

Logo non trovato
(Logo.jpg)

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

Redattori	Alberto Autiero
Verificatori	[Nome Cognome]
Uso	Interno
Destinatari	Prof. Tullio Vardanega, Prof. Riccardo Cardin

Indice

Introduzione	4
A	5
Accoppiamento	5
Agile	5
Aggregazione	5
Amministratore	5
Analisi dei Requisiti	5
Analista	5
Approvazione	5
Associazione	5
Attore	6
Attore principale	6
Attore secondario	6
B	7
Back-end	7
Baseline	7
Best Practice	7
C	8
Capitolato	8
Casi d'uso	8
Cerimonia	8
Committente	8
Composizione	8
Cruscotto/Dashboard	8
D	9
Decisione esterna	9
Decisione interna	9
Diagrammi delle classi	9
Diagrammi dei casi d'uso	9
Diagrammi di attività	9
Diagrammi di Gantt	9
Diagrammi di PERT	9
Dipendenza	9
Dominio d'uso	9
E	10
Economicità	10
Efficacia	10
Efficienza	10
Ereditarietà	10
Ereditarietà di classe	10
Ereditarietà dell'interfaccia	10
F	11
Front-end	11
Funzionalità	11
G	12
Gestione dei rischi	12
GitHub	12
Glossario	12

I	13
Implementazione	13
Incapsulamento	13
Information Hiding	13
Issue	13
L	14
Latex	14
M	15
Milestone	15
N	16
Norme di progetto	16
O	17
Ora produttiva vs ora di orologio	17
P	18
Polimorfismo	18
Post-condizione	18
Pre-condizione	18
Preventivo	18
Progettista	18
Programmatore	18
Progetto	18
Proponente	18
Processi organizzativi	18
Processi primari	19
Processi di supporto	19
R	20
Redattore	20
Repository	20
Requisito	20
Requisiti desiderabili	20
Requisiti funzionali	20
Requisiti non funzionali	20
Requisiti obbligatori	20
Requisiti opzionali	20
Requirements Baseline	20
Requisito software	21
Requisito utente	21
Responsabile	21
S	22
Scenario	22
Scenario alternativo	22
Scenario principale	22
SCRUM	22
SEMAT	22
Sottotipizzazione	22
Specifica Tecnica	22
Sprint	22
Stima dei costi	22
T	23
Technology Baseline	23

Test	23
V	24
Validatore	24
Validazione	24
Verifica	24
Verificatore	24
W	25
Way of Working	25

Versioni del documento

Versione	Data	Descrizione	Redatto	Verificatori	Approvato
0.5.0	29/10/2025	Filtro termini e aggiunta termini mancanti	Alberto Autiero	-	-
0.4.0	28/10/2025	Aggiunti i termini: Way of Working, Economicità, Processi primari, Processi di supporto, Processi organizzativi, Specializzazione di processi, Progetto, Preventivo, Ora produttiva vs ora di orologio, Diagrammi di Gantt, Diagrammi di PERT, Allocazione delle risorse, Stima dei costi, Gestione dei rischi, RTB, SEMAT, Ereditarietà, Polimorfismo, Information hiding, Classe, Incapsulamento, Oggetto, Dipendenza, Accoppiamento, Associazione, Aggregazione, Composizione, Attore, Scenario, Brainstorming, Cerimonia, Glossario, Requisiti obbligatori, Requisiti desiderabili, Requisiti opzionali, Technology baseline	Alberto Autiero	-	-
0.3.0	24/10/2025	Aggiunti i seguenti termini: Attore principale, Attore secondario, Dominio d'uso, Piano di qualifica, Post-condizione, Pre-condizione, Requirements baseline, Scenario alternativo, Scenario principale, Use case (casi d'uso), Validazione, Verifica	Alberto Autiero	-	-
0.2.0	23/10/2025	Aggiunti i seguenti termini: Committente, Diagrammi UML, Efficienza, Requisito, Requisito software, Requisito utente	Alberto Autiero	-	-
0.1.0	22/10/2025	Prima stesura del documento	Alberto Autiero	-	-

1 Introduzione

Questo documento nasce con lo scopo di evitare qualsiasi tipo di ambiguità o dubbi riguardanti la terminologia adoperata all'interno dei documenti del progetto. Per questo motivo, vengono di seguito esposte le definizioni dei termini specifici, degli acronimi e delle parole ambigue utilizzati nella documentazione, adottando una struttura alfabetica per facilitare la navigazione del documento. Il glossario ha lo scopo di garantire una comprensione univoca dei termini da parte di tutti i membri del team e dei destinatari della documentazione.

