Piano di Qualifica

7DOS - 28 Aprile 2019

Informazioni sul documento

Versione	4.0.0
Responsabile	Nicolò Tartaggia
Verifica	Andrea Trevisin Lorenzo Busin
Redazione	Giovanni Sorice Giacomo Barzon
Stato	Approvato
Uso	Esterno
Destinato a	Prof.Tullio Vardanega Prof.Riccardo Cardin Zucchetti 7DOS
Email	7dos.swe@gmail.com

Descrizione

Questo documento descrive le operazioni di verifica e validazione relative al progetto *G&B*.

Diario delle modifiche

Versione	Data	Descrizione	Autore	Ruolo
4.0.0	2019-4-28	*Approvazione del documento per la RA*	Nicolò Tartaggia	Responsabile
3.3.0	2019-04-27	*Verifica del documento*	Andrea Trevisin	Verificatore
3.2.0	2019-04-26	*Verifica del documento*	Lorenzo Busin	Verificatore
3.1.0	2019-04-22	*Revisione §B.1 - (vedi Verbale 2019-04-24)*	Giacomo Barzon	Amministratore
3.0.0	2019-03-22	*Approvazione del documento per la RQ*	Giovanni Sorice	Responsabile
2.7.0	2019-03-21	*Verifica del documento*	Nicolò Tartaggia	Verificatore
2.6.0	2019-03-20	*Verifica del documento*	Marco Costantino	Verificatore
2.5.0	2019-03-19	*Fine stesura §C.4*	Lorenzo Busin	Amministratore
2.4.3	2019-03-19	*Stesura §C.3*	Andrea Trevisin	Amministratore
2.4.2	2019-03-19	*Stesura §C.2*	Giacomo Barzon	Amministratore
2.4.1	2019-03-19	*Inizio stesura §C.4*	Lorenzo Busin	Amministratore
2.4.0	2019-03-19	*Stesura §C.1 - (vedi Verbale del 2019-03-18)*	Michele Roverato	Amministratore
2.3.0	2019-03-18	*Stesura §4*	Giacomo Barzon	Amministratore
2.2.1	2019-03-18	*Inseriti grafici misurazioni in §B.2 - (vedi Verbale del 2019-03-18)*	Lorenzo Busin	Amministratore
2.2.0	2019-03-18	*Stesura §A*	Giacomo Barzon	Amministratore
2.1.0	2019-03-18	*Spostata specifica dei test in §4*	Andrea Trevisin	Amministratore
2.0.1	2019-03-18	*Rimossi contenuti che attenevano alle Norme da §2 e §3 - (vedi Verbale del 2019-03-18)*	Michele Roverato	Amministratore
2.0.0	2019-02-07	*Approvazione del documento per la RP*	Marco Costantino	Responsabile

Versione	Data	Descrizione	Autore	Ruolo
1.8.0	2019-02-06	*Verifica del documento*	Giacomo Barzon	Verificatore
1.7.0	2019-02-05	*Verifica del documento*	Michele Roverato	Verificatore
1.6.1	2019-02-04	*Miglioramento sezione §4.3*	Lorenzo Busin	Amministratore
1.6.0	2019-02-02	*Correzioni a §2.3*	Nicolò Tartaggia	Amministratore
1.5.0	2019-02-01	*Miglioramenti a §2.1.5 e §2.1.6*	Lorenzo Busin	Amministratore
1.4.0	2019-01-30	*Verifica del documento*	Giacomo Barzon	Verificatore
1.3.0	2019-01-29	*Verifica del documento*	Michele Roverato	Verificatore
1.2.0	2019-01-21	*Spostati contenuti in §A*	Andrea Trevisin	Amministratore
1.1.0	2019-01-21	*Rimossi contenuti che attengono alle Norme*	Giovanni Sorice	Amministratore
1.0.0	2019-01-01	*Approvazione del documento per la RR*	Marco Costantino	Responsabile
0.5.0	2019-01-01	*Verifica del documento*	Michele Roverato	Verificatore
0.4.0	2018-12-29	*Verifica del documento*	Lorenzo Busin	Verificatore
0.3.1	2018-12-28	*Miglioramenti a §A.1*	Giovanni Sorice	Amministratore
0.3.0	2018-12-26	*Stesura §A*	Giovanni Sorice	Amministratore
0.2.2	2018-12-13	*Completamento stesura §2*	Nicolò Tartaggia	Amministratore
0.2.1	2018-12-13	*Miglioramenti a §2.3*	Giacomo Barzon	Analista
0.2.0	2018-12-05	*Stesura §2.3*	Giacomo Barzon	Analista
0.1.0	2018-12-05	*Inizio stesura sezione §2*	Nicolò Tartaggia	Amministratore
0.0.2	2018-12-05	*Stesura della sezione §1*	Andrea Trevisin	Amministratore
0.0.1	2018-12-05	*Stesura dello scheletro del documento*	Andrea Trevisin	Amministratore

Indice

1 Introduzione **8**
1.1 Scopo del documento 8
1.2 Scopo del prodotto 8
1.3 Glossario 8
1.4 Maturità del documento 8
1.5 Riferimenti 9
1.5.1 Normativi 9
1.5.2 Informativi 9

2 Qualità di processo **10**
2.1 Pianificazione e controllo 10
2.1.1 Obiettivi 10
2.1.2 Metriche 10
2.2 Gestione dei rischi 11
2.2.1 Obiettivi 11
2.2.2 Metriche 11
2.3 Gestione dei test 11
2.3.1 Obiettivi 11
2.3.2 Metriche 12
2.4 Versionamento e build 12
2.4.1 Obiettivi 12
2.4.2 Metriche 12
2.5 Riassunto delle metriche di processo 13

3 Qualità di prodotto **14**
3.1 Qualità dei documenti 14
3.1.1 Obiettivi 14
3.1.2 Metriche 14
3.2 Qualità del software 15
3.2.1 Functional Suitability 15
3.2.1.1 Obiettivi 15
3.2.1.2 Metriche 15
3.2.2 Performance Efficiency 15
3.2.2.1 Obiettivi 15
3.2.2.2 Metriche 15
3.2.3 Usability 16
3.2.3.1 Obiettivi 16
3.2.3.2 Metriche 16
3.2.4 Reliability 16
3.2.4.1 Obiettivi 16
3.2.4.2 Metriche 16
3.2.5 Maintainability 17
3.2.5.1 Obiettivi 17
3.2.5.2 Metriche 17

3.3 Riassunto delle metriche di prodotto . . . 18

4 Specifica dei test . . . 18
4.1 Test di accettazione . . . 18
4.2 Test di sistema . . . 21
4.3 Test di integrazione . . . 24
4.4 Test di unità . . . 25

Appendici . . . 30

A Copertura dei requisiti . . . 30
A.1 Resoconto copertura dei requisiti . . . 33

B Resoconto delle attività di verifica . . . 35
B.1 Revisioni . . . 35
B.1.1 Revisione dei Requisiti . . . 35
B.1.2 Revisione di Progettazione . . . 35
B.1.2.1 Consolidamento dei requisiti . . . 35
B.1.2.2 Progettazione architetturale . . . 37
B.1.3 Revisione di Qualifica . . . 37
B.1.3.1 Progettazione di dettaglio e codifica . . . 37
B.1.4 Revisione di Accettazione . . . 38
B.1.4.1 Verifica e validazione . . . 38
B.2 Misurazioni . . . 40
B.2.1 Schedule Variance . . . 40
B.2.2 Budget Variance . . . 40
B.2.3 Numero rischi non previsti . . . 41
B.2.4 Indisponibilità dei servizi esterni . . . 41
B.2.5 Percentuale di test eseguiti . . . 42
B.2.6 Percentuale test case passati . . . 42
B.2.7 Percentuale test case falliti . . . 42
B.2.8 Code coverage . . . 43
B.2.9 Tempo medio necessario al team per risolvere un errore . . . 43
B.2.10 Efficienza nella progettazione dei test . . . 44
B.2.11 Percentuale di errori corretti . . . 44
B.2.12 Media commit a settimana . . . 45
B.2.13 Media build Travis a settimana . . . 45
B.2.14 Percentuale build Travis superate . . . 46
B.2.15 Gunning fog index . . . 46
B.2.16 Indice di Gulpease . . . 47
B.2.17 Numero di errori grammaticali . . . 48
B.2.18 Functional Implementation Completeness . . . 48
B.2.19 Average Functional Implementation Correctness . . . 49
B.2.20 Tempo di risposta . . . 49
B.2.21 Average Learning Time . . . 50
B.2.22 Failure Density . . . 50
B.2.23 Operazioni con gestione errori . . . 51

B.2.24 Failure Analysis 51
B.2.25 Comment Ratio 52

C Specifica dei test 53
C.1 Test di accettazione 53
C.2 Test di sistema 63
C.3 Test di integrazione 79
C.4 Test di unità 83

Elenco delle tabelle

2 Riassunto delle metriche di processo . . . 13
3 Riassunto delle metriche di prodotto . . . 18
4 Riassunto test di accettazione . . . 19
5 Riassunto test di sistema . . . 22
6 Riassunto test di integrazione . . . 24
7 Riassunto test di unità . . . 28
8 Copertura requisiti funzionali . . . 32
9 Maturità dei processi, Consolidamento . . . 36
10 Maturità dei processi, Progettazione Architetturale . . . 37
11 Maturità dei processi, Progettazione di dettaglio e codifica . . . 38
12 Maturità dei processi, Verifica e validazione . . . 39
13 Specifica test di accettazione . . . 62
14 Specifica test di sistema . . . 78
15 Specifica test di integrazione . . . 82
16 Specifica test di unità . . . 143

Elenco delle figure

1 Resoconto test di accettazione . . . 20
2 Resoconto test di sistema . . . 23
3 Resoconto test di integrazione . . . 24
4 Resoconto test di unità . . . 29
5 Resoconto requisiti funzionali obbligatori . . . 33
6 Resoconto requisiti funzionali desiderabili . . . 33
7 Resoconto requisiti funzionali opzionali . . . 34
8 Misurazione Schedule Variance . . . 40
9 Misurazione Budget Variance . . . 40
10 Misurazione numero rischi non previsti . . . 41
11 Misurazione indisponibilità dei servizi esterni . . . 41
12 Misurazione percentuale di test eseguiti . . . 42
13 Misurazione percentuale test case passati . . . 42
14 Misurazione percentuale test case falliti . . . 43
15 Misurazione code coverage . . . 43
16 Misurazione tempo medio necessario per risolvere un errore . . . 44
17 Misurazione efficienza nella progettazione dei test . . . 44
18 Misurazione percentuale di errori corretti . . . 45
19 Misurazione media commit a settimana . . . 45
20 Misurazione media build Travis a settimana . . . 46
21 Misurazione percentuale build Travis superate . . . 46
22 Misurazione Gunning fog index . . . 47
23 Misurazione indice di Gulpease . . . 47
24 Misurazione numero di errori grammaticali . . . 48
25 Misurazione Functional Implementation Completeness . . . 48
26 Misurazione Average Functional Implementation Correctness . . . 49
27 Misurazione tempo di risposta . . . 49
28 Misurazione Average Learning Time . . . 50
29 Misurazione Failure Density . . . 50
30 Misurazione . . . 51
31 Misurazione Failure Analysis . . . 51
32 Misurazione Comment Ratio . . . 52

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di esporre dettagliatamente le norme, le metodologie e gli standard che il gruppo 7DOS intende adottare per assicurare che ogni *prodotto*$_g$, di natura documentale o applicativa, aderisca ai vincoli di *qualità*$_g$ stabiliti dal proponente. Per garantire il rispetto di tali vincoli si prevede un continuo *processo*$_g$ di verifica delle attività svolte dal gruppo, al fine di individuare eventuali problematiche nel minor tempo possibile permettendo immediati interventi risolutivi.

1.2 Scopo del prodotto

Il prodotto da realizzare consiste in un *plug-in*$_g$ per il software di monitoraggio *Grafana*$_g$, da sviluppare in linguaggio *JavaScript*$_g$. Il prodotto dovrà svolgere almeno le seguenti funzioni:

- Leggere la definizione di una *rete Bayesiana*$_g$, memorizzata in formato *JSON*$_g$;
- Associare dei nodi della rete Bayesiana ad un flusso di dati presente nel sistema di Grafana;
- Ricalcolare i valori delle probabilità della rete secondo regole temporali prestabilite;
- Derivare nuovi dati dai nodi della rete non collegati al flusso di dati, e fornirli al sistema di Grafana;
- Visualizzare i dati mediante il sistema di creazione di grafici e *dashboard*$_g$ a disposizione.

L'azienda proponente prevede di utilizzare il prodotto per il monitoraggio di sistemi gestionali in Cloud; tuttavia, dato l'obiettivo di rendere il prodotto open-source, esso dovrà essere utilizzabile indipendentemente dal particolare sistema che si desidera monitorare.

1.3 Glossario

Per rendere la lettura del documento più semplice, chiara e comprensibile viene allegato il *Glossario v4.0.0* nel quale sono contenute le definizioni dei termini tecnici, dei vocaboli ambigui, degli acronimi e delle abbreviazioni. La presenza di un termine all'interno del Glossario è segnalata con una "g" posta come pedice (esempio: *Glossario*$_g$).

1.4 Maturità del documento

Il presente documento ha raggiunto un buon grado di dettaglio e completezza, ciò non esclude che potrebbe essere soggetto ad incrementi futuri. Tutto ciò che riguarda la pianificazione degli incrementi, può essere trovato nel *Piano di Progetto v4.0.0* in §4.

1.5 Riferimenti

1.5.1 Normativi

- **Norme di Progetto:** *Norme di Progetto v4.0.0*;
- **Capitolato d'appalto C3**: G&B monitoraggio intelligente di processi DevOps https://www.math.unipd.it/~tullio/IS-1/2018/Progetto/C3.pdf;
- **Verbali:** *Verbale del 2018-12-11*, *Verbale del 2019-03-18*.

1.5.2 Informativi

- **ISO/IEC 12207**: https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf;
- **ISO/IEC 25010**: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en.;
- **ISO/IEC 29119**: https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:29119:-1:ed-1:v1:en;
- **Slide dell'insegnamento Ingegneria del Software 2018-2019 - Processi Software**: https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L03.pdf;
- **Slide dell'insegnamento Ingegneria del Software 2018-2019 - Verifica e validazione: introduzione**: https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L16.pdf;
- **Slide dell'insegnamento Ingegneria del Software 2018-2019 - Verifica e validazione: analisi statica**: https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L17.pdf;
- **Slide dell'insegnamento Ingegneria del Software 2018-2019 - Verifica e validazione: analisi dinamica**: https://www.math.unipd.it/~tullio/IS-1/2018/Dispense/L18.pdf;
- ***Grafana*$_g$ Code Styleguide:** http://docs.grafana.org/plugins/developing/code-styleguide/;
- ***Angular TypeScript*$_g$ Code Styleguide:** https://angular.io/guide/styleguide;
- **Software Engineering - Ian Sommerville - 10th Edition**(Capitolo 24).

2 Qualità di processo

È impossibile creare prodotti di alta qualità se il proprio *way of working*$_g$ è scadente: risulta quindi fondamentale che i processi attuati, *in primis*, garantiscano un elevato livello qualitativo. Il gruppo 7DOS ha deciso di adottare la normativa *ISO/IEC 15504*$_g$, anche nota come *SPICE*$_g$, e di seguire il principio di miglioramento continuo (*PDCA*$_g$).

Per il controllo della qualità di processo il gruppo 7DOS utilizzerà l'approccio a maturità di processo, in quanto previsto dalle buone pratiche di *management*$_g$ e più adatto ad un gruppo inesperto.

2.1 Pianificazione e controllo

Questo processo ha come scopo la pianificazione dello sviluppo del progetto, andando a definire suddivisione delle attività da svolgere, organizzazione oraria del lavoro, pianificazione e controllo dei costi e standard di supporto e accertandosi che il team abbia padronanza delle tecnologie utilizzate.

2.1.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Pianificazione**: organizzazione oraria del lavoro, prestando attenzione nell'assegnazione dei compiti ai vari membri del gruppo e attenendosi ai costi preventivati;
- **Budget**: controllo e verifica della pianificazione monetaria cercando di attenersi ai costi preventivati;
- **Standard**: attenersi a standard ben precisi, affinché il gruppo abbia sempre una linea guida di supporto;
- **Preparazione del personale**: assicurarsi che ciascun membro del personale abbia familiarità con le tecnologie richieste in ogni *task*$_g$, in modo tale da garantire produttività dell'ambiente lavorativo.

2.1.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v4.0.0*, sono le seguenti:

- *Schedule Variance* (*SV*$_g$);
- *Budget Variance* (*BV*$_g$).

2.2 Gestione dei rischi

Questo processo ha come scopo il riconoscimento dei rischi a cui il progetto può incorrere. Il gruppo 7DOS mira, così facendo, a ridurre il più possibile il verificarsi di essi.

2.2.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Identificazione nuovi rischi**: ad ogni fase del progetto il gruppo identificherà i possibili nuovi rischi;
- **Analisi dei rischi**: i rischi individuati verranno analizzati in modo da poter fornire procedure automatiche per prevenire che uno di essi si verifichi.

2.2.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v4.0.0*, sono le seguenti:

- *Numero rischi non previsti;*
- *Indisponibilità servizi esterni.*

2.3 Gestione dei test

Questo processo ha come scopo la definizione di misurazioni sull'esecuzione dell'analisi dinamica, in modo tale da garantire una gestione efficace di essa.

2.3.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Efficienza dei test**: i test devono essere progettati per individuare il maggior numero di difetti possibili. In questo modo è possibile realizzare un prodotto di qualità che soddisfa i requisiti specificati;
- **Risoluzione tempestiva dei problemi riscontrati**: le problematiche evidenziate dai test devono essere risolte nel minor tempo possibile dal gruppo;
- **Eseguire tutti i test necessari**: il sistema si evolve. Molti dei difetti precedentemente segnalati vengono corretti. Ogni volta che viene risolto un errore o è stata aggiunta una nuova funzionalità, è necessario eseguire tutti i test per assicurarsi che il nuovo software modificato abbia effettivamente corrette gli errori noti e non abbia introdotto nuovi errori.

2.3.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v4.0.0*, sono le seguenti:

- *Percentuale test case passati*;
- *Percentuale test case falliti*;
- *Tempo medio necessario al team per risolvere un errore*;
- *Efficienza nella progettazione dei test*;
- *Percentuale di errori corretti*;
- *Percentuale dei test eseguiti*;
- *Code coverage*.

2.4 Versionamento e build

Questo processo ha come scopo la quantificazione e il controllo dei $Commit_g$, in modo tale da verificare l'effettivo sviluppo di versioni sempre più complete e stabili del prodotto. Inoltre, lo strumento di integrazione continua *Travis* CI_g permetterà di eseguire le $build_g$ in modo automatico, permettendo una verifica della correttezza del codice rispetto alle norme.

2.4.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Commit frequenti**: ogni membro del gruppo deve garantire una buona frequenza di commit, affinché il prodotto sia sempre aggiornato con le ultime modifiche;
- **Modifiche di piccole dimensioni**: ogni commit deve comportare una quantità di modifiche limitata. In questo modo, ogni versione del prodotto è monitorabile e verificabile;
- **Buona riuscita delle build**: ogni membro deve accertarsi che ogni commit comporti una build positiva, in modo tale evitare la diffusione di errori.

2.4.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v4.0.0*, sono le seguenti:

- *Media commit a settimana;*
- *Media build Travis a settimana;*
- *Percentuale build Travis superate.*

2.5 Riassunto delle metriche di processo

Nome Metrica	Range accettabile	Range ottimale
Schedule Variance	≥-5 giorni	≥0 giorni
Budget Variance	≥-10%	≥0%
Numero rischi non previsti	≤3	0
Indisponibilità servizi esterni	≥-5	≥0
Percentuale test case passati	100%	100%
Percentuale test case falliti	0%	0%
Tempo medio necessario al team per risolvere un errore	0h-4h	0h-2h
Efficienza nella progettazione dei test	0.5h-3h	0.5h-2h
Percentuale dei difetti corretti	90%-100%	95%-100%
Percentuale dei test eseguiti	85%-100%	95%-100%
Code coverage	80%-100%	90%-100%
Media commit a settimana	≥20	≥30
Media build Travis a settimana	≥20	≥30
Percentuale build Travis superate	70%-100%	80%-100%

Tabella 2: Riassunto delle metriche di processo

3 Qualità di prodotto

Per poter garantire che il prodotto realizzato sia di alta qualità, è necessario definire un modello per la valutazione di quest'ultima; il team 7DOS, per questo motivo, ha scelto di adottare il modello di qualità delineato nello standard *ISO/IEC 25010*$_g$, anche noto come *SQuaRE*$_g$.

Tale modello comprende 8 caratteristiche (ciascuna divisa in sotto-caratteristiche, per un totale di 31) che vanno prese in considerazione durante lo sviluppo del progetto per garantire un'elevata qualità complessiva del prodotto finale.

Per praticità e rilevanza ai fini del prodotto, sono state selezionate 5 caratteristiche da considerare e per ciascuna sono state individuate le sotto-caratteristiche più rilevanti al progetto, da perseguire come obiettivi prioritari. In particolare, sono state scartate: *Compatibility*$_g$, in quanto andando a realizzare un plug-in (di natura integrato in un sistema preesistente) è stata giudicata superflua; *Security*$_g$, in quanto il plug-in non dovrà gestire autenticazione o raccolta di dati; ed infine *Portability*$_g$ in quanto essendo il prodotto un plug-in per un determinato sistema, non è rilevante la sua portabilità ad altri ambienti.

3.1 Qualità dei documenti

I documenti prodotti devono possedere caratteristiche consistenti. In particolare, essi devono essere leggibili, comprensibili e corretti a livello ortografico e sintattico.

3.1.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Leggibilità**: i documenti prodotti dovranno essere leggibili e comprensibili da persone con almeno una licenza di istruzione media;
- **Correttezza ortografica**: i documenti prodotti dovranno essere privi di errori ortografici.

3.1.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v4.0.0*, sono le seguenti:

- *Gunning fog index*;
- *Indice di Gulpease*;
- *Numero di errori grammaticali*.

3.2 Qualità del software

3.2.1 Functional Suitability

Caratteristica che esprime il grado di soddisfacimento dei requisiti espliciti ed impliciti da parte di un prodotto o servizio.

3.2.1.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Functional Completeness**: l'insieme delle funzioni rispettano le aspettative;
- **Functional Correctness**: i risultati ottenuti sono corretti e presentano livelli di precisione desiderati.

3.2.1.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v4.0.0*, sono le seguenti:

- *Functional Implementation Completeness*;
- *Average Functional Implementation Correctness*.

3.2.2 Performance Efficiency

Caratteristica che esprime le prestazioni relative al sistema come quantità di risorse utilizzate per eseguire una determinata funzionalità.

3.2.2.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Time Behaviour**: tempi di risposta ed elaborazione assumono valori adeguati;
- **Resource Utilization**: le risorse disponibili utilizzate durante l'esecuzione di una funzionalità sono adeguate rispetto alla funzionalità stessa.

3.2.2.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v4.0.0*, sono le seguenti:

- *Tempo di risposta*.

3.2.3 Usability

Caratteristica che esprime il grado con cui un prodotto o sistema può essere usato da un determinato utente.

3.2.3.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Learnability**: il prodotto è comprensibile dall'utente e il tempo per conoscere a pieno le sue funzionalità è accettabile;
- **Operability**: le funzionalità offerte devono essere coerenti con ciò che l'utente si aspetta;
- **User Error Protection**: il sistema presenta procedure per proteggere gli utenti dal commettere errori.

3.2.3.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v4.0.0*, sono le seguenti:

- *Average Learning Time*.

3.2.4 Reliability

Caratteristica che esprime il grado con cui un prodotto esegue correttamente determinate funzioni mentre è in uso.

3.2.4.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Maturity**: evitare il più possibile malfunzionamenti in caso di $fault_g$;
- **Fault Tolerance**: se si verificano errori, vengono attivate procedure di gestione dell'errore in modo da non influenzare le prestazioni.

3.2.4.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v4.0.0*, sono le seguenti:

- *Failure Density*;
- *Operazioni con gestione errori*.

3.2.5 Maintainability

Caratteristica che esprime il grado di efficacia ed efficienza con cui il prodotto può essere modificato, tramite azioni di correzione, o aggiornato, tramite azioni di miglioramento.

3.2.5.1 Obiettivi

Gli obiettivi da rispettare durante lo sviluppo del progetto sono i seguenti:

- **Modularity:**: il sistema presenta componenti tali che una modifica ad uno di essi ha il minimo impatto su tutte le altre componenti;
- **Analyzability**: deve essere possibile analizzare l'impatto nel sistema di uno specifico cambiamento ad una o più delle sue parti, ai fini di risalire rapidamente eventuali cause che hanno generato un malfunzionamento;
- **Modifiability**: deve essere possibile eseguire cambiamenti di componenti originali senza introdurre possibili errori;
- **Testability**: deve essere possibile testare il sistema per validare le varie modifiche e per valutare la qualità del sistema.

