Piano di Qualifica

NearYou
Smart custom advertising platform

sevenbits.swe.unipd@gmail.com

Registro modifiche

Versione	Data	Autore	Verificatore	Descrizione
1.0.5	2025-03-28	Leonardo Trolese	Riccardo Piva	Ampliamento del cruscotto con le metriche relative alla revisione PB
1.0.4	2025-03-26	Federico Pivetta	Riccardo Piva	Ampliamento del cruscotto con le metriche relative alla revisione PB
1.0.3	2025-03-26	Alfredo Rubino	Riccardo Piva	Redazione sottosezione Test di Unità
1.0.2	2025-03-25	Leonardo Trolese	Riccardo Piva	Aggiornamento metriche a seguito di inizio stesura codice
1.0.1	2025-03-21	Federico Pivetta	Uncas Peruzzi	Correzione ai test di sistema
1.0.0	2025-02-21	Giovanni Cristellon	Leonardo Trolese	Approvazione documento per RTB
0.4.8	2025-02-21	Leonardo Trolese	Manuel Gusella	Correzione errori minori
0.4.7	2025-02-18	Leonardo Trolese	Pivetta Federico	Aggiunta ulteriori grafici sulle metriche del cruscotto di valutazione
0.4.6	2025-02-16	Leonardo Trolese	Pivetta Federico	Aggiunta ulteriori grafici sulle metriche del cruscotto di valutazione
0.4.5	2025-02-15	Leonardo Trolese	Pivetta Federico	Aggiunta grafici sulle metriche del cruscotto di valutazione
0.4.4	2025-02-11	Federico Pivetta	Uncas Peruzzi	Correzione ai test di sistema e di accettazione a seguito delle modifiche all'Analisi dei Requisiti
0.4.3	2025-01-14	Federico Pivetta	Leonardo Trolese	Riorganizzazione delle metriche di qualità
0.4.2	2025-01-10	Federico Pivetta	Leonardo Trolese	Aggiunta dei test di sistema e dei test di accettazione
0.4.1	2025-01-08	Riccardo Piva	Uncas Peruzzi	Refactor generale sezione qualità processo e qualità prodotto
0.4.0	2025-01-07	Riccardo Piva	Uncas Peruzzi	Creazione cruscotto
0.3.3	2025-01-03	Riccardo Piva	Uncas Peruzzi	Correzioni minori generali
0.3.2	2024-12-16	Alfredo Rubino	Manuel Gusella	Aggiunta acronimi metriche e correzioni minori
0.3.1	2024-12-13	Riccardo Piva	Alfredo Rubino	Correzione standard IEEE
0.3.0	2024-12-12	Riccardo Piva	Alfredo Rubino	Arricchimento sezioni Qualità di processo, Qualità di prodotto e inizio redazione modalità testing

Versione	Data	Autore	Verificatore	Descrizione
0.2.0	2024-12-06	Manuel Gusella	Alfredo Rubino	Inizio redazione sottosezione Qualità di prodotto
0.1.0	2024-11-21	Uncas Peruzzi	Federico Pivetta	Inizio redazione del documento

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Il seguente documento ha l'obiettivo di garantire la qualità del prodotto e dei processi coinvolti nell'intero progetto. Al fine di assicurare che il prodotto soddisfi le aspettative di qualità attese, il documento verrà aggiornato nel tempo per riflettere eventuali modifiche, integrazioni e i risultati delle verifiche effettuate.

1.2 Glossario

Con l'intento di evitare ambiguità nell'interpretazione del linguaggio utilizzato, viene fornito un glossario che si occupa di esplicitare il significato dei termini che riguardano il contesto del progetto. I termini presenti nel glossario sono contrassegnati con una $_G$ a pedice : Termine$_G$.
Le definizioni sono contenute nell'apposito documento *Glossario*.

1.3 Riferimenti

1.3.1 Riferimenti normativi

- *Norme_di_Progetto_v1.0.0*

- Regolamento del progetto didattico
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/PD1.pdf
  (Consultato: 2025-02-19).

- Capitolato C4 - NearYou - Smart custom advertising platform
  https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C4p.pdf
  (Consultato: 2025-02-19).
  https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C4.pdf
  (Consultato: 2025-02-19).

- Standard ISO/IEC 9126
  https://en.wikipedia.org/wiki/ISO/IEC_9126
  (Consultato: 2025-02-19).

- Standard ISO/IEC/IEEE 12207:1995
  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
  (Consultato: 2025-02-19).

1.3.2 Riferimenti informativi

- Qualità di prodotto
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T07.pdf
  (Consultato: 2025-02-19).

- Qualità di processo
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T08.pdf
  (Consultato: 2025-02-19).

- Verifica e validazione
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T09.pdf
  (Consultato: 2025-02-19).
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T10.pdf
  (Consultato: 2025-02-19).
  https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T11.pdf
  (Consultato: 2025-02-19).

2 Obiettivi metrici di qualità

Per far sì che un prodotto raggiunga uno standard qualitativo, è necessario definire delle metriche precise che permettano di monitorare e indicare il grado di qualità del prodotto e che quindi permettano di definire questo standard. Queste metriche vengono definite nel documento *Norme di Progetto*. Questa sezione si occuperà di definire i parametri di accettazione e ottimalità delle relative metriche.

2.1 Qualità di prodotto

La qualità di prodotto è intesa come valutazione del software, e più precisamente per la determinazione del grado di conformità alle attese.
Si rivolge l'attenzione su aspetti come Usabilità, Affidabilità e Manutenibilità, ma più in generale alla qualità esterna (funzionale) ed interna (strutturale) del prodotto software.
Quindi non basta che il software implementi le funzionalità richieste dal proponente, ma le esegua secondo specifici standard di qualità.
In seguito sono presenti le metriche definite dallo standard ISO/IEC 9126 che il gruppo si impegna a soddisfare per la qualità del prodotto software.

2.1.1 Funzionalità

Metrica	Descrizione	Valore accettazione	Valore ideale
MPD01	Requisiti Obbligatori Soddisfatti	100%	100%
MPD02	Requisiti Desiderabili Soddisfatti	$\geq$ 0%	100%
MPD03	Requisiti Opzionali Soddisfatti	$\geq$ 0%	100%

Table 2: Funzionalità - Qualità di prodotto

2.1.2 Affidabilità

Metrica	Descrizione	Valore accettazione	Valore ideale
MPD04	Line coverage	$\geq$ 80%	100%
MPD05	Branch coverage	$\geq$ 80%	100%

Table 3: Affidabilità - Qualità di prodotto

2.1.3 Efficienza

Metrica	Descrizione	Valore accettazione	Valore ideale
MPD06	Tempo medio di risposta	$\leq$ 10 secondi	$\leq$ 4 secondi

Table 4: Efficienza - Qualità di prodotto

2.1.4 Usabilità

Metrica	Descrizione	Valore accettazione	Valore ideale
MPD07	Facilità di utilizzo	$\leq$ 7 click	$\leq$ 5 click
MPD08	Tempo medio di apprendimento	$\leq$ 5 minuti	$\leq$ 2 minuti

Table 5: Usabilità - Qualità di prodotto

2.1.5 Manutenibilità

Metrica	Descrizione	Valore accettazione	Valore ideale
MPD09	Linee di codice per metodo	≤ 50	≤ 25
MPD10	Attributi per classe	≤ 7	≤ 5
MPD11	Structure Fan IN	-	-
MPD12	Structure Fan OUT	-	va minimizzato

Table 6: Manutenibilità - Qualità di prodotto

2.1.6 Portabilità

Metrica	Descrizione	Valore accettazione	Valore ideale
MPD14	Versioni browser supportati	$\geq 80\%$	100%

Table 7: Portabilità - Qualità di prodotto

2.2 Qualità di processo

La qualità di processo è intesa come valutazione delle attività svolte per la realizzazione del prodotto. Seguendo delle buone pratiche e delle linee guida nello sviluppo software, si può garantire che il prodotto finale avrà rispettato a sua volta degli standard qualitativi rendendolo così un prodotto di qualità.
Qui sotto divideremo le metriche di qualità di processo seguendo lo standard ISO/IEC 12207:1995 in tre categorie: Processi primari, Processi di supporto e Processi organizzativi.

2.2.1 Processi Primari

I processi primari si possono dividere in parti primarie e una parte primaria è quella che inizia o esegue lo sviluppo, l'operazione o la manutenzione di prodotti software.

2.2.1.1 Fornitura

La fornitura è il processo che si occupa di consegnare il prodotto software al cliente.
Serve per garantire che il prodotto soddisfi i requisiti di tempi e costi definiti con il cliente.

Metrica	Descrizione	Valore accettazione	Valore ideale
MPC01	Estimated at completion (EAC)	$\pm 5\%$ rispetto al $(\text{BAC})_G$	Budget at completion $(\text{BAC})_G$
MPC02	Estimate to complete (ETC)	≥ 0	$\leq \text{EAC}_G$
MPC03	Actual cost (AC)	≥ 0	$\leq \text{EAC}_G$
MPC04	Earned value (EV)	≥ 0	$\leq \text{EAC}_G$
MPC05	Planned value (PV)	≥ 0	$\leq$ Budget at completion $(\text{BAC})_G$
MPC06	Schedule variance (SV)	$\geq$ -5% rispetto al $(\text{BAC})_G$	$\geq 0\%$
MPC07	Cost variance (CV)	$\geq$ -5% rispetto al $(\text{BAC})_G$	$\geq 0\%$
MPC08	Cost Performance Index (CPI)	≥ 0.9	≥ 1.0

Table 8: Processi primari - Fornitura

2.2.1.2 Sviluppo

Lo sviluppo è il processo riguardante la scrittura del codice del prodotto software.
Questa metrica serve a garantire che il software rispetti le richieste del cliente e che la codifica avvenga in modo efficiente

Metrica	Descrizione	Valore accettazione	Valore ideale
MPC09	Requisiti Obbligatori Soddisfatti (ROS)	100%	100%
MPC10	Requirements Stability Index (RSI)	$\geq$ 80%	100%

Table 9: Processi primari - Codifica

2.2.2 Processi di Supporto

Un processo di supporto è un processo che supporta un altro processo come parte integrante con uno scopo distinto e contribuisce al successo e alla qualità del progetto software.
Un processo di supporto è impiegato ed eseguito, se necessario, da un altro processo.

2.2.2.1 Documentazione

La documentazione è essenziale per la comprensione del prodotto e per la sua manutenzione.
Di conseguenza è essenziale che questa sia chiara, comprensibile e corretta.

Metrica	Descrizione	Valore accettazione	Valore ideale
MPC11	Indice Gulpease	$\geq$ 40%	$\geq$ 60%
MPC12	Correttezza ortografica	0 errori	0 errori

Table 10: Processi di supporto - Documentazione

2.2.2.2 Verifica

La verifica serve a garantire che il prodotto software sia conforme alle specifiche e non contenga errori.

Metrica	Descrizione	Valore accettazione	Valore ideale
MPC13	Line coverage	$\geq$ 80%	100%
MPC14	Passed test cases percentage	$\geq$ 80%	100%

Table 11: Processi di supporto - Verifica

2.2.2.3 Gestione della qualità

La gestione della qualità è necessaria per garantire che tutte le metriche di qualità vengano effettivamente soddisfatte.

Metrica	Descrizione	Valore accettazione	Valore ideale
MPC15	Metriche di qualità soddisfate	$\geq$ 85%	100%

Table 12: Processi di supporto - Gestione della qualità

2.2.3 Processi Organizzativi

I processi organizzativi servono per creare un sottostruttura per il ciclo di vita e per garantire che i processi principali e i loro processi di supporto siano ben strutturati e vengano continuamente migliorati.

