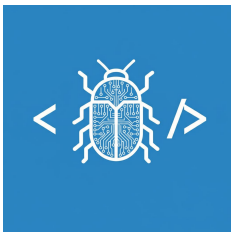

BugBusters

Email: bugbusters.unipd@gmail.com

Gruppo: 4

Università degli Studi di Padova

Laurea in Informatica

Corso: Ingegneria del Software

Anno Accademico: 2025/2026

Glossario

Versione 1.0.0

Stato	In redazione
Redattori	Alberto Autiero
Verificatori	
Uso	Interno ed esterno
Destinatari	Prof. Tullio Vardanega, Prof. Riccardo Cardin, BugBusters, Eggon

Registro delle modifiche

Versione	Data	Descrizione	Redatto	Verificato	Approvato
1.0.0		Approvazione del documento	-	-	
0.1.0		Verifica del documento	-		-
0.0.6	12/12/2025	Sono stati aggiunti nuovi termini	Alberto Autiero	-	-
0.0.5	06/12/2025	Sono stati aggiunti nuovi termini	Alberto Autiero	-	-
0.0.4	30/11/2025	Sono stati aggiunti nuovi termini	Alberto Autiero	-	-
0.0.3	23/11/2025	Sono stati aggiunti nuovi termini	Alberto Autiero	-	-
0.0.2	17/11/2025	Sono stati aggiunti nuovi termini	Alberto Autiero	-	-
0.0.1	06/11/2025	Prima stesura del documento con i termini del capitolato aggiudicato (5) e diapositive dei docenti	Alberto Autiero	-	-

Indice

A

Abstract Class

Classe nella programmazione orientata agli oggetti che non può essere istanziata direttamente e che serve come base per l'ereditarietà. Può contenere sia metodi astratti (senza implementazione, da ridefinire obbligatoriamente nelle sottoclassi) sia metodi concreti (con implementazione condivisa). Definisce un'interfaccia comune e un comportamento parziale che le classi derivate devono completare o specializzare, favorendo il riuso del codice e l'applicazione del polimorfismo.

Accoppiamento

Misura del grado di dipendenza tra componenti software. Indica la probabilità che modifiche a un componente richiedano modifiche ad altri componenti. L'accoppiamento stretto (tight coupling) implica forte interdipendenza e maggiore fragilità del sistema, mentre l'accoppiamento debole (loose coupling) minimizza le dipendenze, favorendo manutenibilità, riusabilità e testabilità. La dipendenza è direttamente proporzionale alla quantità di codice condiviso (SLOC) e all'ampiezza dello scope di tale codice.

Affidabilità (Reliability)

Capacità di un sistema software di mantenere il proprio livello di prestazioni in condizioni specificate per un determinato periodo di tempo.

Aggregazione

Relazione strutturale in OOP che rappresenta un legame "parte-tutto" dove i componenti possono esistere indipendentemente dall'oggetto che li contiene. È una forma di associazione con condivisione dei riferimenti.

Agile

Metodologia di sviluppo del software iterativa e incrementale che si basa su principi come la collaborazione continua con il cliente, la consegna frequente di software funzionante e la capacità di rispondere ai cambiamenti dei requisiti.

AI (Artificial Intelligence)

Campo dell'informatica che si occupa di sviluppare sistemi in grado di svolgere compiti che normalmente richiedono l'intelligenza umana, come il riconoscimento vocale, la visione artificiale, l'elaborazione del linguaggio naturale e il processo decisionale.

AI Co-Pilot

Modulo di supporto automatizzato per gli studi dei Consulenti del Lavoro (CdL), che assiste nei processi di gestione, riconoscimento e distribuzione documentale attraverso tecniche di intelligenza artificiale.

AI generativa

Tipo di intelligenza artificiale in grado di creare autonomamente nuovi contenuti (testo, immagini, audio, codice) basandosi su pattern appresi dai dati di addestramento.

Amministratore

Figura responsabile della gestione dell'infrastruttura, degli strumenti di sviluppo e dei processi del progetto. Si occupa di configurare e mantenere gli ambienti di lavoro e garantire che il team abbia a disposizione gli strumenti necessari.

Analisi dei capitolati

Processo di studio e valutazione dei diversi capitolati d'appalto proposti dalle aziende proponenti, finalizzato a selezionare il progetto più adatto alle competenze, agli interessi e alle risorse del gruppo. Include la comprensione dei requisiti, delle tecnologie richieste e dei vincoli di progetto.

Analisi dei Requisiti

Processo sistematico volto a identificare, documentare, classificare e validare i bisogni e i vincoli del progetto. Trasforma i requisiti utente (espressi nel capitolato, dal punto di vista del committente) in requisiti software (specificati tecnicamente, dal punto di vista dello sviluppatore), definendo "cosa" il sistema deve fare senza specificare "come". Produce il documento di Analisi dei Requisiti (AdR) e costituisce la base per la progettazione architetturale. Include attività di studio del dominio, definizione di casi d'uso, classificazione dei requisiti (obbligatori, desiderabili, opzionali; funzionali e non-funzionali) e tracciamento.

Analisi dinamica

Tecnica di verifica che prevede l'esecuzione del software per osservarne il comportamento in fase di runtime, al fine di identificare errori, verificare le prestazioni e validare le funzionalità. Comprende attività di testing e profiling.

Analista

Figura professionale specializzata nell'analisi dei requisiti e nella specifica delle funzionalità del sistema. Collabora con gli stakeholder per comprendere lesigenze e tradurle in specifiche tecniche.

Angular

Framework per applicazioni web sviluppato da Google, basato su TypeScript, utilizzato per costruire applicazioni client-side single-page (SPA).

API (Application Programming Interface)

Insieme di definizioni, protocolli e strumenti per la costruzione e l'integrazione di software applicativo. Consente a diversi sistemi di comunicare tra loro.

Approvazione

Processo formale attraverso il quale un documento o una funzionalità viene accettata e considerata completa dopo aver superato le verifiche e validazioni necessarie.

Architettura a livelli

Pattern architetturale che organizza il sistema in livelli gerarchici, onde ogni livello fornisce servizi al livello superiore e utilizza servizi del livello inferiore.