3.2.5.2 Metriche

Le metriche utilizzate, definite nel documento *Norme di Progetto v4.0.0*, sono le seguenti:

- *Failure Analysis*;
- *Comment Ratio*.

3.3 Riassunto delle metriche di prodotto

Nome Metrica	Intervallo limite	Range accettabile	Range ottimale
Gunning fog index	/	12-15	0-12
Indice di Gulpease	0%-100%	40%-100%	60%-100%
Numero di errori grammaticali	/	0	0
Functional Implementation Completeness	0%-100%	75%-100%	100%
Average Functional Implementation Correctness	0%-100%	80%-100%	95%-100%
Tempo di risposta	/	0-8 sec	0-3 sec
Average Learning Time	/	0-30	0-15
Failure Density	0%-100%	0%-10%	0%
Operazioni con gestione errori	0%-100%	80%-100%	95%-100%
Failure Analysis	0%-100%	60%-100%	80%-100%
Comment Ratio	0%-100%	60%-100%	80%-100%

Tabella 3: Riassunto delle metriche di prodotto

4 Specifica dei test

4.1 Test di accettazione

Test	Requisito	Stato
TA-1	R0F1	NI
TA-1.1	R1F1.1	NI
TA-2	R0F2	NI
TA-2.1	R0F2.1	NI
TA-2.2	R0F2.2	NI
TA-2.3	R1F2.3	NI

Test	Requisito	Stato
TA-3	R0F3	NI
TA-3.1	R1F3.1	NI
TA-4	R0F4	NI
TA-4.1	R1F4.1	NI
TA-5	R0F5	NI
TA-5.1	R1F5.1	NI
TA-5.2	R1F5.2	NI
TA-5.3	R1F5.3	NI
TA-5.4	R1F5.4	NI
TA-5	R1F5.5	NI
TA-5.6	R1F5.6	NI
TA-5.7	R1F5.7	NI
TA-6	R1F6	NI
TA-6.1	R1F6.1	NI
TA-6.2	R1F6.2	NI
TA-7	R2F8	NI
TA-8	R2F10	NI
TA-8.1	R2F10.1	NI
TA-8.2	R2F10.2	NI
TA-8.3	R2F10.4	NI
TA-8.4	R2F10.5	NI
TA-8.5	R2F10.6	NI

Tabella 4: Riassunto test di accettazione

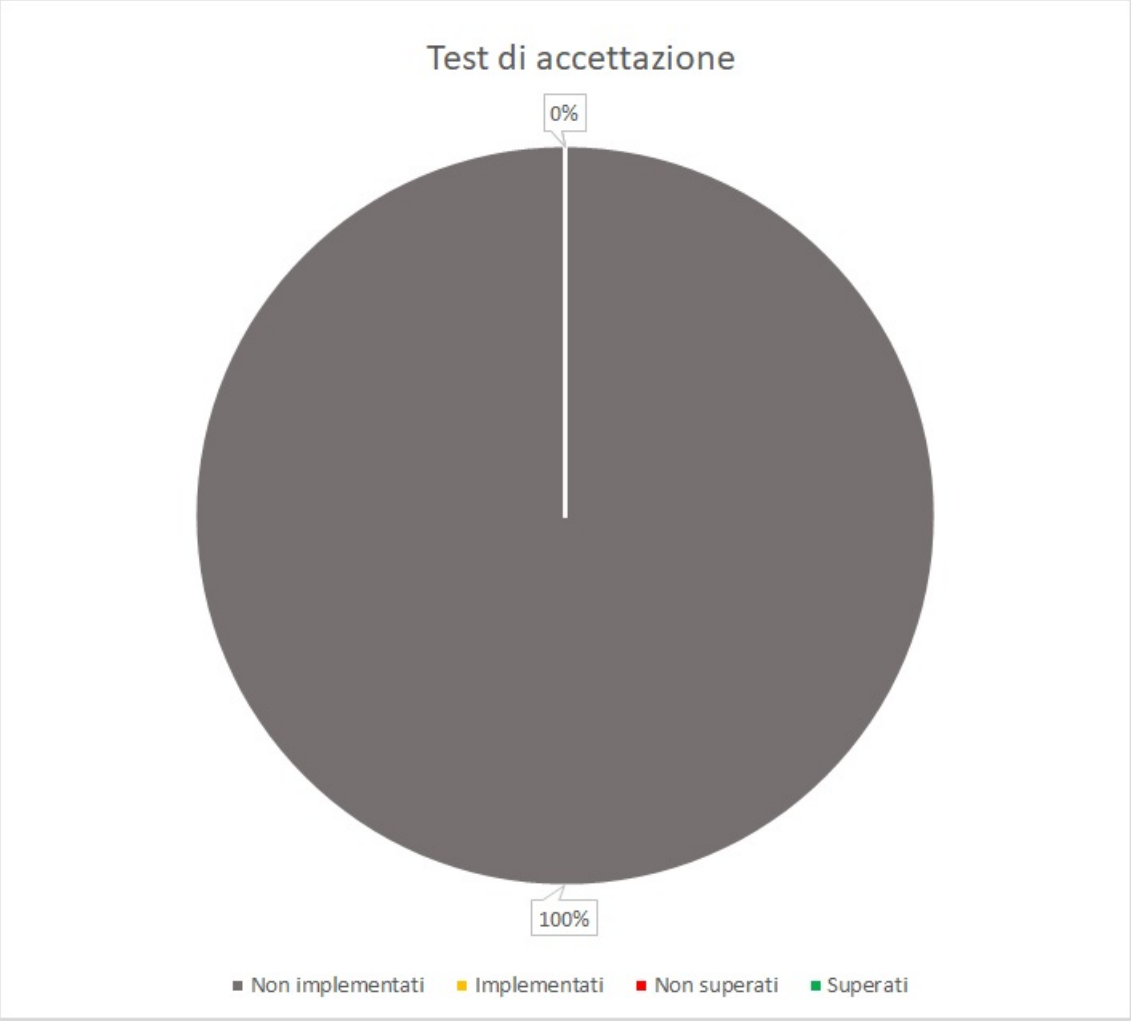

Figura 1: Resoconto test di accettazione

4.2 Test di sistema

Test	Requisito	Stato
TS-1	R0F1	S
TS-1.1	R1F1.1	S
TS-2	R0F2	S
TS-2.1	R0F2.1	S
TS-2.2	R0F2.2	S
TS-2.3	R1F2.3	S
TS-3	R0F3	S
TS-3.1	R1F3.1	S
TS-4	R0F4	S
TS-4.1	R1F4.1	S
TS-5	R0F5	S
TS-5.1	R1F5.1	S
TS-5.2	R1F5.2	S
TS-5.3	R1F5.3	S
TS-5.4	R1F5.4	S
TS-5.5	R1F5.5	S
TS-5.6	R1F5.6	S
TS-5.6.1	R1F5.6.1	S
TS-5.6.2	R1F5.6.2	S
TS-5.6.3	R1F5.6.3	S
TS-5.7	R1F5.7	S
TS-5.7.1	R1F5.7.1	S
TS-6	R1F6	S
TS-6.1	R1F6.1	S
TS-6.1.1	R1F6.1.1	S
TS-6.1.2	R1F6.1.2	S
TS-6.1.3	R1F6.1.3	S
TS-6.2	R1F6.2	S
TS-7	R1F7	NI
TS-8	R2F8	NI
TS-9	R2F9	NI
TS-10	R2F10	S
TS-10.1	R2F10.1	S
TS-10.2	R2F10.2	S
TS-10.3	R2F10.3	S
TS-10.3.1	R2F10.3.1	S
TS-10.3.2	R2F10.3.2	S
TS-10.3.3	R2F10.3.3	S
TS-10.3.4	R2F10.3.4	S
TS-10.3.5	R2F10.3.5	S

Test	Requisito	Stato
TS-10.4	R2F10.4	**S**
TS-10.4.1	R2F10.4.1	**S**
TS-10.4.2	R2F10.4.2	**S**
TS-10.4.3	R2F10.4.3	**S**
TS-10.4.3.1	R2F10.4.3.1	**S**
TS-10.4.3.2	R2F10.4.3.2	**S**
TS-10.4.3.3	R2F10.4.3.3	**S**
TS-10.5	R2F10.5	**S**
TS-10.6	R2F10.6	**S**

Tabella 5: Riassunto test di sistema

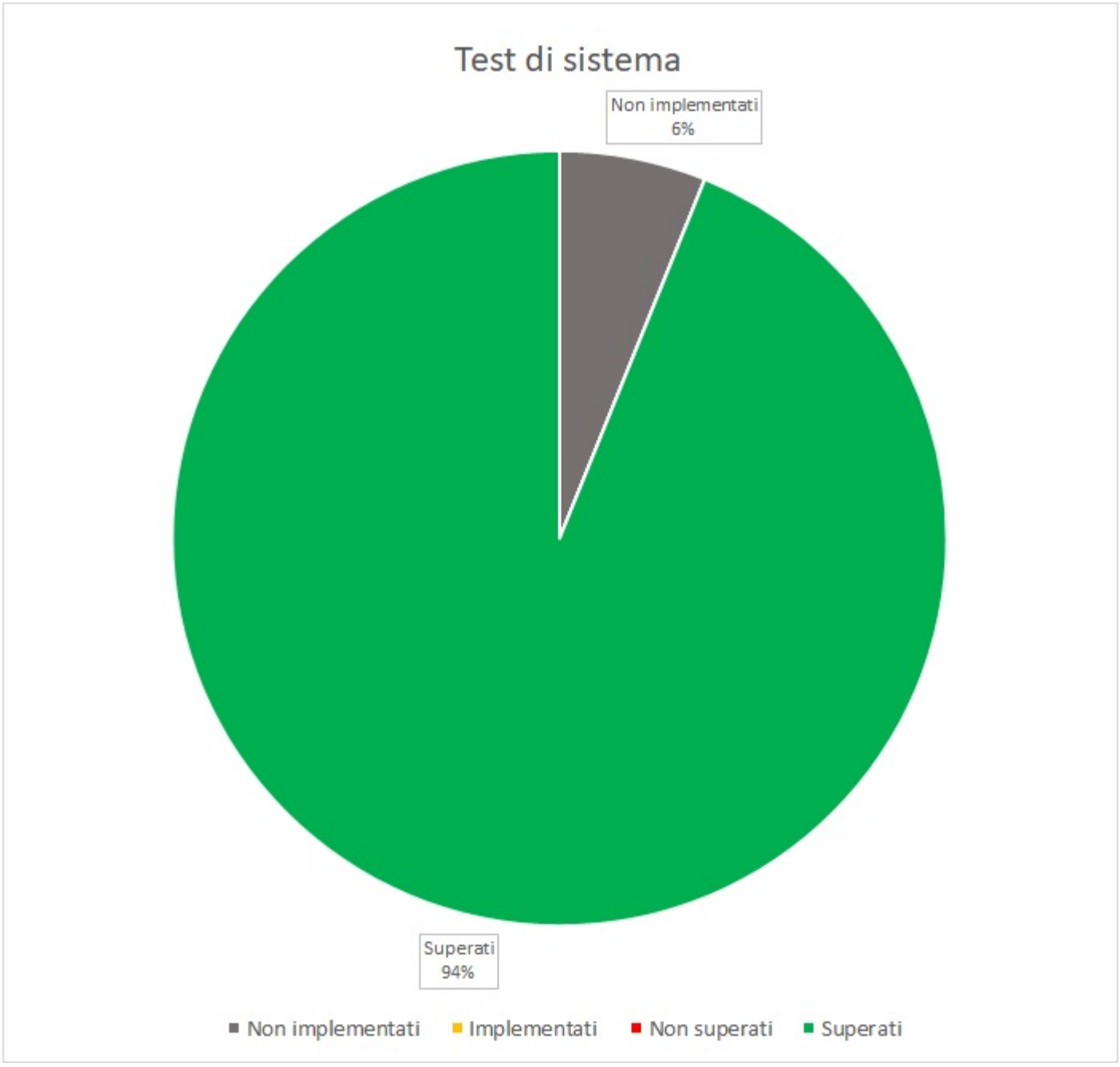

Figura 2: Resoconto test di sistema

4.3 Test di integrazione

Test	Componenti	Stato
TI-1	NetReader NetUpdater.	S
TI-2	NetUpdater NetWriter	S
TI-3	NetReader NetUpdater NetWriter	S
TI-4	NetworkAdapter NetUpdater	S

Tabella 6: Riassunto test di integrazione

Figura 3: Resoconto test di integrazione

4.4 Test di unità

Test	Stato
TU-1	S
TU-2	S
TU-3	S
TU-4	S
TU-5	S
TU-6	S
TU-7	S
TU-8	S
TU-9	S
TU-10	S
TU-11	S
TU-12	S
TU-13	S
TU-14	S
TU-15	S
TU-16	S
TU-17	S
TU-18	S
TU-19	S
TU-20	S
TU-21	S
TU-22	S
TU-23	S
TU-24	S
TU-25	S
TU-26	S
TU-27	S
TU-28	S
TU-29	S
TU-30	S
TU-31	S
TU-32	S
TU-33	S
TU-34	S
TU-35	S
TU-36	S
TU-37	S
TU-38	S
TU-39	S
TU-40	S

Test	Stato
TU-41	S
TU-42	S
TU-43	S
TU-44	S
TU-45	S
TU-46	S
TU-47	S
TU-48	S
TU-49	S
TU-50	S
TU-51	S
TU-52	S
TU-53	S
TU-54	S
TU-55	S
TU-56	S
TU-57	S
TU-58	S
TU-59	S
TU-60	S
TU-61	S
TU-62	S
TU-63	S
TU-64	S
TU-65	S
TU-66	S
TU-67	S
TU-68	S
TU-69	S
TU-70	S
TU-71	S
TU-72	S
TU-73	S
TU-74	S
TU-75	S
TU-76	S
TU-77	S
TU-78	S
TU-79	S
TU-80	S
TU-81	S
TU-82	S

Test	Stato
TU-83	S
TU-84	S
TU-85	S
TU-86	S
TU-87	S
TU-88	S
TU-89	S
TU-90	S
TU-91	S
TU-92	S
TU-93	S
TU-94	S
TU-95	S
TU-96	S
TU-97	S
TU-98	S
TU-99	S
TU-100	S
TU-101	S
TU-102	S
TU-103	S
TU-104	S
TU-105	S
TU-106	S
TU-107	S
TU-108	S
TU-109	S
TU-110	S
TU-111	S
TU-112	S
TU-113	S
TU-114	S
TU-115	S
TU-116	S
TU-117	S
TU-118	S
TU-119	S
TU-120	S
TU-121	S
TU-122	S
TU-123	S
TU-124	S

Test	Stato
TU-125	S
TU-126	S
TU-127	S
TU-128	S
TU-129	S
TU-130	S
TU-131	S
TU-132	S
TU-133	S
TU-134	S
TU-133	S
TU-135	S
TU-136	S
TU-137	S
TU-138	S
TU-139	S
TU-140	S
TU-141	S
TU-142	S
TU-143	S
TU-144	S
TU-145	S
TU-146	S
TU-147	S
TU-148	S
TU-149	S
TU-150	S
TU-151	S
TU-152	S
TU-153	S
TU-154	S
TU-155	S
TU-156	S
TU-157	S
TU-158	S

Tabella 7: Riassunto test di unità

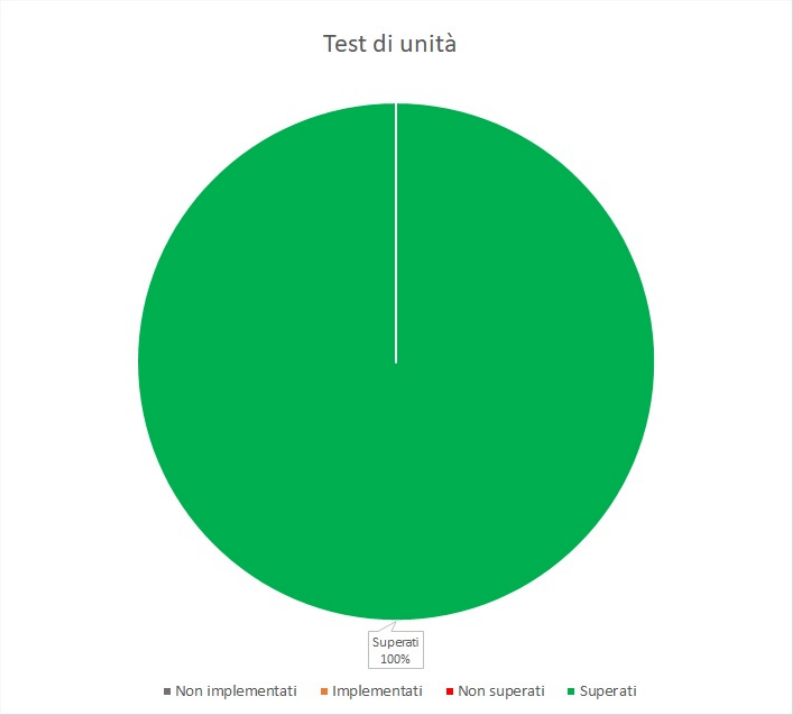

Figura 4: Resoconto test di unità

A Copertura dei requisiti

Requisito	Stato
R0F1	Implementato
R1F1.1	Implementato
R0F2	Implementato
R0F2.1	Implementato
R0F2.2	Implementato
R1F2.3	Implementato
R0F3	Implementato
R1F3.1	Implementato
R0F4	Implementato
R1F4.1	Implementato
R0F5	Implementato
R1F5.1	Implementato
R1F5.2	Implementato
R1F5.3	Implementato
R1F5.4	Implementato
R1F5.5	Implementato
R1F5.6	Implementato
R1F5.6.1	Implementato
R1F5.6.2	Implementato
R1F5.6.3	Implementato
R1F5.7	Implementato
R1F5.7.1	Implementato
R1F6	Implementato
R1F6.1	Implementato
R1F6.1.1	Implementato
R1F6.1.2	Implementato

Requisito	Stato
R1F6.1.3	Implementato
R1F6.2	Implementato
R1F7	Non implementato
R1F7.1	Non implementato
R1F7.1.1	Non implementato
R1F7.1.2	Non implementato
R1F7.1.3	Non implementato
R1F7.1.4	Non implementato
R1F7.1.4.1	Non implementato
R1F7.1.4.2	Non implementato
R1F7.1.4.3	Non implementato
R1F7.2	Non implementato
R1F7.2.1	Non implementato
R1F7.2.2	Non implementato
R1F7.2.2.1	Non implementato
R1F7.2.2.2	Non implementato
R1F7.2.2.3	Non implementato
R1F7.2.2.3.1	Non implementato
R1F7.2.2.3.2	Non implementato
R1F7.2.2.4	Non implementato
R1F7.3	Non implementato
R1F7.4	Non implementato
R1F7.4.1	Non implementato
R1F7.4.2	Non implementato
R1F7.5	Non implementato
R1F7.6	Non implementato
R1F7.6.1	Non implementato
R1F7.6.2	Non implementato

Requisito	Stato
R1F7.7	Non implementato
R1F7.7.1	Non implementato
R1F7.7.2	Non implementato
R1F7.7.3	Non implementato
R1F7.7.4	Non implementato
R2F8	Non implementato
R2F9	Non implementato
R2F10	Implementato
R2F10.1	Implementato
R2F10.2	Implementato
R2F10.3	Implementato
R2F10.3.1	Implementato
R2F10.3.2	Implementato
R2F10.3.3	Implementato
R2F10.3.4	Implementato
R2F10.3.5	Implementato
R2F10.4	Implementato
R2F10.4.1	Implementato
R2F10.4.2	Implementato
R2F10.4.3	Implementato
R2F10.4.3.1	Implementato
R2F10.4.3.2	Implementato
R2F10.4.3.3	Implementato
R2F10.5	Implementato
R2F10.6	Implementato

Tabella 8: Copertura requisiti funzionali

A.1 Resoconto copertura dei requisiti

- **Requisiti obbligatori**

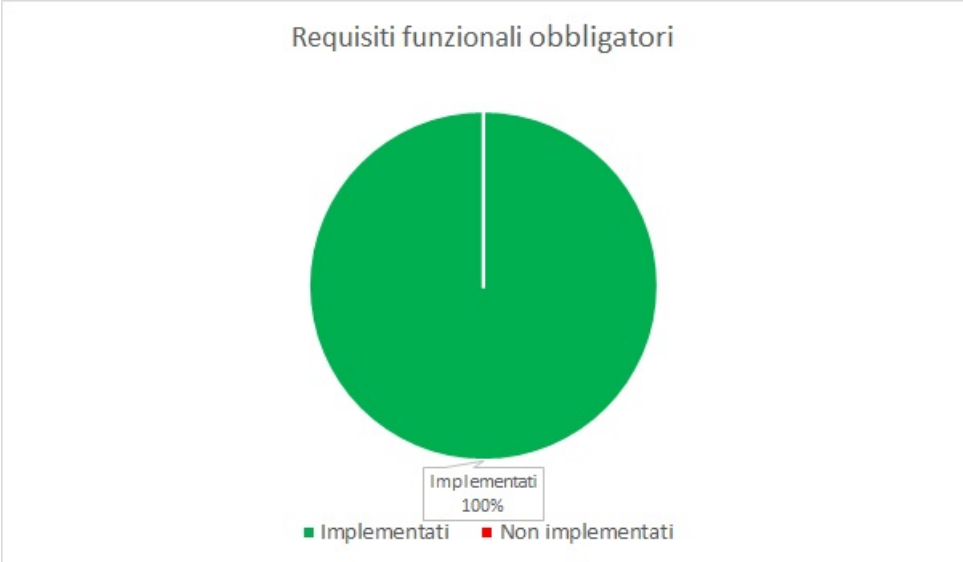

Figura 5: Resoconto requisiti funzionali obbligatori

- **Requisiti desiderabili**

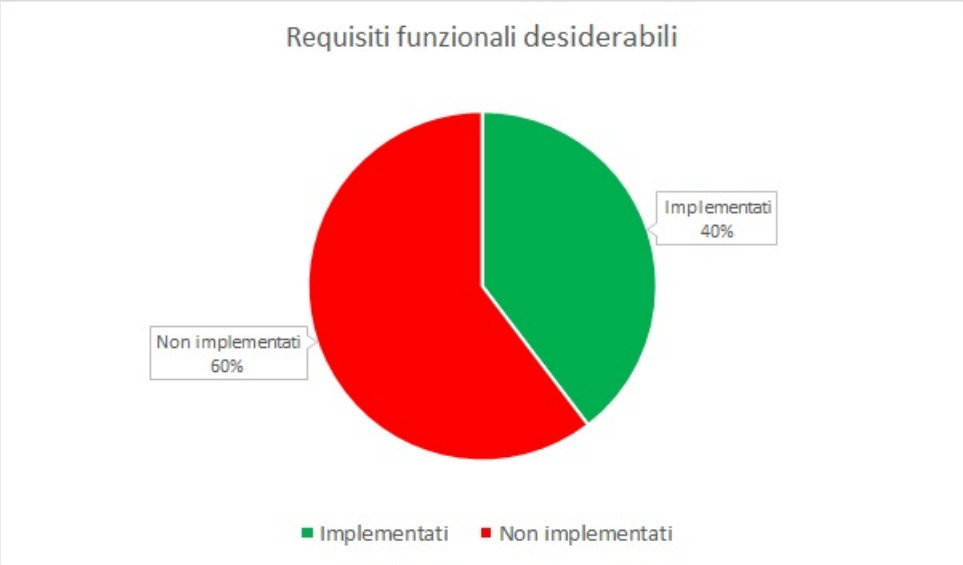

Figura 6: Resoconto requisiti funzionali desiderabili

- **Requisiti opzionali**

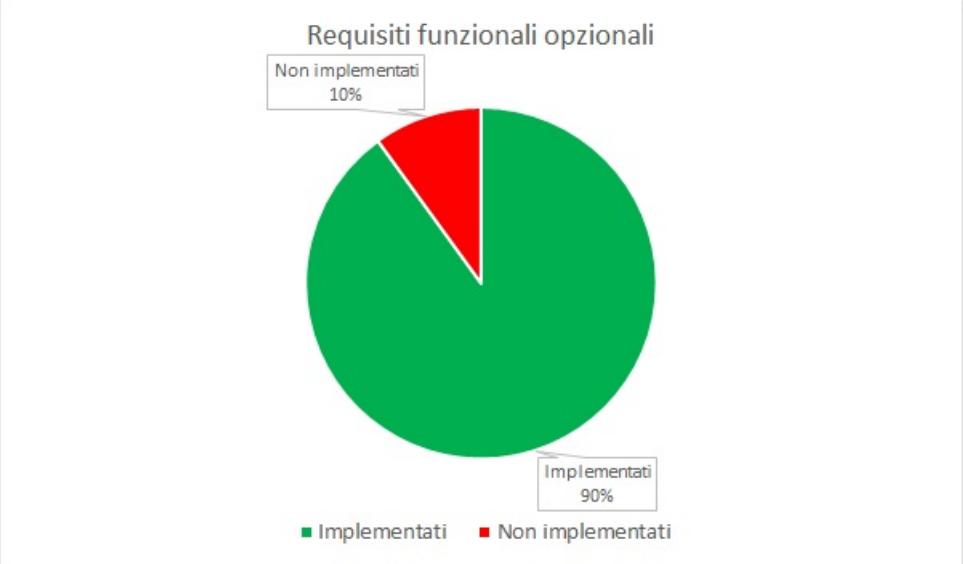

Figura 7: Resoconto requisiti funzionali opzionali

B Resoconto delle attività di verifica

B.1 Revisioni

Questa sezione riporta il resoconto delle attività di verifica svolte prima di ciascuna delle quattro revisioni stabilite dal committente (Revisione dei Requisiti, R. di Progettazione, R. di Qualifica e R. di Accettazione). Al termine di ogni revisione il committente segnalerà le problematiche riscontrate attraverso una valutazione globale dell'andamento del progetto ed una dettagliata per ciascun documento, permettendo al gruppo di eliminare problemi e criticità nel progetto per poi procedere su una base verificata e il più possibile corretta.

