



Norme di Progetto

SnakeByte (Gruppo 1):

Valeria Baleanu, Leonardo Pellizzon, Filippo Venzo, Giuseppe De Fina,
Francesco Pasqual, Christian Libralato, Luca Granziero
(2109911, 2111006, 2113705, 2113187, 2103119, 2101047, 2075512)

Informazioni documento			
Versione	Data	Stato	Destinatari
0.3.0	27/12/2025	Verificato	SnakeByte, prof. Vardanega Tullio, prof. Cardin Riccardo

Registro delle modifiche					
Versione	Data	Autore	Verifica	Approvazione	Descrizione
0.3.0	27/12/2025	V. Baleanu	F. Pasqual	-	Aggiunta metriche di qualità di processo e di prodotto.
0.2.0	08/12/2025	C. Libralato	L. Pellizzon	-	Aggiunte a processo di verifica e approvazione (sez. 3.1.4, 3.1.5). Aggiunta Sviluppo con Analisi Requisiti ai processi primari (sez. 2.2)
0.1.6	29/11/2025	F. Pasqual	V. Baleanu	-	Aggiunto processo di verifica tramite <i>pull request</i>
0.1.5	8/11/2025	Filippo Venzo	-	-	Aggiunta tabella attività completate nella struttura dei verbali
0.1.4	30/10/2025	Filippo Venzo	Luca Granziero	-	Modifica struttura tabella attività, modifiche tipografiche
0.1.3	26/10/2025	Filippo Venzo	Luca Granziero	-	Aggiunta sezioni e termini glossario, modifica convenzione date nei nomi dei file
0.1.2	23/10/2025	Filippo Venzo	Luca Granziero	-	Correzione errori ortografici e aggiunta link
0.1.1	22/10/2025	Filippo Venzo	Luca Granziero	-	Aggiunta sezioni e sotto sezioni
0.1.0	17/10/2025	Filippo Venzo	Luca Granziero	-	Prima stesura

Indice

1 Introduzione	4
1.1 Finalità del documento	4
1.2 Glossario	4
1.3 Riferimenti Normativi	4
1.4 Riferimenti Informativi	4
2 Processi primari	4
2.1 Fornitura	4
2.1.1 Attività	5
2.1.2 Documentazione risultante	5
2.2 Sviluppo	5
2.2.1 Attività	5
2.2.2 Documentazione risultante	6
2.2.3 Analisi dei Requisiti	6
2.2.3.1 Introduzione	6
2.2.3.2 Descrizione del prodotto	6
2.2.3.3 Elenco <i>casi d'uso</i>	6
2.2.3.4 Elenco requisiti	6
3 Processi di supporto	7
3.1 Documentazione	7
3.1.1 Struttura generale dei documenti	7
3.1.1.1 Prima pagina	7
3.1.1.2 Intestazione	7
3.1.1.3 Registro delle modifiche	7
3.1.1.4 Indice	8
3.1.2 Struttura dei verbali	8
3.1.2.1 Tabella delle decisioni	8
3.1.2.2 Tabella delle attività da svolgere	8
3.1.2.3 Tabella delle attività svolte	9
3.1.3 Redazione dei documenti	9
3.1.3.1 Redazione dei verbali	9
3.1.3.2 Redazione dell'analisi dei requisiti	9
3.1.4 Verifica dei documenti	9
3.1.4.1 Processo di verifica tramite <i>pull request_G</i>	9
3.1.5 Approvazione dei documenti	10
3.1.5.1 Processo di approvazione tramite <i>pull request</i>	10
3.1.6 Nomenclatura dei documenti	11
3.1.6.1 Acronimi	11
3.1.6.2 Convenzione	11
3.2 Gestione della configurazione	11
3.2.1 Numeri di versione documenti	11
3.2.2 Repository	11
3.2.2.1 Strumenti	12
4 Processi organizzativi	12
4.1 Processo di gestione	12
4.1.1 Pianificazione	12
4.1.1.1 Assegnazione delle responsabilità	12
4.1.1.2 Responsabile	12
4.1.1.3 Amministratore	12
4.1.1.4 Progettista	13
4.1.1.5 Analista	13

4.1.1.6	Verificatore	13
4.1.1.7	Programmatore	13
4.1.2	Coordinamento	14
4.1.2.1	Riunione interna fissata	14
4.1.2.2	Comunicazioni interne	14
4.1.2.3	Comunicazioni esterne	14
5	Metriche di qualità di processo	14
5.1	Processi primari	14
5.1.1	Fornitura	14
5.1.1.1	MPC1 - Planned Value (PV)	14
5.1.1.2	MPC2 - Earned Value (EV)	14
5.1.1.3	MPC3 - Actual Cost (AC)	15
5.1.1.4	MPC4 - Cost Performance Index (CPI)	15
5.1.1.5	MPC5 - Schedule Performance Index (SPI)	16
5.1.1.6	MPC6 - Estimate At Completion (EAC)	16
5.1.1.7	MPC7 - Estimate To Complete (ETC)	17
5.1.2	Sviluppo	18
5.1.2.1	MPC8 - Requirements Stability Index (RSI)	18
5.1.2.2	MPC9 - Requirements Completion Velocity (RCV)	18
5.1.2.3	MPC10 - Percentuale Attività in Ritardo (PAR)	18
5.2	Processi di supporto	19
5.2.1	Documentazione	19
5.2.1.1	MPC11 - Indice di Gulpease (IG)	19
5.2.1.2	MPC12 - Densità Errori Ortografici (DEO)	20
5.2.1.3	MPC13 - Indice di Frammentazione (IF)	20
5.2.2	Verifica	21
5.2.2.1	MPC14 - Test Success Rate (TSR)	21
5.2.3	Gestione della qualità	21
5.2.3.1	MPC15 - Percentuale Metriche Soddisfatte (PMS)	21
5.3	Processi organizzativi	22
5.3.1	Gestione dei processi	22
5.3.1.1	MPC16 - Percentuale Rischi Inattesi (PRI)	22
5.3.1.2	MPC17 - Labor Efficiency (LE)	22
6	Metriche di qualità di prodotto	23
6.1	Funzionalità	23
6.1.0.1	MPD1 - Percentuale Requisiti Obbligatori Soddisfatti (PROS)	23
6.1.0.2	MPD2 - Percentuale Requisiti Opzionali Soddisfatti (PRPS)	23
6.1.0.3	MPD3 - Percentuale Requisiti Desiderabili Soddisfatti (PRDS)	24
6.2	Affidabilità	24
6.2.0.1	MPD4 - Line Coverage (LC)	24
6.2.0.2	MPD5 - Branch Coverage (BC)	25
6.2.0.3	MPD6 - Statement Coverage (SC)	25
6.3	Usabilità	25
6.3.0.1	MPD7 - Facilità di Apprendimento (FA)	25
6.4	Efficienza	26
6.4.0.1	MPD8 - Tempo Medio di Risposta (TMR)	26
6.5	Manutenibilità	26
6.5.0.1	MPD9 - Densità dei Commenti (DC)	26
6.5.0.2	MPD10 - Complessità Ciclomatica (CC)	27

1 Introduzione

1.1 Finalità del documento

Il presente documento intende fissare le linee guida che il gruppo *SnakeByte* si impegna a rispettare ed attuare per perseguire la migliore efficienza ed efficacia nel processo di realizzazione del progetto didattico.