2.2.3.1 Gestione dei processi

La gestione dei processi indica come vengono gestiti i processi all'interno del progetto.

Metrica	**Descrizione**	**Valore accettazione**	**Valore ideale**
MPC16	Rischi non previsti	≤ 3	0
MPC17	Efficienza temporale (ET)	≤ 5.0	≥ 1.0

Table 13: Processi organizzativi - Gestione dei processi

3 Modalità di Testing

Qui sotto sono elencati i vari test che vengono eseguiti automaticamente sul prodotto software.
Questo serve a garantire che il prodotto soddisfi i requisiti e le aspettative indicate nel documento *Analisi dei Requisiti*.
I test sono divisi in quattro categorie: Test di unità, Test di sistema, Test di integrazione e Test di accettazione.
Per indicarne lo stato, come definito nel documento *Norme di Progetto* vengono utilizzate le seguenti abbreviazioni:

- **P**: Passato
- **NP**: Non Passato
- **NI**: Non Implementato

3.1 Test di unità

I test di unità servono a verificare che ogni singola unità del software funzioni correttamente.

Codice	Descrizione	Stato
TU1	Verificare che la classe `ClickhouseActivityRepository` restituisca correttamente le attività entro un determinato raggio geografico e i dettagli di una specifica attività a partire dal nome. Se l'attività non viene trovata, deve restituire un oggetto vuoto.	P
TU2	Verificare che la classe `ClickhouseMessageRepository` restituisca correttamente l'ultimo messaggio inviato da un utente, includendo tutte le informazioni associate. In caso di assenza di messaggi, deve restituire un oggetto vuoto con valori di default.	P
TU3	Verificare che la classe `ClickhouseUserRepository` gestisca correttamente l'assegnazione e il recupero delle informazioni sugli utenti. In particolare, deve aggiornare lo stato di un utente come occupato, recuperare un utente libero, restituire i dettagli di un utente associato a un sensore e gestire correttamente i casi in cui nessun utente venga trovato.	P
TU4	Verificare che la classe `CustomPrompt` generi correttamente un prompt formattato in base alle informazioni dell'utente e alle attività disponibili. Deve garantire che i dati forniti siano integrati correttamente nel testo generato e che venga mantenuta una struttura valida anche in caso di dati mancanti.	P

Codice	Descrizione	Stato
TU5	Verificare che la classe `DatabaseConnection` gestisca correttamente la connessione e la disconnessione al database. Inoltre, si verifica che la classe `DatabaseConfigParameters` inizializzi correttamente i valori di default dei parametri di configurazione.	P
TU6	Verificare che le classi DTO (`MessageDTO`, `UserDTO`, `ActivityDTO`) inizializzino correttamente i dati forniti e gestiscano i valori di default quando i parametri non vengono specificati.	P
TU7	Verificare che la classe `FilterMessageAlreadyDisplayed` filtri correttamente i messaggi già visualizzati dall'utente, escludendo i messaggi con le stesse coordinate dell'ultimo messaggio inviato, i messaggi con coordinate (0,0) e quelli che risultano identici dopo l'arrotondamento delle coordinate.	P
TU8	Verificare che la classe `FlinkJobManager` inizializzi e gestisca correttamente il flusso di dati in Apache Flink. Deve configurare correttamente le operazioni di `map` e `filter`, collegarsi a sorgenti e sink e avviare l'esecuzione del job tramite `execute`.	P
TU9	Verificare che la classe `GroqLLMService` gestisca correttamente l'inizializzazione, la configurazione della chat e la generazione di risposte strutturate. In particolare, deve caricare la chiave API dall'ambiente, impostare correttamente il modello `ChatGroq` con i parametri previsti e restituire un output coerente con la richiesta ricevuta.	P
TU10	Verificare che la classe `JsonRowDeserializationAdapter` inizializzi correttamente lo schema di deserializzazione JSON, memorizzando correttamente il tipo di dati fornito e creando un'istanza di `JsonRowDeserializationSchema` con i parametri appropriati.	P
TU11	Verificare che la classe `JsonRowSerializationAdapter` inizializzi correttamente lo schema di serializzazione JSON. In particolare, deve memorizzare il tipo di dati fornito e creare un'istanza di `JsonRowSerializationSchema` configurata con i parametri appropriati.	P

Codice	Descrizione	Stato
TU12	Verificare che la classe `KafkaMessageWriter` inizializzi correttamente il writer Kafka. In particolare, deve configurare lo schema di serializzazione, impostare i parametri del sink Kafka e restituire correttamente l'istanza del writer tramite il metodo `get_message_writer`.	P
TU13	Verificare che la classe `KafkaPositionReceiver` configuri correttamente la sorgente Kafka. In particolare, deve impostare i parametri di connessione, il gruppo di consumo, il deserializzatore e le proprietà di gestione degli offset, restituendo correttamente l'istanza della sorgente tramite il metodo `get_position_receiver`.	P
TU14	Verificare che la classe `KafkaSourceConfiguration` inizializzi correttamente i parametri di configurazione con i valori di default previsti, inclusi server, topic, gruppo di consumo e gestione degli offset.	P
TU15	Verificare che la classe `KafkaWriterConfiguration` inizializzi correttamente i parametri di configurazione con i valori di default previsti. In particolare, deve garantire che il server, il topic scrivibile e la struttura del `key_type` siano definiti correttamente.	P
TU16	Verificare che la classe `MessageSerializer` crei correttamente un oggetto `Row` a partire da un'istanza di `MessageDTO`. In particolare, deve garantire che i dati siano convertiti correttamente e mantenere la struttura prevista con il numero corretto di campi.	P
TU17	Verificare che la classe `PositionToMessageProcessor` elabori correttamente le posizioni degli utenti e generi messaggi pubblicitari basati sulle attività nelle vicinanze. In particolare, deve gestire i casi in cui non ci siano attività disponibili e interagire con i repository e il servizio AI per elaborare i dati.	P
TU18	Verificare che la classe `BycicleSimulationStrategy` inizializzi correttamente i parametri di simulazione e calcoli correttamente il percorso, la velocità e il tempo tra le posizioni. In particolare, deve interagire con il grafo per ottenere un percorso valido e restituire valori coerenti per velocità e intervallo di tempo.	P

Codice	Descrizione	Stato
TU19	Verificare che la classe `DatabaseConfigParameters` inizializzi correttamente i parametri di configurazione con i valori di default previsti.	P
TU20	Verificare che la classe `DatabaseConnection` gestisca correttamente la connessione e la disconnessione dal database.	P
TU21	Verificare che la classe `GeoPosition` inizializzi correttamente i parametri e restituisca i valori attesi tramite i metodi di accesso. In particolare, deve garantire che `sensor_id`, latitudine, longitudine e timestamp siano correttamente memorizzati e recuperabili.	P
TU22	Verificare che la classe `GpsSensor` gestisca correttamente la simulazione della posizione. In particolare, deve inizializzare correttamente i parametri, generare e inviare posizioni simulate basate sulla strategia di simulazione e restituire i valori attesi per identificativo del sensore e tempo di aggiornamento.	P
TU23	Verificare che la classe `GraphWrapper` inizializzi e recuperi correttamente la mappa stradale utilizzando `osmnx`. In particolare, deve chiamare `graph_from_point` con i parametri corretti per ottenere il grafo stradale basato sulla posizione, il raggio e il tipo di rete specificati.	P
TU24	Verificare che la classe `KafkaConfigParameters` inizializzi correttamente i parametri di configurazione con i valori di default previsti, inclusi il server Kafka e il topic di origine.	P
TU25	Verificare che la classe `KafkaConfluentAdapter` invii correttamente i dati al broker Kafka. In particolare, deve serializzare correttamente le posizioni, produrre i messaggi con la chiave corretta e gestire la sincronizzazione dei thread tramite un meccanismo di lock.	P
TU26	Verificare che la classe `PositionJsonAdapter` serializzi correttamente un oggetto `GeoPosition` in formato JSON, mantenendo la struttura e i valori attesi per identificativo, coordinate e timestamp.	P

Codice	Descrizione	Stato
TU27	Verificare che la classe `SensorFactory` crei correttamente istanze di `GpsSensor`. In particolare, deve garantire che ogni sensore sia un'istanza valida di `SensorSubject` e `GpsSensor`, che gli UUID assegnati siano validi e univoci, e che il metodo per la creazione di una lista di sensori generi il numero corretto di istanze.	P
TU28	Verificare che la classe `SensorRepository` gestisca correttamente l'assegnazione e il recupero dei sensori. In particolare, deve aggiornare lo stato di un sensore come occupato e recuperare un sensore non occupato dal database.	P
TU29	Verificare che la classe `SensorSimulationAdministrator` gestisca correttamente l'avvio della simulazione dei sensori. In particolare, deve inizializzare correttamente il registro dei sensori, avviare la simulazione tramite un `ThreadPool` e gestire eventuali errori durante l'esecuzione.	P
TU30	Verificare che la classe `UserRepository` gestisca correttamente l'assegnazione e il recupero degli utenti. In particolare, deve aggiornare lo stato di un utente come occupato e recuperare un utente libero dal database.	P
TU31	Verificare che la classe `UserSensorService` assegni correttamente un sensore a un utente disponibile. In particolare, deve recuperare un utente e un sensore non occupati, aggiornare il loro stato nel database e gestire correttamente i casi in cui non siano disponibili utenti o sensori.	P

Table 14: Test di unità

3.2 Test di sistema

I test di sistema servono a verificare la completa copertura dei requisiti concordati nel documento *Analisi dei Requisiti*.

Codice	Descrizione	Requisito	Stato
TS1	Verificare che l'utente privilegiato possa visualizzare la Dashboard$_G$ composta da una mappa interattiva con i vari Marker$_G$ su di essa.	RF01	NI
TS2	Verificare che l'utente privilegiato possa visualizzare dei Marker$_G$ che rappresentano i vari Percorsi$_G$ effettuati in tempo reale dagli utenti presenti nel Sistema$_G$	RF02	NI

Codice	Descrizione	Requisito	Stato
TS3	Verificare che l'utente privilegiato possa visualizzare un Marker$_G$ che rappresenta un Percorso$_G$ effettuato in tempo reale da un utente presente nel Sistema$_G$	RF03	NI
TS4	Verificare che l'utente privilegiato possa visualizzare tutti i punti di interesse riconosciuti dal Sistema$_G$.	RF04	NI
TS5	Verificare che l'utente privilegiato possa visualizzare un Marker$_G$ che rappresenta un punto di interesse riconosciuto dal Sistema$_G$.	RF05	NI
TS6	Verificare che l'utente privilegiato possa visualizzare gli annunci pubblicitari provenienti da un determinato punto di interesse.	RF06	NI
TS7	Verificare che l'utente privilegiato possa visualizzare un singolo annuncio pubblicitario tramite un Marker$_G$.	RF07	NI
TS8	Verificare che l'utente privilegiato possa visualizzare una Dashboard$_G$ relativa ad un singolo utente quando seleziona un Marker$_G$ utente nella Dashboard$_G$ principale.	RF08	NI
TS9	Verificare che l'utente privilegiato possa visualizzare dei Marker$_G$ che rappresentano lo storico delle posizioni dell'utente a cui è riferita la Dashboard$_G$ di singolo utente.	RF09	NI
TS10	Verificare che l'utente privilegiato possa visualizzare un Marker$_G$ che rappresenta la posizione dell'utente in un determinato istante nella Dashboard$_G$ di singolo utente.	RF10	NI
TS11	Verificare che l'utente privilegiato possa visualizzare, nella Dashboard$_G$ di singolo utente, tutti i punti di interesse riconosciuti dal Sistema$_G$.	RF11	NI
TS12	Verificare che l'utente privilegiato possa visualizzare, nella Dashboard$_G$ di singolo utente, un Marker$_G$ che rappresenta un punto di interesse riconosciuto dal Sistema$_G$.	RF12	NI
TS13	Verificare che l'utente privilegiato possa visualizzare lo storico degli annunci pubblicitari generati per l'utente a cui è riferita la Dashboard$_G$ singolo utente.	RF13	NI
TS14	Verificare che l'utente privilegiato possa visualizzare un singolo annuncio pubblicitario tramite un Marker$_G$ nella Dashboard$_G$ di singolo utente.	RF14	NI
TS15	Verificare che l'utente privilegiato possa visualizzare un pannello apposito contenente le informazioni dell'utente, a cui è riferita la Dashboard$_G$ di singolo utente, in forma tabellare.	RF15	NI