Architettura dei microservizi

Approccio architetturale che struttura un'applicazione come una collezione di servizi piccoli, autonomi e indipendenti, ciascuno responsabile di una specifica funzionalità di business.

Architettura di dettaglio

Fase della progettazione software in cui si specificano nel dettaglio i componenti, le interfacce e le relazioni tra di essi, partendo dall'architettura logica.

Architettura logica

Fase della progettazione software in cui si definisce la struttura del sistema a livello di componenti e delle loro interazioni, senza scendere nel dettaglio implementativo.

Architettura multilivello

Pattern architetturale che separa le responsabilità dell'applicazione in livelli logici distinti, tipicamente presentazione, logica di business e persistenza dati.

Architettura Pipe-and-Filter

Pattern architetturale in cui i componenti (filtri) elaborano i dati in sequenza, passandoli attraverso connessioni (pipe).

Architettura Three-Tier

Architettura software che divide l'applicazione in tre livelli: presentazione, logica di business e dati.

Associazione

Relazione strutturale in OOP che rappresenta una connessione duratura tra oggetti di classi diverse, dove gli oggetti interagiscono per un periodo di tempo prolungato condividendo riferimenti.

Attore

Ruolo svolto da un utente o sistema esterno che interagisce con il sistema software per raggiungere obiettivi specifici. Gli attori possono essere primari (beneficiari diretti) o secondari (fornitori di servizi).

Attore principale

Utente o sistema esterno che interagisce direttamente con il sistema software per raggiungere un obiettivo specifico nel caso d'uso. È il protagonista dell'interazione e trae beneficio diretto dall'esecuzione del caso d'uso.

Attore secondario

Utente o sistema esterno che fornisce servizi o supporto all'attore principale durante l'esecuzione di un caso d'uso. Partecipa all'interazione ma non è il beneficiario principale del risultato, spesso fornendo funzionalità accessorie o di supporto.

Availability (Disponibilità)

Misura della percentuale di tempo in cui un sistema è operativo e accessibile agli utenti.

AWS (Amazon Web Services)

Piattaforma di cloud computing che offre servizi di calcolo, storage, database e altre funzionalità per supportare lo sviluppo e il deployment di applicazioni.

B

Back-end

Parte di un'applicazione software che gestisce la logica di business, l'elaborazione dei dati e la comunicazione con il database. Opera sul server ed è inaccessibile direttamente all'utente finale.

Baseline

Versione approvata e formalmente controllata di un documento o di un componente software che serve come riferimento per sviluppi successivi. Le modifiche successive richiedono procedure formali di controllo. Può riferirsi a diverse fasi (Requirements Baseline, Design Baseline, Product Baseline) e garantisce stabilità e tracciabilità.

Best Practice

Insieme di tecniche, metodi e procedure che sono state riconosciute come le più efficaci ed efficienti per raggiungere un obiettivo specifico in un determinato contesto.

Branch Coverage

Metrica di copertura del testing strutturale (white-box) che misura la percentuale di rami decisionali (branch) del flusso di controllo attraversati almeno una volta durante l'esecuzione dei test. Un ramo rappresenta un possibile percorso di esecuzione derivante da una struttura condizionale (if-then-else, switch-case). Il coverage del 100

Builder

Pattern creazionale che separa la costruzione di un oggetto complesso dalla sua rappresentazione, in modo che lo stesso processo di costruzione possa creare diverse rappresentazioni. Il pattern Builder è composto da Director, Builder e Product.

C

Candidatura

Processo con cui un gruppo di studenti presenta la propria offerta per aggiudicarsi un capitolato d'appalto nel corso di Ingegneria del Software, proponendo una soluzione che soddisfi i requisiti del proponente.

Capitolato

Documento contrattuale che specifica i requisiti, le caratteristiche tecniche e le condizioni di un progetto software proposto da un'azienda proponente per il corso di Ingegneria del Software.

Casi d'uso

Tecnica di specifica dei requisiti che descrive le interazioni tra gli attori (utenti o sistemi esterni) e il sistema software per raggiungere un obiettivo specifico.

Cedolini

Documenti retributivi che attestano la retribuzione corrisposta al dipendente per un determinato periodo di paga.

Cedolini Massivi

File contenenti più documenti retributivi aggregati, da suddividere e assegnare ai singoli destinatari tramite riconoscimento automatico.

Cerimonia

Evento formale o informale nel framework Scrum che segue un agenda prestabilita e ha uno scopo specifico, come la Pianificazione dello Sprint, il Daily Stand-up, la Revisione dello Sprint o la Retrospettiva.

Committente

Soggetto che commissiona il progetto, definendone gli obiettivi, i vincoli e i requisiti, e che recepisce il prodotto finale. Nel contesto del corso di Ingegneria del Software, può essere un'azienda proponente o un docente.

Complessità ciclomatica

Metrica software introdotta da Thomas J. McCabe che misura la complessità strutturale di un programma. È calcolata dal grafo di controllo del flusso e indica il numero di cammini linearmente indipendenti all'interno del codice. È utilizzata per identificare moduli ad alto rischio e per pianificare il testing (in particolare per determinare il numero minimo di casi di test necessari per la copertura dei cammini).

Composizione

Relazione strutturale in OOP che rappresenta un legame "parte-tutto" forte dove gli componenti non possono esistere indipendentemente dall'oggetto che li contiene. Implica ownership esclusiva e distruzione concomitante.

Comprensibilità

Capacità di un sistema software di essere facilmente compreso dai suoi utilizzatori, riducendo lo sforzo cognitivo necessario per il suo utilizzo.

Condition/Decision Coverage

Metrica di copertura del testing strutturale (white-box) che richiede che ogni condizione elementare all'interno di una decisione e l'intera decisione stessa assumano sia vero che falso durante l'esecuzione dei test. È più forte della branch coverage perché richiede anche la copertura delle singole condizioni.

Constructor Injection

Tecnica di Dependency Injection in cui le dipendenze vengono fornite a un componente attraverso il suo costruttore.

Consuntivo

Documento di contabilità che riporta i costi effettivamente sostenuti durante una fase del progetto, confrontandoli con le stime iniziali (preventivo). Viene utilizzato per monitorare l'andamento economico del progetto e per apportare eventuali correzioni nelle fasi successive.