Il documento è strutturato secondo le norme dello Standard ISO/IEC 12207:1995 e segue quanto descritto nel *Regolamento del progetto didattico (A.a. 2025/2026)*. Presenta una descrizione dei *processi* del ciclo di vita del *software* e delle *attività* di cui sono composti. A sua volta, ogni attività, è composta da una serie di procedure metodiche dotate di obiettivi e strumenti ben definiti.

È importante notare che il documento in questione è in continua evoluzione fino al suo ritiro, poiché le norme contenute al suo interno vengono costantemente revisionate, ottimizzate ed aggiornate, seguendo un approccio incrementale.

Ogni attività svolta nell'interesse del progetto didattico e nei suoi materiali è regolamentata precedentemente all'esecuzione della stessa.

1.2 Glossario

Il documento cita alcuni termini la cui definizione può risultare ambigua. Per questo, è possibile consultare il *glossario_G* il quale contiene le definizioni di tali espressioni, che saranno marcate da una lettera *G* a pedice.

1.3 Riferimenti Normativi

- **Standard ISO/IEC 12207:1995:**

https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
(consultato il 30/10/2025);

- **Regolamento del progetto didattico:**

<https://www.math.unipd.it/~tullio/IS-1/2025/Dispense/PD1.pdf>
(consultato il 23/10/2025).

1.4 Riferimenti Informativi

- **Sito dedicato alla documentazione**

<https://snakebyteteam.github.io>
(consultato il 23/10/2025)

- **Glossario:**

<https://snakebyteteam.github.io/glossary.html>
(consultato il 30/10/2025)

2 Processi primari

Le attività che compongono i processi primari considerate in questa sede sono un sottoinsieme proprio delle attività previste dallo Standard ISO/IEC 12207:1995.

2.1 Fornitura

Il processo di fornitura, come specificato nello Standard ISO/IEC 12207:1995, definisce le attività dell'organizzazione che fornisce il prodotto software all'acquirente, dalla concezione fino alla consegna del prodotto. Viene istanziato conseguentemente alla redazione della *Valutazione dei capitoli_G*.

2.1.1 Attività

Il processo di fornitura si compone delle seguenti attività

- **Avviamento:** revisione delle proposte dei richiedenti. Per i capitolati di maggiore interesse vengono mandate delle comunicazioni via mail per eventuali approfondimenti;
- **Preparazione della risposta:** viene scelto il capitolato per cui ci vuole candidare sulla base delle considerazioni fatte nella fase precedente e viene preparato il documento di **candidatura**.

2.1.2 Documentazione risultante

La documentazione prodotta durante le attività di fornitura, la quale verrà consegnata ai committenti_G quali prof. Tullio Vardanega, prof. Riccardo Cardin e all'azienda proponente è la seguente

- **Valutazioni dei capitolati** contenente
 - Titolo del capitolato e nome dell'azienda proponente;
 - Una breve descrizione del capitolato e dei suoi obiettivi;
 - Considerazioni del gruppo;
- **Dichiarazione degli impegni** contenente
 - Impegni orari e suddivisione dei ruoli;
 - Preventivo dei costi totali del progetto (calcolato secondo il *Regolamento del progetto didattico*);
 - Data prevista di consegna.
- **Lettera di presentazione** contenente
 - Scelta del capitolato;
 - Motivazione della scelta;
 - Riassunto costi complessivi e data prevista di consegna.

2.2 Sviluppo

Il processo di Sviluppo consiste in un'insieme di attività necessarie per la realizzazione del prodotto software, queste sono particolarmente orientate verso analisi dei requisiti, design, codifica, integrazione e testing; si tratta infatti del processo primario principale che guida la realizzazione del prodotto software per la quasi totalità del ciclo di vita.

2.2.1 Attività

Il processo di sviluppo si compone delle seguenti attività:

- **System requirements analysis;**
- **Software requirements analysis;**
- **Software architectural design;**
- **Software detailed design;**
- **Software coding and testing;**
- **Software qualification testing;**
- **Software installation;**

2.2.2 Documentazione risultante

La documentazione prodotta durante il processo di Sviluppo consiste in:

- **Diagrammi UML_G**: diagrammi realizzati secondo lo standard *UML2.5G* utili alla definizione dei *casi d'uso_G* all'interno dell'Analisi dei Requisiti.
- **Analisi dei Requisiti**: documento che racchiude i risultati della *System requirements analysis* (requisiti funzionali) e *Software requirements analysis* (requisiti non funzionali).

2.2.3 Analisi dei Requisiti

Il suddetto documento, la cui redazione è affidata al ruolo dell'Analista, è necessario al fine di determinare e tracciare l'insieme di requisiti che l'applicativo deve soddisfare.

L'analisi è strutturata in maniera tale da ricavare i requisiti a partire dai *casi d'uso*, che consistono in una serie di scenari che condividono uno scopo per un utente che interagisce con il sistema. La ricerca dei casi d'uso avviene tramite *brainstorming_G* interno al gruppo e si avvale di *feedback_G* da parte della Proponente.

Il documento è strutturato nel seguente modo:

2.2.3.1 Introduzione

Una breve introduzione descrive le finalità del documento, i Riferimenti Informativi e Normativi.

2.2.3.2 Descrizione del prodotto

In questa sezione viene descritto lo scopo e l'obiettivo principale del prodotto, oltre che le principali funzionalità che deve possedere e l'insieme di utenti a cui è destinato.

2.2.3.3 Elenco casi d'uso

I *casi d'uso* vengono elencati seguendo la numerazione associata, sono presentati attraverso un diagramma *UML* e una descrizione testuale in forma tabellare contenente sia le informazioni all'interno del diagramma sia quelle non rappresentabili da esso, tra cui Pre-condizioni e le Post-condizioni.

La struttura è la seguente:

Campo	Descrizione
Attori	Coloro che partecipano attivamente al caso d'uso per raggiungere un preciso obiettivo
Pre-condizioni	Condizioni che devono essere soddisfatte prima dello scenario descritto dal caso d'uso
Post-condizioni	Condizioni che risultano soddisfatte dopo il completamento dello scenario principale del caso d'uso. Se viene completato uno scenario alternativo, saranno soddisfatte le Post-condizioni di quest'ultimo
Trigger	La motivazione che porta l'utente a svolgere i passi del caso d'uso
Scenario principale	Sequenza di passi che l'utente deve seguire per completare il caso d'uso
Scenari alternativi	Scenario divergente dal principale per il verificarsi di una particolare condizione
Estensioni	Casi d'uso ulteriori eseguiti al verificarsi di una particolare condizione nel caso d'uso primario. Modificano Scenario e Post-condizioni
Inclusioni	Casi d'uso ulteriori eseguiti al fine di completare il caso d'uso principale. Vengono eseguiti tutti incondizionatamente.

La crezione della tabella è facilitata dall'utilizzo del template `templateAdR.sty`.

Non tutti i *casi d'uso* necessitano della tabella nella sua interezza, la presenza dei campi: Trigger, Scenari alternativi, Estensioni e Inclusioni dipende dalla situazione.

2.2.3.4 Elenco requisiti

I requisiti sono identificati da un codice univoco e sono associati ai casi d'uso da cui sono stati generati.

3 Processi di supporto

3.1 Documentazione

3.1.1 Struttura generale dei documenti

Le seguenti sezioni illustrano le componenti che ogni documento creato deve avere. Ogni documento deve essere redatto utilizzando il linguaggio L^AT_EX_G, in particolare il file `template.sty` e `template` forniti nel repository interno.