Codice	Descrizione	Requisito	Stato
TS16	Verificare che l'utente privilegiato possa visualizzare nel pannello apposito di visualizzazione informazioni dell'utente: il nome, il cognome, l'email, il genere, la data di nascita e lo stato civile.	RF16	NI
TS17	Verificare che l'utente privilegiato possa visualizzare i dettagli del Marker$_G$ riguardante una singola posizione di un utente nella rispettiva Dashboard$_G$	RF17	NI
TS18	Verificare che l'utente privilegiato quando visualizza i dettagli del Marker$_G$, riguardante una singola posizione di un utente nella rispettiva Dashboard$_G$, possa vedere la latitudine, la longitudine e l'istante di rilevamento del Marker$_G$	RF18	NI
TS19	Verificare che l'utente privilegiato possa visualizzare l'area di influenza di un punto di interesse selezionato.	RF19	NI
TS20	Verificare che l'utente privilegiato possa visualizzare le informazioni dettagliate di un punto di interesse quando selezionato.	RF20	NI
TS21	Verificare che l'utente privilegiato quando visualizza le informazioni dettagliate di un punto di interesse possa visualizzare la latitudine, la longitudine, il nome, la tipologia e la descrizione del punto di interesse.	RF21	NI
TS22	Verificare che l'utente possa visualizzare l'annuncio pubblicitario proveniente dal punto di interesse situato nell'area che sta attraversando.	RF22	NI
TS23	Verificare che l'utente privilegiato possa visualizzare una tabella contenente le informazioni dei singoli PoI ordinati per la quantità di messaggi inviati nel mese.	RF23	NI
TS24	Verificare che l'utente privilegiato possa visualizzare nella tabella dei PoI un singolo PoI, rappresentato da una riga della tabella.	RF24	NI
TS25	Verificare che l'utente privilegiato possa visualizzare in ogni riga della tabella dei PoI il nome, l'indirizzo, la tipologia (di che ambito si occupa), la descrizione e il numero di messaggi inviati durante il mese di un singolo PoI.	RF25	NI
TS26	Verificare che l'utente privilegiato possa visualizzare i dettagli di un annuncio generato.	RF26	NI

Codice	Descrizione	Requisito	Stato
TS27	Verificare che l'utente privilegiato quando visualizza i dettagli di un annuncio possa visualizzare la latitudine, la longitudine, l'istante di creazione, il nome dell'utente coinvolto, il nome del punto di interesse coinvolto e il contenuto dell'annuncio.	RF27	NI
TS28	Verificare che il sensore possa trasmettere i dati rilevati in tempo reale al Sistema$_G$.	RF28	NI
TS29	Verificare che il sensore possa trasmettere il proprio id, la sua latitudine e longitudine al Sistema$_G$.	RF29	NI

Table 15: Test di sistema

3.3 Test di integrazione

I test di integrazione servono a verificare che le componenti del sistema si integrino correttamente e in maniera efficace. L'obiettivo dei test è identificare eventuali problemi di interoperabilità e integrazione fra le componenti del software.

Codice	Descrizione	Stato
TI1	Verificare che i dati simulati vengano correttamente pubblicati sul broker di messaggi	NI
TI2	Assicurarsi che il modulo di stream processing elabori correttamente i dati ricevuti dal broker e li invii al motore di generative AI	NI
TI3	Verificare che il motore LLM generi messaggi pubblicitari coerenti e contestualizzati in base ai dati forniti	NI
TI4	Controllare che i messaggi generati vengano correttamente salvati nella piattaforma di storage	NI
TI5	Controllare che i dati posizionali generati vengano correttamente salvati nella piattaforma di storage	NI
TI6	Verificare che i dati geospaziali vengano aggiornati in tempo reale sulla dashboard contenente la mappa	NI
TI7	Assicurarsi che i messaggi generati dal motore LLM siano correttamente visualizzati nella dashboard	NI

Table 16: Test di integrazione

3.4 Test di accettazione

I test di accettazione sono finalizzati a verificare che tutte le esigenze concordate con il proponente siano soddisfatte, e di conseguenza saranno svolti al termine del progetto dai membri del gruppo in coordi-

nazione con i componenti dell'azienda SyncLab.

Codice	Descrizione	Stato
TA1	Verificare che l'utente privilegiato possa visualizzare la Dashboard composta da una mappa e interagire con i Marker presenti su di essa.	NI
TA2	Verificare che l'utente privilegiato possa visualizzare sulla mappa tutti i Marker relativi ai Percorsi effettuati in tempo reale dagli utenti, ai punti di interesse riconosciuti dal Sistema e agli annunci pubblicitari associati ai punti di interesse.	NI
TA3	Verificare che l'utente privilegiato possa visualizzare una Dashboard relativa ad un singolo utente quando seleziona un Marker utente nella Dashboard principale	NI
TA4	Verificare che l'utente privilegiato possa visualizzare, nella Dashboard relativa ad un singolo utente, tutti i punti di interesse riconosciuti dal Sistema, lo storico delle posizioni dell'utente, lo storico degli annunci pubblicitari generati per l'utente, un pannello dedicato con le informazioni dell'utente in forma tabellare e i dettagli relativi a ciascuna posizione.	NI
TA5	Verificare che l'utente privilegiato possa visualizzare, nella Dashboard relativa a un singolo utente, i dettagli del pannello, inclusi il nome, il cognome, l'email, il genere, la data di nascita e lo stato civile. Inoltre, verificare che siano visibili i dettagli della singola posizione, inclusi la latitudine, la longitudine e l'istante di rilevamento.	NI
TA6	Verificare che l'utente privilegiato possa visualizzare i dettagli di un punto di interesse, inclusi la latitudine, la longitudine, il nome, la tipologia e la descrizione.	NI
TA7	Verificare che l'utente privilegiato possa visualizzare i dettagli di un annuncio generato, inclusi la latitudine, la longitudine, l'istante di creazione, il nome dell'utente coinvolto, il nome del punto di interesse coinvolto e il contenuto dell'annuncio.	NI
TA8	Verificare che l'utente privilegiato possa visualizzare una tabella contenente l'elenco dei punti di interesse, ordinati per la quantità di messaggi inviati nel mese. Ogni PoI deve essere rappresentato da una riga della tabella, contenente il nome, l'indirizzo, la tipologia, la descrizione e il numero di messaggi inviati.	NI

Codice	Descrizione	Stato
TA9	Verificare che il sensore possa trasmettere in tempo reale al Sistema$_G$ i dati rilevati, inclusi il proprio id, la latitudine e la longitudine.	NI

Table 17: Test di accettazione

4 Cruscotto di valutazione delle qualità

4.1 Qualità di processo - fornitura

4.1.1 MPC01 - Estimated at completion (EAC)

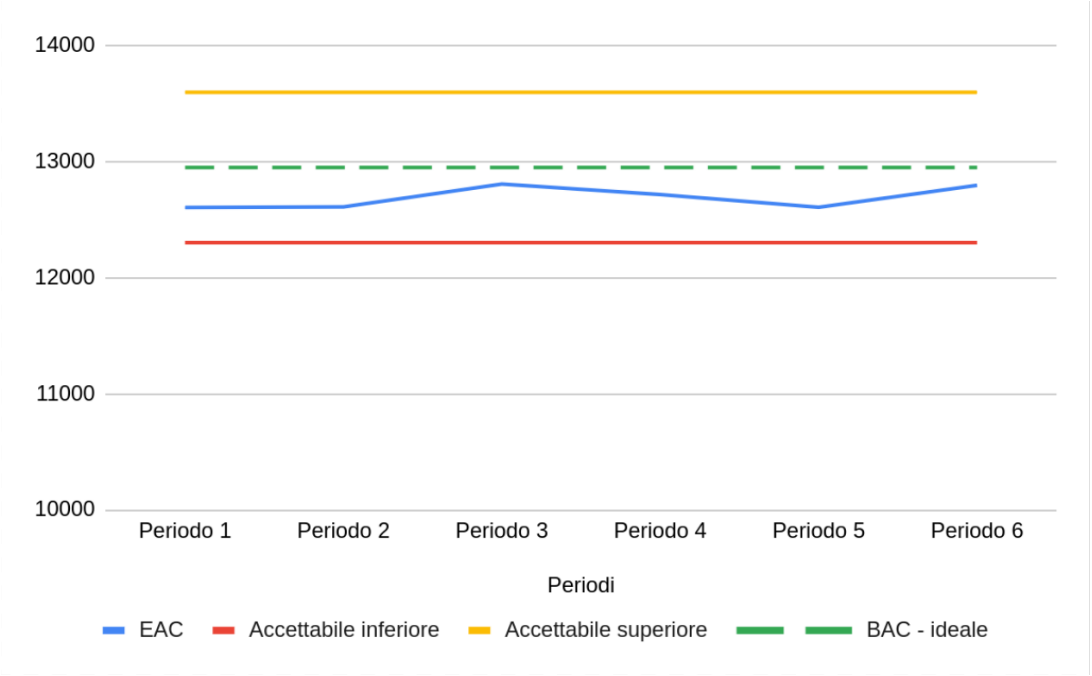

Figure 1: Grafico a linee della metrica EAC

RTB: Il grafico rappresenta la stima aggiornata del costo totale del progetto al completamento. Questo valore è quindi determinato dalla somma dei costi sostenuti fino a un certo momento (in termini di ore produttive svolte), e dei costi stimati al completamento (in termini di ore produttive restanti in riferimento al preventivo iniziale del progetto).
In questo caso emerge dal grafico che il valore di EAC è poco al di sotto del preventivo iniziale (BAC), e in ogni caso entro i valori accettabili definiti per la metrica. Questo indica che il progetto è sufficientemente in linea con le aspettative in termini di costi.
PB:

4.1.2 MPC05 - Planned Value (PV) & MPC04 - Earned Value (EV)

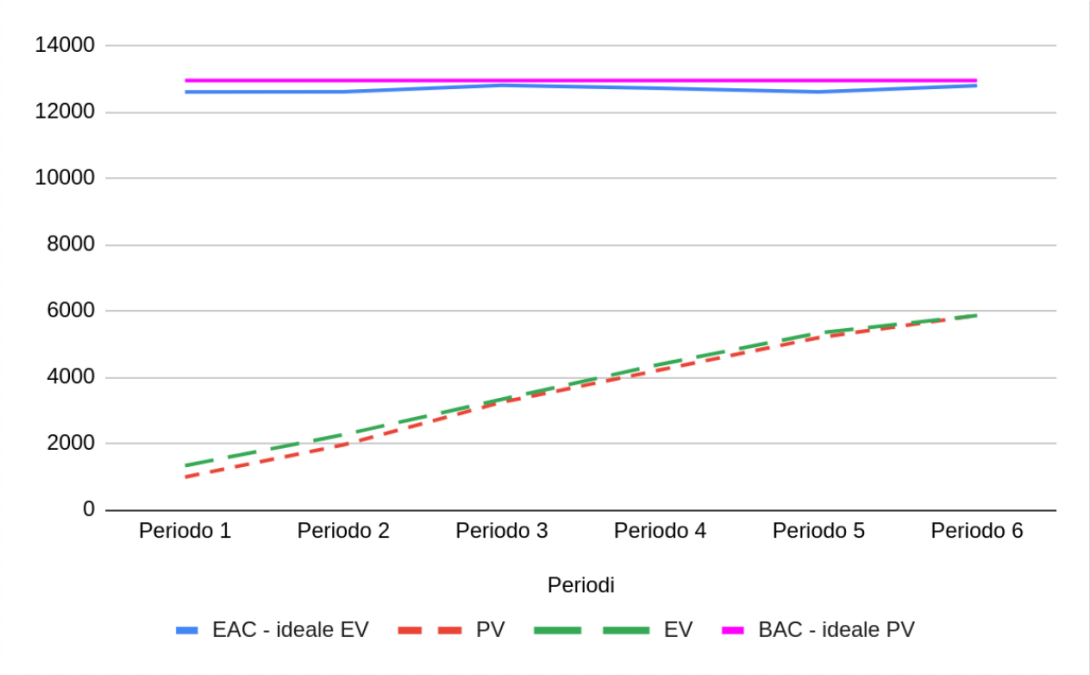

Figure 2: Grafico a linee delle metriche EV e PV

RTB: Il grafico mostra la curva del valore guadagnato (Earned Value), e la curva del valore pianificato (Planned Value). Come si può notare osservando la figura le due curve sono molto vicine, questo indica che il lavoro effettivamente svolto è conforme alla pianificazione; e nello specifico quella dell'EV è sempre leggermente superiore a quella del PV. Ciò indica che il progetto sta producendo un valore maggiore rispetto a quanto pianificato, e quindi che il lavoro svolto è leggermente al di sopra delle aspettative.
PB:

4.1.3 MPC03 - Actual Cost (AC) & MPC02 - Estimate to Complete (ETC)

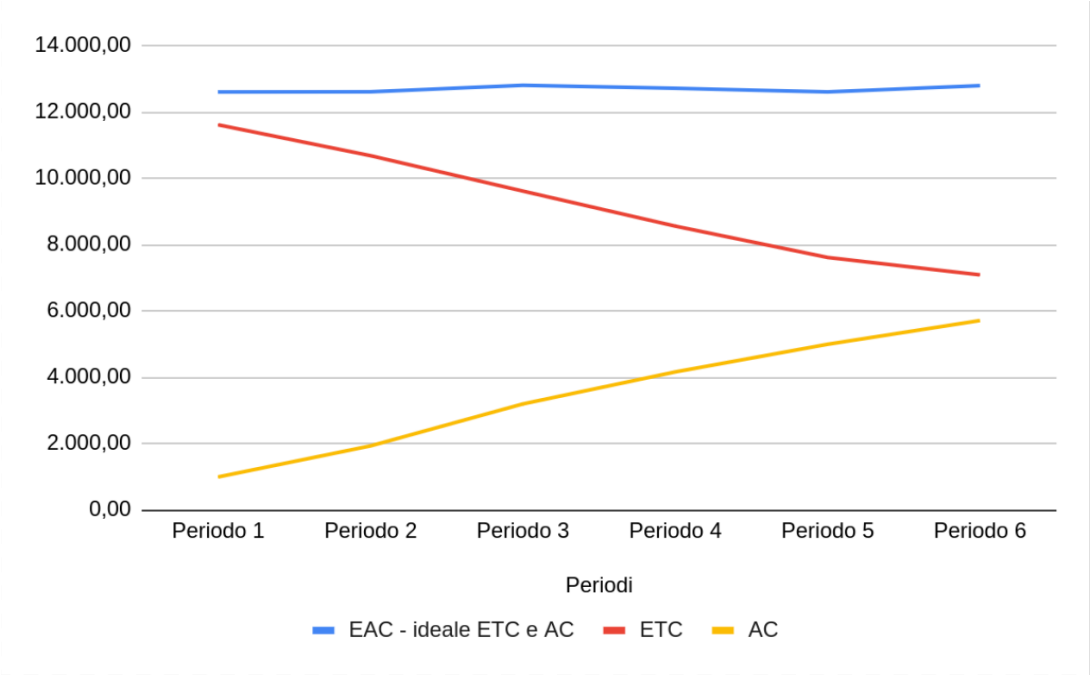

Figure 3: Grafico a linee delle metriche AC e ETC

RTB: Il grafico rappresenta l'Actual Cost (AC), ovvero i costi sostenuti per portare il progetto al suo stato corrente, e l'Estimate to Complete (ETC), cioè la stima del costo rimanente da sostenere per completare il progetto per ogni periodo di misurazione (termine sprint). Entrambe le metriche hanno valore ideale inferiore all'EAC, che viene rispettato in ogni iterazione.
Ovviamente l'ETC tende a diminuire al progredire del progetto poiché questo si avvicina alla sua conclusione, mentre l'AC mostra una crescita proporzionale e inversa rispetto all'ETC, in linea con le aspettative economiche del progetto.
PB:

4.1.4 MPC08 - Cost Performance Index (CPI)

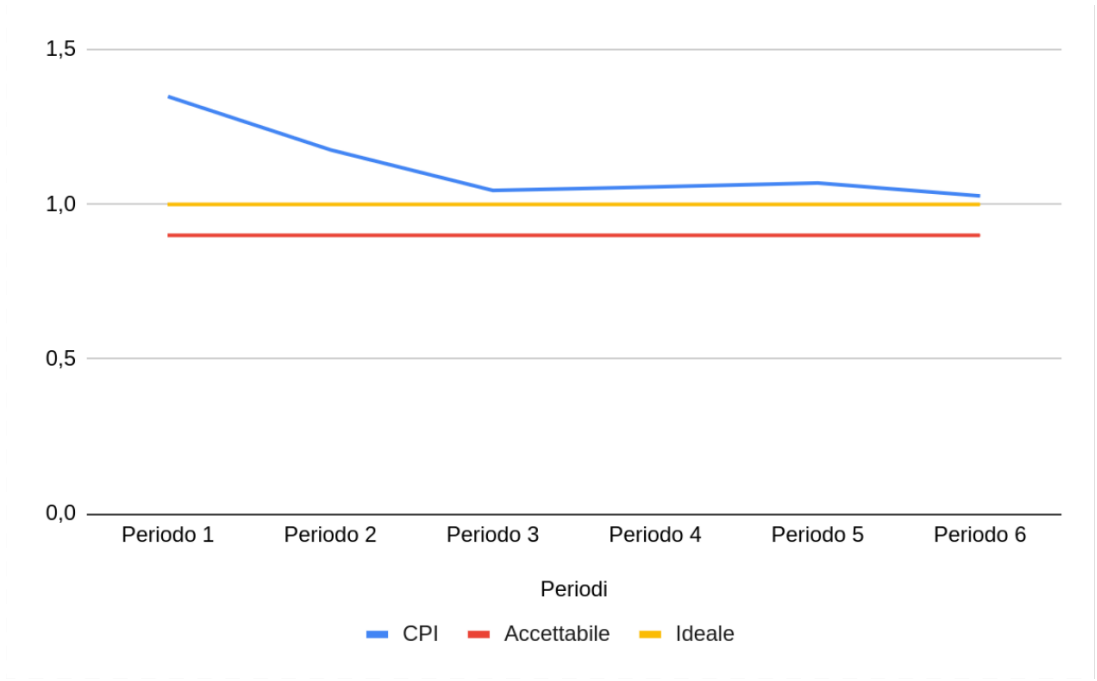

Figure 4: Grafico a linee della metrica CPI

RTB: Il Cost Performance Index (CPI) è una metrica che indica quanti obiettivi sono stati raggiunti rispetto alle spese sostenute in un certo momento del progetto. La metrica è data dal rapporto fra il valore guadagnato (EV) e i costi sostenuti (AC), e infatti il suo valore ideale deve essere maggiore o pari a 1. Il valore accettabile invece è stato scelto essere maggiore o uguale a 0.9.
Il grafico mostra che il CPI ha avuto valore strettamente maggiore di 1 per tutto il progetto, assestandosi fra il terzo e il sesto sprint a un valore pari a circa 1.06. Questo indica che il progetto ha prodotto un valore maggiore rispetto ai costi sostenuti.
PB:

4.1.5 MPC07 - Cost Variance (CV) & MPC06 - Schedule Variance (SV)

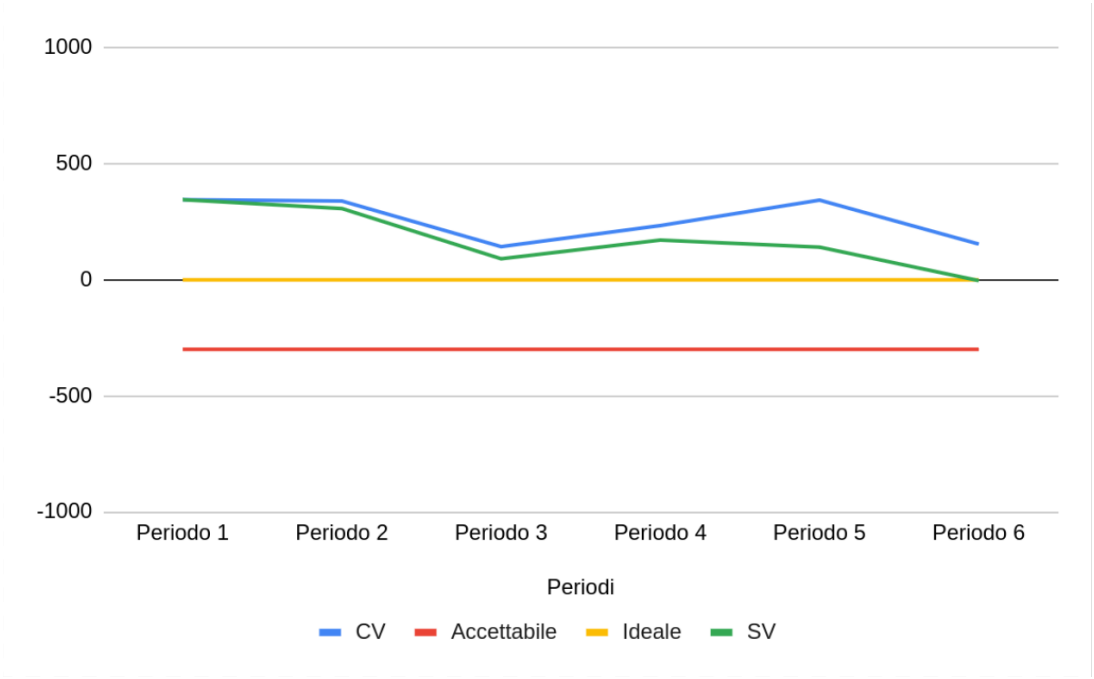

Figure 5: Grafico a linee delle metriche CV e SV

RTB: Il grafico mostra l'andamento della Cost Variance (CV) e della Schedule Variance (SV), che rappresentano rispettivamente: la differenza tra il valore guadagnato (EV) e i costi sostenuti (AC) e la differenza tra il valore guadagnato (EV) e il valore pianificato (PV).
La CV resta sempre positiva per l'intera durata del progetto fino al sesto sprint, e ciò indica che le spese effettuate sono inferiori rispetto al valore prodotto dal team nel corso della prima parte del progetto.
La SV mostra un andamento positivo fino al quinto sprint, e ciò indica che il gruppo ha saputo produrre un valore maggiore di quanto pianificato nel corso delle iterazioni effettuate. Invece per il sesto sprint la SV è andata ad un valore non positivo.