2 A

2.1 Accoppiamento

Grado di interdipendenza tra componenti software. Un accoppiamento stretto indica forte dipendenza, mentre un accoppiamento debole minimizza le dipendenze, favorendo manutenibilità e riutilizzo del codice.

2.2 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.

2.3 Aggregazione

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

2.4 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.

2.5 Analisi dei Requisiti

Processo sistematico volto a identificare, documentare e validare i bisogni, i vincoli e le esigenze del progetto, trasformandoli in requisiti formali che guideranno lo sviluppo del software.

2.6 Analista

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

2.7 Approvazione

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

2.8 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.

2.9 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).

2.10 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.

2.11 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.

3 B

3.1 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.

3.2 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.

3.3 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.

4 C

4.1 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.

4.2 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.

4.3 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.

4.4 Committente

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

4.5 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.

4.6 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.

5 D

5.1 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.

5.2 Decisione interna

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

5.3 Diagrammi delle classi

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

5.4 Diagrammi dei casi d'uso

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

5.5 Diagrammi di attività

Diagrammi UML che modellano il flusso di controllo o il flusso di dati tra attività, utilizzati per descrivere la logica di procedura, di business o di caso d'uso.

5.6 Diagrammi di Gantt

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

5.7 Diagrammi 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).

5.8 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.

5.9 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.

6 E

6.1 Economicità

Principio gestionale che combina efficienza ed efficacia, misurando la capacità di raggiungere obiettivi prefissati (efficacia) impiegando le risorse minime indispensabili (efficienza).

6.2 Efficacia

Capacità di raggiungere gli obiettivi prefissati e produrre i risultati attesi, indipendentemente dalle risorse impiegate.

6.3 Efficienza

Rapporto tra i risultati ottenuti e le risorse impiegate per conseguirli. Un processo è efficiente quando raggiunge i suoi obiettivi utilizzando il minimo di risorse necessarie.

6.4 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.

6.5 Ereditarietà di classe

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

6.6 Ereditarietà dell'interfaccia

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

7 F

7.1 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.

7.2 Funzionalità

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

8 G

8.1 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.

8.2 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.

8.3 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.

9 I

9.1 Implementazione

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

9.2 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.

9.3 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.

9.4 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.

10 L

10.1 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.

11 M

11.1 Milestone

Punto significativo nel ciclo di vita del progetto che segna il completamento di un insieme di attività o il raggiungimento di un obiettivo importante. Serve come punto di verifica del progresso.

12 N

12.1 Norme di progetto

Insieme di regole, procedure e convenzioni stabilite dal team per garantire coerenza, qualità e uniformità nelle attività di sviluppo e nella documentazione prodotta.

13 O

13.1 Ora produttiva vs ora di orologio

Distinzione tra il tempo effettivamente dedicato a compiti produttivi (ore produttive) e il tempo totale trascorso (ore di orologio). Il rapporto tra queste due metriche indica l'efficienza nell'utilizzo del tempo.

14 P

14.1 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.

14.2 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.

14.3 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.

14.4 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.

14.5 Progettista

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

14.6 Programmatore

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

14.7 Progetto

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

14.8 Proponente

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

14.9 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.

14.10 Processi primari

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

14.11 Processi di supporto

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

15 R

15.1 Redattore

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

15.2 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.

15.3 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.

15.4 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.

15.5 Requisiti funzionali

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

15.6 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.

15.7 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.

15.8 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.

15.9 Requirements Baseline

Insieme dei requisiti concordati e formalmente approvati che costituisce il riferimento per lo sviluppo del progetto. Una volta stabilita, qualsiasi modifica alla baseline dei requisiti deve seguire un processo formale di controllo delle modifiche.

15.10 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.

15.11 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.

15.12 Responsabile

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

16 S

16.1 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).

16.2 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.

16.3 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.

16.4 SCRUM

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

16.5 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.

16.6 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.

16.7 Specifica Tecnica

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

16.8 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.

16.9 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.

17 T

17.1 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.

17.2 Test

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

18 V

18.1 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.

18.2 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.

18.3 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.

18.4 Verificatore

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

19 W

19.1 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.