3.1.1.1 Prima pagina

La prima pagina di ogni documento deve riportare, in ordine di posizionamento dall'alto verso il basso

- Logo del gruppo *SnakeByte*;
- Titolo del documento;
- Il nome e il numero del gruppo *SnakeByte*;
- Nome e cognome di ogni componente e relativo numero di matricola UniPD;
- Informazioni generali del documento quali
 - Versione attuale;
 - Data di creazione della versione;
 - Lo stato attuale;
 - I destinatari del documento.
- Contatto email del gruppo *SnakeByte*.

3.1.1.2 Intestazione

Ogni pagina di qualsiasi documento deve riportare come intestazione

- Nome del gruppo *SnakeByte*;
- Titolo del relativo documento.

3.1.1.3 Registro delle modifiche

Tutti i documenti, interni ed esterni, devono riportare a partire dalla seconda pagina il *registro delle modifiche* sottoforma di tabella, la quale deve riassumere

- Versione del documento;
- Data di creazione della versione;
- Autore della versione;
- Verificatore della versione;
- Approvatore della versione;
- Descrizione riassuntiva delle modifiche alla versione precedente.

Ogni modifica ad un documento scatena la creazione di una nuova versione di esso e quindi la compilazione di una nuova riga, verso il basso, della tabella.

3.1.1.4 Indice

Ogni documento deve riportare l'indice dove saranno elencati i titoli di tutte le sezioni e sottosezioni. Ogni titolo deve essere provvisto di link che porta alla sezione associata all'interno dello stesso documento.

Metodologie

- Questo deve essere fatto tramite il pacchetto `hyperref` fornito dal linguaggio L^AT_EX_G

3.1.2 Struttura dei verbali

I verbali sia interni che esterni, oltre alla struttura descritta nel capitolo §3.1.1, devono essere composti dalle seguenti sezioni

- **Informazioni**, contenente
 - Data di svolgimento;
 - Ora inizio;
 - Ora fine;
 - Modalità di svolgimento (Presenza, Online o tramite canali asincroni).
- **Presenze**, contenente, in forma tabellare, le seguenti informazioni
 - Nome e cognome di tutti i membri;
 - Ruolo (ND se non definito);
 - Presenza alla riunione.
- **Ordine del giorno**, con all'interno una lista degli argomenti che vengono trattati all'interno dell'incontro;
- **Approfondimento** degli argomenti ordine del giorno;
- **Decisioni** (Sezione §3.1.2.1);
- **Attività da svolgere** (Sezione §3.1.2.2);
- **Attività svolte** (Sezione §3.1.2.3)

3.1.2.1 Tabella delle decisioni

Per il tracciamento e l'organizzazione di ogni decisione presa collettivamente dal gruppo *SnakeByte*, al termine di ogni verbale, deve essere presente una tabella che riporta le decisioni prese in seguito alla riunione in questione. Per ogni decisione deve essere riportato

- Identificativo alfanumerico della decisione, così composto
`v{i, e}_AAAA_MM_GG.d<numero_decisione>;`
- Descrizione testuale della decisione presa.

3.1.2.2 Tabella delle attività da svolgere

Per il tracciamento delle attività da eseguire, emerse durante l'incontro trattato dal verbale, deve essere presente una tabella che riporta una lista di queste, riassumendo le seguenti informazioni

- Identificativo alfanumerico dell'attività, così composto
`v{i, e}_AAAA_MM_GG.a<numero_attività>;`
- Descrizione testuale dell'attività;
- Id GitHub Issue associata all'attività (carattere “-” se non presente);
- Assegnatario dell'attività;
- Scadenza di completamento.

3.1.2.3 Tabella delle attività svolte

Per il tracciamento delle attività svolte, deve essere presente una tabella che riporta una lista di queste, riassumendo le seguenti informazioni

- Identificativo alfanumerico dell'attività, come definito nella Sezione §3.1.2.2;
- Id GitHub Issue associata all'attività (carattere “-” se non presente);
- Data di completamento.

3.1.3 Redazione dei documenti

La stesura e l'aggiornamento dei documenti è affidata alla persona che ricopre, in quel periodo, il ruolo di *amministratore_G*. Le informazioni che verranno usate per redarre i documenti saranno ricavate dalle riunioni e comunicazioni con gli altri componenti del gruppo.

3.1.3.1 Redazione dei verbali

Differentemente dagli altri documenti, ogni verbale viene redatto da una persona decisa durante la rispettiva riunione.

3.1.3.2 Redazione dell'analisi dei requisiti

Differentemente dagli altri documenti, l'analisi dei requisiti viene redatta dalla persona incaricata in quel momento del ruolo di *analista* (Sezione §4.1.1.5).

3.1.4 Verifica dei documenti

Ogni documento dopo essere stato redatto, deve essere verificato, ovvero la correttezza delle informazioni in esso contenute deve essere confermata.

L'azione di verifica dei documenti viene effettuata dalla persona che ricopre, in quel periodo, il ruolo di *Verificatore*.

3.1.4.1 Processo di verifica tramite *pull request_G*

Il processo di verifica dei documenti adotta il meccanismo delle *pull request* offerto dalla piattaforma *GitHub*. Tale strumento garantisce che nessuna modifica venga integrata nel *branch* principale (**main**) o nel *branch* di sviluppo principale (**develop**) senza un controllo esplicito del verificatore, il quale può approvare il documento o richiedere ulteriori modifiche direttamente all'interno della sezione *pull request* di *GitHub* tramite i commenti.

L'intero ciclo di verifica, incluse eventuali correzioni successive, avviene all'interno di un'unica *pull request*, garantendo la tracciabilità delle discussioni e dei *commit* di modifica.

Di seguito viene definito il *workflow* operativo:

1. Esecuzione modifiche

Il redattore crea un nuovo *branch* denominato `modifica-<nome_documento>` nel suo repository locale (la versione del documento non va inclusa all'interno del nome del *branch* di modifica).

Per definizione, le modifiche su documenti diversi dovranno essere presentate attraverso *branch* di modifica e *pull request* diverse.

2. Apertura della Richiesta

Il redattore, completato il lavoro in locale sul *branch* dedicato, effettua il *push* in remoto del *branch* di modifica e apre una nuova *pull request* verso il *branch* **develop**. In questa fase deve:

- definire come assegnatario e reviewer il verificatore e impostare i *tag*;
- collegare la richiesta alla relativa *Issue* di progetto.

3. Verifica

Il verificatore riceve la notifica ed esamina le modifiche. A questo punto si presentano due scenari:

- (a) **Esito Positivo:** Se il lavoro è corretto e completo, il verificatore approva la *pull request* e procede al *merge* automatico del branch di modifica nel branch di **develop**. L'operazione porta all'eliminazione automatica del branch di modifica previa attivazione dell'impostazione dedicata di GitHub (Settings>General>Automatically delete head branches). Il processo di verifica è concluso.
Spetta all'autore eliminare manualmente il branch di modifica nel suo repository locale.
- (b) **Esito Negativo:** Se sono necessari cambiamenti, il verificatore inserisce commenti *inline* sui punti critici e imposta lo stato della revisione su *Request changes*.

