INGEGNERIA DEL SOFTWARE APPUNTI

## LEZIONE 2 **( SOFTWARE PROCESSES AND SOFTWARE QUALITY )**

Un **software process** è l'insieme di attività, metodi, pratiche e modelli utilizzati per progettare, sviluppare e mantenere il software. Questi processi sono fondamentali per garantire che il software sia realizzato in modo sistematico e controllato. Non esiste un processo universale che funzioni sempre, ed esistono e vengono utilizzati molti processi diversi. Tutti includono, in qualche forma, le seguenti attività fondamentali:

Immagine che contiene testo, Carattere, schermata, numero

Descrizione generata automaticamente

Esistono diversi modelli di sviluppo software, tra cui:

* **Modello a cascata (Waterfall)**: Un processo lineare e sequenziale, in cui ogni fase deve essere completata prima di passare alla successiva. Dove Il risultato di ogni fase è un documento che viene approvato (firmato).

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamente

* REQUIREMENTS ENGINEERING

L'ingegneria dei requisiti è una fase cruciale nello sviluppo del software, il cui obiettivo principale è capire cosa deve fare il software, senza ancora preoccuparsi di come implementarlo tecnicamente. In questa fase, l'attenzione è tutta sul "cosa" e non sul "come". È un processo che richiede un'analisi accurata delle esigenze degli utenti e del dominio del problema che il software intende risolvere.

Il successo di un progetto software dipende in gran parte dalla chiarezza e dalla precisione con cui vengono raccolti e compresi i requisiti. Spesso, le aspettative degli utenti finali possono essere vaghe o mutevoli, ed è qui che entra in gioco l'importanza del coinvolgimento attivo di tutte le parti interessate: clienti, utenti finali e ingegneri del software. Questo dialogo continuo serve a chiarire quali funzioni e caratteristiche il software dovrà avere per soddisfare i bisogni reali degli utenti e i vincoli del contesto in cui sarà utilizzato.

Un aspetto critico dell'ingegneria dei requisiti è l'analisi del dominio del problema, che aiuta gli ingegneri del software a comprendere a fondo l'ambiente in cui il software opererà. Questa comprensione del contesto reale è essenziale per evitare soluzioni fuori luogo o non pratiche.

Il risultato principale di questo processo è il documento di specifica dei requisiti del software. Questo documento non è solo una lista di funzionalità, ma una descrizione dettagliata e formalizzata di tutte le funzionalità, i vincoli e i criteri di accettazione che il software deve soddisfare. Una buona specifica dei requisiti riduce il rischio di malintesi e garantisce che tutte le parti abbiano una visione comune degli obiettivi del progetto.

* SYSTEM AND SOFTWARE/UI DESIGN

La **progettazione di sistemi e software** mira a creare un'architettura adeguata per il software. Si svolge su due livelli principali:

1. **Progettazione del sistema**: definisce l'architettura generale, scomponendo il sistema in moduli e componenti. Si assegnano funzionalità ai moduli e si definiscono le loro interazioni e l'assegnazione ai componenti hardware.
2. **Progettazione del software e dell'interfaccia utente (UI)**: dettaglia come implementare i moduli, includendo la progettazione delle classi necessarie e la prototipazione dell'interfaccia utente.

L'output finale è un insieme di specifiche di progettazione, spesso formalizzato con linguaggi come UML.

* IMPLEMENTATION

L'obiettivo di questa fase dello sviluppo software è quello di "tradurre" le specifiche di progettazione in un linguaggio di programmazione o tecnologia scelti. Non si tratta semplicemente di una traduzione automatica o meccanica, ma di una traduzione di alta qualità che tiene conto di principi di buon design e sviluppo.

Il codice risultante dovrebbe essere:

Leggibile: il codice deve essere chiaro e comprensibile non solo per chi lo scrive, ma anche per altri sviluppatori che potrebbero doverlo leggere o mantenere in futuro. Un codice leggibile facilita la collaborazione e la manutenzione.