B.1.1 Revisione dei Requisiti

- *Norme di Progetto*: sono state effettuate le integrazioni richieste ed è stata aggiornata la struttura del documento in modo da rispettare gli stessi standard per ogni sezione; La sottosezione relativa alla verifica è stata aggiornata in modo da contenere le nuove metriche inserite e le metriche erroneamente inserite nel *Piano di Qualità v1.0.0*;
- *Analisi dei Requisiti*: sono state attuate le opportune modifiche suggerite. Alcuni casi d'uso sono stati leggermente rivisti ed è stato introdotto un caso d'uso più generico per ogni agglomerato di azioni con attitudini simili, modificando il caso d'uso principale;
- *Piano di Progetto*: sono state fatte alcune verifiche nell'uso di alcuni termini e standard come consigliato dal committente. Sono state aggiunte le opportune motivazioni e spiegazioni nelle scelte effettuate. La presentazione dei contenuti è stata rivista in modo da renderla più efficace;
- *Piano di Qualifica*: il documento è stato profondamente rivisto per struttura e contenuti secondo quanto specificato dal committente. Le specifiche degli standard utilizzati e le descrizioni delle metriche sono stati spostati in appendice alle *Norme di Progetto*. È stata aggiunta una sezione riguardante la pianificazione dei test, e le specifiche dei test sono state previste come futura aggiunta in appendice al documento. Infine, la strategia generale per la verifica ha assunto un ruolo centrale nella specifica degli obiettivi di qualità di processo e prodotto.

B.1.2 Revisione di Progettazione

L'attività di verifica è stata svolta per ogni documento, con particolare attenzione alle modifiche e agli incrementi che sono stati effettuati. Sono state eseguite le misurazioni per alcune delle metriche previste.

B.1.2.1 Consolidamento dei requisiti

Processo	Livello di maturità	Considerazioni
Pianificazione e controllo	3	Gestito con l'ausilio di nTask, ha permesso di rispettare le scadenze previste.
Gestione dei rischi	1	Non si sono verificate situazioni di rischio.
Gestione dei test	0	Verrà istanziato una volta presente la necessità di definire e condurre dei test.
Versionamento e build	0	Verrà istanziato una volta presente la necessità di versionare il prodotto software.

Tabella 9: Maturità dei processi e considerazioni

B.1.2.2 Progettazione architetturale

Processo	Livello di maturità	Considerazioni
Pianificazione e controllo	3	Ha permesso di rispettare le scadenze previste.
Gestione dei rischi	1	Si è dimostrato poco stabile al verificarsi di 3 situazioni di rischio (malattia di membri del gruppo), comportando una ridefinizione degli obiettivi per il periodo.
Gestione dei test	1	Sono stati pianificati i primi test di sistema, le cui specifiche tuttavia rimangono da definire.
Versionamento e build	1	Parzialmente automatizzato con l'utilizzo dello strumento *Travis*$_g$.

Tabella 10: Maturità dei processi e considerazioni

B.1.3 Revisione di Qualifica

L'attività di verifica è stata svolta per ogni documento, con particolare attenzione alle modifiche e agli incrementi che sono stati effettuati. Sono state raccolte più misurazioni per ogni metrica prevista, in modo da ottenere delle serie storiche per monitorarne l'andamento.

B.1.3.1 Progettazione di dettaglio e codifica

Processo	Livello di maturità	Considerazioni
Pianificazione e controllo	3	Ha permesso di rispettare le scadenze previste.
Gestione dei rischi	3	Si è dimostrato stabile al verificarsi di alcune situazioni di basso rischio, grazie alle istruzione impartite dal *Responsabile di Progetto*.
Gestione dei test	3	Sono stati specificati ed implementati buona parte dei test. La copertura dei test di unità ha registrato esiti molto positivi.
Versionamento e build	4	Automatizzato il processo tramite *Travis*$_g$. Ogni modifica apportata alla repository di GitHub scaturisce una serie di test sul codice, tenendo traccia dell'esito delle varie build e registrando valori utili per le nostre metriche, come il code coverage.

Tabella 11: Maturità dei processi e considerazioni

B.1.4 Revisione di Accettazione

L'attività di verifica è stata svolta per ogni documento, con particolare attenzione alle modifiche e agli incrementi che sono stati effettuati. Sono state raccolte ulteriori misurazioni delle metriche previste, aggiornando così le serie storiche relative.

B.1.4.1 Verifica e validazione

Processo	Livello di maturità	Considerazioni
Pianificazione e controllo	4	Ha permesso di terminare poco prima delle scadenze previste.
Gestione dei rischi	4	Si è dimostrato stabile al verificarsi di alcune situazioni di basso rischio, grazie alle istruzione impartite dal *Responsabile di Progetto* e anche grazie al fatto che eventi simili si fossero già verificati in precedenza, e quindi di facile mitigazione.
Gestione dei test	4	Sono stati implementati ulteriori test, registrando ottimi esiti per quanto riguarda la copertura.
Versionamento e build	4	Automatizzato il processo tramite *Travis*$_g$. Ogni modifica apportata alla repository di GitHub scaturisce una serie di test sul codice, tenendo traccia dell'esito delle varie build e registrando valori utili per le nostre metriche, come il code coverage.

Tabella 12: Maturità dei processi e considerazioni

B.2 Misurazioni

B.2.1 Schedule Variance

Questa metrica ha raggiunto il valore ottimale.

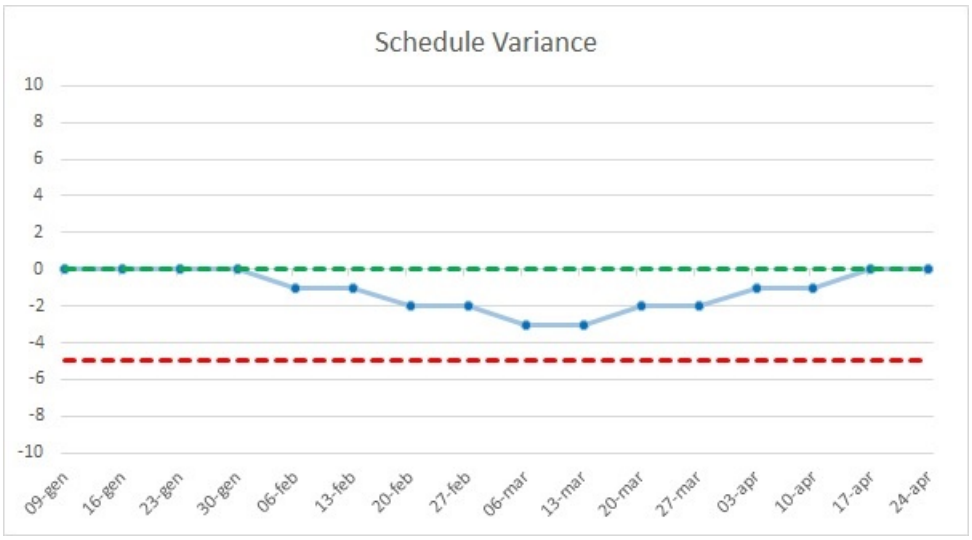

Figura 8: Misurazione Schedule Variance

B.2.2 Budget Variance

Questa metrica è sempre rimasta in linea con il valore ottimale.

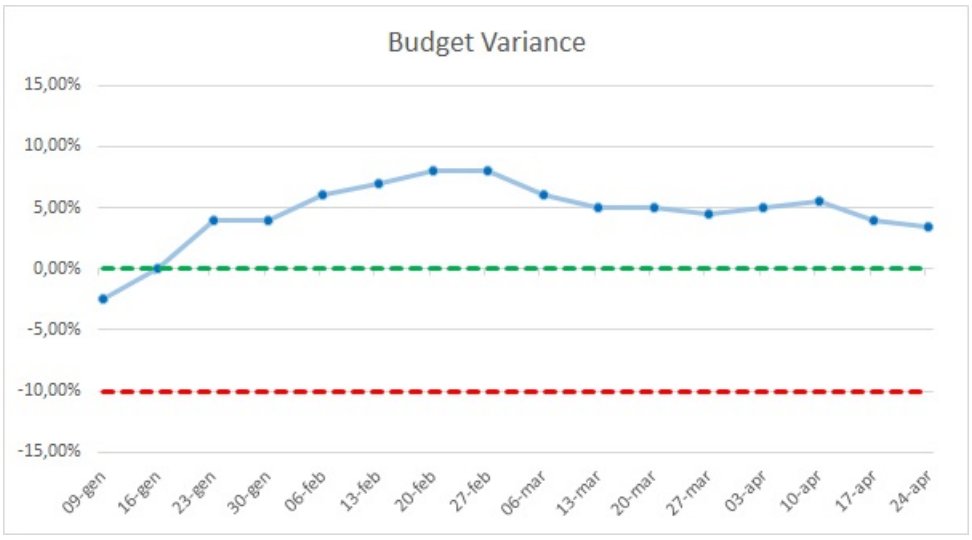

Figura 9: Misurazione Budget Variance

B.2.3 Numero rischi non previsti

Questa metrica è sempre rimasta in linea con il valore accettabile.

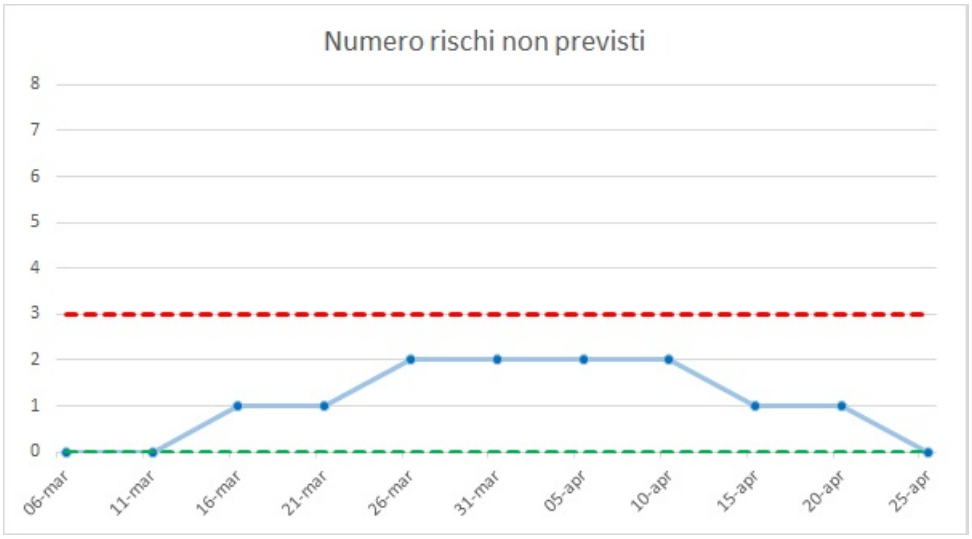

Figura 10: Misurazione numero rischi non previsti

B.2.4 Indisponibilità dei servizi esterni

Questa metrica è sempre rimasta in linea con il valore ottimale, tranne per il verificarsi dell'indisponibilità del servizio *TravisCI*.

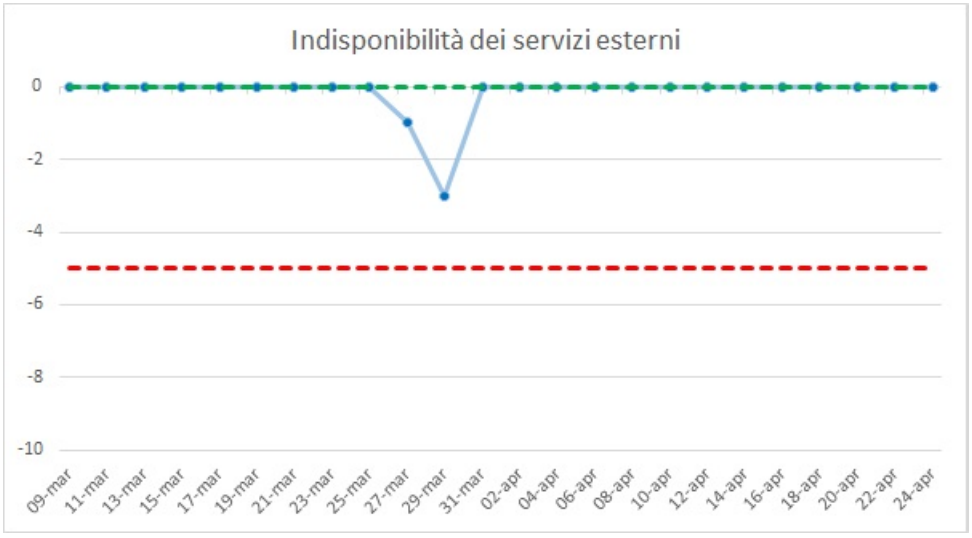

Figura 11: Misurazione indisponibilità dei servizi esterni

B.2.5 Percentuale di test eseguiti

Questa metrica inizialmente ha registrato dei valori non accettabili, ma con lo sviluppo dei test ha raggiunto un valore accettabile.

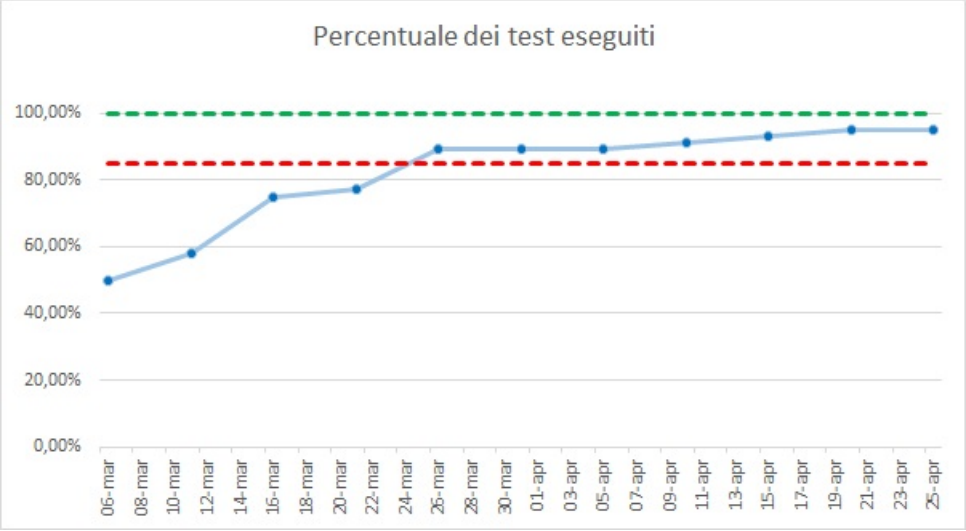

Figura 12: Misurazione percentuale di test eseguiti

B.2.6 Percentuale test case passati

Questa metrica ha quasi raggiunto un valore ottimale.

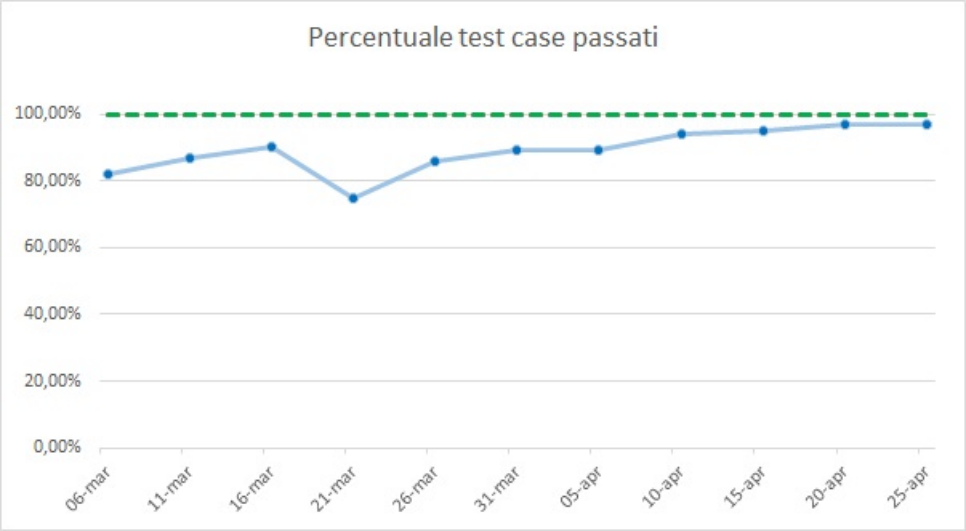

Figura 13: Misurazione percentuale test case passati

B.2.7 Percentuale test case falliti

Questa metrica ha quasi raggiunto un valore ottimale.

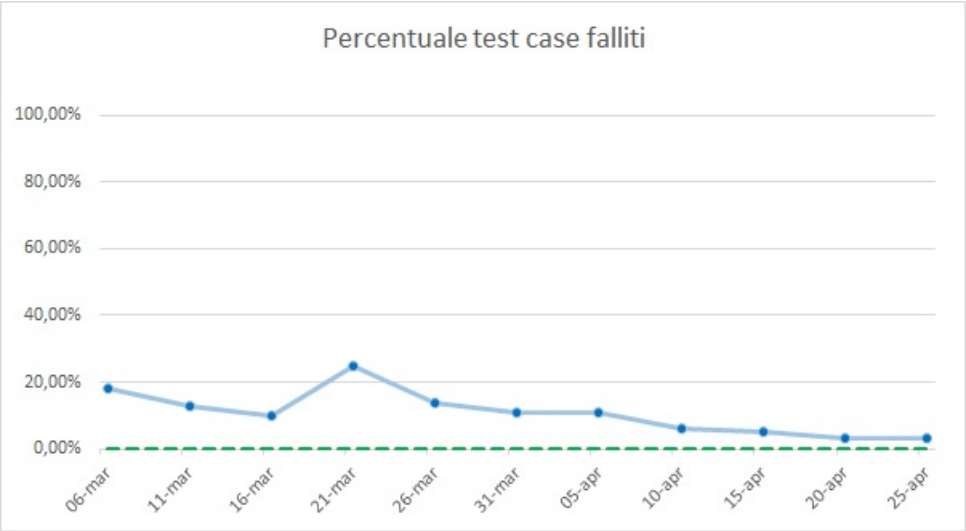

Figura 14: Misurazione percentuale test case falliti

B.2.8 Code coverage

Questa metrica inizialmente ha registrato dei valori non accettabili, ma con lo sviluppo dei test ha raggiunto un valore ottimale.

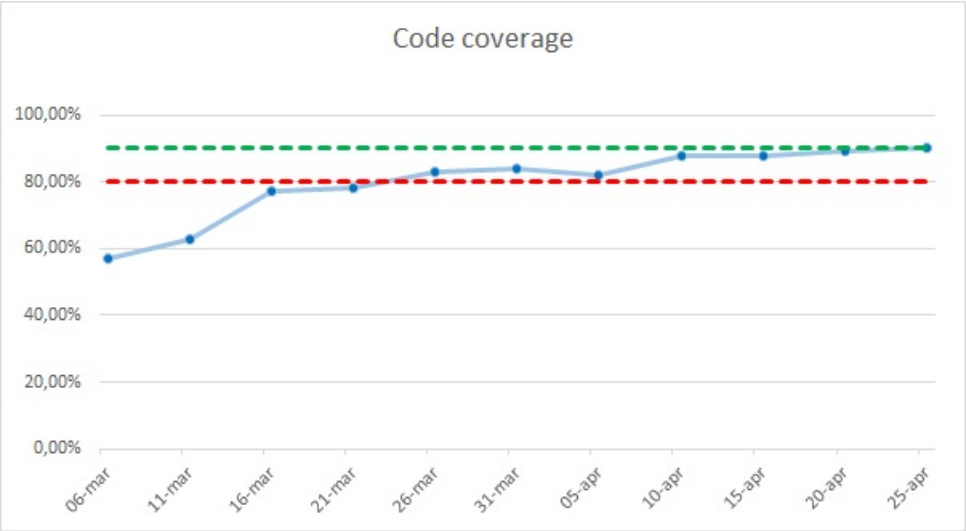

Figura 15: Misurazione code coverage

B.2.9 Tempo medio necessario al team per risolvere un errore

Questa metrica è sempre rimasta in linea con un valore ottimale, tranne in un'occasione causata da un errore consistente.

Figura 16: Misurazione tempo medio necessario per risolvere un errore

B.2.10 Efficienza nella progettazione dei test

Tranne per la fase di sviluppo dei test iniziale, ha sempre registrato un valore ottimale.

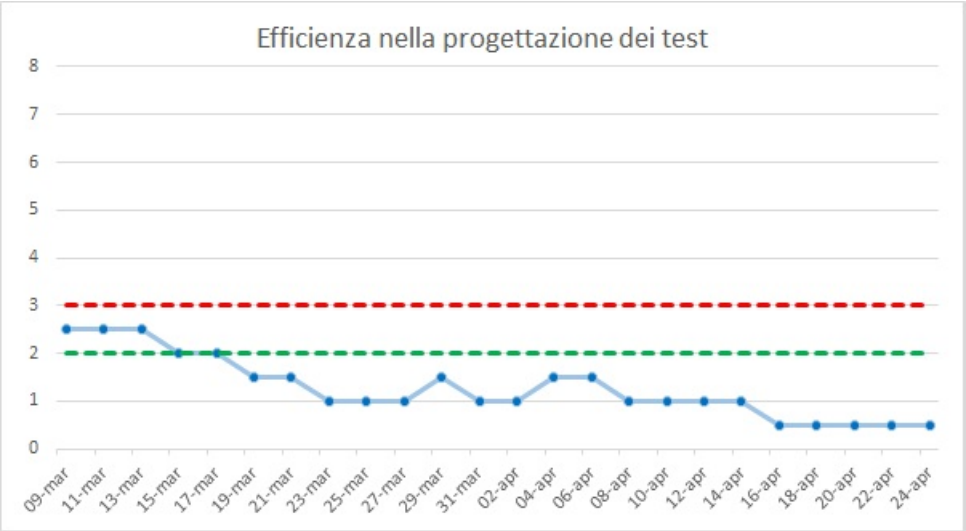

Figura 17: Misurazione efficienza nella progettazione dei test

B.2.11 Percentuale di errori corretti

La presenza di vari errori ha fatto registrare dei valori non accettabili, ma grazie al lavoro dei *Verificatori* questi sono stati corretti, raggiungendo un valore ottimale per questa metrica.

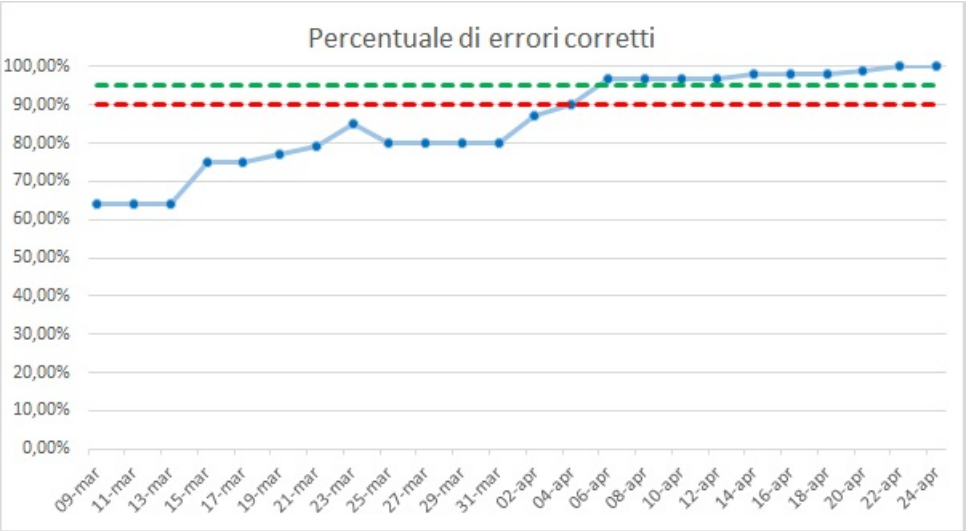

Figura 18: Misurazione percentuale di errori corretti

B.2.12 Media commit a settimana

Il lavoro costante del gruppo ha permesso di registrare dei valori accettabili, a volte anche ottimali, per questa metrica.