Continuous Integration (CI)

Pratica di sviluppo software che prevede l'integrazione frequente (più volte al giorno) del codice prodotto dai membri del team in un repository condiviso. Ogni integrazione è verificata da una build automatizzata e da una serie di test per rilevare errori il più rapidamente possibile. Mira a migliorare la qualità del software e a ridurre il tempo necessario per validare e rilasciare nuovi aggiornamenti.

Controllo di Qualità

Insieme di attività e tecniche volte a garantire che i prodotti software soddisfino gli standard di qualità prestabiliti. Il controllo di qualità si concentra sull'identificazione di difetti nel prodotto realizzato.

Cruscotto di Controllo

Strumento di monitoraggio che fornisce una visualizzazione in tempo reale degli indicatori chiave di prestazione (KPI) e delle metriche di qualità del progetto. Consente di tracciare lo stato delle attività di verifica e validazione.

Cruscotto/Dashboard

Interfaccia utente che presenta in forma grafica e sintetica le metriche, gli indicatori di performance e lo stato corrente del progetto o dell'applicazione.

D

Decisione esterna

Scelta presa da entità esterne al team di progetto (come il committente o il proponente) che vincola le attività del progetto e che il team deve rispettare.

Decisione interna

Scelta presa dal team di progetto riguardante aspetti tecnici, organizzativi o metodologici, documentata per garantire tracciabilità e coerenza nelle attività successive.

Decomposizione funzionale

Processo di scomposizione di un sistema complesso in funzioni o componenti più piccoli e gestibili.

Dependency Injection

Pattern architetturale in cui le dipendenze di un componente vengono fornite dall'esterno, anziché essere create internamente al componente stesso.

Design Pattern

Soluzione progettuale generale e riutilizzabile a un problema ricorrente all'interno di un dato contesto nella progettazione di software. I design pattern non sono progetti finiti che possono essere trasformati direttamente in codice, ma modelli astratti che devono essere adattati al caso specifico.

Design pattern architetturali

Soluzioni progettuali ricorrenti e collaudate per problemi architetturali comuni nei sistemi software.

Design Pattern Comportamentali

Categoria di design pattern che si occupano della comunicazione e dell'assegnazione di responsabilità tra oggetti. Definisce modi efficaci per organizzare il comportamento degli oggetti, migliorando la flessibilità e la riusabilità. Esempi includono Observer, Iterator, Strategy, e Command.

Design Pattern Strutturali

Categoria di design pattern che si occupano della composizione di classi e oggetti per formare strutture più grandi e complesse. Facilitano la progettazione definendo relazioni semplici tra entità per realizzare nuove funzionalità. Esempi includono Adapter, Decorator, Composite, e Proxy.

Diagramma degli stati

Diagramma UML che descrive il comportamento di un oggetto in risposta a eventi, mostrando i diversi stati in cui può trovarsi e le transizioni tra essi.

Diagramma dei casi d'uso

Diagramma UML che descrive le interazioni tra gli attori e il sistema, mostrando i diversi scenari possibili per raggiungere un obiettivo specifico.

Diagramma dei componenti

Diagramma UML che mostra l'organizzazione e le dipendenze tra i componenti software di un sistema.

Diagramma delle classi

Diagramma UML che mostra le classi del sistema, i loro attributi, metodi e le relazioni tra di esse (associazioni, ereditarietà, dipendenze, ecc.).

Diagramma di attività

Diagramma UML che modella il flusso di controllo o il flusso di dati tra attività, utilizzato per descrivere la logica di procedura, di business o di caso d'uso.

Diagramma di deployment

Diagramma UML che mostra la configurazione fisica dei nodi di elaborazione e dei componenti software eseguiti su di essi.

Diagramma di Gantt

Strumento di pianificazione progettuale che visualizza le attività su un asse temporale, mostrando durate, sequenzialità, parallelismo e progressi rispetto alle pianificazioni.

Diagramma di PERT

(Program Evaluation and Review Technique) Strumento di analisi delle dipendenze temporali tra attività di progetto, utilizzato per identificare cammini critici e margini temporali (slack time).

Diagramma di sequenza

Diagramma UML che mostra le interazioni tra oggetti in sequenza temporale, evidenziando l'ordine dei messaggi scambiati.

Dipendenza

Relazione tra componenti software per cui un componente (dipendente) richiede un altro componente (dipenduto) per funzionare correttamente. Le modifiche al componente dipenduto possono influenzare il componente dipendente.

Discord

Piattaforma di comunicazione tramite chat vocale, testuale e video, utilizzata dal gruppo per la comunicazione interna e la collaborazione quotidiana.

Dispatch

Processo di distribuzione automatizzata dei documenti verso i destinatari finali attraverso diversi canali (es. app, portale, email, PEC).

Disponibilità

Capacità di un sistema di essere accessibile e utilizzabile quando richiesto dagli utenti. Misura la percentuale di tempo in cui il sistema è operativo.

Dominio d'uso

Contesto specifico o ambiente operativo in cui il sistema software sarà impiegato, comprendente le caratteristiche degli utenti finali, le condizioni operative, i vincoli tecnologici e le regole di business che definiscono l'ambito di applicazione del prodotto.

E

Economicità

Combinazione ottimale di efficacia ed efficienza. Un sistema economico raggiunge i propri obiettivi (efficacia) senza sprecare risorse (efficienza), massimizzando il valore prodotto rispetto ai costi sostenuti.

Efficacia

Misura della capacità di raggiungere gli obiettivi prefissati e produrre i risultati attesi. Un processo o sistema è efficace quando soddisfa i requisiti e le aspettative degli stakeholder, indipendentemente dalle risorse impiegate. Metrica: grado di raggiungimento degli obiettivi (interni ed esterni).

Efficienza

Misura dell'abilità di raggiungere gli obiettivi impiegando le risorse minime indispensabili. Un processo o sistema è efficiente quando ottimizza l'uso di tempo, persone, denaro e strumenti. Metrica: produttività, ovvero rapporto tra quantità di output prodotto e risorse consumate.

