informazioni siano accessibili solo ad utenti autorizzati
  - **sicurezza a diversi livelli**: tipologia di sicurezza per la quale alcuni tipi di dati e informazioni sono piu segrete di altre quindi necessitano di maggiore segretezza.
  - **anonimato**: proprietà di mantenere segreta l'origine di certi dati
  - **autenticazione**: proprietà di conoscere chi accede ad un servizio
  - **integerità**: proprietà per la quale le risorse non vengono modificate in modo improprio

- le attività del processo sono organizzate in
    - **studio di fattibilità** = produzione documento FSD che valuta costi e benefici della applicazione proposta
    - **specifica** = determinare le funzionalità e le proprietà del SW. Essa produce il RASD. Processo incrementale che richiede continua iterazione col cliente
    - **progettazione** = produzione di un documento contenente la descrizione dell'architettura del SW a livello globale e a livello dei singoli moduli. Produce il Design Document
    - **implementazione e test moduli** = il nome dice tutto…
    - **integrazione e test del sistema** = assemblare il codice prodotto dai diversi programmatori e verificarne l'effettiva compatibilità risolvendo eventuali errori
    - **consegna** = il nome dice tutto…
    - **manutenzione** = il nome dice tutto…

MODELLI CICLO DI VITA DEL SOFTWARE

- modelli di ciclo di vita del software:
    - modelli a **cascata**
        - completamento di una fase prima di passare a quella successiva
        - processo blackbox
        - non tiene in considerazione il fatto che i requisiti cambiano.
            - Prototipazione
    - modelli **iterativi**
        - permettono di avere un maggiore feedback:
        - modello approssimato dell'applicazione che ha lo scopo di fornire il feedback necessario ad individuare caratteristiche del sistema ed eventuali errori di progettazione
            - Modello a fasi di release
                - si parte da sottoinsiemi semplici ma critici sui quali richiedere il feedback del cliente
            - Modello a spirale
                - ottimale per sviluppo di sistemi sicuri
                - modello ciclico tra: analisi rischi, sviluppo, testing e revisione della release
                - ingloba la prototipazione
                - sviluppo continuo
            - Modello a V
                - estende modello a cascata
                - dimostra relazione tra fase di ciclo di vita e fase di testing
    - modelli **agili**
        - coinvolgono quanto piu possibile il committente ottenendo una elevate reattività alle sue richieste

SICUREZZA

- il ciclo di sicurezza segue il seguente corso:
    - definizione **policy di sicurezza** = dice cosa è e cosa non è consentito
        - meccanismi di sicurezza sono metodi/strumenti che applicano le policy
    - **analisi dei rischi** = ha lo scopo di identificare i rischi possibili e valutare strategie per evitarli
        - livello di protezione: funzione della probabilità che un attacco si verifichi e degli effetti dell'attacco qualora succeda
    - **sicurezza e specifica** = la specifica deve integrare eventuali possibili soluzioni a rischi individuati
    - **sicurezza e design**
    - **sicurezza e testing** = si divide in
        - testing funzionale = comporta il mettere alla prova un sistema per determinare se il sistema fa ciò che si suppone debba fare in circostanze normali o critiche
        - testing di sicurezza = comporta il mettere alla prova un sistema allo stesso modo in cui può provarlo un utente malizioso.

- il ciclo di vulnerabilità è il seguente:
    - scoperta vulnerabilità
    - sviluppo di una patch
    - installazione patch

- vi sono diverse classi di valutazioni del prodotto in base alla sicurezza:
    - **D** = **minimal protection** = no meccanismi di protezione
    - **C** = **discretionary protection** = controllo di sicurezza discrezionale
    - **B** = **mandatory protection** = controllo obbligatorio
    - **A** = **verified protection** = uso di metodi formali per la verifica di sicurezza

- **common criteria** = definiscono un insieme di classi e componenti progettate per essere opportunamente combinate per definire profili di protezione per ogni tipo di prodotto IT
- **common evaluation methodology** = definisce le modalità di valutazione di un sistema in base ai criteri comuni
- attacco = qualsiasi atto malevolo contro un sistema o un complesso di sistemi. Le fasi di un attacco sono:
    - goal
    - sottogoal
    - attività
    - eventi
    - conseguenze
    - impatto

- **Graceful degradation** = quando si verifica un guasto il Sistema non si ferma ma continua in modo ristretto o con funzionalita' ridotta
- **fault safely** = in caso di un fallimento un sistema deve terminare in una configurazione sicura
- accettabilità psicologica = utilizzo di modelli mentali e paradigmi tratti dal mondo reale
- **auditability** = deve essere possibile ricostruire la sequenza di eventi che hanno condotto a certe azioni chiave
- **modularizzazione** = suddividere la progettazione in moduli