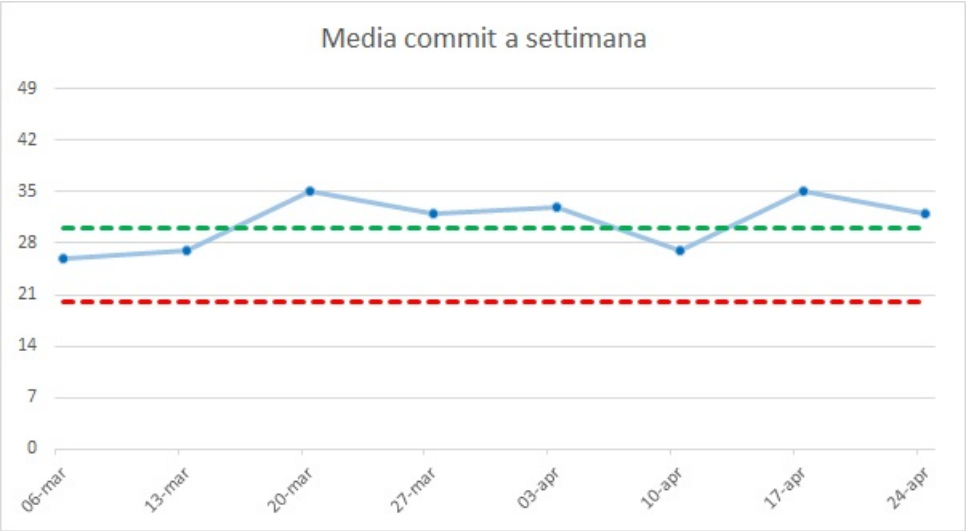

Figura 19: Misurazione media commit a settimana

B.2.13 Media build Travis a settimana

Il lavoro costante del gruppo ha permesso di registrare dei valori accettabili, a volte anche ottimali, per questa metrica.

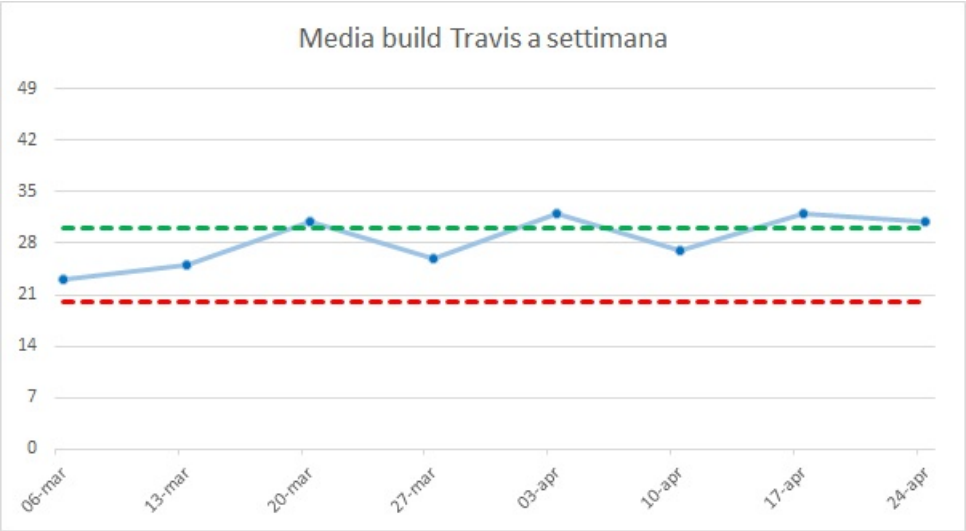

Figura 20: Misurazione media build Travis a settimana

B.2.14 Percentuale build Travis superate

La presenza di vari errori ha fatto registrare dei valori non accettabili inizialmente, ma una volta corrette queste criticità questa metrica ha raggiunto un valore ottimale.

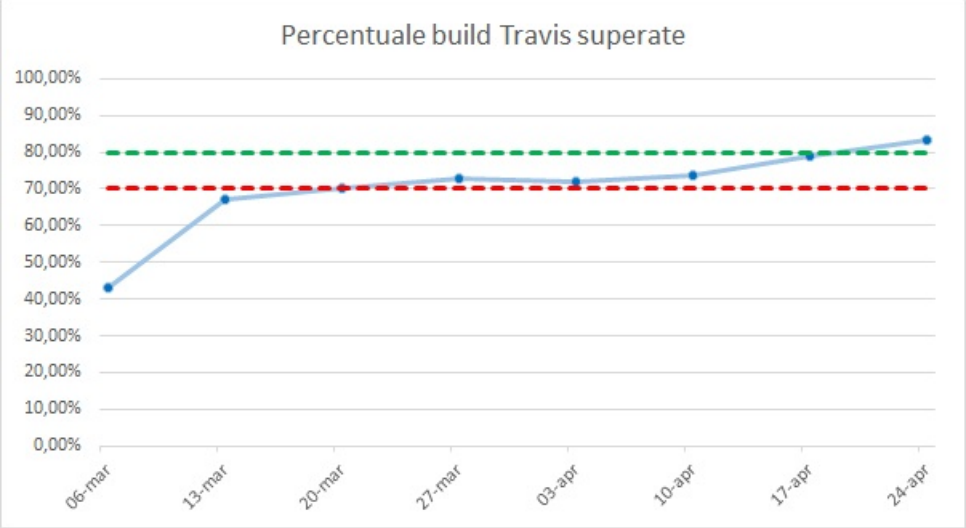

Figura 21: Misurazione percentuale build Travis superate

B.2.15 Gunning fog index

Tutti gli indici Gunning fox dei documenti rientrano nei vincoli dati. Per questo motivo i documenti redatti hanno raggiunto la leggibilità desiderata.

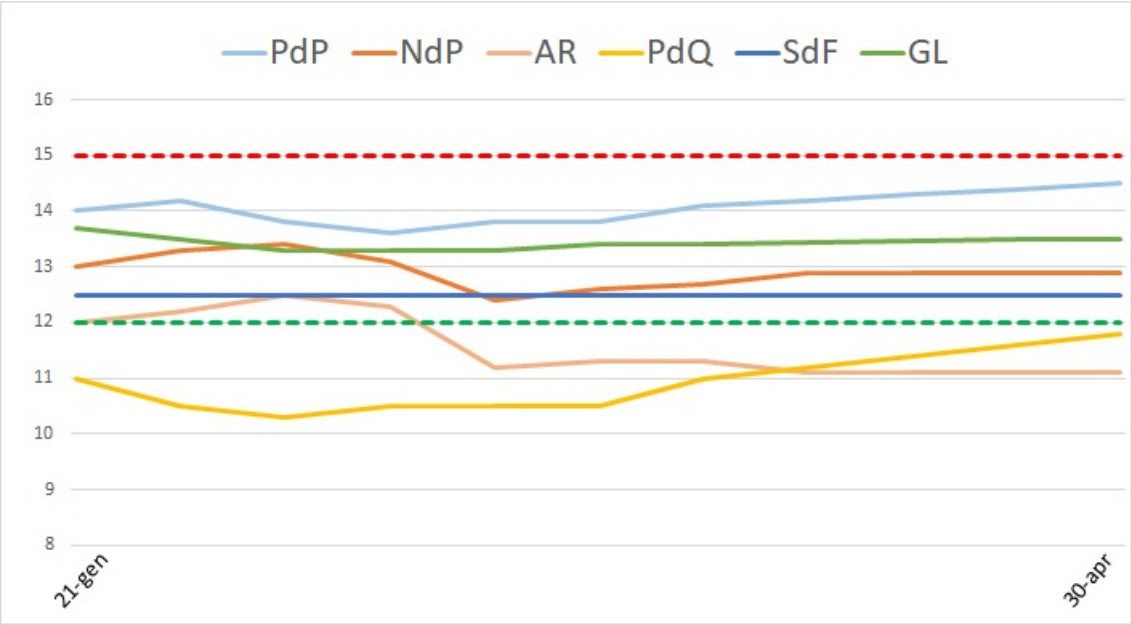

Figura 22: Misurazione Gunning fog index

B.2.16 Indice di Gulpease

Tutti gli indici Gulpease dei documenti rientrano nei vincoli dati. Per questo motivo i documenti redatti hanno raggiunto la leggibilità desiderata.

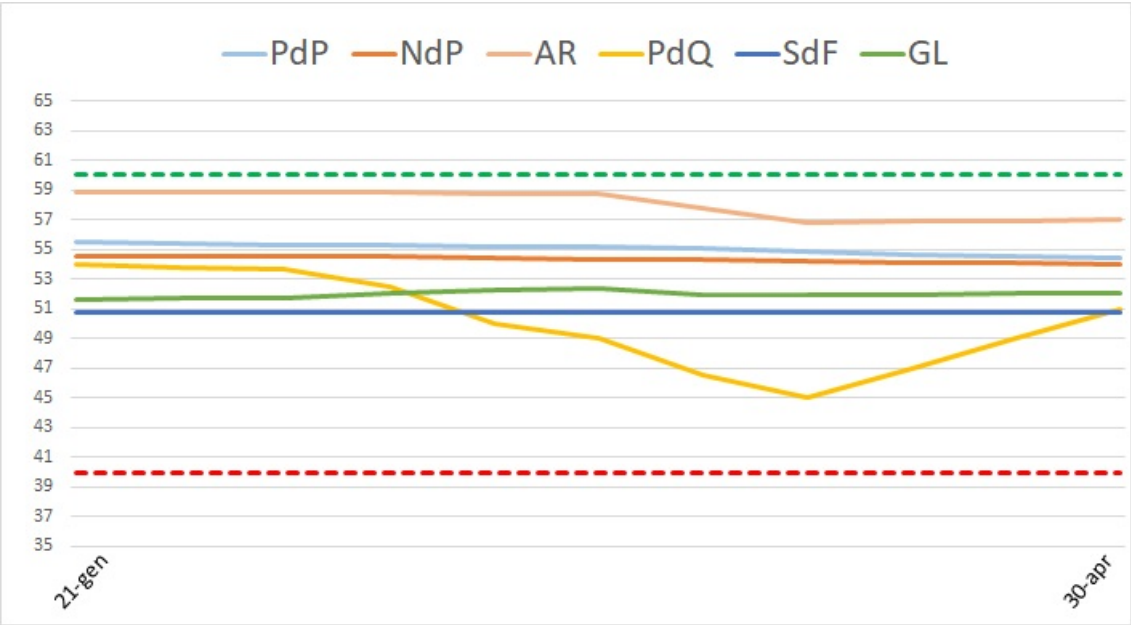

Figura 23: Misurazione indice di Gulpease

B.2.17 Numero di errori grammaticali

Grazie le procedura di verifica e correzione attuate e all'ausilio dello strumento di controllo ortografico integrato in *TexStudio*$_g$, il numero di errori ortografici per ogni documento ha raggiunto il valore ottimale di 0.

Figura 24: Misurazione numero di errori grammaticali

B.2.18 Functional Implementation Completeness

La scarsa copertura delle funzionalità inizialmente ha fatto registrare valori non accettabili, ma grazie al lavoro del team questa metrica ha raggiunto un valore accettabile, anche se non ancora ottimale.

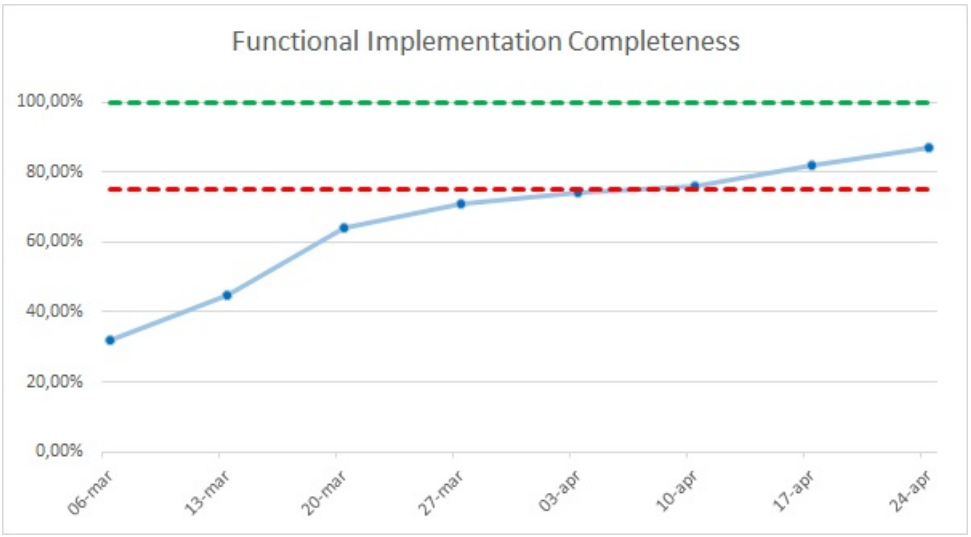

Figura 25: Misurazione Functional Implementation Completeness

B.2.19 Average Functional Implementation Correctness

La presenza di alcune criticità inizialmente ha fatto registrare valori non accettabili, ma grazie al lavoro del team questa metrica ha raggiunto un valore accettabile, anche se non ancora ottimale.

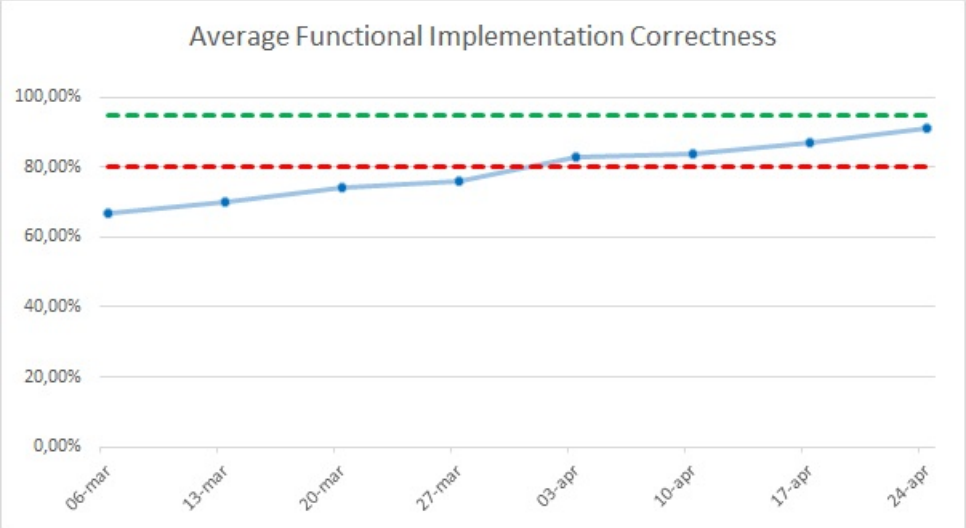

Figura 26: Misurazione Average Functional Implementation Correctness

B.2.20 Tempo di risposta

Il valore di questa metrica è sempre rimasto in linea con il valore ottimale.

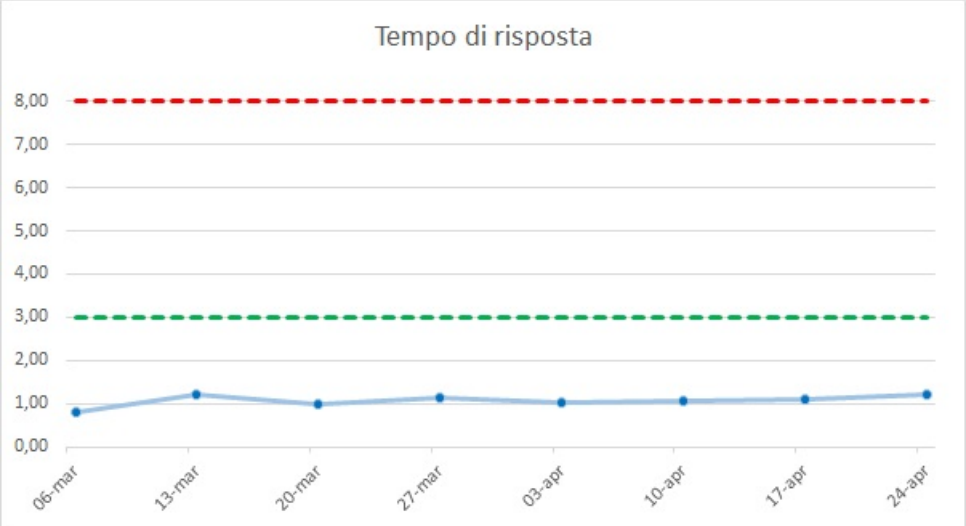

Figura 27: Misurazione tempo di risposta

B.2.21 Average Learning Time

Il valore di questa metrica è sempre rimasto in linea con il valore accettabile.

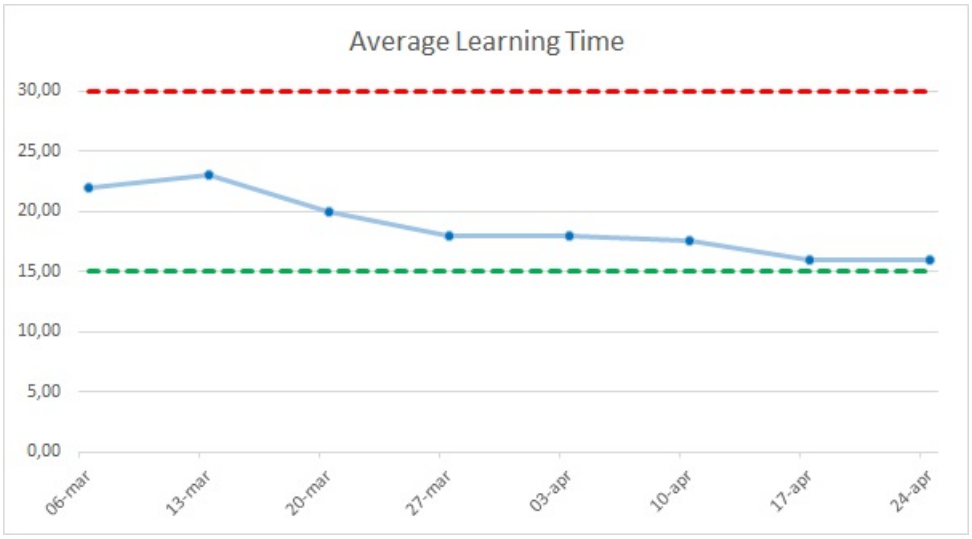

Figura 28: Misurazione Average Learning Time

B.2.22 Failure Density

La presenza di alcune criticità iniziali ha fatto registrare valori non accettabili, ma grazie al lavoro del team questa metrica ha raggiunto un valore accettabile, anche se non ancora ottimale.

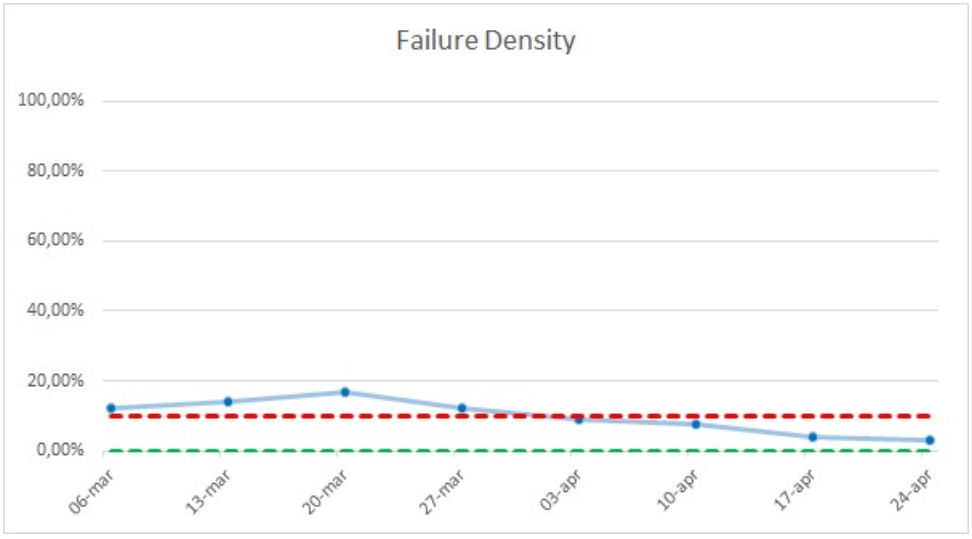

Figura 29: Misurazione Failure Density

B.2.23 Operazioni con gestione errori

Inizialmente abbiamo registrato dei valori non accettabili a causa della scarsa gestione degli errori, ma grazie allo sviluppo dei test siamo riusciti ad ottenere un valore ottimale per questa metrica.

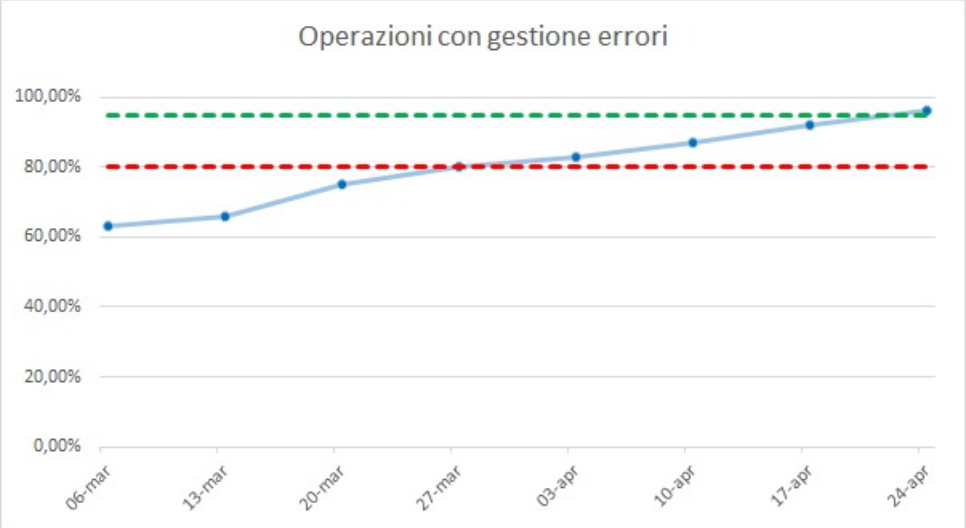

Figura 30: Misurazione

B.2.24 Failure Analysis

Il valore di questa metrica è sempre rimasto in linea con il valore accettabile.

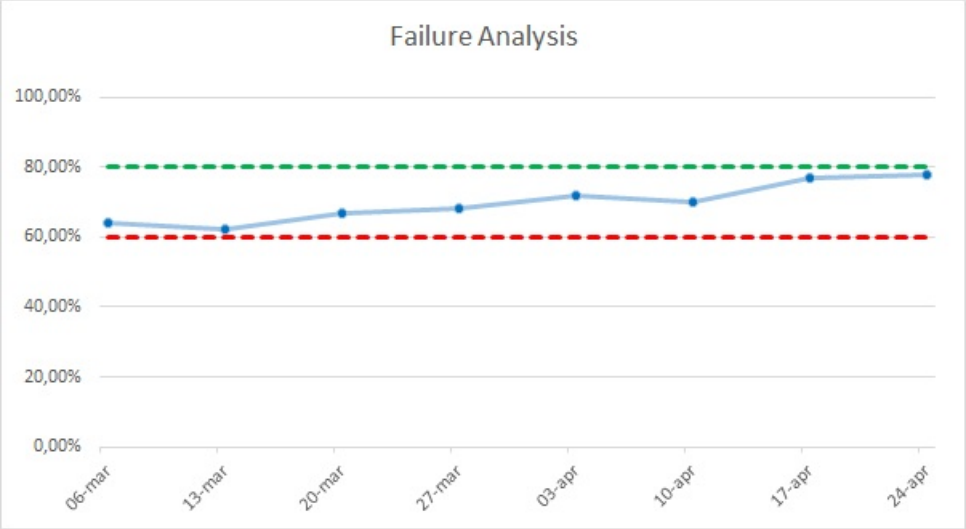

Figura 31: Misurazione Failure Analysis

B.2.25 Comment Ratio

La presenza di pochi commenti nel codice inizialmente ha fatto registrare dei valori non accettabili, ma con il lavoro del team questa metrica ha raggiunto un valore accettabile.