Entity Resolution

Processo di identificazione e associazione di entità (es. persone o aziende) a partire da dati parziali ouplicati, tramite algoritmi di matching e disambiguazione.

Ereditarietà

Meccanismo della programmazione orientata agli oggetti che permette a una classe (sottoclasse) di acquisire attributi e metodi di un'altra classe (superclasse), favorendo il riutilizzo del codice e le relazioni di generalizzazione.

Ereditarietà dell'interfaccia

Meccanismo di ereditarietà in cui una interfaccia eredita da un'altra interfaccia, oppure una classe implementa un'interfaccia.

Ereditarietà di classe

Meccanismo di ereditarietà in cui una classe eredita da un'altra classe (sia classe concreta che astratta).

Error / Errore

Manifestazione di un fault all'interno del sistema durante la sua esecuzione. È uno stato interno del sistema che può portare a un failure se non gestito. Corrisponde a una discrepanza tra il valore calcolato, osservato o misurato e il valore vero, specificato o teoricamente corretto.

Error Handling

Insieme di tecniche e meccanismi di programmazione per gestire le condizioni di errore che possono verificarsi durante l'esecuzione di un software. L'error handling prevede l'identificazione, la segnalazione e il recupero dagli errori, al fine di mantenere la stabilità e l'affidabilità del sistema.

F

Failure / Fallimento

Evento osservabile esternamente in cui il sistema devia dal suo comportamento corretto o atteso, non fornendo il servizio richiesto. Rappresenta l'impatto esterno di un fault o error.

Fault / Guasto

Causa radice di un errore (error) nel software. Può essere un difetto statico nel codice sorgente (es. bug), nella progettazione o nei requisiti. Un fault attivato può portare a un error, che a sua volta può propagarsi e causare un failure.

Flessibilità

Capacità di un sistema software di adattarsi a cambiamenti nei requisiti o nell'ambiente operativo con modifiche minime.

Front-end

Parte di un'applicazione software con cui l'utente interagisce direttamente, responsabile della presentazione dei dati e dell'acquisizione dell'input dell'utente.

Funzionalità

Caratteristica o capacità specifica che un sistema software deve possedere per soddisfare i bisogni degli utenti e gli obiettivi del progetto.

G

GDPR (General Data Protection Regulation)

Regolamento generale sulla protezione dei dati dell'Unione Europea che stabilisce norme per la protezione e la libera circolazione dei dati personali.

Gestione dei rischi

Processo sistematico di identificazione, analisi, pianificazione e controllo dei rischi di progetto, volto a minimizzare la probabilità di occorrenza e l'impatto degli eventi negativi.

GitHub

Piattaforma di hosting per repository Git che offre strumenti per il version control, la collaborazione e la gestione del ciclo di vita del software.

Glossario

Documento che raccoglie e definisce i termini specifici, gli acronimi e le parole ambigue utilizzati nella documentazione di progetto, con lo scopo di garantire una comprensione univoca della terminologia da parte di tutti i membri del team e dei destinatari.

Google Meet

Piattaforma di videoconferenza sviluppata da Google che consente meeting virtuali, condivisione dello schermo e chat. Utilizzata dal gruppo per le riunioni di coordinamento e le presentazioni.

H

Human in the loop

Approccio in cui l'intelligenza artificiale e gli esseri umani collaborano, con l'uomo che supervisiona, corregge o fornisce feedback al sistema AI, particolarmente utile quando la confidenza del sistema è bassa.

I

Implementazione

Processo di traduzione di un design in codice eseguibile, realizzando le specifiche funzionali e non funzionali.

Incapsulamento

Principio della programmazione orientata agli oggetti che consiste nel racchiudere in un'unica entità (classe) dati e metodi che operano su di essi, nascondendo i dettagli implementativi all'esterno.

Incapsulamento (Encapsulation)

Principio della programmazione orientata agli oggetti che consiste nel racchiudere in un'unica entità (classe) dati e metodi che operano su di essi, nascondendo i dettagli implementativi all'esterno e esponendo solo un'interfaccia pubblica.

Information Hiding

Principio di progettazione software che consiste nel nascondere i dettagli implementativi di un modulo, esponendo solo le interfacce necessarie, per ridurre l'accoppiamento e aumentare la manutenibilità.

Interfaccia

In programmazione orientata agli oggetti, un'interfaccia è un tipo astratto che contiene una serie di metodi dichiarati senza implementazione. Le classi che implementano un'interfaccia devono fornire un'implementazione per tutti i suoi metodi. Le interfacce definiscono un contratto che le classi devono seguire.

Issue

Segnalazione di un problema, un bug o una richiesta di miglioramento nel sistema di tracking del progetto. Ogni issue viene tracciata, assegnata e gestita fino alla risoluzione.

Iterator

Design pattern comportamentale che fornisce un modo per accedere sequenzialmente agli elementi di una collezione senza esporne la rappresentazione interna. Separa l'algoritmo di iterazione dalla struttura dati, permettendo diversi tipi di iterazione su una stessa collezione.

K

KPI (Key Performance Indicator)

Indicatore chiave di performance che misura l'efficacia di un processo, un'attività o un'organizzazione nel raggiungere i propri obiettivi.

L

Latex

Sistema di composizione tipografica utilizzato per la produzione di documentazione tecnica e scientifica di alta qualità, particolarmente adatto per documenti complessi con formule matematiche.

Lettera di candidatura

Documento formale con cui un gruppo di studenti esprime ufficialmente il proprio interesse a realizzare un capitolato d'appalto specifico, presentando le motivazioni, le competenze del gruppo e una proposta preliminare di approccio al progetto. Viene indirizzata ai docenti e al proponente.

Livelli aperti

Architettura a livelli in cui un livello può utilizzare servizi di qualsiasi livello sottostante, non solo di quello immediatamente inferiore.

Livelli chiusi

Architettura a livelli in cui un livello può utilizzare solo i servizi del livello immediatamente inferiore.

LLM (Large Language Model)

Modello di linguaggio di grandi dimensioni addestrato su vasti corpus di testo, in grado di generare, comprendere e elaborare linguaggio naturale in modo sofisticato.