PB:

4.2 Qualità di processo - Sviluppo

4.2.1 MPC10 - Requirements Stability Index (RSI)

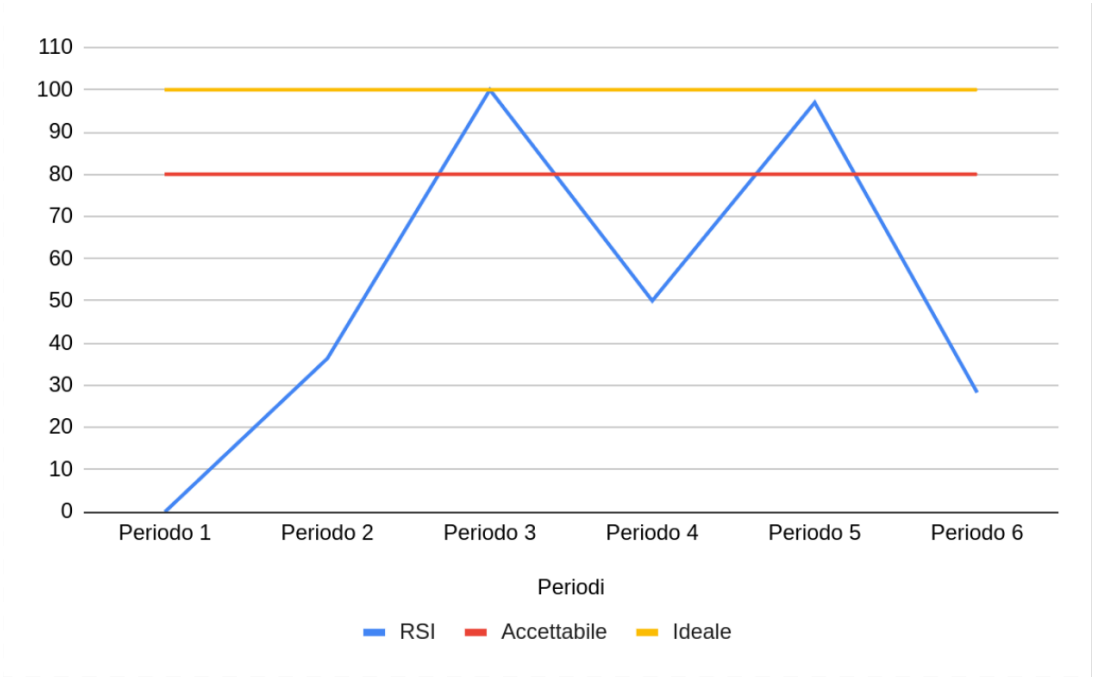

Figure 6: Grafico a linee della metrica RSI

RTB: La metrica RSI rappresenta un indice di stabilità dei requisiti individuati nel corso del progetto. Questo indice è in percentuale e assume valore tanto più alto quanto maggiore è la stabilità dei requisiti. In altre parole, il valore è alto se pochi requisiti sono cambiati rispetto all'ultima misurazione, mentre è basso se molti requisiti sono stati modificati, aggiunti o rimossi rispetto all'ultima misurazione.
Come si può evincere dal grafico, l'RSI è partito da valore nullo (in maniera coerente rispetto alla formula usata per il calcolo di tale metrica, poiché nel primo sprint sono stati definiti i primi requisiti); crescendo nella seconda iterazione (mentre il primo tentativo di identificazione dei requisiti era ancora in corso); e ha poi mostrato un andamento altalenante, con picchi di stabilità nel terzo e quinto periodo e cali significativi (inferiori al valore accettabile) in corrispondenza della terza e sesta iterazione. In entrambi questi casi il valore basso è seguito a significative modifiche dei requisiti, successive a degli incontri di chiarimento organizzati con il professor Cardin, che hanno evidenziato degli errori nell'approccio del gruppo all'analisi dei requisiti.
Complessivamente il gruppo si aspetta una maggiore stabilità in futuro, in caso di un buon esito della revisione RTB, mentre potrebbero essere necessari ulteriori significativi cambiamenti in caso di esito negativo.

PB:

4.3 Qualità di processo - Gestione dei processi

4.3.1 MPC16 - Rischi non previsti

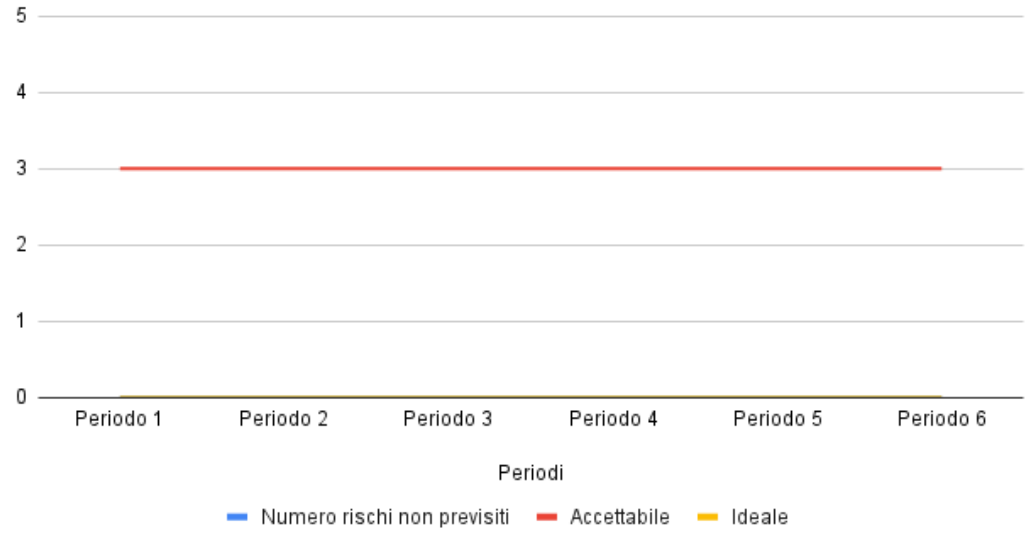

Figure 7: Grafico a linee della metrica "Rischi non previsti"

RTB: Nel corso del progetto il team ha senz'altro avuto modo di affrontare svariati dei rischi emersi durante l'analisi dei rischi, tuttavia non sono stati riscontrati rischi non previsti nel corso dei primi sei periodi del progetto.
PB:

4.3.2 MPC17 - Efficienza temporale (ET)

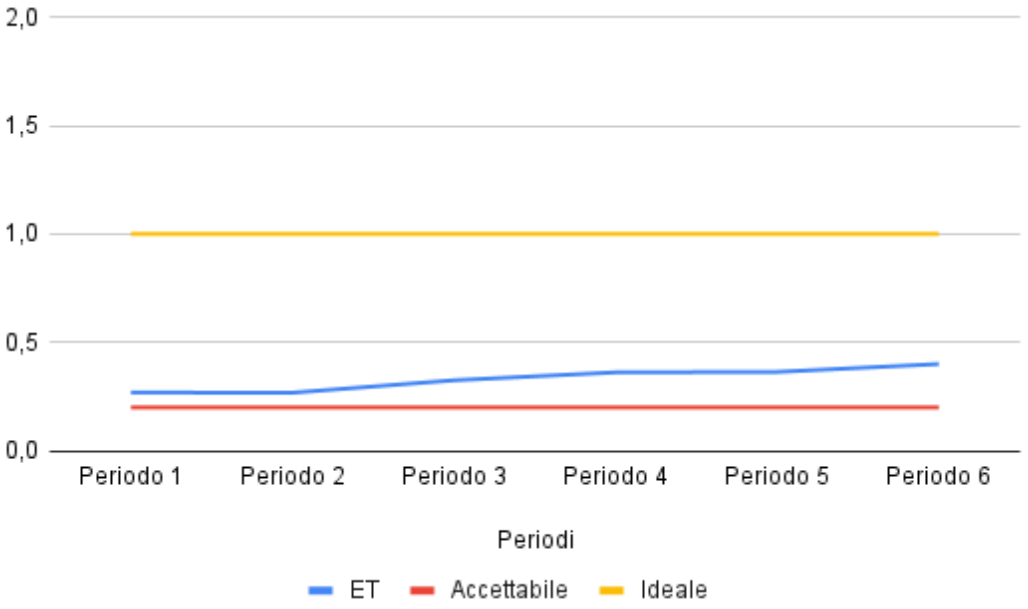

Figure 8: Grafico a linee della metrica ET

RTB: La metrica di efficienza temporale è data dal rapporto fra il numero di ore produttive e totali per ogni sprint, e indica quante delle ore dedicate al progetto sono state effettivamente usate per la pro-

duzione di materiale (software o di altra natura), rispetto alle ore effettivamente consumate dai membri del gruppo. Le ore produttive sono state accuratamente rendicontate per l'intera durata del progetto, mentre quelle totali (che includono anche le produttive), sono state misurate in maniera più precisa possibile, ma senza pretese di assoluta esattezza, per via della natura di queste ultime.
Dal grafico emerge che la ET si mantiene, per l'intera durata delle prime sei iterazioni del progetto, poco al di sopra del valore minimo accettabile (pari a 0.2), ma al di sotto del valore ideale (pari o superiore a 1.0).

PB:

4.4 Qualità di processo - Documentazione

4.4.1 MPC01 - Errori ortografici

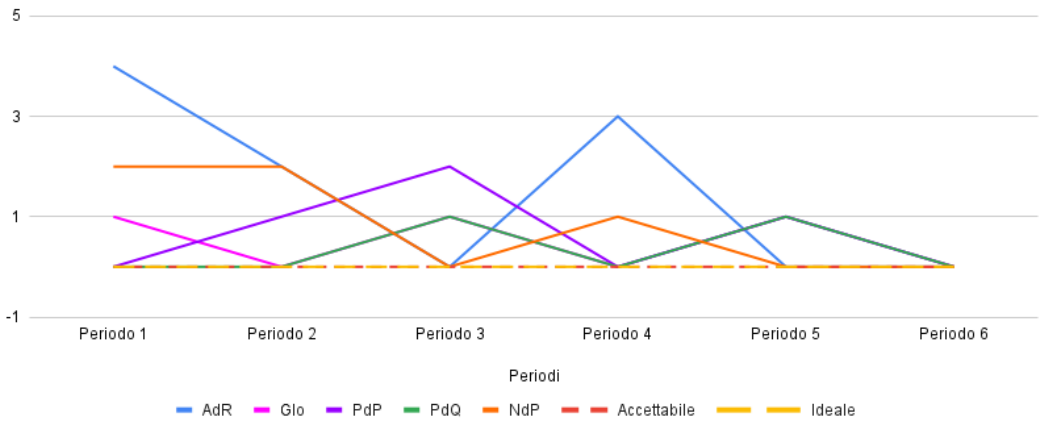

Figure 9: Grafico a linee della metrica "Errori ortografici"

RTB: La metrica rappresenta un indice della qualità della documentazione e fa riferimento al numero di errori ortografici individuati in fase di verifica dei documenti prodotti.
Sebbene il valore ideale non sia mai rispettato nel corso dei primi 5 sprint per tutta la documentazione, è bene evidenziare che il team ha mantenuto comunque dei valori bassi, che hanno presentato dei prevedibili aumenti in fasi di grande produzione di documentazione. L'obiettivo del gruppo è comunque stato quello di consegnare al termine del sesto periodo dei documenti privi di errori ortografici, e ciò è stato raggiunto, dato che almeno da parte nostra non ne sono stati individuati.