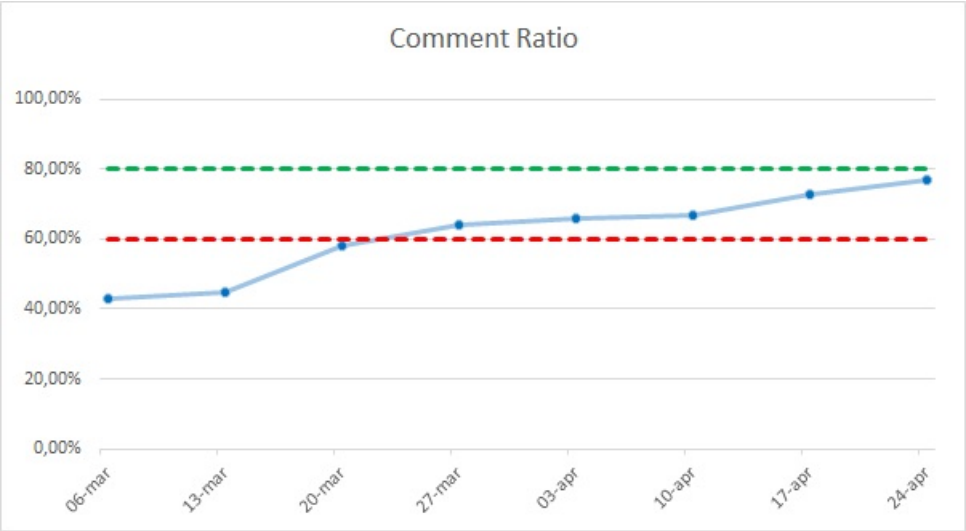

Figura 32: Misurazione Comment Ratio

C Specifica dei test

C.1 Test di accettazione

Test	Specifica
TA-1	• **Progettazione**: verifica che il sistema permetta di leggere la definizione di una rete Bayesiana importando un file in formato JSON; • **Caso**: – **Input**: file JSON con la definizione di rete; – **Risultato atteso**: il sistema importa la definizione della rete mostrandone il contenuto. • **Procedura**: – L'utente crea un nuovo 7DOS panel; – L'utente si posiziona sulla scheda di *"Edit"*; – L'utente si posiziona sul *"Manage network"* tab; – L'utente preme il pulsante per importare un file JSON; – L'utente seleziona il file da importare; – Conferma la selezione del file.
TA-2	• **Progettazione**: verifica che il sistema permetta la gestione della connessione tra i nodi della rete ed il flusso di dati; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema effettua la connessione tra flusso dati e nodi della rete. • **Procedura**: – L'utente si posiziona sulla scheda di *"Edit"*; – L'utente si posiziona sul *"Connect nodes"* tab; – L'utente seleziona il flusso di dati da monitorare.

Test	Specifica
TA-2.1	• **Progettazione**: verifica che il sistema permetta di connettere un nodo della rete al flusso di dati; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema connette il nodo selezionato e lo monitora rispetto al flusso dati. • **Procedura**: – L'utente si posiziona sulla scheda di *"Edit"*; – L'utente si posiziona sul *"Connect nodes"* tab; – L'utente seleziona il nodo da monitorare rispetto al flusso di dati.
TA-2.2	• **Progettazione**: verifica che il sistema permetta di disconnettere un nodo della rete al flusso di dati; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema disconnette il nodo selezionato. • **Procedura**: – L'utente si posiziona sulla scheda di *"Edit"*; – L'utente si posiziona sul *"Connect nodes"* tab; – L'utente deseleziona il nodo da monitorare rispetto al flusso di dati.
TA-2.3	• **Progettazione**: verifica che il sistema permetta di modificare un nodo della rete connesso al flusso di dati; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema apporta le modifiche richieste. • **Procedura**: – L'utente si posiziona sulla scheda di *"Edit"*; – L'utente si posiziona sul *"Connect nodes"* tab; – L'utente modifica il nodo da monitorare rispetto al flusso di dati.

Test	Specifica
TA-3	• **Progettazione**: verifica che il sistema permetta di applicare il ricalcolo delle probabilità della rete secondo regole temporali stabilite dall'utente; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema effettua il ricalcolo delle probabilità della rete aggiornandone lo stato dei nodi. • **Procedura**: – Il sistema legge i dati provenienti dal flusso; – Il sistema li elabora calcolando le probabilità dei nodi; – Il sistema aggiorna lo stato dei nodi.
TA-3.1	• **Progettazione**: verifica che il sistema permetta all'utente di modificare le regole temporali per effettuare il ricalcolo delle probabilità della rete; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema apporta le modifiche richieste alle regole temporali per il ricalcolo delle probabilità. • **Procedura**: – L'utente si posiziona sulla scheda di *"Edit"*; – L'utente si posiziona sul *"Manage monitoring"* tab; – L'utente modifica le regole temporali per il ricalcolo delle probabilità della rete.
TA-4	• **Progettazione**: verifica che il sistema permetta di fornire nuovi dati a Grafana derivati dai nodi della rete non direttamente collegati al flusso di dati; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema elabora i dati proveniente dai nodi non connessi al flusso. • **Procedura**: – Il sistema legge i dati in ingresso dal flusso; – Il sistema elabora i dati; – Il sistema usa i risultati ottenuti per calcolare le probabilità dei nodi non connessi.

Test	Specifica
TA-4.1	• **Progettazione**: verifica che il sistema permetta l'aggiornamento dei dati secondo una frequenza stabilita dall'utente; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema effettua l'aggiornamento costante dei dati. • **Procedura**: – Il sistema, dopo un tempo predefinito, legge nuovi dati in ingresso; – Il sistema elabora i nuovi dati.
TA-5	• **Progettazione**: verifica che il sistema permetta di leggere i dati provenienti dal flusso ed elaborarli, visualizzandone il risultato attraverso un grafico; • **Caso**: – **Input**: flusso di dati; – **Risultato atteso**: grafico derivato dall'elaborazione dei dati. • **Procedura**: – Il sistema legge i dati in ingresso dal flusso; – Il sistema elabora i dati ricevuti; – Il sistema costruisce un grafico sulla base dei risultati ottenuti.
TA-5.1	• **Progettazione**: verifica che il sistema aggiorni i dati presenti nel grafico in base alla frequenza stabilita dall'utente; • **Caso**: – **Input**: nuovi dati dal flusso; – **Risultato atteso**: il sistema aggiorna i dati nel grafico. • **Procedura**: – Il sistema legge i nuovi dati proveniente dal flusso, in base alla frequenza stabilita; – Il sistema effettua il ricalcolo delle probabilità della rete aggiornando lo stato dei nodi, in base alla frequenza stabilita; – Il sistema elabora i risultati e li mostra attraverso un grafico.

Test	Specifica
TA-5.2	• **Progettazione**: verifica che il sistema permetta all'utente di creare un nuovo panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema crea il nuovo panel e lo mostra all'interno della dashboard. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente clicca sulla voce *"Add panel"*; – L'utente sceglie la tipologia di panel da creare.
TA-5.3	• **Progettazione**: verifica che il sistema permetta all'utente di spostare un panel all'interno della dashboard; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema sposta panel nella nuova posizione scelta dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente trascina il panel che vuole spostare nella posizione desiderata.
TA-5.4	• **Progettazione**: verifica che il sistema permetta all'utente di cancellare un panel all'interno della dashboard; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema cancella panel scelto dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente clicca sull'intestazione del panel; – L'utente seleziona la voce *"Remove"*; – Conferma l'eliminazione del panel.

Test	Specifica
TA-5.5	• **Progettazione**: verifica che il sistema permetta all'utente di minimizzare un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema modifica la dimensione del panel come scelto dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente si posiziona sull'angolo in basso a destra del panel; – L'utente modifica minimizza il panel.
TA-5.6	• **Progettazione**: verifica che il sistema permetta all'utente configurare un panel dopo la sua creazione; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema configura le opzioni del panel come stabilito dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente configura le opzioni del panel come desidera.
TA-5.7	• **Progettazione**: verifica che il sistema permetta all'utente di modificare un panel all'interno della dashboard; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema modifica le opzioni del panel come stabilito dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente modifica le opzioni del panel come desidera.

Test	Specifica
TA-6	• **Progettazione**: verifica che il sistema permetta all'utente di definire alert in base a livelli di soglia raggiunti dai nodi non collegati al flusso dei dati; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema crea un nuovo alert. • **Procedura**: – L'utente si posiziona all'interno di un panel di tipo *graph*; – L'utente si posiziona sulla schermata di *edit*; – L'utente si posiziona sulla scheda *"Alert"*; – L'utente clicca sul pulsante *"Create alert"*.
TA-6.1	• **Progettazione**: verifica che il sistema permetta all'utente di configurare i parametri di un alert; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema configura i parametri dell'alert. • **Procedura**: – L'utente si posiziona sulla scheda *"Alert"*; – L'utente seleziona l'alert da configurare; – L'utente configura i parametri come desidera.
TA-6.2	• **Progettazione**: verifica che il sistema permetta all'utente di impostare il modo in cui viene notificata l'attivazione di un alert; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema imposta il sistema di notifica dell'alert. • **Procedura**: – L'utente si posiziona sulla scheda *"Alert"*; – L'utente seleziona l'alert da configurare; – L'utente imposta come deve venire notificata l'attivazione dell'alert.

Test	Specifica
TA-7	• **Progettazione**: verifica che il sistema permetta di applicare più reti Bayesiane a diversi oggetti di monitoraggio; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema avvia il monitoraggio di diversi oggetti. • **Procedura**: – L'utente importa più reti Bayesiane; – L'utente connette le reti a diversi flussi di dati.
TA-8	• **Progettazione**: verifica che il sistema permetta di condividere grafici presenti in una dashboard o in un singolo panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema condivide il grafico secondo l'opzione scelta dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa alla dashboard o ad un panel.
TA-8.1	• **Progettazione**: verifica che il sistema permetta di visualizzare il link diretto ad una dashboard o ad un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema mostra il link generato con cui l'utente può effettuare la condivisione. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa alla dashboard o ad un panel; – L'utente si posiziona sulla scheda *"Link"*; – L'utente configura le opzioni per generare il link per la condivisione.

Test	Specifica
TA-8.2	• **Progettazione**: verifica che il sistema permetta di visualizzare il codice per l'inclusione di un panel in una pagina web; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: Il sistema mostra il frame generato con cui l'utente può effettuare la condivisione. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa ad un panel; – L'utente si posiziona sulla scheda *"Embed"*; – L'utente configura le opzioni per generare il frame per la condivisione.
TA-8.3	• **Progettazione**: verifica che il sistema permetta di condividere lo snapshot di una dashboard o di un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema effettua la condivisione dello snapshot. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda *"Snapshot"*; – L'utente configura le opzioni per la condivisione dello snapshot.
TA-8.4	• **Progettazione**: verifica che il sistema permetta di visualizzare il codice JSON contenente la definizione di una dashboard; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema mostra il codice JSON con la definizione della dashboard. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share dashboard"*; – L'utente si posiziona sulla scheda *"Export"*; – L'utente clicca sul pulsante per visualizzare il JSON.

Test	Specifica
TA-8.5	• **Progettazione**: verifica che il sistema permetta di salvare il file JSON contenente la definizione di una dashboard; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema effettua il salvataggio del codice JSON con la definizione della dashboard. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share dashboard"*; – L'utente si posiziona sulla scheda *"Export"*; – L'utente clicca sul pulsante per salvare il JSON.

Tabella 13: Specifica test di accettazione

C.2 Test di sistema

Test	Specifica
TS-1	• **Progettazione**: verifica che sia possibile e leggere la definizione della rete Bayesiana da un file in formato JSON; • **Caso**: – **Input**: file JSON con la definizione di una rete Bayesiana; – **Risultato atteso**: il sistema legge il JSON e costruisce la rete. • **Procedura**: – L'utente preme il pulsante per importare un file; – L'utente seleziona il file; – L'utente conferma la selezione.
TS-1.1	• **Progettazione**: verifica che sia possibile validare un file JSON; • **Caso**: – **Input**: file JSON; – **Risultato atteso**: il sistema mostra degli errori se il file non è valido. • **Procedura**: – L'utente seleziona un JSON da importare; – L'utente conferma la selezione; – Il sistema verifica la validità del file.
TS-2	• **Progettazione**: verifica che sia possibile gestire la connessione tra i nodi della rete ai rispettivi flussi di dati; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema effettua la connessione tra flusso dati e nodi della rete. • **Procedura**: – L'utente si posiziona sul *"Network"* panel; – L'utente seleziona il flusso dati da monitorare.

Test	Specifica
TS-2.1	• **Progettazione**: verifica che sia possibile connettere un nodo della rete ad un flusso di dati; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema connette il nodo selezionato al flusso dati. • **Procedura**: – L'utente si posiziona sul *"Network"* panel; – L'utente sceglie quale nodo monitorare rispetto al flusso dati.
TS-2.2	• **Progettazione**: verifica che sia possibile disconnettere un nodo della rete da un flusso di dati.; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema disconnette il nodo selezionato dal flusso dati. • **Procedura**: – L'utente si posiziona sul *"Network"* panel; – L'utente sceglie quale nodo disconnettere rispetto al flusso dati.
TS-2.3	• **Progettazione**: verifica che sia essere possibile modificare il flusso di dati connesso ad un nodo; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema seleziona un altro flusso di dati da associare alla rete. • **Procedura**: – L'utente si posiziona sul *"Network"* panel; – L'utente modifica la fonte del flusso dati.

Test	Specifica
TS-3	• **Progettazione**: verifica che sia possibile applicare il ricalcolo delle probabilità della rete secondo regole temporali prestabilite; • **Caso**: – **Input**: flusso di dati; – **Risultato atteso**: il sistema effettua il ricalcolo delle probabilità della rete aggiornandone lo stato dei nodi. • **Procedura**: – Il sistema legge i dati provenienti dal flusso; – Il sistema li elabora calcolando le probabilità dei nodi; – Il sistema aggiorna lo stato dei nodi.
TS-3.1	• **Progettazione**: verifica che sia possibile modificare le suddette regole temporali; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema apporta le modifiche richieste alle regole temporali per il ricalcolo delle probabilità. • **Procedura**: – L'utente si posiziona sulla scheda di *"Edit"*; – L'utente si posiziona sul *"Network"* panel; – L'utente modifica le regole temporali per il ricalcolo delle probabilità della rete.
TS-4	• **Progettazione**: verifica che sia possibile fornire nuovi dati al sistema di Grafana derivati dai nodi della rete non collegati al flusso di monitoraggio; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema elabora i dati proveniente dai nodi non connessi al flusso. • **Procedura**: – Il sistema legge i dati in ingresso dal flusso; – Il sistema elabora i dati; – Il sistema usa i risultati ottenuti per calcolare le probabilità dei nodi non connessi.

Test	Specifica
TS-4.1	• **Progettazione**: verifica che sia possibile aggiornare i dati in base alla frequenza stabilita; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema effettua l'aggiornamento costante dei dati. • **Procedura**: – Il sistema, dopo un tempo predefinito, legge nuovi dati in ingresso; – Il sistema elabora i nuovi dati.
TS-5	• **Progettazione**: verifica che i dati siano disponibili al sistema di creazione di grafici e dashboard per la loro visualizzazione; • **Caso**: – **Input**: flusso di dati; – **Risultato atteso**: grafico derivato dall'elaborazione dei dati. • **Procedura**: – Il sistema legge i dati in ingresso dal flusso; – Il sistema elabora i dati ricevuti; – Il sistema costruisce un grafico sulla base dei risultati ottenuti.
TS-5.1	• **Progettazione**: verifica che sia possibile aggiornare la dashboard in base alla frequenza stabilita; • **Caso**: – **Input**: nuovi dati dal flusso; – **Risultato atteso**: il sistema aggiorna i dati nel grafico. • **Procedura**: – Il sistema legge i nuovi dati proveniente dal flusso, in base alla frequenza stabilita; – Il sistema effettua il ricalcolo delle probabilità della rete aggiornando lo stato dei nodi, in base alla frequenza stabilita; – Il sistema elabora i risultati e li mostra attraverso un grafico.

Test	Specifica
TS-5.2	• **Progettazione**: verifica che sia possibile creare un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema crea il nuovo panel e lo mostra all'interno della dashboard. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente clicca sulla voce *"Add panel"*; – L'utente sceglie la tipologia di panel da creare.
TS-5.3	• **Progettazione**: verifica che sia possibile spostare un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema sposta panel nella nuova posizione scelta dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente trascina il panel che vuole spostare nella posizione desiderata.
TS-5.4	• **Progettazione**: verifica che sia possibile cancellare un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema cancella panel scelto dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente clicca sull'intestazione del panel; – L'utente seleziona la voce *"Remove"*; – Conferma l'eliminazione del panel.
TS-5.5	• **Progettazione**: verifica che sia possibile minimizzare un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema modifica la dimensione del panel come scelto dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente si posiziona sull'angolo in basso a destra del panel; – L'utente modifica minimizza il panel.

Test	Specifica
TS-5.6	• **Progettazione**: verifica che sia possibile configurare un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema configura le opzioni del panel come stabilito dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente configura le opzioni del panel come desidera.
TS-5.6.1	• **Progettazione**: verifica che sia possibile selezionare un flusso dati; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema associa i dati in ingresso al flusso. • **Procedura**: – L'utente si posiziona sul *Network* panel; – L'utente seleziona la fonte del flusso dati.
TS-5.6.2	• **Progettazione**: verifica che sia possibile selezionare un nodo della rete; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: . • **Procedura**: – L'utente si posiziona sul *Network* panel; – L'utente seleziona il nodo da monitorare.
TS-5.6.3	• **Progettazione**: verifica che sia possibile selezionare un intervallo di tempo; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema mostra nel grafico i dati elaborati in quell'intervallo di tempo. • **Procedura**: – L'utente si posiziona nelle impostazioni della dashboard; – L'utente seleziona l'intervallo di tempo.

Test	Specifica
TS-5.7	• **Progettazione**: verifica che sia possibile modificare un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema modifica le opzioni del panel come stabilito dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente modifica le opzioni del panel come desidera.
TS-5.7.1	• **Progettazione**: verifica che sia possibile usare le modifiche standard di Grafana su un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema apporta le modifiche richieste. • **Procedura**: – L'utente si posiziona nelle impostazioni del panel; – L'utente apporta le modifiche che desidera.
TS-6	• **Progettazione**: verifica che sia possibile definire alert in base a livelli di soglia raggiunti dai nodi non collegati al flusso dei dati; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema crea un nuovo alert. • **Procedura**: – L'utente si posiziona all'interno di un panel di tipo *graph*; – L'utente si posiziona sulla schermata di *edit*; – L'utente si posiziona sulla scheda *"Alert"*; – L'utente clicca sul pulsante *"Create alert"*.
TS-6.1	• **Progettazione**: verifica che sia possibile configurare i parametri di un alert; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema configura i parametri dell'alert. • **Procedura**: – L'utente si posiziona sulla scheda *"Alert"*; – L'utente seleziona l'alert da configurare; – L'utente configura i parametri come desidera.

Test	Specifica
TS-6.1.1	• **Progettazione**: verifica che sia possibile inserire il nome di un alert; • **Caso**: – **Input**: stringa col nome dell'alert; – **Risultato atteso**: il sistema inserisce il nome dell'alert. • **Procedura**: – L'utente si posiziona nelle impostazioni del panel; – L'utente scrive il nome dell'alert. – L'utente conferma il nome scelto.
TS-6.1.2	• **Progettazione**: verifica che sia possibile inserire l'intervallo di verifica di un alert; • **Caso**: – **Input**: numero intero; – **Risultato atteso**: il sistema inserisce il tempo per l'intervallo di verifica. • **Procedura**: – L'utente si posiziona nelle impostazioni del panel ; – L'utente seleziona il tempo per l'intervallo di verifica. – L'utente conferma la selezione.
TS-6.1.3	• **Progettazione**: verifica che sia possibile inserire la condizione di attivazione di un alert; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema inserisce la condizione di attivazione. • **Procedura**: – L'utente si posiziona nelle impostazioni del panel ; – L'utente inserisce la condizione di attivazione. – L'utente conferma l'inserimento.

Test	Specifica
TS-6.2	• **Progettazione**: verifica che sia possibile impostare il modo in cui viene notificata l'attivazione di un alert; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema imposta il sistema di notifica dell'alert. • **Procedura**: – L'utente si posiziona sulla scheda *"Alert"*; – L'utente seleziona l'alert da configurare; – L'utente imposta come deve venire notificata l'attivazione dell'alert.
TS-7	• **Progettazione**: verifica che sia possibile disegnare la rete Bayesiana con un editor grafico specializzato; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: la rete Bayesiana è stata disegnata. • **Procedura**: – L'utente avvia l'editor grafico; – L'utente inizia a disegnare una rete Bayesiana;
TS-8	• **Progettazione**: verifica che sia possibile applicare più reti Bayesiane in oggetti di monitoraggio diversi; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema avvia il monitoraggio di diversi oggetti. • **Procedura**: – L'utente importa più reti Bayesiane; – L'utente connette le reti a diversi flussi di dati.

Test	Specifica
TS-9	• **Progettazione**: verifica che sia possibile creare una rete Bayesiana a partire dai dati raccolti sul campo anziché svilupparla con la collaborazione degli esperti del settore; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema crea une rete Bayesiana. • **Procedura**: – Il sistema raccoglie i dati provenienti dal flusso; – Il sistema li elabora per costruire una rete Bayesiana.
TS-10	• **Progettazione**: verifica che sia possibile condividere un grafico; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema condivide il grafico secondo l'opzione scelta dall'utente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa alla dashboard o ad un panel.
TS-10.1	• **Progettazione**: verifica che sia possibile visualizzare il link diretto ad una dashboard o ad un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema mostra il link generato con cui l'utente può effettuare la condivisione. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa alla dashboard o ad un panel; – L'utente si posiziona sulla scheda *"Link"*; – L'utente configura le opzioni per generare il link per la condivisione.

Test	Specifica
TS-10.2	• **Progettazione**: verifica che sia possibile visualizzare il codice per l'inclusione di un panel in una pagina web; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema mostra il frame generato con cui l'utente può effettuare la condivisione. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa ad un panel; – L'utente si posiziona sulla scheda *"Embed"*; – L'utente configura le opzioni per generare il frame per la condivisione.
TS-10.3	• **Progettazione**: verifica che sia possibile selezionare le opzioni di visualizzazione per la condivisione dei grafici; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema seleziona le opzioni richieste. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa ad un panel o una dashboard; – L'utente configura le opzioni di visualizzazione.
TS-10.3.1	• **Progettazione**: verifica che sia possibile selezionare la visualizzazione dell'intervallo di tempo corrente in un grafico; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema mostra l'intervallo di tempo corrente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa ad un panel o una dashboard; – L'utente seleziona la visualizzazione dell'intervallo di tempo.

Test	Specifica
TS-10.3.2	• **Progettazione**: verifica che sia possibile deselezionare la visualizzazione dell'intervallo di tempo corrente in un grafico; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema nasconde l'intervallo di tempo corrente. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa ad un panel o una dashboard; – L'utente deseleziona la visualizzazione dell'intervallo di tempo.
TS-10.3.3	• **Progettazione**: verifica che sia possibile selezionare la visualizzazione di variabili di template in un grafico; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema mostra le variabili di template. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa ad un panel o una dashboard; – L'utente seleziona la visualizzazione delle variabili di template.
TS-10.3.4	• **Progettazione**: verifica che sia possibile deselezionare la visualizzazione di variabili di template in un grafico; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema nasconde le variabili di template. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa ad un panel o una dashboard; – L'utente deseleziona la visualizzazione delle variabili di template.

Test	Specifica
TS-10.3.5	• **Progettazione**: verifica che sia possibile selezionare il tema di un grafico; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema mostra il tema selezionato. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa ad un panel o una dashboard; – L'utente seleziona il tema.
TS-10.4	• **Progettazione**: verifica che sia possibile condividere uno snapshot di una dashboard o di un panel; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema effettua la condivisione dello snapshot. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda *"Snapshot"*; – L'utente configura le opzioni per la condivisione dello snapshot.
TS-10.4.1	• **Progettazione**: verifica che sia possibile pubblicare uno snapshot sull'istanza locale dell'utente; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema pubblica lo snapshot sull'istanza. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda *"Snapshot"*; – L'utente preme il pulsante per pubblicare lo snaphsot sulla sua istanza.