Manutenibile: il codice deve essere facile da aggiornare e modificare quando necessario, sia per correggere bug, sia per aggiungere nuove funzionalità, senza dover riscrivere intere parti del sistema.

Riutilizzabile: i componenti del software dovrebbero essere progettati in modo tale da poter essere riutilizzati in contesti diversi. Il riuso del codice riduce la duplicazione e migliora l'efficienza nello sviluppo.

Estendibile: il software dovrebbe essere costruito con un'architettura flessibile, che permetta di aggiungere nuove funzionalità senza compromettere la struttura esistente. Questo richiede l'adozione di design pattern e principi di architettura che promuovono l'estensibilità.

Testabile: ogni parte del codice deve poter essere facilmente testata, sia manualmente che tramite strumenti automatici (come test unitari). Un codice testabile facilita l'individuazione di errori e garantisce una maggiore affidabilità del software.

In sintesi, il risultato dovrebbe essere un codice "pulito", ovvero conforme ai principi del "clean code"

* TESTING

La **Verification and Validation (V&V)** è un processo fondamentale nello sviluppo del software che ha l'obiettivo di assicurarsi che l'implementazione realizzata soddisfi le esigenze degli utenti e sia conforme alle specifiche definite.

* **Verifica (Verification)**: si concentra sul verificare che il sistema sia conforme alle sue specifiche. In altre parole, ci chiediamo: *"Abbiamo costruito il sistema nel modo giusto?"*. Questo avviene generalmente attraverso il **testing del programma**, eseguendo il software in un ambiente controllato per verificare che il suo comportamento sia corretto rispetto alle specifiche definite in precedenza.
* **Validazione (Validation)**: ha lo scopo di accertare se il sistema soddisfi le aspettative dei clienti. In questo caso, la domanda è: *"Abbiamo costruito il sistema giusto?"*. La validazione si esegue definendo e conducendo **test di accettazione** che verificano se il software risponde correttamente ai bisogni e alle aspettative degli utenti finali.

In sintesi, la verifica controlla che il prodotto sia tecnicamente corretto secondo le specifiche, mentre la validazione si assicura che risolva il problema per cui è stato progettato, soddisfacendo le necessità degli utenti.

* OPERATION AND MAINTENANCE

Una volta completato lo sviluppo del software, si entra nella fase di **Operation**, in cui il sistema viene distribuito e installato per i clienti, ed è messo in **uso reale**. Questo rappresenta il momento in cui il software inizia a funzionare nel contesto di utilizzo per cui è stato progettato, permettendo agli utenti di trarne beneficio.

Tuttavia, il ciclo di vita del software non si conclude qui. Inizia infatti la fase di **manutenzione**, una parte inevitabile e fondamentale. Il software, infatti, dovrà inevitabilmente subire modifiche nel tempo, per diverse ragioni:

* **Bisogni del cliente in evoluzione**: le esigenze dei clienti possono cambiare man mano che nuove funzionalità vengono richieste o le condizioni di mercato cambiano.
* **Cambiamenti nel contesto d'uso**: l'ambiente in cui opera il software potrebbe mutare, come ad esempio l'introduzione di nuove leggi che impongono cambiamenti ai processi aziendali o nuove regolamentazioni che richiedono aggiornamenti del sistema.
* **Correzione di bug**: nonostante la fase di verifica e validazione, è possibile che alcuni bug non rilevati emergano durante l'uso reale, richiedendo interventi di correzione e miglioramento.

La gestione di queste modifiche fa parte di discipline più avanzate come la **Gestione dei Progetti Software e l'Evoluzione del Software**, che si occupano di come mantenere, adattare e far evolvere un sistema software nel tempo. Queste competenze sono trattate a un livello più approfondito in corsi di specializzazione come quelli di un Master in **Software Project Management and Evolution**.