PB:

4.4.2 MPC11 - Indice Gulpease

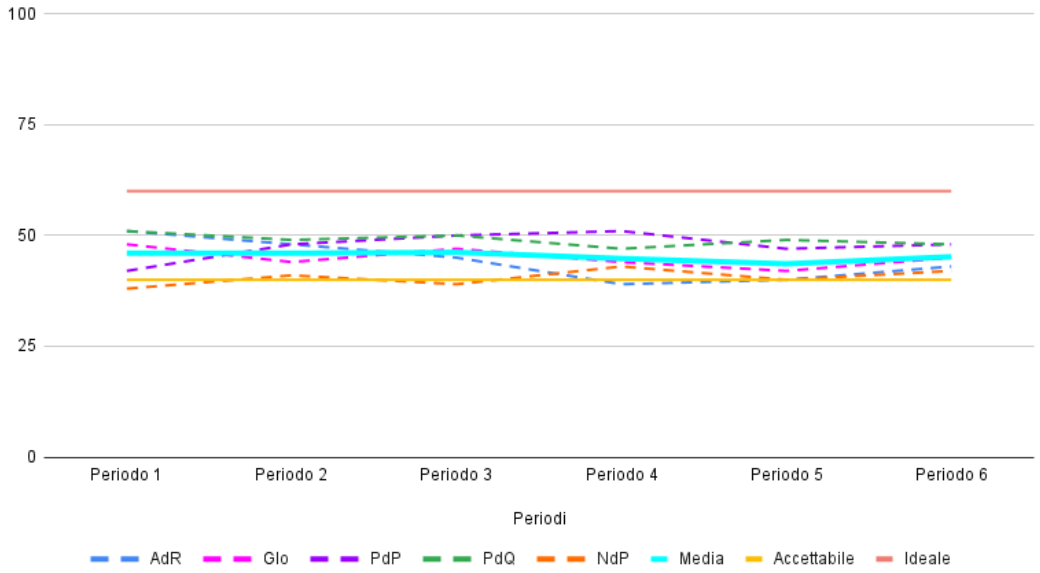

Figure 10: Grafico a linee della metrica "Indice di Gulpease"

RTB: Emerge da un'analisi del grafico che il valore di accettazione dell'indice di Gulpease (valore ideale *geq* 60; valore accettabile *geq* 40), è stato quasi sempre rispettato in tutti i documenti redatti per cui si è scelto di misurarlo; con alcune eccezioni per il documento *Analisi dei Requisiti* (nel quarto periodo), e per il documento *Norme di Progetto* (nella prima e nella terza iterazione).
Appare tuttavia chiaro che il gruppo ha avuto non poche difficoltà a raggiungere il valore ideale per questa metrica, poiché è stata sottovalutata la difficoltà dello scrivere documentazione chiara e con linguaggio semplice.
A questo proposito il gruppo si porrà in futuro l'obiettivo di migliorare il valore di questa metrica, cercando di esprimere i concetti con frasi più brevi e semplici.

PB:

4.5 Qualità di processo - Gestione della qualità

4.5.1 MPC15 - Metriche di qualità soddisfatte

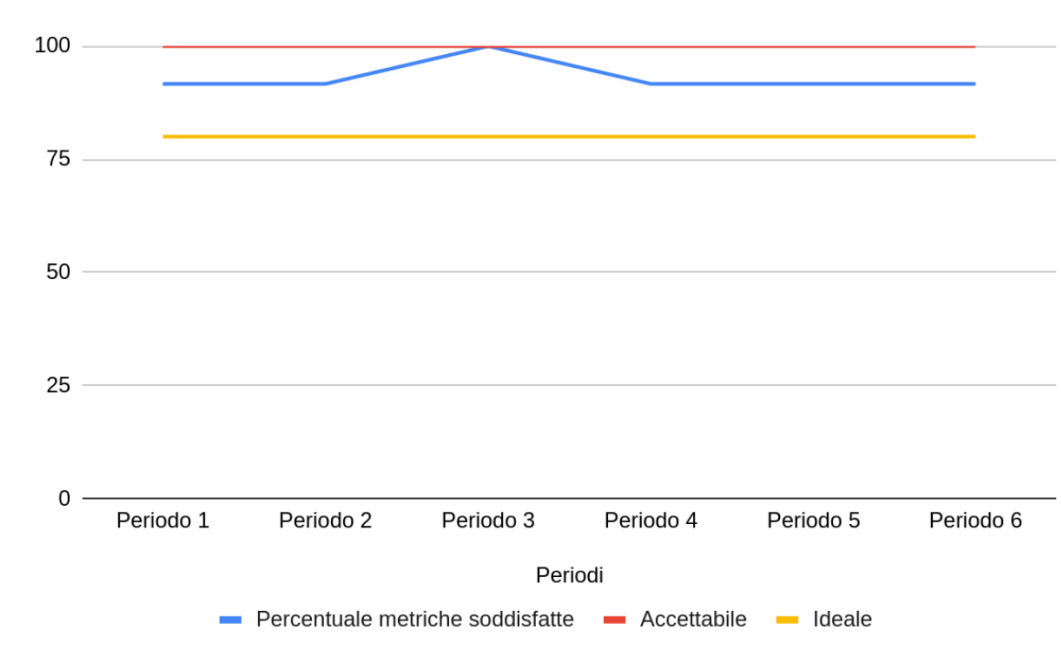

Figure 11: Grafico a linee della metrica "metriche di qualità soddisfatte"

RTB: Questa metrica rispecchia la capacità del team di rispettare i valori accettabili delle altre metriche misurata nel corso del progetto. Dall'analisi del suo grafico emerge che il gruppo è sempre riuscito a garantire il rispetto di più dell'90% delle metriche misurate. Ciò è un segnale chiaramente positivo che dimostra un buon lavoro del gruppo dal punto di vista della gestione della qualità.
Detto ciò è però bene notare che il gruppo, in solo uno dei precedenti sei periodi, ha raggiunto il valore ideale (100%), quindi ci sono comunque importanti margini di miglioramento da perseguire.
Un'ultima nota va aggiunta per specificare che il gruppo ha scelto di considerare la metrica relativa all'indice di Gulpease, che si applica a cinque documenti distinti, come una metrica unica facendo la media dei cinque valori misurati.

PB:

4.6 Qualità di processo - Analisi dei Requisiti

4.6.1 MPC09 - Requisiti Obbligatori Soddisfatti (ROS)

Figure 12: Grafico a linee della metrica ROS

PB: Questa metrica riflette la percentuale di requisiti obbligatori soddisfatti durante ogni iterazione. L'analisi del grafico mostra un andamento positivo, partendo dall'84% raggiunto alla revisione RTB e progredendo costantemente attraverso i periodi (87%, 90%, 93%...) fino a raggiungere il valore ideale del 100% alla revisione PB.
La costante progressione verso il 100% dei requisiti obbligatori soddisfatti, culminata nel raggiungimento del valore ideale, è un segnale estremamente positivo. Dimostra una solida comprensione delle esigenze del progetto e un'efficace esecuzione da parte del team nel tradurre tali requisiti in funzionalità complete.

4.7 Qualità di processo - Verifica

4.7.1 MPD04 - Line Coverage

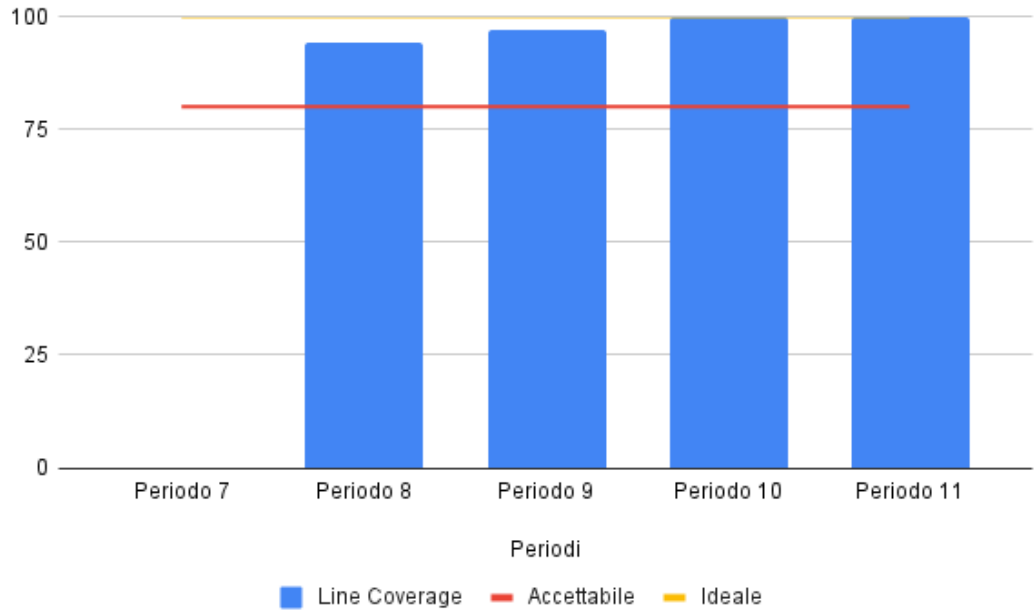

Figure 13: Grafico a linee della metrica "Line coverage"

PB: La metrica rappresenta la percentuale di linee di codice coperte dai test automatici. Il valore ideale è 100%, mentre il valore accettabile è 90%.
Il grafico mostra che il valore di questa metrica è sempre stato al di sopra del valore accettabile, e ha raggiunto il valore ideale in prossimità della revisione PB.
Da notare che non c'è alcun valore registrato in corrispondenza della settima iterazione poiché nessun test era ancora stati implementato durante quest'ultima.