Load balancer

Componente che distribuisce il carico di lavoro tra più server per migliorare le prestazioni e l'affidabilità del sistema.

M

Manutenibilità

Capacità di un prodotto software di essere modificato per correggere errori, migliorare le prestazioni o adattarsi a un ambiente in cambiamento. La manutenibilità è uno degli attributi di qualità del software e influenza il costo del ciclo di vita del prodotto.

MCDC (Modified Condition/Decision Coverage)

Metodo di copertura del testing strutturale (white-box) usato spesso in contesti di sicurezza critica (es. avionica, automotive). Richiede che ogni condizione elementare all'interno di una decisione dimostri di influenzare indipendentemente l'esito della decisione. Ogni condizione deve essere testata con valori che ne causino sia vero che falso, e per ogni condizione deve esistere un test in cui la modifica solo di quella condizione cambia l'esito dell'intera decisione.

Microservizio

Approccio architetturale in cui un'applicazione è composta da piccoli servizi indipendenti, ciascuno eseguibile autonomamente e comunicante tramite API.

Microsoft Teams

Piattaforma di collaborazione sviluppata da Microsoft che offre chat, videoconferenze, archiviazione file e integrazione con applicazioni. Utilizzata dal gruppo per le riunioni di coordinamento e la condivisione di documenti.

Milestone

Data di calendario che fissa un punto di avanzamento atteso nel tempo di progetto, definendo obiettivi specifici e misurabili da raggiungere. Il raggiungimento di una milestone è dimostrato dalla realizzazione di una baseline approvata. Le milestone devono essere SMART: Specifiche (obiettivi chiari), Misurabili (risultati verificabili), Achievable (realisticamente raggiungibili), Rilevanti (significative per il progetto e gli stakeholder), Time-bound (con scadenza definita). Vengono identificate procedendo a ritroso dalla milestone finale verso l'inizio del progetto, determinando le attività e le risorse necessarie per ogni periodo.

Model

Componente in un pattern architetturale che rappresenta i dati e la logica di business dell'applicazione, indipendentemente dall'interfaccia utente.

Model-View-Controller (MVC)

Pattern architetturale che separa l'applicazione in tre componenti interconnessi: Model (dati e logica di business), View (interfaccia utente) e Controller (gestisce l'input dell'utente e media tra Model e View).

Model-View-Presenter (MVP)

Pattern architetturale derivato da MVC che separa l'applicazione in Model, View e Presenter, dove il Presenter contiene la logica di presentazione e agisce da intermediario tra View e Model.

Model-View-ViewModel (MVVM)

Pattern architetturale che separa l'applicazione in Model, View e ViewModel, utilizzando il data binding per sincronizzare automaticamente View e ViewModel.

Modularità

Grado in cui un sistema è composto da componenti separati che possono essere sviluppati, testati e mantenuti indipendentemente.

Modulo

Componente software autonomo e sostituibile che incapsula una specifica funzionalità e fornisce un'interfaccia ben definita. I moduli favoriscono la modularità, il riuso e la manutenibilità del sistema.

N

Norme di progetto

Documento che definisce le regole, le convenzioni, le procedure e gli standard adottati dal gruppo per la realizzazione del progetto. Include norme per la redazione dei documenti, la gestione del versionamento, le convenzioni di codifica, le procedure di verifica e validazione, e le modalità di comunicazione interna ed esterna.

O

Observer

Design pattern comportamentale che definisce una dipendenza uno-a-molti tra oggetti, in modo che quando un oggetto (soggetto) cambia stato, tutti i suoi dipendenti (osservatori) vengano notificati e aggiornati automaticamente. È fondamentale per implementare sistemi di eventi e notifiche.

OCR (Optical Character Recognition)

Tecnologia che converte immagini di testo scritto o stampato in testo digitale machine-readable.

Oracolo

In contesto di testing, meccanismo, processo o risorsa (umana o automatizzata) che determina se il sistema sotto test ha superato o fallito un caso di prova. L'oracolo confronta l'output effettivo del sistema con l'output atteso. Può essere un semplice confronto di valori, un modello di riferimento, o un giudizio umano.

P

Pattern Creazionali

Categoria di design pattern che si occupano dei meccanismi di creazione degli oggetti, cercando di creare oggetti in modo adatto alla situazione. Esempi includono Factory Method, Abstract Factory, Builder, Singleton, etc.

PB (Product Baseline)

Baseline che definisce lo stato del prodotto software in un determinato momento, tipicamente al termine di una fase di sviluppo o di uno sprint. Include le specifiche del prodotto, la documentazione e il codice sorgente, e serve come riferimento per le fasi successive o per il rilascio.

Piano di progetto

Documento di pianificazione che descrive come il progetto sarà condotto, gestito e controllato. Definisce gli obiettivi, le attività, le risorse, i tempi, i costi, i rischi e le modalità di monitoraggio e comunicazione del progetto.

Piano di qualifica

Documento che definisce le strategie, le risorse e le attività per garantire la qualità del processo di sviluppo e del prodotto software. Include la definizione degli standard di qualità, le metriche per la misurazione, i processi di verifica e validazione, e le responsabilità del team per il controllo della qualità.

Piano di Qualifica

Documento che definisce le modalità, le risorse e le tempistiche per le attività di verifica e validazione del software. Descrive gli standard di qualità da adottare, le metriche per la misurazione della qualità, i processi di controllo e le responsabilità del team per garantire la qualità del prodotto.

POC (Proof of Concept)

Realizzazione preliminare e limitata di un'idea o di un concetto, finalizzata a verificarne la fattibilità tecnica, l'adeguatezza rispetto ai requisiti o la validità di un'architettura proposta. Un POC viene solitamente sviluppato con risorse minime per valutare rapidamente se un'approccio o una tecnologia siano promettenti prima di impegnare risorse significative in uno sviluppo completo.

Polimorfismo

Principio della programmazione orientata agli oggetti che permette a oggetti di classi diverse di rispondere allo stesso messaggio (metodo) in modo specifico per la propria classe, favorendo flessibilità ed estensibilità del codice.

Post-condizione

Condizione o stato del sistema che deve essere vero dopo il completamento di un caso d'uso. Definisce il risultato atteso e le garanzie che il sistema fornisce al termine dell'esecuzione del caso d'uso.