SPECIFICHE

- le specifiche possono essere
  - **Informali** = definite mediante linguaggio naturale
  - **semi formali** = possibilmente grafici (ad es. UML)
  - **formali** = tramite formalismi operazionali e formalismi dichiarativi
- a loro volta vi sono diversi formalismi:
  - formalismi **operazionali**
    - definiscono il sistema descrivendone il comportamento come se eseguito da una macchina astratta
  - formalismi **dichiarativi**
    - definiscono il sistema dichiarando le proprietà che esso deve avere
- specifica dei requisiti = descrizione completa e non ambigua dei requisiti. è un accordo tra il produttore di un servizio e il suo committente.
- I requisiti possono essere
  - Funzionali
  - Non funzionali
  - del processo e manutenzione

- una specifica deve avere le seguenti caratteristiche:
  - **Chiarezza**
  - **Non ambiguità**
  - **Consistenza** = no contraddizioni
  - **Completezza** = le modalità di funzionamento descritte dalla specifica debbono essere definite in modo completo e dettagliato
  - **Incrementalità** = sviluppata in piu passi
  - **comprensività** = intuitiva e comprensibile per il cliente
- UML = Linguaggio di modellazione unificato

TESTING

- il test/collaudo software di un programma consiste nell'eseguirlo con alcuni casi di test e controllare che il comportamento sia corretto. Esso deve essere
    - automatizzato
    - riguardare ogni fase di sviluppo
    - esteso a tutti i componenti di un sistema
    - essere pianificato
    - seguire standard e metodologie

- un failure è il funzionamento non corretto del programma
- un bug è un elemento del programma sorgente non corrispondente alle aspettative
- un errore è un fattore umano che causa una deviazione tra il software prodotto e il programma ideale
- ci sono diversi tipi di testing:
    - accettazione: comportamento del sw è confrontato con i requisiti dell'utente finale
    - conformità: comportamento del sw è confrontato con le specifiche dei requisiti
    - sistema: comportamento dell'intero sistema come monolitico
    - integrazione: controllo sul modo di cooperazione delle unità
    - unità: test del comportamento delle singole unità. Per ogni metodo testato si introducono
        - test driver: metodo che chiama il test unit con opportuni parametri
        - test stub: metodo che sostituisce eventuali metodi usati dal test unit per testare in modo isolato e controllato
    - regressione: test del comportamento di release successive
- in base al tipo di accessibilità si parla di
    - **whitebox testing**
        - si assume che il programma sorgente sia disponibile
    - **blackbox testing**
        - si assume che non si guardi il programma sorgente ma solo quello che dovrebbe fare
- un programma P è una funzione da un dominio D ad un codominio R
    - OK(P,d) con d in D se P è corretto per input d
    - OK(P) è corretto se per ogni d in D OK(P,d)
    - un **test set T** è un sottoinsieme finito di D
    - un **test** è un elemento di D
    - un **test criteria** è quindi una funzione C che dato un programma P, la sua specifica S e un test set T restituisce vero o falso. Può essere visto come un generatore di test set dato P ed S. Possono essere

- Teorema di **Goodenough e Gerhart** = dato un criterio C un programma P se C è affidabile e T test set selezionato con C e T non trova malfunzionamenti in P allora P è corretto
- i criteri possono essere
    - **AFFIDABILI** = Un criterio C si definisce affidabile se, per ogni coppia (T1, T2) di test set adeguati al criterio C, se T1 trova un malfunzionamento nel programma, anche T2 trova il malfunzionamento e viceversa.
    - **VALIDI** = criterio che mi permette di definire almeno un test set che mi permetta di trovare il problema
    - **IDEALI** = se sono sia validi che affidabili

- un test set è adeguato per testare un progamma P secondo il criterio di **copertura delle istruzioni/statement coverage** se per ogni istruzione s di P esiste un caso di test in T che esegue s
- un test set soddisfa il criterio di **copertura degli archi** o **branch coverage** di P se e solo se **ogni arco** del grafo di P viene percorso almeno una volta

- un test set è adeguato per testare un programma P secondo il criterio di **copertura delle condizioni** se per ogni condizione di P esiste
    - un caso di test T in cui la condizione è vera
    - un caso di test T in cui la condizione è false
- un test set è adeguato per testare un programma P secondo il criterio di **copertura delle decisioni** se per ogni decisione di P esiste
    - un caso di test T in cui la decisione è presa
    - un caso di test T in cui la decisione non è presa
- un test soddisfa **MCC (Multiple Condition Coverage)** se testa **ogni possibile combinazione** dei valori di verità delle condizioni atomiche in ogni decisione
- secondo **MCDC (Modified Condition Decision Coverage)** il test set deve essere preso in modo che ogni condizione all'interno di una decisione deve far variare in modo indipendente il valore finale della decisione