4.7.2 MPD05 - Branch Coverage

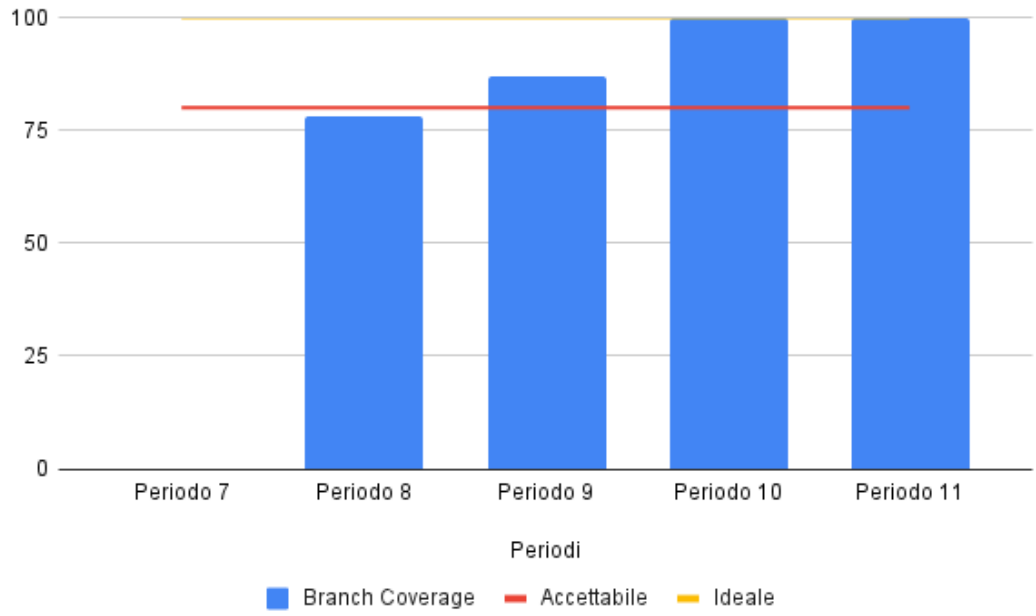

Figure 14: Grafico a linee della metrica "Branch coverage"

PB: La metrica rappresenta la percentuale di rami di codice coperti dai test automatici. Il valore ideale è 100%, mentre il valore accettabile è 90%.
Il grafico mostra che il valore di questa metrica è sempre stato al di sopra del valore accettabile, e ha raggiunto il valore ideale in prossimità della revisione PB.
Da notare che non c'è alcun valore registrato in corrispondenza della settima iterazione poiché nessun test era ancora stati implementato durante quest'ultima.

4.7.3 MPD10 - Linee di codice per metodo

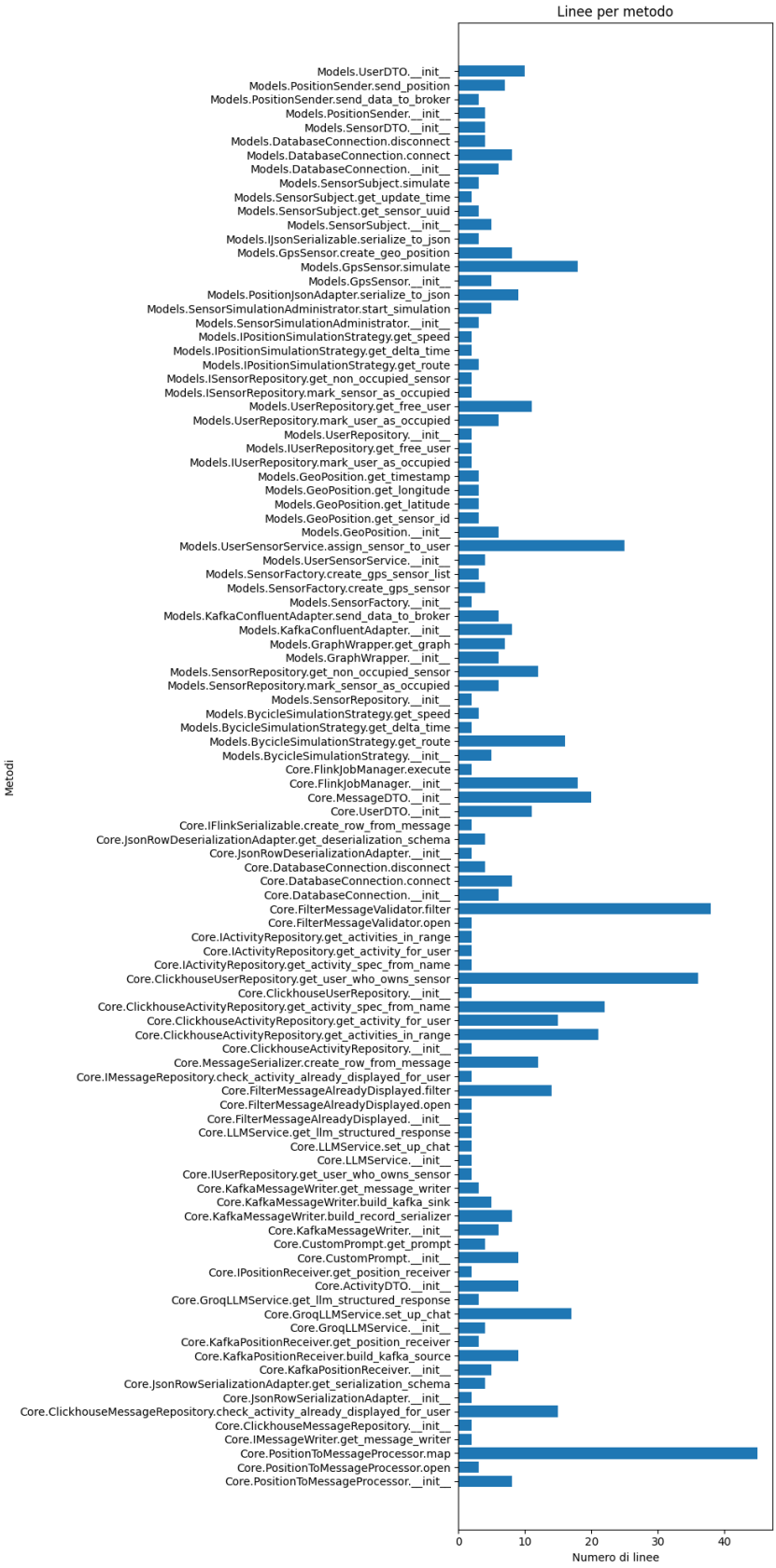

Figure 15: Grafico a barre della metrica "Linee di codice per metodo"

PB: Questo grafico rappresenta il numero di linee di codice per ciascun metodo scritto durante la fase di codifica.
Rispetto alla maggior parte delle altre metriche il grafico non rappresenta un andamento temporale, ma mostra il numero di linee di codice per ogni metodo allo stato corrente del prodotto software; ciò perché non sarebbe stato possibile rappresentare in maniera immediata e facilmente leggibile tale metrica altrimenti.
In ogni caso dal grafico si evince che il valore ideale (inferiore a 25) è quasi sempre stato rispettato, ad esclusione di tre casi, in cui è comunque stato registrato un valore accettabile (inferiore a 50).

4.7.4 MPD10 - Attributi per classe

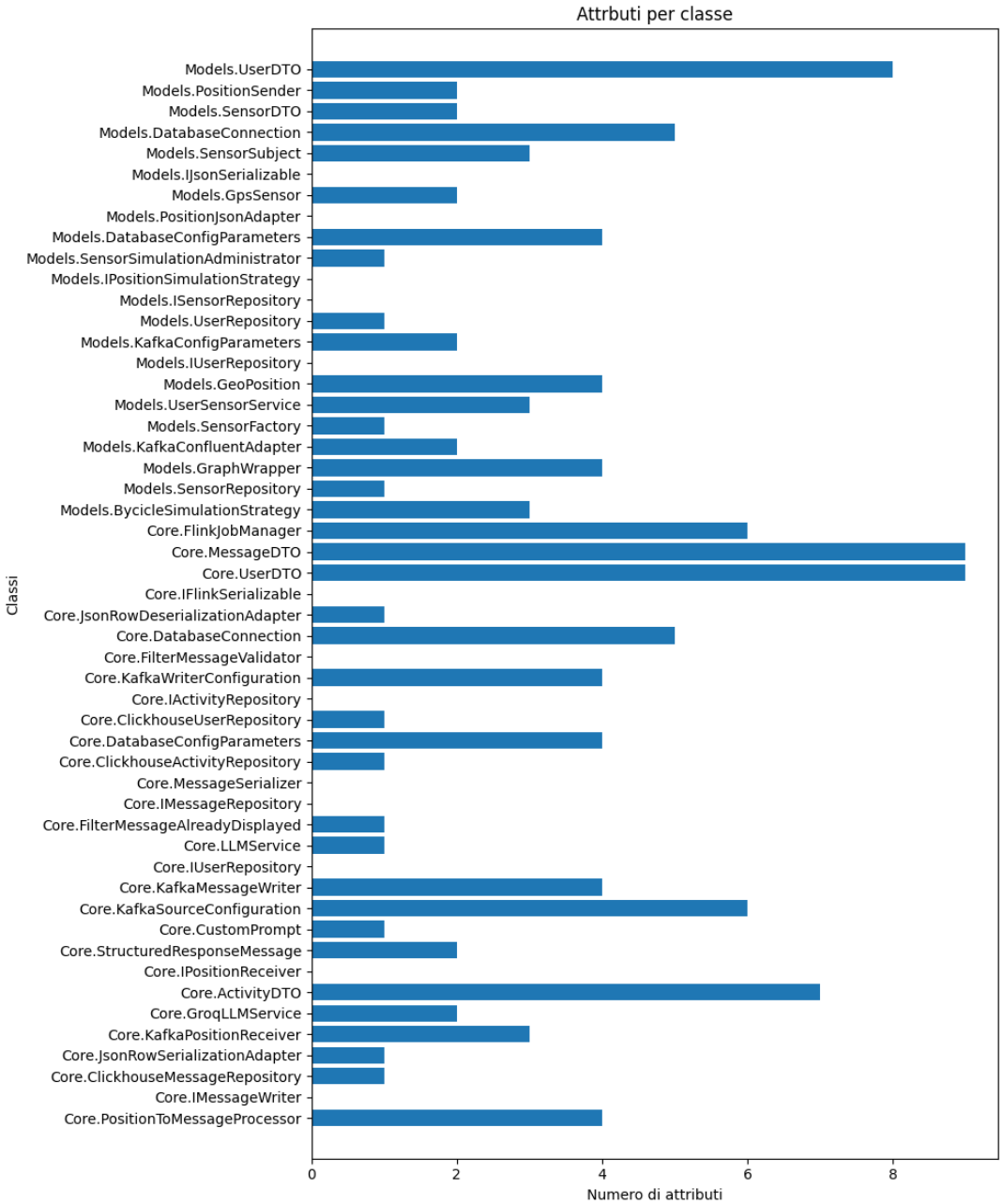

Figure 16: Grafico a barre della metrica "Attributi per classe"

PB: Questo grafico rappresenta il numero di attributi per ogni classe scritta durante la fase di codifica.
Rispetto alla maggior parte delle altre metriche il grafico non rappresenta un andamento temporale, ma

mostra il numero di attributi per ogni classe allo stato corrente del prodotto software; ciò perché non sarebbe stato possibile rappresentare in maniera immediata e facilmente leggibile tale metrica altrimenti. Analizzando il grafico si evince che il valore ideale (inferiore a 5) è sempre stato rispettato, ad esclusione di un caso, in cui è comunque stato registrato un valore accettabile (inferiore a 7).