Pre-condizione

Condizione o stato del sistema che deve essere vero prima che un caso d'uso possa iniziare. Definisce i prerequisiti necessari per l'esecuzione corretta del caso d'uso.

Preventivo

Documento di pianificazione che stima i costi e le risorse necessarie per lo svolgimento del progetto, basato sulle attività pianificate e sulle risorse disponibili.

Procedimento agile

Approccio iterativo e incrementale allo sviluppo software che enfatizza la flessibilità e la collaborazione con il cliente.

Procedimento bottom-up

Approccio allo sviluppo software che parte dai componenti di basso livello per costruire gradualmente sistemi più complessi.

Procedimento top-down

Approccio allo sviluppo software che parte da una visione d'insieme per scomporre gradualmente il sistema in componenti più piccoli.

Processi di supporto

Processi che supportano i processi primari, includendo documentazione, gestione della configurazione, accertamento della qualità, verifica, validazione e risoluzione dei problemi.

Processi organizzativi

Processi trasversali rispetto ai singoli progetti che riguardano la gestione dei processi, delle infrastrutture, del miglioramento continuo e della formazione del personale nell'organizzazione.

Processi primari

Processi fondamentali che agiscono direttamente sul ciclo di vita del prodotto software, includendo acquisizione, fornitura, sviluppo, operazione e manutenzione.

Prodotto (Product)

Nel contesto dei design pattern creazionali, il Product è l'oggetto complesso che viene costruito. In particolare, nel pattern Builder, il Product è l'oggetto finale che viene assemblato dal Director utilizzando il Builder.

Progettazione di dettaglio

Fase della progettazione software in cui si specificano nel dettaglio i componenti, le interfacce e le relazioni tra di essi.

Progettazione logica

Fase della progettazione software in cui si definisce l'architettura del sistema senza considerare i dettagli implementativi specifici.

Progettista

Figura responsabile della progettazione dell'architettura software e delle soluzioni tecniche, garantendo che soddisfino i requisiti e siano realizzabili efficientemente.

Progetto

Insieme di attività che devono raggiungere obiettivi specifici a partire da date specificate, con un inizio e una fine fissate in calendario, disponendo di risorse limitate (persone, tempo, denaro, strumenti) e consumando risorse nel loro svolgersi.

Programmatore

Figura che implementa il codice sorgente secondo le specifiche tecniche, seguendo le best practice e gli standard di qualità definiti nel progetto.

Prompt

Input (testo e/o altri dati) fornito a un modello di intelligenza artificiale per ottenere una risposta o un output specifico.

Proponente

Azienda o organizzazione che propone un capitolato d'appalto per il progetto del corso di Ingegneria del Software, definendone requisiti e obiettivi.

Pull Model

Pattern architetturale in cui il client richiede attivamente i dati al server quando necessario.

Push Model

Pattern architetturale in cui il server invia automaticamente i dati al client senza che quest'ultimo li abbia esplicitamente richiesti.

PWA (Progressive Web App)

Applicazione web che utilizza tecnologie web moderne per offrire un'esperienza simile a quella di un'app nativa, funzionando offline e potendo essere installata sul dispositivo.

Q

Qualità

Insieme delle caratteristiche di un prodotto software che gli conferiscono la capacità di soddisfare le esigenze espresse e implicite degli stakeholder. La qualità del software si misura in base a attributi interni (visibili agli sviluppatori) ed esterni (visibili agli utenti).

Qualità architetturale

Insieme delle caratteristiche che definiscono la bontà di un'architettura software in termini di modularità, riusabilità, manutenibilità e altre proprietà desiderabili.

R

Rating System

Sistema di valutazione che assegna un punteggio o un giudizio a un'entità (es. prodotto, servizio, contenuto) in base a criteri prestabiliti.

RBAC (Role-Based Access Control)

Modello di controllo degli accessi in cui i permessi sono assegnati a ruoli specifici, e gli utenti ottengono i permessi attraverso l'assegnazione a questi ruoli.

Redattore

Membro del team responsabile della stesura e della produzione dei documenti di progetto, garantendo chiarezza, completezza e conformità alle norme stabilite.

Regression test / Test di regressione

Test eseguiti dopo una modifica al software (es. correzione di un bug, aggiunta di una nuova funzionalità) per garantire che le modifiche non abbiano introdotto nuovi difetti o abbiano rotto funzionalità esistenti che precedentemente funzionavano. L'insieme di test di regressione (regression test suite) tende a crescere nel tempo e viene spesso automatizzato.

Repository

Archivio centrale in cui vengono memorizzati e versionati i file sorgente, la documentazione e le risorse del progetto utilizzando un sistema di controllo versione.

Requisiti desiderabili

Requisiti che sono importanti ma non essenziali per il funzionamento base del sistema. La loro implementazione apporta valore aggiunto ma la loro assenza non compromette il progetto.

Requisiti funzionali

Specificano cosa il sistema deve fare, descrivendo le funzionalità, i comportamenti e le interazioni che il software deve supportare.

Requisiti non funzionali

Definiscono come il sistema deve comportarsi in termini di prestazioni, sicurezza, affidabilità, usabilità e altri attributi di qualità, senza riguardo alle funzionalità specifiche.

Requisiti obbligatori

Requisiti che devono essere necessariamente soddisfatti e la cui mancata implementazione comporterebbe il fallimento del progetto. Sono critici per il successo del sistema.

Requisiti opzionali

Requisiti che sono utili ma non necessari, e la cui implementazione dipende dalla disponibilità di risorse e tempo. Possono essere considerati per versioni future del prodotto.

Requisito

Condizione o capacità che deve essere posseduta da un sistema o componente software per soddisfare un contratto, standard, specifica o altro documento formalmente imposto.

Requisito software

Requisito specificato in termini tecnici, destinato agli sviluppatori, che descrive in modo dettagliato e misurabile una funzionalità o un vincolo del sistema software.

Requisito utente

Requisito espresso dal punto di vista dell'utente finale, descritto in linguaggio naturale e senza dettagli tecnici, focalizzato su ciò che l'utente si aspetta che il sistema faccia.