4. Risoluzione e Nuova Verifica

Nel caso siano state richieste modifiche, il redattore:

- (a) viene notificato;
- (b) applica le correzioni in locale ed effettua un nuovo *push* sullo stesso branch (aggiornando automaticamente la *pull request* esistente);
- (c) risponde ai commenti o li risolve e richiede una nuova revisione (*Re-request review*);
- (d) in caso di esito di verifica negativo itera il punto 4.

3.1.5 Approvazione dei documenti

Ogni documento, dopo essere stato verificato, deve essere approvato per poter essere rilasciato come documentazione ufficiale del progetto nel branch **main**. L'approvazione avviene in maniera simile alla verifica da chi, in quel periodo, ricopre la figura di *responsabile_G*.

3.1.5.1 Processo di approvazione tramite *pull request*

I documenti vengono approvati in maniera progressiva al fine di evitare l'accumulo di un carico di lavoro insostenibile per il responsabile quando si raggiunge una baseline. Una volta che tutti i documenti contenuti nel branch **develop** sono stati approvati si può procedere con il *merge* del branch **develop** verso il branch **main**.

Il momento opportuno per sottoporre un documento ad approvazione non è a seguito della verifica ma è definito in base alle tempistiche imposte dalle baselines.

L'approvazione di un singolo documento consiste nei seguenti passi:

1. Aggiornamento versione

Il redattore crea un branch locale denominato **approvazione-<nome_documento>** (la versione del documento non va inclusa nel nome) in cui modifica **esclusivamente** la versione aumentando l'indice *Major* e azzerando gli altri (X.0.0).

2. Apertura Richiesta

Viene effettuata la *push* e aperta una *pull request* verso il branch **develop**. Devono essere impostati i *tag*, l'assegnatario e il *reviewer* devono coincidere con il responsabile.

3. Approvazione

Il responsabile viene notificato, controlla il documento e fornisce un verdetto:

- (a) **Esito Positivo:** il documento è approvato con successo, il responsabile approva la *pull request* e il *merge* sul branch **develop**.
Spetta al redattore eliminare manualmente il branch di approvazione in locale.
- (b) **Esito Negativo:** il documento non è adatto e richiede variazioni che vengono comunicate come commenti, lo stato revisione viene impostato a *Request Changes*.

4. Risoluzione e Nuova Approvazione

In caso di esito di approvazione negativo, il redattore:

- (a) viene notificato;

- (b) applica le modifiche e ripete il processo di verifica (Sezione §3.1.4.1);
- (c) rinnova la *pull request* esistente per l'approvazione;
- (d) in caso di esito negativo itera il punto 4.

3.1.6 Nomenclatura dei documenti

3.1.6.1 Acronimi

Nella nomenclatura e all'interno dei documenti sono utilizzati i seguenti acronimi

Abbreviazione	Significato
VI	Verbale interno
VE	Verbale esterno
NdP	Norme di Progetto
PB	Product Baseline
RTB	Requirements and Technology Baseline

3.1.6.2 Convenzione

La convenzione in uso per la nomenclatura dei file è la seguente (dove X, Y e Z sono le versione descritte nella sezione §3.2.1)

- **Verbali:** vi_AAAA_MM.GG_vX.Y.Z
- **Generale:** <ACR.DOC.>_vX.Y.Z (dove <ACR.DOC.> è l'acronimo del documento)

3.2 Gestione della configurazione

La gestione della configurazione è l'insieme delle attività volte a identificare, tracciare e controllare le modifiche apportate a qualsiasi elemento nel progetto.

Secondo lo Standard ISO/IEC 12207:1995, le azioni di gestione della configurazione devono controllare le modifiche e i rilasci degli elementi, registrare e segnalare lo stato degli elementi e le richieste di modifica, garantire la completezza, la coerenza e la correttezza degli elementi, e controllare l'immagazzinamento, la movimentazione e la consegna degli elementi.

3.2.1 Numeri di versione documenti

La convenzione per il numero di versione dei documenti è il formato X.Y.Z (*MAJOR.MINOR.PATCH*) ogni documento parte dalla prima versione 0.1.0

- Ogni modifica minore (*patch version*), come correzione di errori grammaticali o aggiunta d'informazioni meno significative, fa avanzare Z di una unità;
- Ogni modifica maggiore (*minor version*), come aggiunta di sezioni o modifiche sostanziali fa avanzare Y di una unità;
- La cifra X avanza solamente quando l'approvazione da parte del responsabile termina con successo (*major version*).

Ogni avanzamento delle cifre X e Y riportano le cifre alla loro destra a 0.

3.2.2 Repository

I repository creati per la gestione della configurazione sono

- **MVP:** repository contenente il Minimum Viable Product del progetto;
- **snakebyte.github.io:** repository contenente il codice sorgente del sito web dedicato alla documentazione del progetto e la documentazione stessa.

3.2.2.1 Strumenti

Gli strumenti adottati per la gestione della configurazione e del versionamento dei file di progetto sono:

- **Git_G**: Version Control System distribuito e OpenSource
- **Github_G**: servizio di hosting per progetti software e implementazione di Git.

4 Processi organizzativi

4.1 Processo di gestione

Il processo di gestione contiene le attività e i compiti generici che possono essere impiegati da qualsiasi ruolo che debba gestire i rispettivi processi.

4.1.1 Pianificazione

4.1.1.1 Assegnazione delle responsabilità

Per tutta la durata del progetto, per ragioni formative, i membri del gruppo ricopriranno a rotazione 6 ruoli fondamentali nel campo dell'ingegneria del software. Ogni componente del gruppo *SnakeByte* dovrà ricoprire almeno una volta tutti i seguenti ruoli:

- Responsabile (Sezione §4.1.1.2)
- Amministratore (Sezione §4.1.1.3)
- Progettista (Sezione §4.1.1.4)
- Analista (Sezione §4.1.1.5)
- Verificatore (Sezione §4.1.1.6)
- Programmatore (Sezione §4.1.1.7)

La rotazione dei ruoli avviene tra la fine di un periodo (*sprint_G*) e l'inizio di quello successivo.

4.1.1.2 Responsabile

Il responsabile (*Project Manager*) governa il *team* e rappresenta il progetto verso l'esterno. Deve avere conoscenze e capacità tecniche per valutare rischi, scelte e alternative.

I compiti principali del responsabile sono

- Prendere scelte nell'interesse del gruppo;
- Approvare il lavoro realizzato dagli altri ruoli;
- Pianificare le attività e gestire le risorse necessarie;
- Coordinare e relazionarsi con l'esterno

Le persone incaricate come responsabile devono essere una per periodo.

4.1.1.3 Amministratore

L'amministratore di sistema (*Sysadmin*) definisce, controlla e manutiene l'ambiente informatico di lavoro.

- Definisce, seleziona e mette in opera le risorse informatiche a supporto delle *Norme di Progetto*;
- Gestisce le segnalazioni (*ticket*) sul non funzionamento dell'infrastruttura.

Le persone incaricate come amministratore devono essere una per periodo.

4.1.1.4 Progettista

Il progettista ha competenze tecniche e tecnologiche aggiornate e per questo ha il ruolo di determinare le scelte realizzative del prodotto. Segue il progetto durante la fase di sviluppo software ma non durante la sua manutenzione.

I principali compiti del progettista sono

- Ideare l'architettura del prodotto in modo da ottenere la migliore efficienza ed efficacia possibile;
- Supervisionare la fase di codifica supportando i programmatori.

La realizzazione del progetto a regola d'arte richiede la presenza di più progettisti, i quali devono aver la possibilità di confrontarsi tra di loro in modo da prendere le scelte attuative in maniera critica.