4.7.5 MPD11 - Structure Fan IN

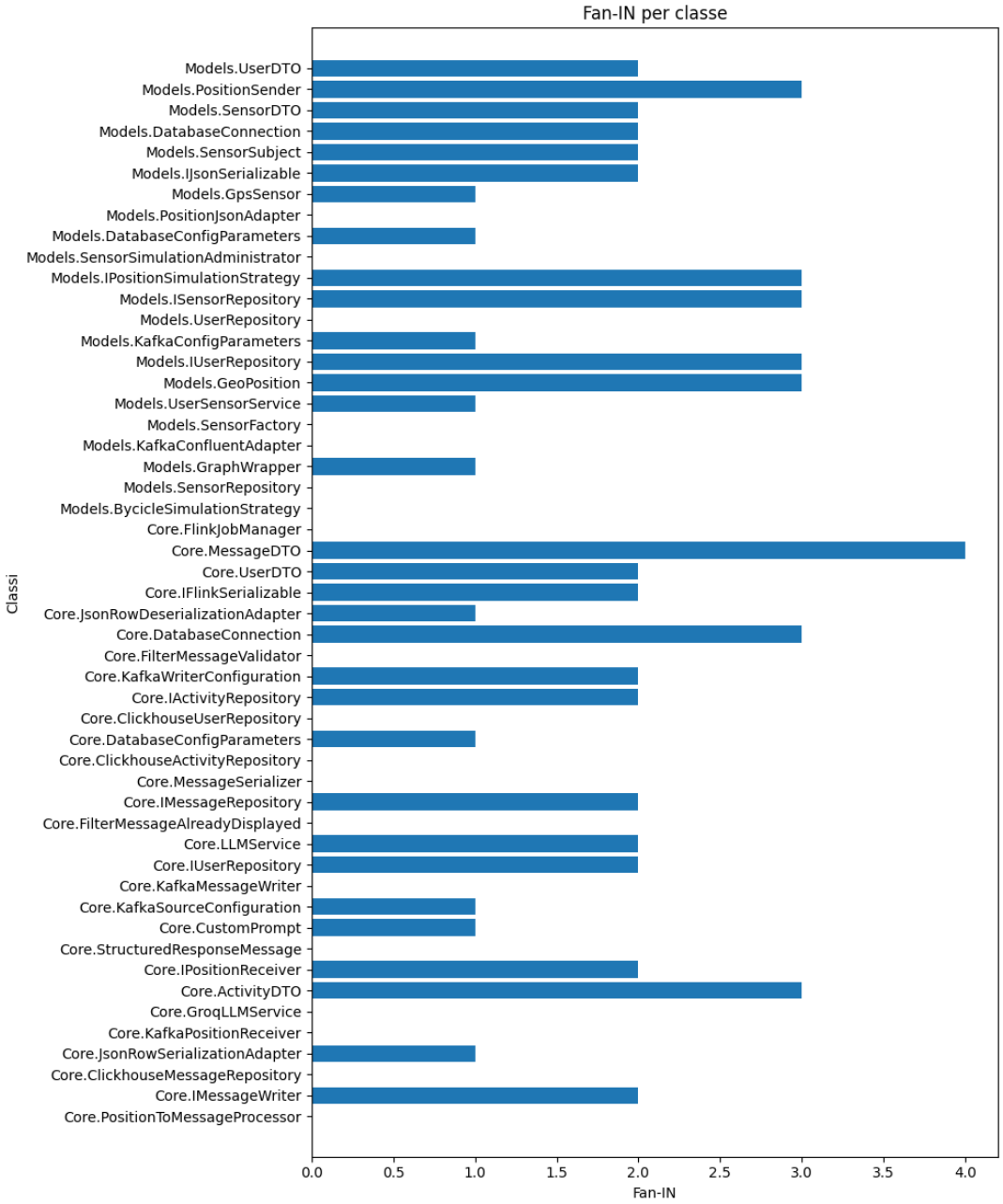

Figure 17: Grafico a barre della metrica "Structure Fan IN"

PB: Questo grafico rappresenta per ogni classe il numero di altre classi che dipendono da essa.
Rispetto alla maggior parte delle altre metriche il grafico non rappresenta un andamento temporale, ma mostra il numero di attributi per ogni classe allo stato corrente del prodotto software; ciò perché non sarebbe stato possibile rappresentare in maniera immediata e facilmente leggibile tale metrica altrimenti. Dall'analisi del grafico emerge che la fan-IN è sempre stata inferiore a 5 e per molti moduli si è registrato un valore nullo. Questa metrica non ha un valore ideale, poiché un alta fan-IN indica che una classe è

molto utilizzata da altre classi, e quindi è un buon segnale di riutilizzo del codice; ma allo stesso tempo un valore elevato indica che una classe crea molto accoppiamento essendo molto utilizzata, e quindi è un segnale negativo.
In generale i valori riscontrati non sono troppo elevati, e sono in linea con le aspettative del gruppo.

4.7.6 MPD12 - Structure Fan OUT

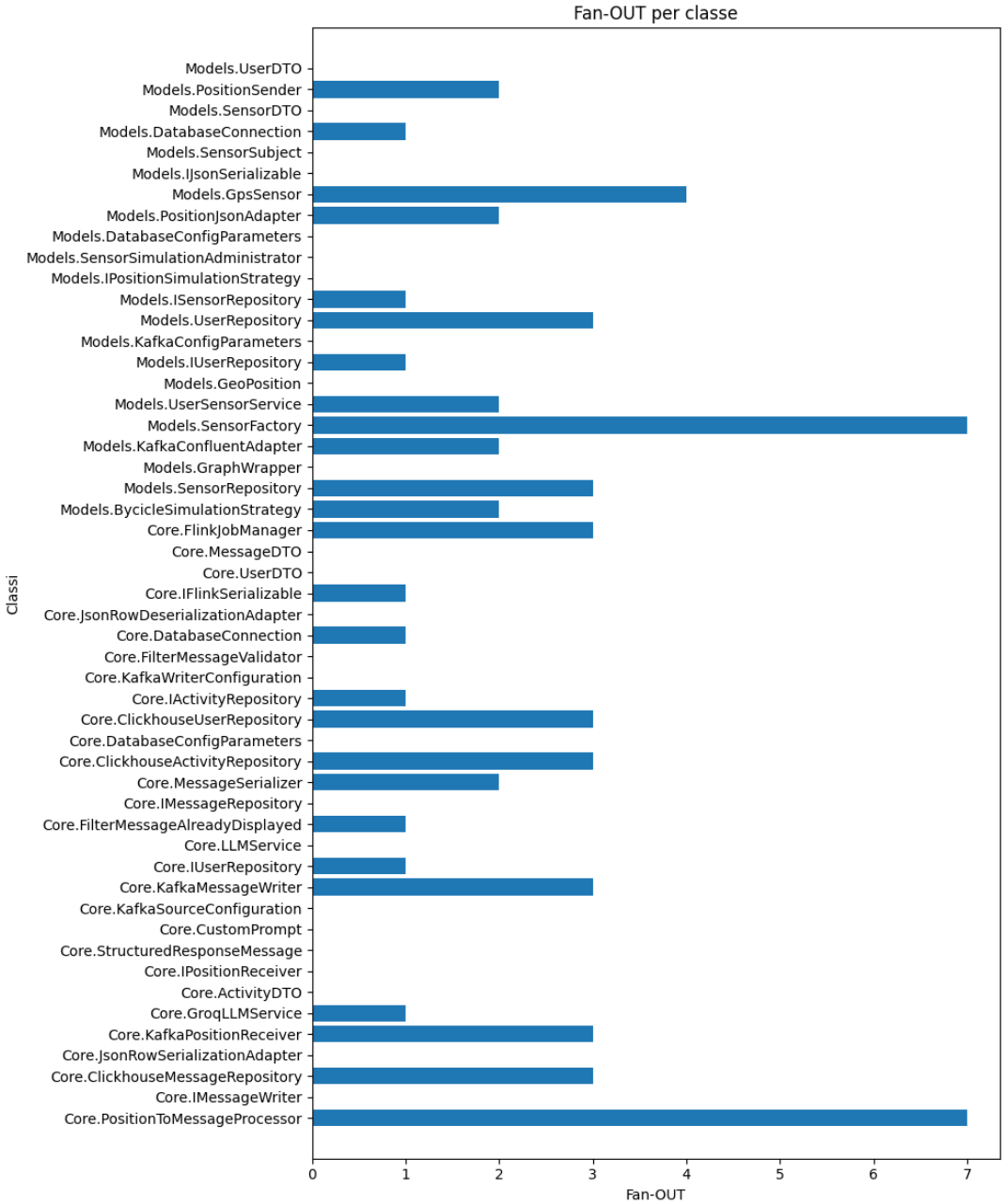

Figure 18: Grafico a barre della metrica "Structure Fan OUT"

PB: Questo grafico rappresenta per ogni classe il numero di altre classi la prima chiama o usa.
Rispetto alla maggior parte delle altre metriche il grafico non rappresenta un andamento temporale, ma mostra il numero di attributi per ogni classe allo stato corrente del prodotto software; ciò perché non sarebbe stato possibile rappresentare in maniera immediata e facilmente leggibile tale metrica altrimenti.
Dall'analisi del grafico emerge che la fan-OUT è sempre stata inferiore a 7 e per molti moduli si è registrato un valore nullo. Questa metrica non ha un valore ideale, poiché un alta fan-OUT indica che una

classe è molto dipendente da altre, cosa che generalmente è negativa (a meno che la classe non sia un controller o orchestratore); ma allo stesso tempo un valore basso, anche se p indice di buona coesione e indipendenza, può anche indicare che una classe è troppo semplice e superflua.
In generale il valore di fan-OUT ad eccezione di alcuni casi, è stato relativamente basso, in linea con le aspettative del gruppo.

4.7.7 MPC14 - Passed test cases percentage

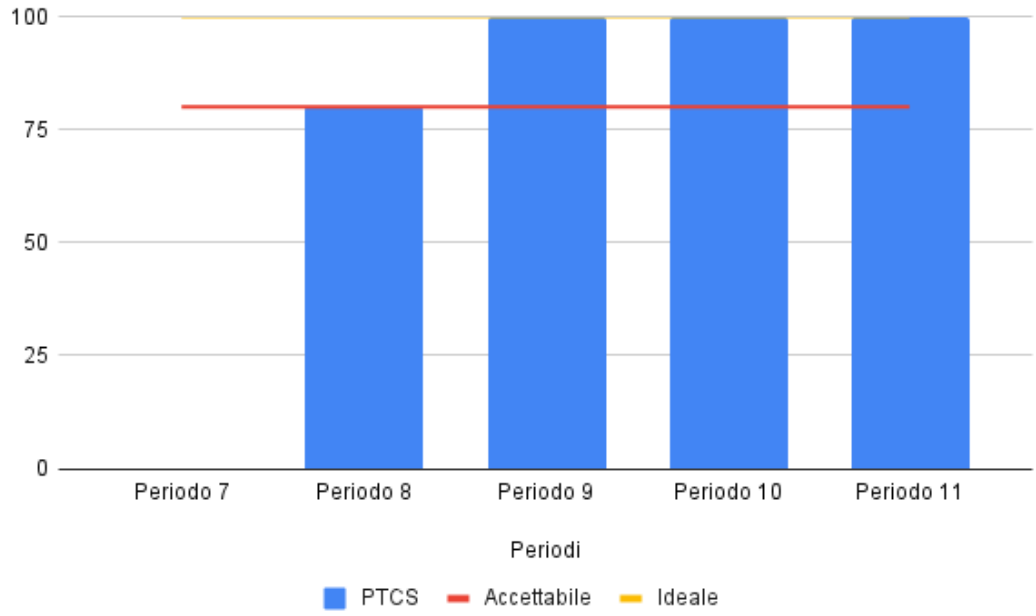

Figure 19: Grafico a linee della metrica "Passed test cases percentage"

PB: Questa metrica rappresenta la percentuale di test superati rispetto al totale dei test eseguiti. Il valore ideale è 100%, mentre il valore accettabile è 90%.
Il grafico mostra che il valore di questa metrica è sempre stato al di sopra del valore accettabile, e ha raggiunto il valore ideale in prossimità della revisione PB.
Da notare che non c'è alcun valore registrato in corrispondenza della settima iterazione poiché nessun test era ancora stati implementato durante quest'ultima.