Responsabile

Figura di riferimento del progetto con compiti di coordinamento, pianificazione, gestione delle risorse e comunicazione con docenti e proponenti.

Retrospettiva

Cerimonia del framework Scrum che si tiene al termine di ogni sprint, durante la quale il team riflette sul processo di lavoro adottato, identifica ciò che ha funzionato bene e ciò che può essere migliorato, e definisce un piano di azione per incrementare l'efficacia e la qualità del lavoro negli sprint successivi.

Riusabilità

Capacità di un componente software di essere utilizzato in diversi contesti o applicazioni senza modifiche sostanziali.

Robustezza

Capacità di un sistema software di funzionare correttamente in condizioni anomali o in presenza di input non validi.

RTB (Requirements and Technology Baseline)

Baseline che combina i requisiti e le tecnologie approvate per il progetto. Definisce l'insieme dei requisiti concordati e formalmente approvati e lo stack tecnologico di riferimento, costituendo la base per lo sviluppo del progetto. Ogni modifica a questa baseline deve seguire un processo formale di controllo delle modifiche.

Ruby

Linguaggio di programmazione dinamico, open source, orientato agli oggetti e interpretato, noto per la sua semplicità e produttività.

Ruby on Rails

Framework per applicazioni web scritto in Ruby, che segue il pattern Model-View-Controller (MVC) e i principi di convenzione sopra configurazione (convention over configuration).

S

Sandbox di Sviluppo

Ambiente isolato per testare e validare funzionalità senza influenzare i sistemi di produzione, utilizzato per sviluppare e verificare nuove feature in sicurezza.

Scalabilità

Capacità di un sistema di gestire un aumento del carico di lavoro aggiungendo risorse, senza modifiche all'architettura.

Scenario

Sequenza di interazioni tra attori e sistema che descrive un percorso specifico attraverso un caso d'uso. Può essere principale (percorso di successo) o alternativo (variazioni ed eccezioni).

Scenario alternativo

Sequenza di interazioni nel caso d'uso che rappresenta un percorso diverso da quello principale, tipicamente gestendo condizioni eccezionali, errori o scelte alternative dell'utente. Descrive come il sistema reagisce in situazioni non standard.

Scenario principale

Sequenza di interazioni tra l'attore principale e il sistema che descrive il percorso di successo del caso d'uso, dove l'obiettivo viene raggiunto senza intoppi o condizioni eccezionali. Rappresenta il flusso ideale e più frequente di esecuzione.

Scope

Ambito di un progetto, che definisce i confini, i deliverables, gli obiettivi e i compiti che sono inclusi nel progetto.

SCRUM

Framework agile per la gestione dello sviluppo software che enfatizza lo sviluppo iterativo, l'adattamento ai cambiamenti e la consegna incrementale di valore.

SEMAT

(Software Engineering Method and Theory) Iniziativa internazionale per rifondare l'ingegneria del software come disciplina rigorosa, basata su un kernel di elementi essenziali comuni a tutti i metodi di sviluppo software.

Sicurezza

Insieme di caratteristiche e meccanismi che proteggono il sistema da accessi non autorizzati, garantiscono la riservatezza, l'integrità e la disponibilità dei dati, e prevengono attacchi e vulnerabilità.

Sottotipizzazione

Relazione tra tipi per cui un tipo (sottotipo) può essere utilizzato in ogni contesto in cui è atteso un altro tipo (supertipo), in accordo con il principio di sostituzione di Liskov.

Specification Tecnica

Documento che descrive in dettaglio l'architettura, il design e le scelte implementative del sistema software, guidando le attività di sviluppo.

Sprint

Periodo di tempo fisso (tipicamente 2-4 settimane) in Scrum durante il quale il team sviluppa e consegna un incremento di prodotto potenzialmente rilasciabile.

Stakeholder

Portatore di interesse, ovvero qualsiasi individuo, gruppo o organizzazione che può influenzare o essere influenzato da un progetto, un'azienda o un sistema.

Stand-alone

Nel contesto del progetto sviluppato dal team BugBusters, si riferisce a un'applicazione o modulo software sviluppato in modo autonomo e indipendente dall'infrastruttura NEXUM esistente. I moduli AI Assistant Generativo e AI Co-Pilot per i CdL saranno realizzati come applicazioni stand-alone che simulano i comportamenti e le interfacce di NEXUM dove necessario, con l'obiettivo di poter essere integrati successivamente nella piattaforma principale. Questo approccio consente uno sviluppo più flessibile e isolato, pur mantenendo la compatibilità con l'architettura e le API di NEXUM.

Stima dei costi

Processo di previsione dei costi associati alle attività di progetto, considerando risorse umane, strumenti, infrastrutture e altri fattori che influenzano il budget complessivo.

T

Technical debt / Debito tecnico

Metafora che descrive le conseguenze a lungo termine di scelte tecniche non ottimali fatte durante lo sviluppo, che potrebbero richiedere lavoro futuro per essere corrette. Il debito tecnico può derivare da scelte consapevoli (per rilasciare prima) o inconsapevoli (mancanza di conoscenza). Accumulare debito tecnico può portare a un aumento dei costi di manutenzione e a una riduzione della qualità.

Technology Baseline

Insieme delle tecnologie, framework, librerie e strumenti di sviluppo selezionati e approvati per il progetto. Definisce lo stack tecnologico di riferimento e costituisce la base per le scelte implementative, garantendo coerenza e standardizzazione nell'architettura software.

Template

Struttura o modello predefinito che definisce il layout e l'aspetto di un'interfaccia utente, separando la presentazione dalla logica di business.

Test

Processo sistematico di verifica che il software soddisfi i requisiti specificati e identifichi difetti, attraverso l'esecuzione controllata di casi di test.

Test case / Caso di prova

Specifica di input, condizioni di esecuzione e risultati attesi per un singolo test. Un caso di prova è un insieme di condizioni o variabili sotto le quali un tester determina se un sistema o una sua parte funziona correttamente.

Test di Accettazione (TA) / Collaudo

Test condotti per determinare se un sistema soddisfa i criteri di accettazione stabiliti dal cliente e per permettere al cliente di decidere se accettare o meno il sistema. Viene eseguito in un ambiente il più possibile simile a quello di produzione.