Parliamo ora di , **SOFTWARE QUALITY**

La qualità del software è un concetto che si riferisce a quanto bene un software soddisfa i requisiti specificati e le aspettative degli utenti. La qualità del software può essere misurata in termini di vari attributi.

La qualità del software non è un concetto unico, ma può essere vista da prospettive diverse, con approcci variegati. Il **ISO/IEC 25002** è uno standard che definisce modelli di qualità del software, offrendo un quadro per valutare e migliorare le proprietà qualitative del prodotto.

**Modelli di qualità nel software secondo lo standard ISO/IEC 25002:**

1. **Product Quality Model**:
   * Questo modello si focalizza sulle proprietà qualitative del prodotto software. È composto da **9 caratteristiche**, ognuna ulteriormente suddivisa in **sotto-caratteristiche**.
   * Queste caratteristiche servono come un **modello di riferimento** per specificare, misurare e valutare la qualità del prodotto.
   * Le caratteristiche coprono vari aspetti del prodotto software, come affidabilità, efficienza, manutenibilità, sicurezza, usabilità, e altro ancora.
2. **Quality-in-use Model**:
   * Questo modello si concentra sull'impatto della qualità del software dal punto di vista dell'utente finale, considerando il contesto d'uso specifico.
   * È composto da **3 caratteristiche**, anch'esse suddivise in sotto-caratteristiche, che influenzano l'esperienza degli **stakeholder** quando il sistema viene utilizzato.
   * Questo modello è importante perché tiene conto di come il prodotto viene effettivamente utilizzato in un ambiente reale e di come supporta gli obiettivi e le esigenze degli utenti.

**Vediamo in dettaglio il PRODUCT QUALITY DI ISO/IEX 25002:**

Immagine che contiene testo, schermata, Carattere, Rettangolo

Descrizione generata automaticamente

1. FUNCTIONAL SUITABILITY

La **funzionalità** (Functional Suitability) si riferisce al grado in cui un componente o un sistema offre funzioni che soddisfano le esigenze esplicite e implicite, quando utilizzato in condizioni specifiche.

Questa caratteristica si suddivide in tre sotto-caratteristiche principali:

1. **Completezza funzionale (Functional Completeness)**: misura il grado in cui le funzioni offerte coprono tutte le attività specificate e gli obiettivi degli utenti. In pratica, il software deve fornire tutte le funzionalità necessarie per soddisfare i requisiti.
2. **Correttezza funzionale (Functional Correctness)**: riguarda il grado in cui il prodotto fornisce risultati accurati quando viene utilizzato dagli utenti previsti. Questo implica che il software deve operare correttamente, senza errori, e rispondere in modo affidabile alle esigenze operative.
3. **Adeguatezza funzionale (Functional Appropriateness)**: valuta il grado in cui le funzioni facilitano il raggiungimento di compiti e obiettivi specifici. In questo caso, l'accento è posto sull'efficienza e sull'utilità delle funzioni nel permettere agli utenti di completare i loro compiti in modo efficace.
4. RELIABILITY

La **affidabilità** (Reliability) rappresenta il grado in cui un sistema esegue le sue funzioni in condizioni specifiche e per un periodo di tempo determinato. È una caratteristica essenziale per garantire che il software possa essere utilizzato con continuità e senza problemi.

Questa caratteristica è suddivisa in quattro sotto-caratteristiche principali:

1. **Assenza di errori (Faultlessness)**: misura il grado in cui un sistema esegue le funzioni specificate senza difetti o malfunzionamenti durante il normale funzionamento. Un sistema affidabile deve essere progettato per ridurre al minimo gli errori durante l'uso.
2. **Disponibilità (Availability)**: indica il grado in cui un sistema è operativo e accessibile quando richiesto per l'uso. La disponibilità è fondamentale per garantire che il software sia sempre pronto a rispondere alle esigenze degli utenti nei momenti necessari.
3. **Tolleranza ai guasti (Fault Tolerance)**: valuta la capacità di un sistema di continuare a funzionare correttamente anche in presenza di guasti hardware o software. Un sistema con una buona tolleranza ai guasti può gestire errori senza compromettere gravemente il suo funzionamento.
4. **Recuperabilità (Recoverability)**: misura la capacità del sistema di riprendersi da un'interruzione o da un malfunzionamento. In caso di guasto, il sistema dovrebbe poter ripristinare le sue funzioni in modo rapido ed efficiente, minimizzando l'impatto sull'operatività.
5. EFFICIENCY