Test	Specifica
TS-10.4.2	• **Progettazione**: verifica che sia possibile pubblicare uno snapshot su Raintank; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema pubblica lo snapshot su Raintank. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda *"Snapshot"*; – L'utente preme il pulsante per pubblicare lo snaphsot su Raintank.
TS-10.4.3	• **Progettazione**: verifica che sia possibile configurare le opzioni di visualizzazione di uno snapshot; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema configura le opzioni selezionate. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda *"Snapshot"*; – L'utente configura le opzioni di visualizzazione.
TS-10.4.3.1	• **Progettazione**: verifica che sia possibile inserire il nome di uno snapshot; • **Caso**: – **Input**: stringa con il nome; – **Risultato atteso**: il sistema inserisce il nome dello snapshot. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda *"Snapshot"*; – L'utente inserisce il nome dello snapshot.

Test	Specifica
TS-10.4.3.2	• **Progettazione**: verifica che sia possibile selezionare il tempo di permanenza di uno snapshot; • **Caso**: – **Input**: numero intero; – **Risultato atteso**: il sistema inserisce il tempo di permanenza. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda *"Snapshot"*; – L'utente inserisce il tempo di permanenza dello snapshot.
TS-10.4.3.3	• **Progettazione**: verifica che sia possibile inserire il tempo massimo per il caricamento dei dati in uno snapshot; • **Caso**: – **Input**: numero intero; – **Risultato atteso**: il sistema inserisce il tempo di timeout. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share"* relativa a una dashboard ad un panel; – L'utente si posiziona sulla scheda *"Snapshot"*; – L'utente inserisce il tempo di timeout dello snapshot.
TS-10.5	• **Progettazione**: verifica che sia possibile visualizzare il codice JSON contenente la definizione di una dashboard; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema mostra il codice JSON con la definizione della dashboard. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share dashboard"*; – L'utente si posiziona sulla scheda *"Export"*; – L'utente clicca sul pulsante per visualizzare il JSON.

Test	Specifica
TS-10.6	• **Progettazione**: verifica che sia possibile salvare il file JSON contenente la definizione di una dashboard; • **Caso**: – **Input**: nessuno; – **Risultato atteso**: il sistema effettua il salvataggio del codice JSON con la definizione della dashboard. • **Procedura**: – L'utente si posiziona all'interno di una dashboard; – L'utente seleziona l'opzione *"Share dashboard"*; – L'utente si posiziona sulla scheda *"Export"*; – L'utente clicca sul pulsante per salvare il JSON.

Tabella 14: Specifica test di sistema

C.3 Test di integrazione

Test	Specifica
Componenti	NetReader - NetUpdater

TI-1	• **Progettazione**: verifica che l'interazione tra NetReader e NetUpdater dati input corretti si svolga correttamente;
	• **Caso**:
	– **Input**: network corretta;
	– **Risultato atteso**: False,
	$0.18 < update\_result[0] < 0.22$,
	$0.58 < update\_result[1] < 0.62$.
	• **Procedura**:
	`const reader: NetReader = new NetReader(network);`
	`const updater: NetUpdater = new NetUpdater(network);`
	`const error: boolean = false;`
	`// read`
	`const read_result: InputResultAggregate = reader`
	`.read()`
	`.catch((e) => { error = true };`
	`// update`
	`const update_result: CalcResultAggregate;`
	`update_res = updater.updateNet(read_result);`
	`// check`
	`expect(error).to.equal(false);`
	`expect(update_result[0]).to.be.between(0.18, 0.22);`
	`expect(update_result[1]).to.be.between(0.58, 0.62);`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>Componenti</td><td>NetUpdater - NetWriter</td></tr>
<tr><td>TI-2</td><td>

Progettazione: verifica che l'interazione tra NetUpdater e NetWriter dati input corretti si svolga correttamente;
Caso:

Input: network corretta;
Risultato atteso: False.

Procedura:


```
const updater: NetUpdater = new NetUpdater(network);
const writer: SingleNetWriter = new SingleNetWriter(
   new ConcreteWriteClientFactory()
      .makeInfluxWriteClient(...));
const arrayResult: Array<InputResult>;
arrayResult.push(new InputResult(nodeAdapter, "0"));
const error: boolean = false;
// update
const update_res: CalcResultAggregate = updater
   .updateNet(read_res);
// write
writer.write(new InputResultAggregate(arrayResult))
   .catch(()=> {error = true});
// check
expect(error).to.equal(false);
```

</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>Componenti</td><td>NetManager: NetReader - NetUpdater - NetWriter</td></tr>
<tr><td>TI-3</td><td>

Progettazione: verifica che l'interazione tra NetReader, NetUpdater e NetWriter dati input corretti si svolga correttamente;
Caso:

Input: network corretta;
Risultato atteso: False.

Procedura:


```
const reader: NetReader = new NetReader(network);
const updater: NetUpdater = new NetUpdater(network);
const writer: SingleNetWriter = new SingleNetWriter(
   new ConcreteWriteClientFactory()
      .makeInfluxWriteClient(...));
const readerError: boolean = false;
const writerError: boolean = false;
// read
const read_res: InputResultAggregate = reader
   .read()
   .catch((e) => { readerError = true };
// update
const update_res: CalcResultAggregate = updater
   .updateNet(read_res);
// write
writer.write(new InputResultAggregate(arrayResult))
   .catch((e) => {writerError = true});
// check
expect(readerError).to.equal(false);
expect(writerError).to.equal(false);
```
</td></tr>
</table>

Test	Specifica
Componenti	NetworkAdapter - NetUpdater

TI-4	• **Progettazione**: verifica che l'interazione tra NetworkAdapter, AbstractValue e NetUpdater dati input corretti si svolga correttamente;
	• **Caso**:
	– **Input**: network corretta;
	– **Risultato atteso**: False, $0.18 < update\_result[0] < 0.22$, $0.58 < update\_result[1] < 0.62$.
	• **Procedura**:
	`const network: NetworkAdapter;`
	`// parse`
	`network = new ConcreteNetworkFactory()`
	`.parseNetwork(network, networkSchema);`
	`const networkUpdater: NetUpdater = new NetUpdater(`
	`network);`
	`// update`
	`const update_result: Array<CalcResult>;`
	`update_result = networkUpdater`
	`.updateNet(results);`
	`// check`
	`expect(update_result[0]).to.be.between(0.18, 0.22);`
	`expect(update_result[1]).to.be.between(0.58, 0.62);`

Tabella 15: Specifica test di integrazione

C.4 Test di unità

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>Scope</td><td>BoolValue::constructor</td></tr>
<tr><td>TU-1</td><td>

- **Progettazione**: verifica che il sistema rilevi una stringa non valida per il costruttore di *BoolValue()*;
- **Caso**:
  - **Input**: stringa non valida;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"invalid string parameter"*.
- **Procedura**:

```
let str: string;
expect(() => new BoolValue(true, str)).to.throw(TypeError, "↩
    invalid string parameter");
```

</td></tr>
<tr><td>TU-2</td><td>

- **Progettazione**: verifica che il sistema rilevi un valore booleano non valido per il costruttore di *BoolValue()*;
- **Caso**:
  - **Input**: boolean non valido;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"invalid boolean parameter"*.
- **Procedura**:

```
let str: boolean;
expect(() => new BoolValue(str, "name1")).to.throw(TypeError, ↩
    "invalid boolean parameter");
```

</td></tr>
<tr><td>Scope</td><td>BoolValue::isValueType</td></tr>
<tr><td>TU-3</td><td>

- **Progettazione**: verifica il funzionamento di *isValueType()*;
- **Caso**:
  - **Input**: oggetto *BoolValue* valido;
  - **Risultato atteso**: True.
- **Procedura**:

```
const boolV = new BoolValue(true, "name2");
const result: boolean = boolV.isValueType("true");
expect(result).to.equal(true);
```

</td></tr>
</table>

Test	Specifica

TU-4	• **Progettazione**: verifica il funzionamento di *isValueType()* con un parametro errato;
	• **Caso**:
	– **Input**: oggetto *BoolValue*;
	– **Risultato atteso**: False.
	• **Procedura**:
	`const boolV = new BoolValue(false, "name3");`
	`const result: boolean = boolV.isValueType("true");`
	`expect(result).to.equal(false);`

TU-5	• **Progettazione**: verifica il funzionamento di *isValueType()* con un parametro casuale;
	• **Caso**:
	– **Input**: oggetto *BoolValue*;
	– **Risultato atteso**: False.
	• **Procedura**:
	`const boolV = new BoolValue(false, "name4");`
	`const result: boolean = boolV.isValueType("brzcld");`
	`expect(result).to.equal(false);`

TU-6	• **Progettazione**: verifica il corretto funzionamento di *isValueType()* senza parametro;
	• **Caso**:
	– **Input**: oggetto *BoolValue*;
	– **Risultato atteso**: False.
	• **Procedura**:
	`const boolV = new BoolValue(true, "name5");`
	`const result: boolean = boolV.isValueType("");`
	`expect(result).to.equal(false);`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-7</td><td>

- **Progettazione**: verifica il funzionamento di *isValueType()* con un parametro valido ma con la lettera maiuscola;
- **Caso**:
  - **Input**: oggetto *BoolValue*;
  - **Risultato atteso**: True.
- **Procedura**:

```
const boolV = new BoolValue(true, "name6");
const result: boolean = boolV.isValueType("True");
expect(result).to.equal(true);
```

</td></tr>
<tr><td>TU-8</td><td>

- **Progettazione**: verifica il funzionamento di *isValueType()* con un parametro non definito;
- **Caso**:
  - **Input**: oggetto *BoolValue*;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"invalid parameter"*.
- **Procedura**:

```
const boolV = new BoolValue(true, "name7");
let str: string;
expect(() => boolV.isValueType(str)).to.throw(TypeError, "↩
    invalid parameter");
```

</td></tr>
<tr><td>Scope</td><td>BoolValue::getValueName</td></tr>
<tr><td>TU-9</td><td>

- **Progettazione**: verifica il funzionamento di *getValueName()*;
- **Caso**:
  - **Input**: oggetto *BoolValue*;
  - **Risultato atteso**: stringa con nome dell'oggetto.
- **Procedura**:

```
const name: string = "name8";
const boolV = new BoolValue(true, name);
boolV.isValueType("true");
boolV.isValueType("false");
const result: string = boolV.getValueName();
expect(result).to.equal(name);
```

</td></tr>
<tr><td>Scope</td><td>RangeValue::constructor</td></tr>
</table>

Test	Specifica

TU-10	• **Progettazione**: verifica che il sistema rilevi un nome come parametro non valido per il costruttore di *RangeValue()*;
	• **Caso**:
	– **Input**: stringa col nome non valida;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"invalid name parameter"*.
	• **Procedura**:
	`let str: string;`
	`expect(() => new RangeValue(30, 70, str)).to.throw(TypeError, "invalid name parameter");`

TU-11	• **Progettazione**: verifica che il sistema rilevi dei valori di range come parametri non validi per il costruttore di *RangeValue()*;
	• **Caso**:
	– **Input**: valori del range non validi;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"invalid range parameter"*.
	• **Procedura**:
	`let n1: number;`
	`let n2: number;`
	`expect(() => new RangeValue(n1, n2, "name1")).to.throw(TypeError, "invalid range parameter");`

TU-12	• **Progettazione**: verifica che il sistema rilevi parametri invertiti del range per il costruttore di *RangeValue()*; • **Caso**: – **Input**: *minRange* e *maxRange* invertiti; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"maxRange is less then minRange"*. • **Procedura**: `expect(() => new RangeValue(70, 30, "name1")).to.throw(TypeError, "maxRange is less then minRange");`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-13</td><td>

Progettazione: verifica il funzionamento del costruttore di RangeValue() con gli stessi parametri di range;
Caso:

Input: minRange e maxRange uguali a 50;
Risultato atteso: stringa con nome dell'oggetto.

Procedura:

<pre>
const name: string = "name1";
const rangeV: RangeValue = new RangeValue(50, 50, name);
rangeV.getValueName();
expect(rangeV.getValueName()).to.equal(name);
</pre>
</td></tr>
<tr><td>Scope</td><td>RangeValue::isValueType</td></tr>
<tr><td>TU-14</td><td>

Progettazione: verifica il funzionamento di isValueType();
Caso:

Input: oggetto RangeValue;
Risultato atteso: True.

Procedura:

<pre>
const rangeV = new RangeValue(30, 70, "name2");
const result: boolean = rangeV.isValueType("50");
expect(result).to.equal(true);
</pre>
</td></tr>
<tr><td>TU-15</td><td>

Progettazione: verifica il funzionamento di isValueType() con parametro il limite del range;
Caso:

Input: oggetto RangeValue;
Risultato atteso: True.

Procedura:

<pre>
const rangeV = new RangeValue(30, 70, "name3");
const result: boolean = rangeV.isValueType("70");
expect(result).to.equal(true);
</pre>
</td></tr>
</table>

Test	Specifica

TU-16	• **Progettazione**: verifica il funzionamento di *isValueType()* con parametro casuale;
	• **Caso**:
	– **Input**: oggetto *RangeValue*;
	– **Risultato atteso**: False.
	• **Procedura**:
	`const rangeV = new RangeValue(30, 70, "name4");`
	`const result: boolean = rangeV.isValueType("brzcld");`
	`expect(result).to.equal(false);`

TU-17	• **Progettazione**: verifica il funzionamento di *isValueType()* con parametro vuoto;
	• **Caso**:
	– **Input**: oggetto *RangeValue*;
	– **Risultato atteso**: False.
	• **Procedura**:
	`const rangeV = new RangeValue(30, 70, "name5");`
	`const result: boolean = rangeV.isValueType("");`
	`expect(result).to.equal(false);`

TU-18	• **Progettazione**: verifica il funzionamento di *isValueType()* con parametro errato;
	• **Caso**:
	– **Input**: oggetto *RangeValue*;
	– **Risultato atteso**: False.
	• **Procedura**:
	`const rangeV = new RangeValue(30, 70, "name6");`
	`const result: boolean = rangeV.isValueType("20");`
	`expect(result).to.equal(false);`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-19</td><td>

- **Progettazione**: verifica il funzionamento di *isValueType()* con parametro non definito;
- **Caso**:
  - **Input**: oggetto *RangeValue*;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"invalid string parameter"*..
- **Procedura**:

```
const rangeV = new RangeValue(30, 70, "name7");
let str: string;
expect(() => rangeV.isValueType(str)).to.throw(TypeError, "↩
    invalid string parameter");
```

</td></tr>
<tr><td>TU-20</td><td>

- **Progettazione**: verifica il funzionamento di *isValueType()* su un oggetto con gli stessi valori di range;
- **Caso**:
  - **Input**: oggetto *RangeValue* con *minRange* e *maxRange* uguali a 50;
  - **Risultato atteso**: True.
- **Procedura**:

```
const rangeV = new RangeValue(50, 50, "name8");
const result: boolean = rangeV.isValueType("50");
expect(result).to.equal(true);
```

</td></tr>
<tr><td>Scope</td><td>RangeValue::getValueName</td></tr>
<tr><td>TU-21</td><td>

- **Progettazione**: verifica il funzionamento di *getValueName()*;
- **Caso**:
  - **Input**: oggetto *RangeValue*;
  - **Risultato atteso**: stringa con nome dell'oggetto.
- **Procedura**:

```
const name: string = "name9";
const rangeV = new RangeValue(30, 70, name);
rangeV.isValueType("true");
rangeV.isValueType("false");
const result: string = rangeV.getValueName();
expect(result).to.equal(name);
```

</td></tr>
<tr><td>Scope</td><td>StringValue::constructor</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-22</td><td>

- **Progettazione**: verifica che il sistema rilevi un valore come parametro non valido per il costruttore di *StringValue()*;
- **Caso**:
  - **Input**: stringa con value non valida;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"invalid value parameter"*.
- **Procedura**:

```
let str: string;
expect(() => new StringValue("value1", str)).to.throw(↩
    TypeError, "invalid value parameter");
```

</td></tr>
<tr><td>TU-23</td><td>

- **Progettazione**: verifica che il sistema rilevi un nome come parametro non valido per il costruttore di *StringValue()*;
- **Caso**:
  - **Input**: stringa con nome non valida;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"invalid name parameter"*.
- **Procedura**:

```
let str: string;
expect(() => new StringValue(str, "name1")).to.throw(TypeError↩
    , "invalid name parameter");
```

</td></tr>
<tr><td>Scope</td><td>StringValue::isValueType</td></tr>
<tr><td>TU-24</td><td>

- **Progettazione**: verifica il funzionamento di *isValueType()*;
- **Caso**:
  - **Input**: oggetto *StringValue*;
  - **Risultato atteso**: True.
- **Procedura**:

```
const stringV = new StringValue("value2", "name2");
const result: boolean = stringV.isValueType("value2");
expect(result).to.equal(true);
```

</td></tr>
</table>

Test	Specifica

TU-25	• **Progettazione**: verifica il funzionamento di *isValueType()* con parametro errato;
	• **Caso**:
	– **Input**: oggetto *StringValue*;
	– **Risultato atteso**: False.
	• **Procedura**:
	`const stringV = new StringValue("value3", "name3");`
	`const result: boolean = stringV.isValueType("true");`
	`expect(result).to.equal(false);`

TU-26	• **Progettazione**: verifica il funzionamento di *isValueType()* con parametro casuale;
	• **Caso**:
	– **Input**: oggetto *StringValue*;
	– **Risultato atteso**: False.
	• **Procedura**:
	`const stringV = new StringValue("value4", "name4");`
	`const result: boolean = stringV.isValueType("brzcld");`
	`expect(result).to.equal(false);`

TU-27	• **Progettazione**: verifica il funzionamento di *isValueType()* con parametro vuoto;
	• **Caso**:
	– **Input**: oggetto *StringValue*;
	– **Risultato atteso**: False.
	• **Procedura**:
	`const stringV = new StringValue("value4", "name4");`
	`const result: boolean = stringV.isValueType("brzcld");`
	`expect(result).to.equal(false);`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-28</td><td>

- **Progettazione**: verifica il funzionamento di *isValueType()* con parametro vuoto;
- **Caso**:
  - **Input**: oggetto *StringValue*;
  - **Risultato atteso**: False.
- **Procedura**:

```
const stringV = new StringValue("value5", "name5");
const result: boolean = stringV.isValueType("");
expect(result).to.equal(false);
```

</td></tr>
<tr><td>TU-29</td><td>

- **Progettazione**: verifica il funzionamento di *isValueType()* con parametro corretto ma con la lettera maiuscola;
- **Caso**:
  - **Input**: oggetto *StringValue*;
  - **Risultato atteso**: False.
- **Procedura**:

```
const stringV = new StringValue("value6", "name6");
const result: boolean = stringV.isValueType("VALUE6");
expect(result).to.equal(false);
```

</td></tr>
<tr><td>TU-30</td><td>

- **Progettazione**: verifica il funzionamento di *isValueType()* con parametro non definito;
- **Caso**:
  - **Input**: oggetto *StringValue*;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"invalid parameter"*.
- **Procedura**:

```
const stringV = new StringValue("value7", "name7");
let str: string;
expect(() => stringV.isValueType(str)).to.throw(TypeError, "↩
    invalid parameter");
```

</td></tr>
<tr><td>Scope</td><td>StringValue::getValueName</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-31</td><td>

- **Progettazione**: verifica il funzionamento di *getValueName()*;
- **Caso**:
  - **Input**: oggetto *StringValue*;
  - **Risultato atteso**: stringa con nome dell'oggetto.
- **Procedura**:

```
const name: string = "name8";
const stringV = new StringValue("value8", name);
stringV.isValueType("true");
stringV.isValueType("false");
const result: string = stringV.getValueName();
expect(result).to.equal(name);
```

</td></tr>
<tr><td>Scope</td><td>ConcreteNodeAdapter::constructor</td></tr>
<tr><td>TU-32</td><td>

- **Progettazione**: verifica che il sistema rilevi parametri non validi per il costruttore di *ConcreteNodeAdapter()*;
- **Caso**:
  - **Input**: parametro *node* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *TypeError* mostrando il messaggio *"invalid node parameter"*.
- **Procedura**:

```
let node: JNode;
const values: Array<AbstractValue> = new Array<AbstractValue>();
expect(() => new ConcreteNodeAdapter(node, values)).to.throw(TypeError, "invalid node parameter");
```

</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-33</td><td>

Progettazione: verifica che il sistema rilevi parametri non validi per il costruttore di ConcreteNodeAdapter();
Caso:

Input: parametro values non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo TypeError mostrando il messaggio "invalid values parameter".

Procedura:


```
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
let values: Array<AbstractValue>;
expect(() => new ConcreteNodeAdapter(n1, values)).to.throw(↩
    TypeError, "invalid values parameter");
```
</td></tr>
<tr><td>Scope</td><td>ConcreteNodeAdapter::getSates</td></tr>
<tr><td>TU-34</td><td>

Progettazione: verifica il funzionamento di getStates();
Caso:

Input: oggetto ConcreteNodeAdapter con stati [True, False];
Risultato atteso: [True, False].

Procedura:


```
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const concreteNode = new ConcreteNodeAdapter(n1, new Array<↩
    AbstractValue>());
const states: Array<string> = concreteNode.getStates();
expect(states.toLocaleString()).to.equal("true,false");
```
</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-35</td><td>

Progettazione: verifica il funzionamento di getStates() controllando che venga ritornata una copia profonda dell'oggetto;
Caso:

Input: oggetto ConcreteNodeAdapter con stati [True, False];
Risultato atteso: [True, False].

Procedura:


```
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const concreteNode = new ConcreteNodeAdapter(n1, new Array<↩
    AbstractValue>());
const states = concreteNode.getStates();
states.reverse();
expect(concreteNode.getStates().toLocaleString()).to.equal("↩
    true,false");
```

</td></tr>
<tr><td>TU-36</td><td>

Progettazione: verifica il funzionamento di getStates() con stati di un nodo invertiti;
Caso:

Input: oggetto ConcreteNodeAdapter con stati [True, False];
Risultato atteso: diverso da [False, True].

Procedura:


```
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const concreteNode = new ConcreteNodeAdapter(n1, new Array<↩
    AbstractValue>());
const states = concreteNode.getStates();
expect(states.toLocaleString()).to.not.equal("false,true");
```

</td></tr>
<tr><td>Scope</td><td>ConcreteNodeAdapter::getValueName</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-37</td><td>

- **Progettazione**: verifica il funzionamento di *getValueName()*;
- **Caso**:
  - **Input**: array di *BoolValue*;
  - **Risultato atteso**: nome dei values.
- **Procedura**:

```
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const concreteNodeValues: Array<AbstractValue> = new Array<↩
    AbstractValue>();
concreteNodeValues.push(new BoolValue(false, "boolvalue-1"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-2"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-3"));
const concreteNode = new ConcreteNodeAdapter(n1, ↩
    concreteNodeValues);
const values: Array<AbstractValue> = concreteNode.getValues();
expect(values[0].getValueName()).to.equal("boolvalue-1");
expect(values[1].getValueName()).to.equal("boolvalue-2");
expect(values[2].getValueName()).to.equal("boolvalue-3");
```

</td></tr>
<tr><td>TU-38</td><td>

- **Progettazione**: verifica il funzionamento di *getValueName()* controllando che venga ritornata una copia profonda dell'oggetto;
- **Caso**:
  - **Input**: array di *BoolValue*;
  - **Risultato atteso**: nome dei values.
- **Procedura**:

```
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const concreteNodeValues: Array<AbstractValue> = new Array<↩
    AbstractValue>();
concreteNodeValues.push(new BoolValue(false, "boolvalue-1"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-2"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-3"));
const concreteNode = new ConcreteNodeAdapter(n1, ↩
    concreteNodeValues);
const values: Array<AbstractValue> = concreteNode.getValues();
values.reverse();
const newValues = concreteNode.getValues();
expect(newValues[0].getValueName()).to.equal("boolvalue-1");
expect(newValues[1].getValueName()).to.equal("boolvalue-2");
expect(newValues[2].getValueName()).to.equal("boolvalue-3");
```

</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-39</td><td>

Progettazione: verifica il funzionamento di getValueName();
Caso:

Input: array di BoolValue;
Risultato atteso: diverso da un nome predefinito.

Procedura:


```
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const concreteNodeValues: Array<AbstractValue> = new Array<↩
    AbstractValue>();
concreteNodeValues.push(new BoolValue(false, "boolvalue-1"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-2"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-2"));
const concreteNode = new ConcreteNodeAdapter(n1, ↩
    concreteNodeValues);
const values: Array<AbstractValue> = concreteNode.getValues();
let str: string;
expect(values[0].getValueName()).to.not.equal("test");
expect(values[1].getValueName()).to.not.equal(str);
expect(values[2].getValueName()).to.not.equal("");
```
</td></tr>
<tr><td>Scope</td><td>concreteNodeValues::findValue</td></tr>
<tr><td>TU-40</td><td>

Progettazione: verifica il funzionamento di findValue();
Caso:

Input: array di BoolValue;;
Risultato atteso: primo value con la stessa chiave di ricerca.