4.1.1.5 Analista

L'analista conosce il dominio del problema ed ha competenze avanzate sull'analisi dei requisiti. Ha molta influenza sul successo del prodotto. Non segue il progetto fino alla consegna, ma il suo intervento è richiesto solo nei momenti di bisogno.

I suoi compiti principali sono

- Analizzare i requisiti e il dominio applicativo;
- Definire fattibilità e obiettivi delle attività;
- Redigere il documento *Analisi dei requisiti*.

Più membri possono essere occupati in questo ruolo contemporaneamente.

4.1.1.6 Verificatore

Il compito principale del verificatore è analizzare la conformità agli obiettivi del lavoro svolto dagli altri ruoli presenti nel progetto. Ha competenze tecniche e profonda conoscenza delle *Norme di Progetto*.

I principali compiti del verificatore sono

- Verificare che gli artefatti software e di documentazione siano aderenti alle *Norme di Progetto* e agli obiettivi;
- Segnalare eventuali errori da correggere;
- Redigere test di verifica e di aderenza ai requisiti per i prodotti software.

Più membri possono essere occupati in questo ruolo contemporaneamente.

4.1.1.7 Programmatore

Il programmatore è la figura responsabile della realizzazione e manutenzione del prodotto software. Ha competenze tecniche ampie ma deleghe limitate.

I principali compiti del programmatore sono

- Scrivere codice secondo le specifiche del *progettista* e che persegue gli obiettivi e requisiti definiti dall'*analista*;
- Redigere la documentazione tecnica (*Manuale*) del prodotto.

È il ruolo in cui vengono occupati più membri contemporaneamente.

4.1.2 Coordinamento

4.1.2.1 Riunione interna fissata

Il gruppo *SnakeByte* ha scelto di fissare un giorno all'interno della settimana lavorativa in cui si svolge una riunione in presenza a cui prendono parte i componenti del gruppo. La partecipazione dei membri è richiesta ma non tassativa. Inoltre nel caso di impossibilità della maggior parte del gruppo la riunione viene annullata.

Il giorno attuale in cui si tiene la riunione è il *Lunedì* alle ore *12.15* circa.

Altre eventuali riunioni possono essere fissate tramite confronto e accordo tra i componenti del gruppo su giorno e ora.

4.1.2.2 Comunicazioni interne

Per le comunicazioni interne, il gruppo *SnakeByte* ha individuato come mezzo per la trasmissione di informazioni in formato sincrono e asincrono la piattaforma *Discord_G*.

4.1.2.3 Comunicazioni esterne

Per le comunicazioni esterne, il gruppo *SnakeByte* ha individuato come mezzo per la trasmissione di informazioni in formato asincrono i messaggi email, esclusivamente tramite l'indirizzo ufficiale del gruppo (*snakebyteteam@gmail.com*). Mentre per le comunicazioni sincrone l'applicazione di videoconferenze *Google Meet*.

5 Metriche di qualità di processo

In questa sezione vengono definite le metriche di qualità adottate per monitorare e valutare in modo quantitativo l'efficacia e l'efficienza dei processi utilizzati durante lo sviluppo del progetto.

Ogni metrica di qualità di processo è identificata da un codice univoco della forma *MPCX*, dove *X* rappresenta un numero intero progressivo.

5.1 Processi primari

5.1.1 Fornitura

5.1.1.1 MPC1 - Planned Value (PV)

Codice	MPC1
Formula	$PV_k = BAC \cdot \text{Percentuale completamento pianificato allo sprint } k$ Dove: <ul style="list-style-type: none"> • <i>BAC</i> = Budget totale previsto (<i>Budget At Completion</i>); • <i>k</i> = Numero dello sprint di riferimento.
Descrizione	Valore pianificato del lavoro che, secondo il piano, dovrebbe essere completato fino alla fine di un determinato <i>sprint</i> . Il <i>Planned Value</i> viene calcolato al termine di ciascuno <i>sprint</i> per monitorare l'avanzamento del progetto rispetto alla pianificazione. Questa metrica è utile perché fornisce un punto di riferimento temporale per valutare se il progetto sta procedendo secondo la schedulazione prevista, permettendo di identificare tempestivamente eventuali ritardi o anticipi rispetto al piano originale.

5.1.1.2 MPC2 - Earned Value (EV)

Codice	MPC2
--------	------

Formula	$EV_k = BAC \cdot \text{Percentuale completamento effettivo allo sprint } k$ Dove: <ul style="list-style-type: none">• BAC = Budget totale previsto (<i>Budget At Completion</i>);• k = Numero dello sprint di riferimento.
Descrizione	Valore effettivo del lavoro che è stato completato fino alla fine di un determinato <i>sprint</i> . L' <i>Earned Value</i> viene calcolato al termine di ciascuno sprint per monitorare il progresso reale del progetto rispetto al piano. Questa metrica è utile per confrontare il lavoro completato sia con i costi sostenuti (tramite CPI) che con la pianificazione temporale (tramite SPI), costituendo quindi la base per tutte le analisi di performance.

5.1.1.3 MPC3 - Actual Cost (AC)

Codice	MPC3
Formula	$AC_k = \sum_{i=1}^k \text{Costo sostenuto nello sprint } i$ Dove: <ul style="list-style-type: none">• k = Numero dello sprint di riferimento.
Descrizione	Rappresenta il costo effettivo, cioè il costo reale sostenuto per il lavoro svolto fino ad un determinato <i>sprint</i> . Viene calcolato alla fine di ogni <i>sprint</i> . Questa metrica è utile per confrontare le spese effettive con il budget pianificato e con il valore prodotto, consentendo di rilevare tempestivamente sforamenti di budget o inefficienze nell'utilizzo delle risorse.

5.1.1.4 MPC4 - Cost Performance Index (CPI)

Codice	MPC4
Formula	$CPI_k = \frac{EV_k}{AC_k}$ Dove: <ul style="list-style-type: none">• EV_k = <i>Earned Value</i> allo <i>sprint</i> k (si veda §5.1.1.2);• AC_k = <i>Actual Cost</i> allo <i>sprint</i> k (si veda §5.1.1.3);• k = Numero dello sprint di riferimento.

Descrizione	Rappresenta l'efficienza con cui il budget viene utilizzato e corrisponde al rapporto tra il valore del lavoro completato e il costo reale sostenuto fino ad un determinato <i>sprint</i> . Questa metrica è utile per quantificare l'efficienza nell'uso delle risorse e di prevedere se il progetto terminerà entro il budget previsto. Un monitoraggio costante del CPI permette di identificare tempestivamente inefficienze e di adottare misure correttive per evitare sforamenti significativi. Interpretazione dei valori: <ul style="list-style-type: none"> • CPI > 1: il progetto è sotto il budget; • CPI < 1: il progetto è sopra il budget; • CPI = 1: il progetto è in linea con il budget pianificato.
--------------------	---