L'**efficienza** (Efficiency) rappresenta il grado in cui un prodotto esegue le sue funzioni rispettando i vincoli di risorse specificati, utilizzando in modo efficiente risorse come CPU, memoria, spazio di archiviazione ed energia. L'efficienza è fondamentale per garantire che il software funzioni bene senza sovraccaricare o esaurire le risorse disponibili.

Questa caratteristica si suddivide in tre sotto-caratteristiche principali:

1. **Comportamento temporale (Time Behaviour)**: misura il grado in cui i tempi di risposta e le velocità di elaborazione di un sistema soddisfano i requisiti. Un sistema efficiente deve rispondere rapidamente alle richieste degli utenti e mantenere un buon throughput (velocità di elaborazione dei dati).
2. **Utilizzo delle risorse (Resource Utilization)**: valuta il grado in cui la quantità e i tipi di risorse utilizzati da un sistema soddisfano i requisiti. Un buon software deve ottimizzare l'uso delle risorse, garantendo che non vengano consumate inutilmente CPU, memoria o altre risorse critiche.

* **Capacità (Capacity)**: misura il grado in cui i limiti massimi di un parametro del sistema (come la quantità di dati gestibili, il numero di utenti supportati, ecc.) soddisfano i requisiti. Un sistema efficiente deve poter operare vicino ai suoi limiti massimi senza degrado delle prestazioni.

1. USABILITY

L'**usabilità** (Usability) si riferisce al grado in cui un prodotto o un sistema può essere utilizzato e interagito dagli utenti in modo efficace e soddisfacente. È una componente cruciale che determina quanto facilmente gli utenti possano sfruttare le funzionalità del sistema per soddisfare le loro esigenze.

Le principali sotto-caratteristiche dell'usabilità includono:

1. **Riconoscibilità (Recognizability)**: misura il grado in cui gli utenti possono riconoscere se il sistema è adatto alle loro esigenze. Un sistema con una buona riconoscibilità consente agli utenti di capire immediatamente se può risolvere i loro problemi.
2. **Apprendibilità (Learnability)**: valuta quanto facilmente e rapidamente gli utenti possono imparare a utilizzare le funzioni di un prodotto. Un sistema facile da imparare riduce il tempo di formazione e rende gli utenti più produttivi in breve tempo.
3. **Operabilità (Operability)**: si riferisce alla facilità con cui un prodotto o un sistema può essere utilizzato e controllato. Un sistema operabile è intuitivo e consente agli utenti di svolgere le loro attività senza sforzo.
4. **Protezione dagli errori degli utenti (User Error Protection)**: misura il grado in cui il sistema previene gli errori degli utenti o li aiuta a evitarli. Questo migliora l'esperienza utente riducendo gli errori e minimizzando le conseguenze di eventuali azioni sbagliate.
5. **Inclusività (Inclusivity)**: valuta quanto un sistema possa essere utilizzato da persone provenienti da vari contesti (età, abilità, culture, etnie, lingue, generi, situazioni economiche, ecc.). Un sistema inclusivo garantisce che le persone con diverse esigenze o capacità possano accedere e interagire con il prodotto senza barriere
6. SECURITY

La sicurezza informatica di un sistema si riferisce al grado in cui il sistema si difende dagli attacchi di attori malintenzionati e protegge le informazioni e i dati, applicando adeguati meccanismi di autorizzazione. È un aspetto cruciale per garantire la protezione delle risorse digitali e la fiducia degli utenti nel sistema.