Procedura:


```
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const concreteNodeValues: Array<AbstractValue> = new Array<↩
    AbstractValue>();
concreteNodeValues.push(new BoolValue(false, "boolvalue-1"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-2"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-3"));
const concreteNode = new ConcreteNodeAdapter(n1, ↩
    concreteNodeValues);
const value: AbstractValue = concreteNode.findValue("true");
expect(value.getValueName()).to.equal("boolvalue-2");
```
</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-41</td><td>

Progettazione: verifica il funzionamento di findValue() su un oggetto non presente nella lista;
Caso:

Input: array di BoolValue;;
Risultato atteso: NULL.

Procedura:

<pre>
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const concreteNodeValues: Array<AbstractValue> = new Array<AbstractValue>();
concreteNodeValues.push(new BoolValue(false, "boolvalue-1"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-2"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-3"));
const concreteNode = new ConcreteNodeAdapter(n1, concreteNodeValues);
const value: AbstractValue = concreteNode.findValue("valorechenonce");
expect(value).to.equal(null);
</pre>
</td></tr>
<tr><td>TU-42</td><td>

Progettazione: verifica il funzionamento di findValue() su un oggetto non definito;
Caso:

Input: array di BoolValue;;
Risultato atteso: il sistema lancia un'eccezione di tipo TypeError mostrando il messaggio "invalid parameter".

Procedura:

<pre>
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const concreteNodeValues: Array<AbstractValue> = new Array<AbstractValue>();
concreteNodeValues.push(new BoolValue(false, "boolvalue-1"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-2"));
concreteNodeValues.push(new BoolValue(true, "boolvalue-3"));
const concreteNode = new ConcreteNodeAdapter(n1, concreteNodeValues);
let str: string;
expect(() => concreteNode.findValue(str)).to.throw(TypeError, "invalid parameter");
</pre>
</td></tr>
<tr><td>Scope</td><td>ConcreteNetworkFactory::parseNetwork</td></tr>
</table>

Test	Specifica

TU-43	• **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* controllando il numero di nodi presenti;
	• **Caso**:
	– **Input**: file JSON con definizione di rete con due nodi;
	– **Risultato atteso**: 2.
	• **Procedura**:
	`const json = require("./CorrectNetwork.json");`
	`const jsonString: string = JSON.stringify(json);`
	`const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory().parseNetwork(jsonString);`
	`expect(s.getNodeList().length).to.equal(2);`

TU-44	• **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* con un input nullo;
	• **Caso**:
	– **Input**: NULL;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"JSON file content is null..."*.
	• **Procedura**:
	`let str: string;`
	`expect(() => new ConcreteNetworkFactory().parseNetwork(str)).to.throw(Error, "JSON file content is null...");`

TU-45	• **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* con un JSON vuoto;
	• **Caso**:
	– **Input**: file JSON vuoto;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Bad Json Content!"*.
	• **Procedura**:
	`let str: string = "";`
	`expect(() => new ConcreteNetworkFactory().parseNetwork(str)).to.throw(Error, "Bad Json Content!");`

Test	Specifica

TU-46	• **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* con un JSON non valido;
	• **Caso**:
	– **Input**: file JSON non valido;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Bad Json Content!"*.
	• **Procedura**:
	`let json = require("./InvalidNetwork.json");`
	`const jsonString = JSON.stringify(json);`
	`expect(() => new ConcreteNetworkFactory().parseNetwork(jsonString)).to.throw(Error, "");`

TU-47	• **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* con un JSON che non rispetta la struttura di rete;
	• **Caso**:
	– **Input**: file JSON con un struttura di rete errata;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Bad Json Content!"*.
	• **Procedura**:
	`const jsonString = "{"nodes": {[]}}";`
	`expect(() => new ConcreteNetworkFactory().parseNetwork(jsonString)).to.throw(Error, "Bad Json Content!");`

TU-48	• **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* con un JSON con valori della CPT errati;
	• **Caso**:
	– **Input**: file JSON con valori della CPT errati;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"CPT error!"*.
	• **Procedura**:
	`let json = require("./IncorrectCpt.json");`
	`const jsonString = JSON.stringify(json);`
	`expect(() => new ConcreteNetworkFactory().parseNetwork(jsonString)).to.throw(Error, "CPT error!");`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-49</td><td>

Progettazione: verifica il funzionamento di ConcreteNetworkFactory() con un JSON con range invertiti;
Caso:

Input: file JSON con range invertiti;
Risultato atteso: il sistema lancia un'eccezione di tipo TypeError mostrando il messaggio "maxRange is less then minRange".

Procedura:


```
let json = require("./InvertedMinMax.json");
const jsonString = JSON.stringify(json);
expect(() => new ConcreteNetworkFactory().parseNetwork(↩
    jsonString)).to.throw(TypeError, "maxRange is less then ↩
    minRange");
```

</td></tr>
<tr><td>TU-50</td><td>

Progettazione: verifica il funzionamento di ConcreteNetworkFactory() con un JSON con valori dei nodi errati;
Caso:

Input: file JSON con valori dei nodi errati;
Risultato atteso: il sistema lancia un'eccezione di tipo TypeError mostrando il messaggio "invalid value parameter".

Procedura:


```
let json = require("./IncorrectType.json");
const jsonString = JSON.stringify(json);
expect(() => new ConcreteNetworkFactory().parseNetwork(↩
    jsonString)).to.throw(TypeError, "invalid value parameter"↩
    );
```

</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-51</td><td>

Progettazione: verifica il funzionamento di ConcreteNetworkFactory() con un JSON con un nodo padre non esistente;
Caso:

Input: file JSON con un nodo padre non esistente;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "Node Parent not found in the network!".

Procedura:


```
let json = require("./IncorrectParent.json");
const jsonString = JSON.stringify(json);
expect(() => new ConcreteNetworkFactory().parseNetwork(↩
    jsonString)).to.throw(Error, "Node Parent not found in the↩
     network!");
```
</td></tr>
<tr><td>TU-52</td><td>

Progettazione: verifica il funzionamento di ConcreteNetworkFactory() con un JSON con un riferimento circolare diretto;
Caso:

Input: file JSON con un riferimento circolare diretto;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "Direct circular parenthood".

Procedura:


```
let json = require("./DCircularParenthood.json");
const jsonString = JSON.stringify(json);
new ConcreteNetworkFactory().parseNetwork(jsonString);
expect(() => new ConcreteNetworkFactory().parseNetwork(↩
    jsonString)).to.throw(Error, "Direct circular parenthood")↩
    ;
```
</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-53</td><td>

- **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* con un JSON con un riferimento circolare indiretto;
- **Caso**:
  - **Input**: file JSON con un riferimento circolare indiretto;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Indirect circular parenthood"*.
- **Procedura**:

```
let json = require("./ICircularParenthood.json");
const jsonString = JSON.stringify(json);
new ConcreteNetworkFactory().parseNetwork(jsonString);
expect(() => new ConcreteNetworkFactory().parseNetwork(↩
    jsonString)).to.throw(Error, "Indirect circular parenthood↩
    ");
```

</td></tr>
<tr><td>TU-54</td><td>

- **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* con un JSON con due nodi dallo stesso nome;
- **Caso**:
  - **Input**: file JSON con un nodo dallo stesso nome di un altro nodo;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"The node ExampleNode2 already exist in the network!"*.
- **Procedura**:

```
let json = require("./IncorrectName.json");
const jsonString = JSON.stringify(json);
expect(() => new ConcreteNetworkFactory().parseNetwork(↩
    jsonString)).to.throw(Error, "The node ExampleNode2 ↩
    already exist in the network!");
```

</td></tr>
</table>

Test	Specifica
TU-55	• **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* con un JSON con probabilità della CPT non valide; • **Caso**: – **Input**: file JSON con probabilità CPT non valide; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Incorrect CPT probabilities"*. • **Procedura**: `expect(() => new ConcreteNetworkFactory().parseNetwork(IncorrectCptJsonString, jsonSchemaString)).to.throw(Error, "Incorrect CPT probabilities");`
TU-56	• **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* con un JSON con probabilità della CPT vuote; • **Caso**: – **Input**: file JSON con probabilità CPT vuote; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Empty CPT"*. • **Procedura**: `expect(() => new ConcreteNetworkFactory().parseNetwork(EmptyCptJsonString, jsonSchemaString)).to.throw(Error, "Empty CPT");`
TU-57	• **Progettazione**: verifica il funzionamento di *ConcreteNetworkFactory()* con un JSON le righe della CPT errate; • **Caso**: – **Input**: file JSON con righe CPT errate; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Incorrect CPT number of rows"*. • **Procedura**: `expect(() => new ConcreteNetworkFactory().parseNetwork(IncorrectCptRowsJsonString, jsonSchemaString)).to.throw(Error, "Incorrect CPT number of rows");`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-58</td><td>

Progettazione: verifica il funzionamento di ConcreteNetworkFactory() con un JSON le colonne della CPT errate;
Caso:

Input: file JSON con colonne CPT errate;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "Incorrect CPT number of columns".

Procedura:


```
expect(() => new ConcreteNetworkFactory().parseNetwork(↩
    IncorrectCptColumnsJsonString, jsonSchemaString)).to.throw↩
    (Error, "Incorrect CPT number of columns");
```
</td></tr>
<tr><td>Scope</td><td>ConcreteNetworkAdapter::constructor</td></tr>
<tr><td>TU-59</td><td>

Progettazione: verifica il funzionamento del costruttore di ConcreteNetworkAdapter() con parametri non validi;
Caso:

Input: parametro graph non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid graph parameter".

Procedura:


```
let graph: JGraph;
let list: Array<NodeAdapter> = new Array<NodeAdapter>();
expect(() => new ConcreteNetworkAdapter(graph, list)).to.throw↩
    (Error, "invalid graph parameter");
```
</td></tr>
<tr><td>TU-60</td><td>

Progettazione: verifica il funzionamento del costruttore di ConcreteNetworkAdapter() con parametri non validi;
Caso:

Input: parametro list non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid list parameter".

Procedura:


```
let graph: JGraph = jsbayes.newGraph();
let list: Array<NodeAdapter>;
expect(() => new ConcreteNetworkAdapter(graph, list)).to.throw↩
    (Error, "invalid list parameter");
```
</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-61</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *ConcreteNetworkAdapter()* con parametri non validi;
- **Caso**:
  - **Input**: parametro *list* vuoto;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid list parameter"*.
- **Procedura**:

```
let graph: JGraph = jsbayes.newGraph();
let list: Array<NodeAdapter> = new Array<NodeAdapter>();
expect(() => new ConcreteNetworkAdapter(graph, list)).to.throw(Error, "invalid list parameter");
```

</td></tr>
<tr><td>Scope</td><td>ConcreteNetworkAdapter::observeNode</td></tr>
<tr><td>TU-62</td><td>

- **Progettazione**: verifica il funzionamento di *observeNode()*;
- **Caso**:
  - **Input**: file JSON valido;
  - **Risultato atteso**: TRUE.

</td></tr>
<tr><td>TU-63</td><td>

- **Progettazione**: verifica il funzionamento di *observeNode()* quando il nome di un nodo non è definito;
- **Caso**:
  - **Input**: file JSON valido, nome nodo vuoto;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid node parameter"*.
- **Procedura**:

```
const network: NetworkAdapter = new ConcreteNetworkFactory().parseNetwork(correctJsonString, jsonSchemaString);
expect(()=>network.observeNode("", "Example of string value")).to.throw(Error, "invalid node parameter");
```

</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-64</td><td>

Progettazione: verifica il funzionamento di observeNode() quando il nome di un nodo non è presente;
Caso:

Input: file JSON valido, nome nodo errato;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "Node FakeNode is not present in the network".

Procedura:

<pre>
const network: NetworkAdapter = new ConcreteNetworkFactory().parseNetwork(correctJsonString, jsonSchemaString);
expect(()=>network.observeNode("FakeNode", "Example of string value")).to.throw(Error, "Node FakeNode is not present in the network");
</pre>
</td></tr>
<tr><td>TU-65</td><td>

Progettazione: verifica il funzionamento di observeNode() quando il valore di un nodo non è presente;
Caso:

Input: file JSON valido, valore nodo errato;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "Node Example has not a value called FakeValue".

Procedura:

<pre>
const network: NetworkAdapter = new ConcreteNetworkFactory().parseNetwork(correctJsonString, jsonSchemaString);
expect(()=>network.observeNode("Example", "FakeValue")).to.throw(Error, "Node Example has not a value called FakeValue");
</pre>
</td></tr>
<tr><td>Scope</td><td>ConcreteNetworkAdapter::unobserveNode</td></tr>
<tr><td>TU-66</td><td>

Progettazione: verifica il funzionamento di unobserveNode();
Caso:

Input: file JSON valido;
Risultato atteso: TRUE.

</td></tr>
</table>

Test	Specifica

TU-67	• **Progettazione**: verifica il funzionamento di *unobserveNode()* quando il nome del nodo non è definito;
	• **Caso**:
	– **Input**: file JSON valido, nome nodo non definito;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid node parameter"*.
	• **Procedura**:
	`const network: NetworkAdapter = new ConcreteNetworkFactory().parseNetwork(correctJsonString, jsonSchemaString);`
	`let name: string;`
	`expect(()=>network.unobserveNode(name)).to.throw(Error, "invalid node parameter");`

TU-68	• **Progettazione**: verifica il funzionamento di *unobserveNode()* quando il nome del nodo non è vuoto;
	• **Caso**:
	– **Input**: file JSON valido, nome nodo vuoto;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid node parameter"*.
	• **Procedura**:
	`const network: NetworkAdapter = new ConcreteNetworkFactory().parseNetwork(correctJsonString, jsonSchemaString);`
	`expect(()=>network.unobserveNode("")).to.throw(Error, "invalid node parameter");`

TU-69	• **Progettazione**: verifica il funzionamento di *unobserveNode()* quando il nome del nodo non è vuoto;
	• **Caso**:
	– **Input**: file JSON valido, nome nodo vuoto;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid node parameter"*.
	• **Procedura**:
	`const network: NetworkAdapter = new ConcreteNetworkFactory().parseNetwork(correctJsonString, jsonSchemaString);`
	`expect(()=>network.unobserveNode("")).to.throw(Error, "invalid node parameter");`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-70</td><td>

Progettazione: verifica il funzionamento di unobserveNode() quando il nome del nodo non è presente;
Caso:

Input: file JSON valido, nome nodo errato;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "Node FakeNode is not present in the network".

Procedura:

<pre>
const network: NetworkAdapter = new ConcreteNetworkFactory().↩
    parseNetwork(correctJsonString, jsonSchemaString);
expect(()=>network.unobserveNode("FakeNode")).to.throw(Error, ↩
    "Node FakeNode is not present in the network");
</pre>
</td></tr>
<tr><td>Scope</td><td>ConcreteNetworkAdapter::sampleNetwork</td></tr>
<tr><td>TU-71</td><td>

Progettazione: verifica il funzionamento di sampleNetwork() quando il numero non è definito;
Caso:

Input: file JSON valido, numero non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid number parameter".

Procedura:

<pre>
const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory()↩
    .parseNetwork(correctJsonString, jsonSchemaString);
let undNumber: number;
expect(()=>s.sampleNetwork(undNumber)).to.throw(Error, "↩
    invalid number parameter");
</pre>
</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-72</td><td>

Progettazione: verifica il funzionamento di sampleNetwork() quando il numero è minore di zero;
Caso:

Input: file JSON valido, numero < 0;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid number parameter".

Procedura:


```
const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory()↩
    .parseNetwork(correctJsonString, jsonSchemaString);
let negNumber: number = -50;
expect(()=>s.sampleNetwork(negNumber)).to.throw(Error, "↩
    invalid number parameter");
```
</td></tr>
<tr><td>Scope</td><td>ConcreteNetworkAdapter::getNodeProbs</td></tr>
<tr><td>TU-73</td><td>

Progettazione: verifica il funzionamento di getNodeProbs() quando il nome del nodo non è definito;
Caso:

Input: file JSON valido, nome nodo non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid node parameter".

Procedura:


```
const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory()↩
    .parseNetwork(correctJsonString, jsonSchemaString);
let undName: string;
expect(()=>s.getNodeProbs(undName)).to.throw(Error, "invalid ↩
    node parameter");
```
</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-74</td><td>

- **Progettazione**: verifica il funzionamento di *getNodeProbs()* quando il nome del nodo è vuoto;
- **Caso**:
  - **Input**: file JSON valido, nome nodo vuoto;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid node parameter"*.
- **Procedura**:

```
const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory()
    .parseNetwork(correctJsonString, jsonSchemaString);
expect(()=>s.getNodeProbs("")).to.throw(Error, "invalid node
    parameter");
```

</td></tr>
<tr><td>TU-75</td><td>

- **Progettazione**: verifica il funzionamento di *getNodeProbs()* quando il nome del nodo è presente;
- **Caso**:
  - **Input**: file JSON valido, nome nodo errato;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Node IncorrectName is not present in the network"*.
- **Procedura**:

```
const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory()
    .parseNetwork(correctJsonString, jsonSchemaString);
expect(()=>s.getNodeProbs("IncorrectName")).to.throw(Error, "
    Node IncorrectName is not present in the network");
```

</td></tr>
<tr><td>Scope</td><td>ConcreteNetworkAdapter::getNodeList</td></tr>
<tr><td>TU-76</td><td>

- **Progettazione**: verifica il funzionamento di *getNodeList()*;
- **Caso**:
  - **Input**: file JSON valido;
  - **Risultato atteso**: *"Example2"*.
- **Procedura**:

```
const s: ConcreteNetworkAdapter = new ConcreteNetworkFactory()
    .parseNetwork(correctJsonString, jsonSchemaString);
const nodeOne = s.getNodeList()[1];
expect(nodeOne.getName()).to.equal("Example2");
```

</td></tr>
<tr><td>Scope</td><td>InputFlow::constructor</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-77</td><td>

Progettazione: verifica il funzionamento del costruttore di InfluxInputFlow() quando database non è definito;
Caso:

Input: database non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid db parameter".

Procedura:


```
let db: string;
expect(()=>
new InfluxInputFlow(db, "query",
new ConcreteReadClientFactory().makeInfluxReadClient("http://↩
    localhost", "9957", ["user", "password"]))
).to.throw(Error, "invalid db parameter");
```
</td></tr>
<tr><td>TU-78</td><td>

Progettazione: verifica il funzionamento del costruttore di InfluxInputFlow() quando query non è definito;
Caso:

Input: query non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid query parameter".

Procedura:


```
let query: string;
expect(()=>
new InfluxInputFlow("db", query,
new ConcreteReadClientFactory().makeInfluxReadClient("http://↩
    localhost", "9957", ["user", "password"]))
).to.throw(Error, "invalid query parameter");
```
</td></tr>
</table>

Test	Specifica

TU-79	• **Progettazione**: verifica il funzionamento del costruttore di *InfluxInputFlow()* quando *client* non è definito;
	• **Caso**:
	– **Input**: *client* non definito;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid client parameter"*.
	• **Procedura**:
	`let readClient: ReadClient;`
	`expect(()=>`
	`new InfluxInputFlow("db", "query", readClient)`
	`).to.throw(Error, "invalid client parameter");`

TU-80	• **Progettazione**: verifica il funzionamento del costruttore di *InfluxInputFlow()* quando *database* è vuoto;
	• **Caso**:
	– **Input**: *database* vuoto;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid db parameter"*.
	• **Procedura**:
	`expect(()=>`
	`new InfluxInputFlow("", "query",`
	`new ConcreteReadClientFactory().makeInfluxReadClient("http://localhost", "9957", ["user", "password"]))`
	`).to.throw(Error, "invalid db parameter");`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-81</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *InfluxInputFlow()* quando *query* è vuoto;
- **Caso**:
  - **Input**: *query* vuoto;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid query parameter"*.
- **Procedura**:

```
        expect(()=>
new InfluxInputFlow("db", "",
new ConcreteReadClientFactory().makeInfluxReadClient("http://↩
    localhost", "9957", ["user", "password"]))
).to.throw(Error, "invalid query parameter");
```

</td></tr>
<tr><td>TU-82</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *InfluxInputFlow()*;
- **Caso**:
  - **Input**: parametri corretti;
  - **Risultato atteso**: nessuna eccezione viene viene lanciata.
- **Procedura**:

```
expect(()=>
new InfluxInputFlow("db", "query",
new ConcreteReadClientFactory().makeInfluxReadClient("http://↩
    localhost", "9957", ["user", "password"]))
).to.not.throw(Error);
```

</td></tr>
<tr><td>Scope</td><td>InfluxInputFlow::getResult</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-83</td><td>

- **Progettazione**: verifica il funzionamento di *getResult()*;
- **Caso**:
  - **Input**: oggetto *InfluxInputFlow*;
  - **Risultato atteso**: viene ritornato il risultato.
- **Procedura**:

```
1 const influxInput: InfluxInputFlow = new InfluxInputFlow("↩
    prova", "SELECT field1 FROM burglary",
2 new ConcreteReadClientFactory().makeInfluxReadClient("http://↩
    localhost", "8086", ["root", "root"]));
3 influxInput.getResult().then(function(result){
4 expect(result.toString()).to.not.equal(null);
```

</td></tr>
<tr><td>TU-84</td><td>

- **Progettazione**: verifica il funzionamento di *getResult()* quando c'è un errore nalla query;
- **Caso**:
  - **Input**: query errata;
  - **Risultato atteso**: il sistema mostra il messaggio *"Error: Query"*.
- **Procedura**:

```
1 const influxInput: InfluxInputFlow = new InfluxInputFlow("↩
    prova", "SELECT field FROM measurement",
2 new ConcreteReadClientFactory().makeInfluxReadClient("http://↩
    localhost", "8086", ["user", "password"]));
3 influxInput.getResult().then(function(){}).catch(function(e){
4 expect(e.toString()).to.contain("Error: Query");
5 });
```

</td></tr>
<tr><td>Scope</td><td>Datasource::constructor</td></tr>
</table>

Test	Specifica
TU-85	• **Progettazione**: verifica il funzionamento del costruttore di *Datasource()* quando *url* è vuoto; • **Caso**: – **Input**: *url* vuoto; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid url parameter"*. • **Procedura**: `expect(()=> expect(() => new DataSource("", "database")).to.throw(Error, "invalid url parameter");`
TU-86	• **Progettazione**: verifica il funzionamento del costruttore di *Datasource()* quando *database* è vuoto; • **Caso**: – **Input**: *database* vuoto; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid db parameter"*. • **Procedura**: `expect(() => new DataSource("http://localhost", "")).to.throw(Error, "invalid db parameter");`
TU-87	• **Progettazione**: verifica il funzionamento del costruttore di *Datasource()* quando *username* è vuoto; • **Caso**: – **Input**: *username* vuoto; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid username parameter"*. • **Procedura**: `expect(() => new DataSource("http://localhost", "database", "")).to.throw(Error, "invalid username parameter");`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-88</td><td>

Progettazione: verifica il funzionamento del costruttore di Datasource() quando password è vuoto;
Caso:

Input: password vuoto;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid password parameter".

Procedura:

<pre><code>expect(() => new DataSource("http://localhost", "database", "username", "")).to.throw(Error, "invalid password parameter");</code></pre>
</td></tr>
<tr><td>TU-89</td><td>

Progettazione: verifica il funzionamento del costruttore di Datasource() quando type è vuoto;
Caso:

Input: type vuoto;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid type parameter".

Procedura:

<pre><code>expect(() => new DataSource("http://localhost", "database", "username", "password", "")).to.throw(Error, "invalid type parameter");</code></pre>
</td></tr>
<tr><td>TU-90</td><td>

Progettazione: verifica il funzionamento del costruttore di Datasource() quando name è vuoto;
Caso:

Input: name vuoto;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid name parameter".