5.1.1.5 MPC5 - Schedule Performance Index (SPI)

Codice	MPC5
Formula	$SPI_k = \frac{EV_k}{PV_k}$ <p>Dove:</p> <ul style="list-style-type: none"> • EV_k = Earned Value allo sprint k (si veda §5.1.1.2); • PV_k = Planned Value allo sprint k (si veda §5.1.1.1); • k = Numero dello sprint di riferimento.
Descrizione	Rappresenta l'efficienza con cui il lavoro viene completato rispetto alla pianificazione e corrisponde al rapporto tra il valore del lavoro completato e il valore pianificato del lavoro fino ad un determinato <i>sprint</i> . Questa metrica è utile per valutare se il gruppo ha rispettato le scadenze previste e permette di stimare se il progetto terminerà nei tempi pianificati. Interpretazione dei valori: <ul style="list-style-type: none"> • SPI > 1: il progetto è in anticipo rispetto alla pianificazione; • SPI < 1: il progetto è in ritardo rispetto alla pianificazione; • SPI = 1: il progetto procede secondo i tempi previsti. <p>Un SPI costantemente inferiore a 1 segnala la necessità di rivedere la pianificazione o di allocare risorse aggiuntive per recuperare i ritardi accumulati.</p>

5.1.1.6 MPC6 - Estimate At Completion (EAC)

Codice	MPC6
---------------	------

Formula	$EAC_k = AC_k + \frac{BAC - EV_k}{CPI_k \cdot SPI_k}$ <p>Dove:</p> <ul style="list-style-type: none"> • AC_k = Actual Cost allo sprint k (si veda §5.1.1.3); • BAC = Budget totale previsto (<i>Budget At Completion</i>); • EV_k = Earned Value allo sprint k (si veda §5.1.1.2); • CPI_k = Cost Performance Index allo sprint k (si veda §5.1.1.4); • SPI_k = Schedule Performance Index allo sprint k (si veda §5.1.1.5); • k = Numero dello sprint di riferimento.
Descrizione	<p>Stima del costo totale finale del progetto al completamento, calcolata proiettando le performance attuali (sia in termini di costi che di schedulazione) sul lavoro rimanente, al termine di un determinato <i>sprint</i>. La metrica <i>EAC</i> fornisce una previsione realistica del budget necessario considerando le inefficienze o efficienze riscontrate durante l'esecuzione del progetto. La formula si compone di due elementi:</p> <ul style="list-style-type: none"> • Costi già sostenuti (AC): il totale delle spese effettivamente sostenute fino allo sprint k; • Stima del lavoro rimanente ($\frac{BAC - EV}{CPI \cdot SPI}$): il valore del lavoro ancora da completare (BAC - EV), corretto in base alle performance attuali di costo (CPI) e schedulazione (SPI). <p>Il denominatore $CPI \cdot SPI$ rappresenta l'efficienza complessiva del progetto: valori inferiori a 1 indicano che il lavoro rimanente costerà più del previsto, mentre valori superiori a 1 suggeriscono risparmi potenziali. Questa metrica consente di identificare tempestivamente potenziali sforamenti di budget, permettendo di applicare azioni correttive prima che gli scostamenti diventino critici.</p>

5.1.1.7 MPC7 - Estimate To Complete (ETC)

Codice	MPC7
Formula	$ETC_k = EAC_k - AC_k$ <p>Dove:</p> <ul style="list-style-type: none"> • EAC_k = Estimate at Completion allo sprint k (si veda §5.1.1.6); • AC_k = Actual Cost allo sprint k (si veda §5.1.1.3); • k = Numero dello sprint di riferimento.
Descrizione	<p>Stima del costo necessario per completare il lavoro rimanente del progetto, calcolata al termine di un determinato <i>sprint</i>. Rappresenta la proiezione delle risorse economiche ancora necessarie per portare a termine tutte le attività non ancora completate e viene calcolata come la differenza tra il costo finale stimato e il costo sostenuto. Questa metrica è utile perché permette di valutare se le risorse economiche residue sono sufficienti per completare il progetto.</p>

5.1.2 Sviluppo

5.1.2.1 MPC8 - Requirements Stability Index (RSI)

Codice	MPC8
Formula	$RSI = \frac{R_{iniziali} - (R_{modificati} + R_{cancellati})}{R_{iniziali} + R_{aggiunti}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $R_{iniziali}$ = Numero di requisiti definiti all'inizio del progetto; • $R_{modificati}$ = Numero di requisiti iniziali che hanno subito modifiche; • $R_{cancellati}$ = Numero di requisiti iniziali che sono stati rimossi; • $R_{aggiunti}$ = Numero di nuovi requisiti aggiunti durante il progetto.
Descrizione	<p>Indice che misura la stabilità dei requisiti del progetto durante il suo ciclo di vita, quantificando la percentuale di requisiti che non hanno subito modifiche, aggiunte o cancellazioni rispetto al totale. Interpretazione dei valori:</p> <ul style="list-style-type: none"> • $RSI \geq 85\%$: requisiti molto stabili, eccellente analisi iniziale; • $60\% \leq RSI < 85\%$: stabilità accettabile, alcune modifiche sono normali; • $40\% \leq RSI < 60\%$: elevata volatilità, problemi nella definizione dei requisiti; • $RSI < 40\%$: instabilità critica.

5.1.2.2 MPC9 - Requirements Completion Velocity (RCV)

Codice	MPC9
Formula	$RCV = \frac{\sum_{i=1}^k RC_i}{k}$ <p>Dove:</p> <ul style="list-style-type: none"> • RC_i = Numero di requisiti completati nello <i>sprint</i> i; • k = Numero dello <i>sprint</i> di riferimento.
Descrizione	Rappresenta il numero medio di requisiti completati per <i>sprint</i> , calcolato su un periodo di riferimento. Questa metrica è utile per la pianificazione futura: permette di stimare realisticamente quanti requisiti possono essere completati negli <i>sprint</i> rimanenti e di prevedere la data di completamento del progetto. Una RCV stabile indica un processo di sviluppo maturo e prevedibile, mentre variazioni significative possono segnalare problemi di stima o complessità variabile dei requisiti.

5.1.2.3 MPC10 - Percentuale Attività in Ritardo (PAR)

Codice	MPC10
---------------	-------

Formula	$PAR_k = \frac{ACR_k}{ACT_k} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • ACR_k = Numero di attività completate in ritardo nello <i>sprint</i> k; • ACT_k = Numero di attività completate in totale nello <i>sprint</i> k; • k = Numero dello <i>sprint</i> di riferimento.
Descrizione	<p>Rappresenta la percentuale di attività completate in ritardo rispetto alla scadenza prevista durante un determinato <i>sprint</i>. Questa metrica è fondamentale per valutare l'efficacia della pianificazione e l'aderenza alle tempistiche: un valore elevato indica problemi sistematici nella stima dei tempi, nella gestione delle priorità o nella disponibilità delle risorse. Permette di identificare problematiche nel processo di sviluppo e di adottare azioni correttive per migliorare la prevedibilità delle consegne e il rispetto delle <i>milestone</i> di progetto. Interpretazione dei valori:</p> <ul style="list-style-type: none"> • PAR = 0%: tutte le attività sono state completate nei tempi previsti; • PAR \leq 15%: buona pianificazione, anche se con alcuni ritardi; • 15% < PAR \leq 30%: problemi moderati di stima o esecuzione; • PAR > 30%: problemi significativi, necessario rivedere il processo di pianificazione.

5.2 Processi di supporto

5.2.1 Documentazione

5.2.1.1 MPC11 - Indice di Gulpease (IG)

Codice	MPC11
Formula	$IG = 89 + \frac{300 \cdot N_{frasi} - 10 \cdot N_{lettere}}{N_{parole}}$ <p>Dove:</p> <ul style="list-style-type: none"> • N_{frasi} = Numero di frasi nel testo; • $N_{lettere}$ = Numero di lettere nel testo; • N_{parole} = Numero totale di parole nel testo.