Le principali sotto-caratteristiche della sicurezza includono:

1. **Confidenzialità (Confidentiality)**: misura il grado in cui un sistema garantisce che i dati siano accessibili solo a chi è autorizzato. La confidenzialità protegge le informazioni sensibili impedendo accessi non autorizzati.
2. **Integrità (Integrity)**: valuta il grado in cui un sistema assicura che il suo stato e i suoi dati siano protetti da modifiche o cancellazioni non autorizzate. L'integrità garantisce che le informazioni rimangano accurate e affidabili nel tempo.
3. **Non ripudio (Non-repudiation)**: si riferisce al grado in cui le azioni o gli eventi possono essere provati come effettivamente avvenuti, impedendo che possano essere negati o contestati successivamente. È fondamentale per la responsabilità e la tracciabilità.
4. **Responsabilità (Accountability)**: misura il grado in cui le azioni di un'entità (utente o sistema) possono essere tracciate in modo univoco fino a quell'entità. La responsabilità garantisce che ogni azione eseguita nel sistema possa essere attribuita al corretto esecutore.
5. **Autenticità (Authenticity)**: valuta il grado in cui l'identità di un soggetto o di una risorsa può essere verificata come autentica. L'autenticità garantisce che gli utenti o le risorse siano effettivamente quelli che dichiarano di essere, proteggendo contro l'accesso fraudolento.
6. COMPATIBILITY

La **compatibilità** (Compatibility) rappresenta il grado in cui un sistema è in grado di scambiare informazioni con altri prodotti, sistemi o componenti e/o di eseguire le sue funzioni richieste condividendo lo stesso ambiente e le risorse comuni con altri sistemi. In altre parole, un sistema compatibile può coesistere ed essere integrato efficacemente con altri sistemi senza creare problemi.

Le principali sotto-caratteristiche della compatibilità includono:

1. **Co-esistenza (Co-existence)**: misura il grado in cui un prodotto può eseguire le sue funzioni richieste in modo efficiente, condividendo un ambiente comune e risorse (come memoria o processi) con altri prodotti, senza avere un impatto negativo su nessuno di essi. Un sistema con una buona co-esistenza non interferisce con altri sistemi e continua a funzionare correttamente.
2. **Interoperabilità (Interoperability)**: valuta il grado in cui un sistema, prodotto o componente può scambiare informazioni con altri prodotti e utilizzare reciprocamente le informazioni scambiate. Un sistema interoperabile è in grado di comunicare e lavorare insieme ad altri sistemi, facilitando l'integrazione e la collaborazione tra più tecnologie o piattaforme.
3. MAINTAINABILITY

La manutenibilità rappresenta il grado di efficacia ed efficienza con cui un prodotto o un sistema può essere modificato per migliorarlo, correggerlo o adattarlo ai cambiamenti dell'ambiente e dei requisiti.

Questa caratteristica è composta dalle seguenti sottocaratteristiche:

- Modularità - Grado in cui un software è composto da componenti discreti, in modo che una modifica a un componente abbia un impatto minimo sugli altri.

- Riutilizzabilità - Grado in cui un software o un modulo può essere utilizzato come risorsa in più di un sistema.

- Modificabilità - Grado in cui un prodotto o un sistema può essere modificato in modo efficace ed efficiente senza introdurre difetti o degradare la qualità.

- Testabilità - Grado in cui è possibile stabilire i criteri di test per un sistema ed eseguire test per determinare se tali criteri sono stati soddisfatti

1. FLEXIBILITY

La flessibilità è il grado di adattamento di un prodotto ai cambiamenti dei requisiti, dei contesti d'uso o dell'ambiente operativo.

Questa caratteristica è composta dalle seguenti sottocaratteristiche:

- Adattabilità : Grado in cui un sistema può essere efficacemente ed efficientemente adattato o trasferito a diversi ambienti hardware, software o altri ambienti operativi o di utilizzo.