Test di Integrazione (TI)

Test che verificano l'interazione e la corretta comunicazione tra i moduli o i componenti integrati del sistema. L'obiettivo è individuare difetti nelle interfacce e nelle interazioni tra i componenti.

Test di Sistema (TS)

Test di tipo black-box che verificano il sistema software nel suo complesso, in un ambiente il più possibile simile a quello di produzione, per accertare che soddisfi tutti i requisiti software specificati nel documento di Analisi dei Requisiti (AdR). Comprende sia test funzionali (verifica delle funzionalità) sia test non-funzionali (prestazioni, sicurezza, usabilità, affidabilità). Viene

eseguito dopo il completamento del test di integrazione e prima del collaudo (test di accettazione). Misura il requirements coverage, ovvero la percentuale di requisiti software coperti e soddisfatti. Prepara la validazione finale con il committente, accertando internamente che il prodotto sia pronto per il collaudo.

Test di Unità (TU)

Test che verificano il corretto funzionamento delle singole unità (moduli, classi, funzioni) del software in isolamento. Sono tipicamente scritti e eseguiti dagli sviluppatori durante la fase di implementazione.

Test funzionale (black-box)

Approccio di testing che valida il comportamento esterno del software senza considerare la struttura interna. I test sono progettati basandosi sui requisiti funzionali e sull'esperienza dell'utente. L'obiettivo è verificare che il sistema risponda correttamente a input e condizioni specifiche.

Test strutturale (white-box)

Approccio di testing che valida la struttura interna e il flusso di controllo del software. I test sono progettati basandosi sulla conoscenza del codice sorgente, dell'architettura e della logica di implementazione. L'obiettivo è garantire che tutti i percorsi e le strutture del codice siano esercitati.

Test suite

Insieme organizzato di casi di test, tipicamente eseguiti in sequenza, per verificare un componente o un sistema. Una test suite può essere progettata per testare una specifica funzionalità o un insieme di funzionalità.

Tracciamento (dei requisiti)

Processo di documentazione e gestione delle relazioni tra i requisiti e gli elementi del sistema (come componenti, test, casi d'uso) che li soddisfano. Garantisce che tutti i requisiti siano presi in carico e permette di valutarne l'impatto in caso di modifiche.

Transazioni distribuite

Transazioni che coinvolgono multiple risorse distribuite in diversi nodi di un sistema, richiedendo meccanismi di coordinamento per garantire atomicità e consistenza.

U

Unità architetturali

Componenti fondamentali che costituiscono l'architettura di un sistema software, come moduli, componenti, connettori e dati.

V

V-Model / Modello a V

Modello di sviluppo software che estende il modello a cascata, enfatizzando la verifica e la validazione in ogni fase. Nel modello a V, per ogni fase di sviluppo (es. definizione dei requisiti, progettazione architetturale, progettazione di dettaglio) esiste una corrispondente fase di test (es. test di sistema, test di integrazione, test di unità). Il modello rappresenta graficamente la relazione tra le fasi di definizione e le fasi di test.

Validatore

Membro del team responsabile di eseguire attività di validazione, accertando che il prodotto software soddisfi le effettive esigenze del cliente e gli obiettivi di business.

Validazione

Processo che accerta che il prodotto software sviluppato soddisfi le effettive esigenze del cliente e gli obiettivi di business per i quali è stato realizzato. Risponde alla domanda "Stiamo costruendo il prodotto giusto?" e viene tipicamente effettuata attraverso test di accettazione con il cliente.

Verbale

Documento formale che registra quanto discusso, deciso e pianificato durante una riunione. Include informazioni come data, partecipanti, ordine del giorno, punti discussi, decisioni prese e azioni da intraprendere.

Verbale esterno

Verbale redatto in seguito a una riunione con soggetti esterni al gruppo di progetto, come il proponente o i docenti. Oltre ai contenuti standard di un verbale, può includere specifiche decisioni o accordi presi con le parti esterne.

Verbale interno

Verbale redatto in seguito a una riunione interna al gruppo di progetto. Documenta le discussioni e le decisioni prese dai membri del team, e viene utilizzato per tracciare lo stato di avanzamento, assegnare compiti e coordinare le attività.

Verifica

Processo sistematico che determina se i prodotti di lavoro (documenti, codice, componenti) soddisfano i requisiti e le specifiche definite per loro. Risponde alla domanda "Stiamo costruendo il prodotto nel modo giusto?" e include attività come revisioni, ispezioni e test.

Verifica vs. Validazione

La verifica è il processo di valutazione dei prodotti di lavoro di un progetto per accertare che soddisfino i requisiti specificati e le condizioni imposte all'inizio dello sviluppo. Risponde alla

domanda "Stiamo costruendo il prodotto nel modo giusto?". La validazione, invece, accerta che il prodotto finale soddisfi le effettive esigenze del cliente e gli obiettivi di business. Risponde alla domanda "Stiamo costruendo il prodotto giusto?". In sintesi, la verifica si concentra sulla correttezza del processo di sviluppo, mentre la validazione sulla correttezza del prodotto finale.

Verificatore

Membro del team responsabile di controllare che documenti, codice e altri prodotti di lavoro rispettano gli standard di qualità definiti, le norme di progetto e siano privi di errori, incoerenze o ambiguità.

Versionamento/versioning

Sistema per gestire le diverse versioni di file, codice sorgente o documenti, permettendo di tracciare le modifiche, collaborare in team e revertire a versioni precedenti.

ViewModel

Componente nel pattern MVVM che espone i dati e i comandi del Model in modo da essere facilmente legati alla View, spesso implementando notifiche di cambiamento per aggiornare automaticamente la View.

W

Way of Working

Insieme di processi, metodologie, strumenti e pratiche adottati dal team per organizzare e svolgere le attività di progetto in modo coordinato ed efficiente. Definisce come il team collabora, comunica e gestisce il lavoro quotidiano.

Workflow

Sequenza di attività che definiscono un processo di business, dove compiti, informazioni o documenti passano da un partecipante all'altro secondo regole prestabilite.