Descrizione	Indice che misura la leggibilità di un testo in lingua italiana, basato sulla lunghezza delle parole e delle frasi. Questa metrica è utile per garantire la qualità della documentazione di progetto: assicura che i documenti siano comprensibili, riducendo ambiguità e incomprensioni che potrebbero causare errori o ritardi. Interpretazione dei valori: <ul style="list-style-type: none"> • $IG \geq 80$: testo molto facile, comprensibile da studenti di scuola elementare; • $60 \leq IG < 80$: testo facile, comprensibile da studenti di scuola media; • $40 \leq IG < 60$: testo abbastanza difficile, comprensibile da studenti di scuola superiore; • $IG < 40$: testo difficile, richiede istruzione universitaria.
--------------------	--

5.2.1.2 MPC12 - Densità Errori Ortografici (DEO)

Codice	MPC12
Formula	$DEO = \frac{N_{errori}}{N_{parole}} \cdot 1000$ <p>Dove:</p> <ul style="list-style-type: none"> • N_{errori} = Numero totale di errori ortografici rilevati nel documento; • N_{parole} = Numero totale di parole nel documento.
Descrizione	Rappresenta il numero di errori ortografici presenti ogni 1000 parole di documentazione prodotta. Questa metrica è fondamentale per garantire la professionalità della documentazione. Interpretazione dei valori: <ul style="list-style-type: none"> • $DEO = 0$: il testo non presenta alcun errore; • $0 < DEO \leq 5$: il testo presenta pochi errori; • $5 < DEO \leq 10$: il testo presenta alcuni errori, è necessario revisionare il documento; • $DEO > 10$: il testo presenta troppi errori, è necessario correggere il documento.

5.2.1.3 MPC13 - Indice di Frammentazione (IF)

Codice	MPC13
Formula	$IF = \frac{N_{sezioni} + N_{paragrafi}}{N_{linee}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $N_{sezioni}$ = Numero di sezioni e sottosezioni del documento; • $N_{paragrafi}$ = Numero di paragrafi del documento; • N_{linee} = Numero totale di linee di testo del documento.

Descrizione	Misura il grado di suddivisione del testo in unità logiche (sezioni e paragrafi). Un valore bilanciato indica che il documento non presenta "muri di testo" eccessivi, facilitando la lettura veloce e la memorizzazione dei concetti. Interpretazione dei valori: <ul style="list-style-type: none"> • IF < 5%: il testo è molto denso e difficile da leggere; • 5% ≤ IF ≤ 20%: bilanciamento ottimale tra contenuto e suddivisione spaziale; • IF > 20%: il testo è eccessivamente frammentato, rischiando di perdere coesione.
--------------------	--

5.2.2 Verifica

5.2.2.1 MPC14 - Test Success Rate (TSR)

Codice	MPC14
Formula	$TSR = \frac{T_{successo}}{T_{totali}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $T_{successo}$ = Numero di test eseguiti con esito positivo; • T_{totali} = Numero di test eseguiti.
Descrizione	Percentuale di casi di test che sono stati eseguiti con esito positivo rispetto al numero totale di casi di test eseguiti. Questa metrica fornisce una misura della correttezza del software e dell'efficacia dell'attività di <i>testing</i> . Interpretazione dei valori: <ul style="list-style-type: none"> • TSR = 100%: qualità eccellente, tutti i test sono stati superati; • 80% ≤ TSR < 100%: qualità accettabile; • 70% ≤ TSR < 80%: qualità discreta, sono necessarie correzioni; • TSR < 70%: qualità inaccettabile, necessaria revisione approfondita del codice.

5.2.3 Gestione della qualità

5.2.3.1 MPC15 - Percentuale Metriche Soddisfatte (PMS)

Codice	MPC15
Formula	$PMS = \frac{M_{soddisfatte}}{M_{totali}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $M_{soddisfatte}$ = Numero di metriche soddisfatte; • M_{totali} = Numero di metriche definite.

Descrizione	Percentuale di metriche di qualità soddisfatte rispetto al numero di metriche totali. Interpretazione dei valori:
	<ul style="list-style-type: none"> • PMS $\geq 90\%$: eccellente, il progetto rispetta gli standard di qualità; • $80\% \leq \text{PMS} < 90\%$: buono, ma alcune aree richiedono miglioramento; • $70\% \leq \text{PMS} < 80\%$: sufficiente, necessarie azioni correttive; • $\text{PMS} < 70\%$: insoddisfacente, il progetto presenta problemi significativi di qualità.

5.3 Processi organizzativi

5.3.1 Gestione dei processi

5.3.1.1 MPC16 - Percentuale Rischi Inattesi (PRI)

Codice	MPC16
Formula	$PRI = \frac{R_{inattesi}}{R_{totali}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $R_{inattesi}$ = Numero di rischi non individuati che si sono verificati; • R_{totali} = Numero di rischi totali che si sono verificati;
Descrizione	Percentuale di rischi non individuati in fase di analisi iniziale che si sono manifestati nel corso del progetto. Questa metrica fornisce una misura del livello di accuratezza dell'attività di analisi e gestione dei rischi. Interpretazione dei valori: <ul style="list-style-type: none"> • PRI = 0: eccellente; • $PRI \leq 5\%$: buono, alcuni rischi non previsti sono normali; • $5\% < PRI \leq 20\%$: accettabile ma migliorabile; • $20\% < PRI \leq 40\%$: insufficiente, è necessario migliorare l'analisi dei rischi; • $PRI > 40\%$: gestione inadeguata, l'attività di gestione dei rischi è inefficace.

5.3.1.2 MPC17 - Labor Efficiency (LE)

Codice	MPC17
Formula	$LE_k = \frac{H_{pianificate}}{H_{effettive}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $H_{pianificate}$ = Somma delle ore stimate per le attività completate nello <i>sprint k</i>; • $H_{effettive}$ = Somma delle ore rendicontate dai membri del gruppo per lo <i>sprint k</i>; • k = Numero dello sprint di riferimento.

Descrizione	Rappresenta il rapporto tra le ore di lavoro pianificate per lo svolgimento di determinate attività e le ore effettivamente impiegate per il loro completamento nello <i>sprint</i> di riferimento. Mentre lo SPI (si veda §5.1.1.5) valuta l'avanzamento in termini di valore economico, questa metrica permette di monitorare direttamente la precisione delle stime temporali effettuate dal team durante la pianificazione. Interpretazione dei valori: <ul style="list-style-type: none"> • LE > 100%: il gruppo è stato più veloce del previsto; • LE < 100%: il team ha impiegato più tempo del previsto; • LE = 100%: le stime sono perfettamente in linea con la capacità produttiva.
--------------------	---

6 Metriche di qualità di prodotto

In questa sezione vengono definite le metriche di qualità adottate per garantire che i prodotti di progetto (documentazione e *software*) soddisfino i requisiti qualitativi stabiliti.

Ogni metrica è identificata da un codice univoco della forma *MPDX*, dove *X* rappresenta un numero intero progressivo.