- Scalabilità : Grado di capacità di un sistema di gestire carichi di lavoro crescenti o decrescenti o di adattare la propria capacità alla variabilità.

- Installabilità : Grado di efficacia ed efficienza con cui un prodotto o un sistema può essere installato e/o disinstallato con successo.

- Sostituibilità : Grado in cui un prodotto può sostituire un altro prodotto software specificato per lo stesso scopo nello stesso ambiente.

1. SAFETY

La sicurezza rappresenta il grado in cui un prodotto evita uno stato in cui la vita umana, la salute, la proprietà o l'ambiente sono in pericolo.

Questa caratteristica comprende, tra le altre, le seguenti sottocaratteristiche:

- Fail safe - Grado in cui un prodotto è in grado di posizionarsi automaticamente in una modalità operativa sicura o di tornare a una condizione di sicurezza in caso di guasto.

- Identificazione del rischio - Grado in cui un prodotto è in grado di identificare un corso di eventi o operazioni che possono portare a un rischio inaccettabile.

- Segnalazione dei pericoli - Grado in cui un sistema fornisce avvisi di rischi inaccettabili alle operazioni o ai controlli interni in modo che possano reagire in tempo utile.

**Vediamo in dettaglio il QUALITY-in-USE MODEL DI ISO/IEX 25002:**

Immagine che contiene testo, schermata, biglietto da visita, Carattere

Descrizione generata automaticamente

Modello della qualità d'uso, composto da 3 caratteristiche (ulteriormente suddivise in sottocaratteristiche) che possono influenzare le parti interessate quando i prodotti o i sistemi vengono utilizzati in un determinato contesto d'uso.

-Misura il grado in cui un prodotto o un sistema può essere utilizzato da utenti specifici per soddisfare le loro esigenze e raggiungere obiettivi specifici con efficacia, efficienza, assenza di rischi e soddisfazione in contesti d'uso specifici.

1. USABILITY

Misura la misura in cui gli utenti riescono a raggiungere i propri obiettivi in modo efficiente e soddisfacente utilizzando il sistema. Questa caratteristica è composta da : efficiacia ( quanto bene gli utenti riescono a completare le attività previste utilizzando il sistema.

Efficienza(le risorse necessarie per portare a termine i compiti

Soddisfazione( il comfort dell’utente e l’esperienza postivia durante l’utilizzo del sistema

1. SAFETY

La sicurezza valuta la capacità del sistema di prevenire danni o pregiudizi alle persone, all’ambiente e agli interessi commerciali.

È composta da queste sottocategorie : Danni commerciali (valuta l’efficiacia del sistema nel prevenire perdite finanziare o danni all’azienda. Salute e sicureza dell’operatore : quanto bene il sistema protegge gli utenti dai rischi per la salute o dai pericoli per la sicurezza durante il suo utilizzo. Danno ambientale ( il sistema dovrebbe evitare o ridurre al minimo gli impatti negativi sull’ambiente

1. FLEXIBILITY

La flessibilità si riferisce alla capacità del sistema di adattarsi e funzionare efficaciamente in contesti o ambienti diversi. Questa caratteristica è composta dalle seguenti sotto caratteristiche: Conformità al contesto( capacità del sistema di espandersi o adattarsi ad ambienti nuovi o mutevoli senza modifiche significative ). Estensibilita del contesto: la possibilità del sistema di espandersi o adattarsi ad ambienti nuovi o mutevoli senza modifiche significative). Accessibilità( indica l’efficacia con cui il sistema puo essere utilizzato da tutte le persone comprese quelle con disabilità, in diversi ambienti.

## LEZIONE 3 (REQUIREMENT ENGINEERING)

Possiamo dire che i requisiti sono descrizioni di cosa il sistema dovrebbe fare :

* Servizi che il sistema provvede a dare agli utenti
* Vincoli di operazioni (operazioni da fare)