6.1 Funzionalità

6.1.0.1 MPD1 - Percentuale Requisiti Obbligatori Soddisfatti (PROS)

Codice	MPD1
Formula	$PROS = \frac{RO_{soddisfatti}}{RO_{totali}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $RO_{soddisfatti}$ = Requisiti obbligatori soddisfatti; • RO_{totali} = Requisiti obbligatori totali.
Descrizione	Percentuale di requisiti obbligatori soddisfatti rispetto al numero totale di requisiti obbligatori definiti. Interpretazione dei valori: <ul style="list-style-type: none"> • PROS = 100%: tutti i requisiti obbligatori sono stati implementati, il prodotto è completo; • $90\% \leq PROS < 100\%$: il prodotto è quasi completo, mancano poche funzionalità; • $70\% \leq PROS < 90\%$: il prodotto è parzialmente funzionante; • $PROS < 70\%$: il prodotto è incompleto, non rilasciabile.

6.1.0.2 MPD2 - Percentuale Requisiti Opzionali Soddisfatti (PRPS)

Codice	MPD2
---------------	------

Formula	$PRPS = \frac{RP_{soddisfatti}}{RP_{totali}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $RP_{soddisfatti}$ = Requisiti opzionali soddisfatti; • RP_{totali} = Requisiti opzionali totali.
Descrizione	Percentuale di requisiti opzionali soddisfatti rispetto al numero totale di requisiti opzionali definiti. L'implementazione di requisiti opzionali aumenta il valore del prodotto ma non è essenziale per il suo funzionamento base.

6.1.0.3 MPD3 - Percentuale Requisiti Desiderabili Soddisfatti (PRDS)

Codice	MPD3
Formula	$PRDS = \frac{RD_{soddisfatti}}{RD_{totali}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $RD_{soddisfatti}$ = Requisiti desiderabili soddisfatti; • RD_{totali} = Requisiti desiderabili totali.
Descrizione	Percentuale di requisiti desiderabili soddisfatti rispetto al numero totale di requisiti desiderabili definiti. I requisiti desiderabili migliorano significativamente l'esperienza utente e la competitività del prodotto.

6.2 Affidabilità

6.2.0.1 MPD4 - Line Coverage (LC)

Codice	MPD4
Formula	$LC = \frac{L_{eseguite}}{L_{totali}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $L_{eseguite}$ = Numero di linee di codice eseguite dai test; • L_{totali} = Numero di linee di codice totali.
Descrizione	Indica la percentuale di linee di codice eseguite dai test automatici rispetto alle linee di codice totali. Questa metrica fornisce una misura delle linee di codice coperte dai casi di test. Interpretazione dei valori: <ul style="list-style-type: none"> • $LC \geq 80\%$: copertura eccellente, codice ben testato; • $60\% \leq LC < 80\%$: copertura buona, ma migliorabile; • $40\% \leq LC < 60\%$: copertura insufficiente, è necessario aumentare i test; • $LC < 40\%$: copertura critica, il codice è poco testato.

6.2.0.2 MPD5 - Branch Coverage (BC)

Codice	MPD5
Formula	$BC = \frac{B_{eseguite}}{B_{totali}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $B_{eseguite}$ = Numero di rami (<i>branch</i>) decisionali eseguiti dai test; • B_{totali} = Numero di rami decisionali totali.
Descrizione	Indica la percentuale di rami decisionali del codice eseguiti dai test. Interpretazione dei valori: <ul style="list-style-type: none"> • $BC \geq 75\%$: copertura eccellente dei percorsi decisionali; • $60\% \leq BC < 75\%$: copertura buona; • $40\% \leq BC < 60\%$: copertura insufficiente; • $BC < 40\%$: molti percorsi decisionali non testati, rischio elevato di errori.

6.2.0.3 MPD6 - Statement Coverage (SC)

Codice	MPD6
Formula	$SC = \frac{S_{eseguite}}{S_{totali}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $S_{eseguite}$ = Numero di istruzioni (<i>statement</i>) eseguite dai test; • S_{totali} = Numero di istruzioni totali.
Descrizione	Indica la percentuale di istruzioni nel codice eseguite dai test. Interpretazione dei valori: <ul style="list-style-type: none"> • $SC \geq 80\%$: copertura eccellente delle istruzioni; • $60\% \leq SC < 80\%$: copertura buona; • $40\% \leq SC < 60\%$: copertura insufficiente; • $SC < 40\%$: molte istruzioni non testate.

6.3 Usabilità

6.3.0.1 MPD7 - Facilità di Apprendimento (FA)

Codice	MPD7
---------------	------

Formula	$FA = \frac{\sum_{i=1}^k T_i}{k}$ <p>Dove:</p> <ul style="list-style-type: none"> • T_i = Tempo impiegato dall'utente i-esimo; • k = Numero di utenti testati;
Descrizione	Tempo medio necessario ad un utente per completare un'operazione specifica, senza supporto esterno.

6.4 Efficienza

6.4.0.1 MPD8 - Tempo Medio di Risposta (TMR)

Codice	MPD8
Formula	$TMR = \frac{\sum_{i=1}^k T_i}{k}$ <p>Dove:</p> <ul style="list-style-type: none"> • T_i = Tempo di risposta per la richiesta i-esima; • k = Numero di richieste testate;
Descrizione	Tempo medio che intercorre tra l'invio di una richiesta al sistema e la ricezione della risposta completa. Interpretazione dei valori: <ul style="list-style-type: none"> • $TMR \leq 1$ secondo: eccellente, risposta immediata; • $1 < TMR \leq 3$ secondi: buono, risposta rapida; • $3 < TMR \leq 5$ secondi: accettabile, ma migliorabile; • $TMR > 5$ secondi: inaccettabile, l'utente percepisce lentezza.

6.5 Manutenibilità

6.5.0.1 MPD9 - Densità dei Commenti (DC)

Codice	MPD9
Formula	$DC = \frac{L_{commenti}}{L_{totali}} \cdot 100$ <p>Dove:</p> <ul style="list-style-type: none"> • $L_{commenti}$ = Linee dedicate ai commenti; • L_{totali} = Linee totali nel codice.

Descrizione	Rapporto tra le righe dedicate ai commenti e le righe di codice totali. Una densità adeguata favorisce maggior manutenibilità. Interpretazione dei valori: <ul style="list-style-type: none"> • $DC < 10\%$: il codice è scarsamente documentato, rendendo difficile l'analisi e la manutenzione futura; • $10\% \leq DC \leq 30\%$: densità ottimale per un codice ben scritto e auto-esplicativo; • $DC > 30\%$: il codice potrebbe essere eccessivamente commentato. Valori molto alti possono indicare la presenza di una logica eccessivamente complessa che richiede troppe spiegazioni.
--------------------	--

6.5.0.2 MPD10 - Complessità Ciclomatica (CC)

Codice	MPD10
Formula	$CC = E - N + 2P$ <p>Dove:</p> <ul style="list-style-type: none"> • E = Numero di archi del grafo di controllo di flusso; • N = Numero di nodi del grafo di controllo di flusso; • P = Numero di componenti connesse.
Descrizione	Misura la complessità strutturale del codice calcolando il numero di percorsi linearmente indipendenti attraverso il codice sorgente. Viene calcolata per ciascuna funzione o metodo del sistema. Una bassa complessità indica codice più leggibile, testabile e manutenibile. Interpretazione dei valori: <ul style="list-style-type: none"> • $CC \leq 10$: complessità bassa, codice semplice e manutenibile; • $10 < CC \leq 20$: complessità moderata, ancora accettabile; • $20 < CC \leq 30$: complessità elevata; • $CC > 30$: complessità critica, il codice è difficile da testare e mantenere.