Procedura:

<pre><code>expect(() => new DataSource("http://localhost", "database", "username", "password", "type", "")).to.throw(Error, "invalid name parameter");</code></pre>
</td></tr>
<tr><td>Scope</td><td>Datasource::copy</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-91</td><td>

- **Progettazione**: verifica il funzionamento del metodo statico copy quando *datasource* è corretto;
- **Caso**:
  - **Input**: *datasource* corretto;
  - **Risultato atteso**: una copia del datasource.
- **Procedura**:

```
1 expect(() => DataSource.copy(new DataSource("http://localhost↩
    :8086").getUrl()).to.equal("http://localhost:8086");
```

</td></tr>
<tr><td>TU-92</td><td>

- **Progettazione**: verifica il funzionamento del metodo statico copy quando *datasource* è null;
- **Caso**:
  - **Input**: null;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid datasource parameter"*.
- **Procedura**:

```
1 expect(() => DataSource.copy(null).to.throw(Error, "invalid ↩
    datasource parameter");
```

</td></tr>
<tr><td>Scope</td><td>Datasource::clone</td></tr>
<tr><td>TU-94</td><td>

- **Progettazione**: verifica il funzionamento del metodo clone;
- **Caso**:
  - **Input**: un datasource ben formato;
  - **Risultato atteso**: un clone del datasource di invocazione.
- **Procedura**:

```
1 expect(new DataSource("http://localhost", "database").clone().↩
    getUrl()).to.equal("http://localhost:");
```

</td></tr>
<tr><td>Scope</td><td>DataSource::getDatabase</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-95</td><td>

- **Progettazione**: verifica il funzionamento del metodo getDatabase;
- **Caso**:
  - **Input**: un datasource ben formato;
  - **Risultato atteso**: la stringa che rappresenta il database passato.
- **Procedura**:

```
1 expect(new DataSource("http://localhost:8086", "database", "↩
    username", "password", "type", "name", 1).getDatabase()).↩
    to.equal("database");
```

</td></tr>
<tr><td>Scope</td><td>DataSource::getGrafanaDatasourceId</td></tr>
<tr><td>TU-96</td><td>

- **Progettazione**: verifica il funzionamento del metodo getGrafanaDatasourceId;
- **Caso**:
  - **Input**: un datasource ben formato;
  - **Risultato atteso**: il getGrafanaDatasourceId passato.
- **Procedura**:

```
1 expect(new DataSource("http://localhost:8086", "database", "↩
    username", "password", "type", "name", 1).↩
    getGrafanaDatasourceId()).to.equal(1);
```

</td></tr>
<tr><td>Scope</td><td>DataSource::getHost</td></tr>
<tr><td>TU-97</td><td>

- **Progettazione**: verifica il funzionamento del metodo getHost;
- **Caso**:
  - **Input**: un datasource ben formato;
  - **Risultato atteso**: la stringa che rappresenta l'host passato.
- **Procedura**:

```
1 expect(new DataSource("http://localhost:8086", "database", "↩
    username", "password", "type", "name", 1).getHost()).to.↩
    equal("http://localhost");
```

</td></tr>
<tr><td>Scope</td><td>DataSource::getName</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-98</td><td>

Progettazione: verifica il funzionamento del metodo getName;
Caso:

Input: un datasource ben formato;
Risultato atteso: la stringa che rappresenta il name passato.

Procedura:

<pre>1 expect(new DataSource("http://localhost:8086", "database", "↩
    username", "password", "type", "name", 1).getName()).to.↩
    equal("name");</pre>
</td></tr>
<tr><td>Scope</td><td>DataSource::getPassword</td></tr>
<tr><td>TU-99</td><td>

Progettazione: verifica il funzionamento del metodo getPassword;
Caso:

Input: un datasource ben formato;
Risultato atteso: la stringa che rappresenta la password passata.

Procedura:

<pre>1 expect(new DataSource("http://localhost:8086", "database", "↩
    username", "password", "type", "name", 1).getPassword()).↩
    to.equal("password");</pre>
</td></tr>
<tr><td>Scope</td><td>DataSource::getPort</td></tr>
<tr><td>TU-100</td><td>

Progettazione: verifica il funzionamento del metodo getPort;
Caso:

Input: un datasource ben formato;
Risultato atteso: la stringa che rappresenta la porta passata.

Procedura:

<pre>1 expect(new DataSource("http://localhost:8086", "database", "↩
    username", "password", "type", "name", 1).getPort()).to.↩
    equal("8086");</pre>
</td></tr>
<tr><td>Scope</td><td>DataSource::getType</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-101</td><td>

- **Progettazione**: verifica il funzionamento del metodo getType;
- **Caso**:
  - **Input**: un datasource ben formato;
  - **Risultato atteso**: la stringa che rappresenta la porta passata.
- **Procedura**:

```
1 expect(new DataSource("http://localhost:8086", "database", "↩
    username", "password", "type", "name", 1).getType()).to.↩
    equal("type");
```

</td></tr>
<tr><td>Scope</td><td>DataSource::getUrl</td></tr>
<tr><td>TU-102</td><td>

- **Progettazione**: verifica il funzionamento del metodo getUrl;
- **Caso**:
  - **Input**: un datasource ben formato;
  - **Risultato atteso**: la stringa che rappresenta l'url passato.
- **Procedura**:

```
1 expect(new DataSource("http://localhost:8086", "database", "↩
    username", "password", "type", "name", 1).getUrl()).to.↩
    equal("http://localhost:8086");
```

</td></tr>
<tr><td>Scope</td><td>DataSource::getUsername</td></tr>
<tr><td>TU-103</td><td>

- **Progettazione**: verifica il funzionamento del metodo getUsername;
- **Caso**:
  - **Input**: un datasource ben formato;
  - **Risultato atteso**: la stringa che rappresenta l'username passato.
- **Procedura**:

```
1 expect(new DataSource("http://localhost:8086", "database", "↩
    username", "password", "type", "name", 1).getUsername()).↩
    to.equal("username");
```

</td></tr>
<tr><td>Scope</td><td>DataSource::cloneWithDB</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-103</td><td>

Progettazione: verifica il funzionamento del metodo cloneWithDB nel caso in cui la stringa passata sia vuota;
Caso:

Input: una stringa vuota;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid database parameter".

Procedura:

<pre>1 expect(datasource.cloneWithDB("")).to.throw(Error, "invalid ↩
    database parameter");</pre>
</td></tr>
<tr><td>TU-105</td><td>

Progettazione: verifica il funzionamento del metodo cloneWithDB nel caso in cui la stringa passata sia corretta;
Caso:

Input: una stringa corretta;
Risultato atteso: un datasource corretto.

Procedura:

<pre>1 expect(datasource.cloneWithDB("database")).to.not.throw(Error,↩
    "invalid database parameter");</pre>
</td></tr>
<tr><td>Scope</td><td>NetReader::constructor</td></tr>
<tr><td>TU-106</td><td>

Progettazione: verifica il funzionamento del costruttore di NetReader() con parametro non definito;
Caso:

Input: parametro non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid parameter".

Procedura:

<pre>1 let networkAdapter: ConcreteNetworkAdapter;
2 expect(()=> new NetReader(networkAdapter)).to.throw(Error, "↩
    invalid parameter");</pre>
</td></tr>
<tr><td>Scope</td><td>NetReader::read</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-107</td><td>

- **Progettazione**: verifica il funzionamento di *read()*;
- **Caso**:
  - **Input**: oggetto *NetReader*;
  - **Risultato atteso**: "Example".
- **Procedura**:

```
const networkReader: NetReader = new NetReader(network);
networkReader.connectNode("Example", new DataSource("http://localhost:8086/"), "SELECT Percent_DPC_Time FROM win_cpu");
networkReader.read().then(function(result){
expect(result.buildIterator().next().value.getNode().getName()).to.equal("Example");
```

</td></tr>
<tr><td>Scope</td><td>NetReader::connectNode</td></tr>
<tr><td>TU-108</td><td>

- **Progettazione**: verifica il funzionamento di *connectNode()* con *node* non definito;
- **Caso**:
  - **Input**: *node* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid node parameter"*.
- **Procedura**:

```
let node: string;
expect(() => networkReader.connectNode(node, new DataSource("http://localhost"), "query")).to.throw(Error, "invalid node parameter");
```

</td></tr>
<tr><td>TU-109</td><td>

- **Progettazione**: verifica il funzionamento di *connectNode()* con *datasource* non definito;
- **Caso**:
  - **Input**: *datasource* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid datasource parameter"*.
- **Procedura**:

```
let dataS: DataSource;
expect(() => networkReader.connectNode("node", dataS, "query").to.throw(Error, "invalid datasource parameter");
```

</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-110</td><td>

Progettazione: verifica il funzionamento di connectNode() con query non definito;
Caso:

Input: query non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid query parameter".

Procedura:

<pre>
let query: string;
expect(() => networkReader.connectNode("node", new DataSource("http://localhost"), query)).to.throw(Error, "invalid query parameter");
</pre>
</td></tr>
<tr><td>TU-111</td><td>

Progettazione: verifica il funzionamento di connectNode() con node vuoto;
Caso:

Input: node vuoto;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid node parameter".

Procedura:

<pre>
expect(() => networkReader.connectNode("", new DataSource("http://localhost"), "query")).to.throw(Error, "invalid node parameter");
</pre>
</td></tr>
<tr><td>TU-112</td><td>

Progettazione: verifica il funzionamento di connectNode() con query vuoto;
Caso:

Input: query vuoto;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid query parameter".

Procedura:

<pre>
expect(() => networkReader.connectNode("node", new DataSource("http://localhost"), "")).to.throw(Error, "invalid query parameter");
</pre>
</td></tr>
<tr><td>Scope</td><td>NetReader::disconnectNode</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-113</td><td>

- **Progettazione**: verifica il funzionamento di *disconnectNode()* con *node* non definito;
- **Caso**:
  - **Input**: *node* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid node parameter"*.
- **Procedura**:

```
expect(() => networkReader.disconnectNode(node)).to.throw(↩
    Error, "[7DOS G&B][NetReader]disconnectNode - Invalid node↩
    ");
```

</td></tr>
<tr><td>Scope</td><td>ReusableReadClientPool::acquireReusable</td></tr>
<tr><td>TU-114</td><td>

- **Progettazione**: verifica il funzionamento di *acquireReusable()* con *datasource* non definito;
- **Caso**:
  - **Input**: *datasource* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid datasource parameter"*.
- **Procedura**:

```
let datasource: DataSource;
expect(() => ReusableReadClientPool.getInstance().↩
    acquireReusable(datasource)).to.throw(Error, "invalid ↩
    datasource parameter");
```

</td></tr>
<tr><td>TU-115</td><td>

- **Progettazione**: verifica il funzionamento di *acquireReusable()* con *datasource* definito;
- **Caso**:
  - **Input**: *datasource* definito;
  - **Risultato atteso**: TRUE.
- **Procedura**:

```
let datasource: DataSource = new DataSource("http://localhost/↩
    ");
let JSONstr: string = JSON.stringify(ReusableReadClientPool.↩
    getInstance().acquireReusable(datasource));
expect(JSONstr.includes("http://localhost/")).to.equal(true);
```

</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-116</td><td>

- **Progettazione**: verifica il funzionamento di *acquireReusable()* con *to_remove* non definito;
- **Caso**:
  - **Input**: *to_remove* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid to_remove parameter"*.
- **Procedura**:

```
let to_remove: ReadClient;
expect(() => ReusableReadClientPool.getInstance().releseReusable(to_remove)).to.throw(Error, "invalid to_remove parameter");
```

</td></tr>
<tr><td>Scope</td><td>CalcResultAggregate::constructor</td></tr>
<tr><td>TU-117</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *CalcResultAggregate()* con *collection* non definito;
- **Caso**:
  - **Input**: *collection* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid collection parameter"*.
- **Procedura**:

```
let collection: Array<CalcResult>;
expect(()=> new CalcResultAggregate(collection)).to.throw(Error, "invalid collection parameter");
```

</td></tr>
<tr><td>TU-118</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *CalcResultAggregate()* con *collection* vuoto;
- **Caso**:
  - **Input**: *collection* vuoto;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid collection parameter"*.
- **Procedura**:

```
  let collection:Array<CalcResult> = new Array<CalcResult>();
  expect(()=> new CalcResultAggregate(collection)).to.throw(Error, "invalid collection parameter");
```

</td></tr>
<tr><td>Scope</td><td>CalcResultAggregate::createIterator</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-119</td><td>

- **Progettazione**: verifica il funzionamento di *createIterator()*;
- **Caso**:
  - **Input**: oggetto *CalcResultItem*, oggetto *CalcResult*;
  - **Risultato atteso**: "nodo".
- **Procedura**:

```
let collection: Array<CalcResult> = new Array<CalcResult>();
let itemsArray: Array<CalcResultItem> = new Array<CalcResultItem>();
itemsArray.push(new CalcResultItem("stringa", 1));
collection.push(new CalcResult("nodo", itemsArray));
expect(new CalcResultAggregate(collection).createIterator().next().value.getNodeName()).to.equal("nodo");
```

</td></tr>
<tr><td>Scope</td><td>CalcResultItem::constructor</td></tr>
<tr><td>TU-120</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *CalcResultItem()* con parametri nulli;
- **Caso**:
  - **Input**: parametri nulli;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid null parameter"*.
- **Procedura**:

```
expect(() => new CalcResultItem(null,null)).to.throw(Error, "invalid null parameter");
```

</td></tr>
<tr><td>TU-121</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *CalcResultItem()* con probabilità negativa;
- **Caso**:
  - **Input**: probabilità < 0;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid probability parameter"*.
- **Procedura**:

```
expect(() => new CalcResultItem("n1",-5)).to.throw(Error, "invalid probability parameter");
```

</td></tr>
</table>

Test	Specifica
TU-122	• **Progettazione**: verifica il funzionamento del costruttore di *CalcResultItem()* con probabilità maggiore di uno; • **Caso**: – **Input**: probabilità > 1; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid probability parameter"*. • **Procedura**: `expect(() => new CalcResultItem("n1",1.1)).to.throw(Error, "invalid parameter");`
Scope	CalcResultItem::getValueName
TU-123	• **Progettazione**: verifica il funzionamento di *getValueName()*; • **Caso**: – **Input**: oggetto *CalcResultItem*; – **Risultato atteso**: "n1". • **Procedura**: `expect(new CalcResultItem("n1",0.5).getValueName()).to.equal("n1");`
Scope	CalcResultItem::getProbValue
TU-124	• **Progettazione**: verifica il funzionamento di *getProbValue()*; • **Caso**: – **Input**: oggetto *CalcResultItem*; – **Risultato atteso**: "0.5". • **Procedura**: `expect(new CalcResultItem("n1",0.5).getProbValue()).to.equal(0.5);`
Scope	CalcResult::constructor

Test	Specifica
TU-125	• **Progettazione**: verifica il funzionamento del costruttore di *CalcResult()* con parametri nulli; • **Caso**: – **Input**: parametri nulli; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid null parameter"*. • **Procedura**: `expect(() => new CalcResult(null,null)).to.throw(Error, "invalid null parameter");`

TU-126	• **Progettazione**: verifica il funzionamento del costruttore di *CalcResult()* con item nulli in array;
	• **Caso**:
	– **Input**: item nulli in array;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid array parameter"*.
	• **Procedura**:
	`let calcItems:Array<CalcResultItem>=new Array<CalcResultItem>();`
	`calcItems.push(new CalcResultItem("1",0.5));`
	`calcItems.push(new CalcResultItem("2",0.6));`
	`calcItems.push(new CalcResultItem("3",0.3));`
	`calcItems.push(null);`
	`expect(() => new CalcResult(null,calcItems)).to.throw(Error, "invalid array parameter");`

TU-127	• **Progettazione**: verifica il funzionamento del costruttore di *CalcResult()*;
	• **Caso**:
	– **Input**: oggetto *CalcResult*;
	– **Risultato atteso**: nessuna eccezione viene lanciata.
	• **Procedura**:
	`let calcItems:Array<CalcResultItem>=new Array<CalcResultItem>();`
	`calcItems.push(new CalcResultItem("1",0.5));`
	`expect(() => new CalcResult("n1",calcItems)).to.not.throw(Error);`

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>Scope</td><td>CalcResult::getNodeName</td></tr>
<tr><td>TU-128</td><td>

- **Progettazione**: verifica il funzionamento di *getNodeName()*;
- **Caso**:
  - **Input**: oggetto *CalcResultItem*;
  - **Risultato atteso**: "n1".
- **Procedura**:

```
let calcItems:Array<CalcResultItem>=new Array<CalcResultItem>();
calcItems.push(new CalcResultItem("1",0.5));
expect(new CalcResult("n1",calcItems).getNodeName()).to.equal("n1");
```

</td></tr>
<tr><td>Scope</td><td>CalcResult::getValueProbs</td></tr>
<tr><td>TU-129</td><td>

- **Progettazione**: verifica il funzionamento di *getValueProbs()*;
- **Caso**:
  - **Input**: oggetto *calcResult*;
  - **Risultato atteso**: ["n5", 0.5], ["n3", 0.6].
- **Procedura**:

```
let calcItems:Array<CalcResultItem>=new Array<CalcResultItem>();
calcItems.push(new CalcResultItem("n5",0.5));
calcItems.push(new CalcResultItem("n3",0.6));
let calcResult=new CalcResult("n1",calcItems);
let probs=calcResult.getValueProbs();
expect(probs[0].getValueName()).to.equals("n5");
expect(probs[0].getProbValue()).to.equals(0.5);
expect(probs[1].getValueName()).to.equals("n3");
expect(probs[1].getProbValue()).to.equals(0.6);
```

</td></tr>
<tr><td>Scope</td><td>InputResultAggregate::constructor</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-130</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *InputResultAggregate()* con *collection* non definito;
- **Caso**:
  - **Input**: *collection* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid collection parameter"*.
- **Procedura**:

```
let collection: Array<InputResult>;
expect(() => new InputResultAggregate(collection)).to.throw(↩
    Error, "invalid collection parameter");
```

</td></tr>
<tr><td>Scope</td><td>InputResultAggregate::buildIterator</td></tr>
<tr><td>TU-131</td><td>

- **Progettazione**: verifica il funzionamento di *buildIterator()*;
- **Caso**:
  - **Input**: oggetto *InputResult*;
  - **Risultato atteso**: "valore".
- **Procedura**:

```
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const values: Array<AbstractValue> = new Array<AbstractValue↩
    >();
values.push(new BoolValue(true, "prova"));
const nodeName: ConcreteNodeAdapter = new ConcreteNodeAdapter(↩
    n1, values);
const currentValue: string = "valore";
const collection: Array<InputResult> = new Array<InputResult↩
    >();
collection.push(new InputResult(nodeName, currentValue));
expect(new InputResultAggregate(collection).buildIterator().↩
    next().value.getCurrentValue()).to.equal("valore");
```

</td></tr>
<tr><td>Scope</td><td>InputResult::constructor</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-132</td><td>

Progettazione: verifica il funzionamento del costruttore di InputResult() con nodeName non definito;
Caso:

Input: nodeName non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid nodeName parameter".

Procedura:


```
let nodeName: ConcreteNodeAdapter;
expect(() => new InputResult(nodeName, "valore")).to.throw(↩
    Error, "invalid nodeName parameter");
```
</td></tr>
<tr><td>TU-133</td><td>

Progettazione: verifica il funzionamento del costruttore di InputResult() con currentValue non definito;
Caso:

Input: currentValue non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid currentValue parameter".

Procedura:


```
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const values: Array<AbstractValue> = new Array<AbstractValue↩
    >();
values.push(new BoolValue(true, "prova"));
const nodeName: ConcreteNodeAdapter = new ConcreteNodeAdapter(↩
    n1, values);
let currentValue: string;
expect(() => new InputResult(nodeName, currentValue)).to.throw↩
    (Error, "invalid currentValue parameter");
```
</td></tr>
<tr><td>Scope</td><td>InputResult::getNode</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-134</td><td>

Progettazione: verifica il funzionamento di getNode();
Caso:

Input: oggetto InputResult;
Risultato atteso: "n1".

Procedura:

<pre>
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const values: Array<AbstractValue> = new Array<AbstractValue>();
values.push(new BoolValue(true, "prova"));
const nodeName: ConcreteNodeAdapter = new ConcreteNodeAdapter(n1, values);
const currentValue: string = "valore";
const name: string = new InputResult(nodeName, currentValue).getNode().getName();
expect(name).to.equal("n1");
</pre>
</td></tr>
<tr><td>Scope</td><td>InputResult::getCurrentValue</td></tr>
<tr><td>TU-135</td><td>

Progettazione: verifica il funzionamento di getCurrentValue();
Caso:

Input: oggetto InputResult;
Risultato atteso: "valore".

Procedura:

<pre>
const graph: JGraph = jsbayes.newGraph();
const n1: JNode = graph.addNode("n1", ["true", "false"]);
const values: Array<AbstractValue> = new Array<AbstractValue>();
values.push(new BoolValue(true, "prova"));
const nodeName: ConcreteNodeAdapter = new ConcreteNodeAdapter(n1, values);
const currentValue: string = "valore";
const getCurrent: string = new InputResult(nodeName, currentValue).getCurrentValue();
expect(getCurrent).to.equal("valore");
</pre>
</td></tr>
<tr><td>Scope</td><td>NetUpdater::constructor</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-136</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *NetUpdater()* con *network* non definito;
- **Caso**:
  - **Input**: *network* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid network parameter"*.
- **Procedura**:

```
let network: NetworkAdapter;
expect(() => new NetUpdater(network)).to.throw(Error, "invalid network parameter");
```

</td></tr>
<tr><td>Scope</td><td>NetUpdater::updateNet</td></tr>
<tr><td>TU-137</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *updateNet()* con *InputResultAggregate* non definito;
- **Caso**:
  - **Input**: *InputResultAggregate* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"invalid InputResultAggregate parameter"*.
- **Procedura**:

```
const network: NetworkAdapter = new ConcreteNetworkFactory().parseNetwork(jsonString, jsonSchemaString);
let results: InputResultAggregate;
const networkUpdater: NetUpdater = new NetUpdater(network);
expect(()=> networkUpdater.updateNet(results)).to.throw(Error,"invalid InputResultAggregate parameter");
```

</td></tr>
<tr><td>Scope</td><td>SingleNetWriter::constructor</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-138</td><td>

Progettazione: verifica il funzionamento del costruttore di SingleNetWriter() con client non definito;
Caso:

Input: client non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid client parameter".

Procedura:


```
let client: WriteClient;
expect(()=> new SingleNetWriter(client)).to.throw(Error, "↩
    invalid client parameter");
```
</td></tr>
<tr><td>TU-139</td><td>

Progettazione: verifica il funzionamento del costruttore di SingleNetWriter() con client definito;
Caso:

Input: oggetto SingleNetWriter;
Risultato atteso: diverso da "null".

Procedura:


```
new ConcreteWriteClientFactory().makeInfluxWriteClient(
"http://localhost", "8086", "prova", ["root", "root"]
).then(function(writeClient){
    const singleWriter: SingleNetWriter = new SingleNetWriter(↩
        writeClient);
    expect(singleWriter).to.not.equal(null);
});
```
</td></tr>
<tr><td>Scope</td><td>SingleNetWriter::write</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-140</td><td>

Progettazione: verifica il funzionamento del costruttore di write() con calcData non definito;
Caso:

Input: calcData non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid calcData parameter".

Procedura:

<pre>
const idb: InfluxDB = new InfluxDB();
const client: WriteClient = new InfluxWriteClient("", "", idb);
const singleWriter: SingleNetWriter = new SingleNetWriter(client);
let calc: CalcResultAggregate;
singleWriter.write(calc).then(function (){})
   .catch(function (e){
      expect(<Error> e.toString()).to.equal("Error: invalid calcData parameter");
});
</pre>
</td></tr>
<tr><td>Scope</td><td>NetManager::constructor</td></tr>
<tr><td>TU-141</td><td>

Progettazione: verifica il funzionamento del costruttore di NetManager() con reader non definito;
Caso:

Input: reader non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid reader parameter".

Procedura:

<pre>
let unReader: NetReader;
expect(() =>new NetManager(unReader, updater, writer)).to.throw(Error, "invalid reader parameter");
</pre>
</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-142</td><td>

Progettazione: verifica il funzionamento del costruttore di NetManager() con updater non definito;
Caso:

Input: updater non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid updater parameter".

Procedura:

<pre>
let unUpdater: NetUpdater;
expect(() =>new NetManager(reader, unUpdater, writer)).to.↩
    throw(Error, "invalid updater parameter");
</pre>
</td></tr>
<tr><td>TU-143</td><td>

Progettazione: verifica il funzionamento del costruttore di NetManager() con writer non definito;
Caso:

Input: writer non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid writer parameter".

Procedura:

<pre>
let unWriter: NetWriter;
expect(() => new NetManager(reader, updater, unWriter)).to.↩
    throw(Error, "invalid writer parameter");
</pre>
</td></tr>
<tr><td>Scope</td><td>NetManager::updateNet</td></tr>
<tr><td>TU-144</td><td>

Progettazione: verifica il funzionamento di updateNet();
Caso:

Input: oggetto NetManager;
Risultato atteso: TRUE.

</td></tr>
<tr><td>Scope</td><td>InfluxWriteClient::constructor</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-145</td><td>

Progettazione: verifica il funzionamento del costruttore di InfluxWriteClient() con dsn non definito;
Caso:

Input: dsn non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid dsn parameter".

Procedura:

<pre>
let dsn: string;
expect(() => new InfluxWriteClient(dsn, "prova", influx)).to.throw(Error, "invalid dsn parameter");
</pre>
</td></tr>
<tr><td>TU-146</td><td>

Progettazione: verifica il funzionamento del costruttore di InfluxWriteClient() con defaultDB non definito;
Caso:

Input: defaultDB non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid defaultDB parameter".

Procedura:

<pre>
let defaultDB: string;
expect(() => new InfluxWriteClient("http://localhost:8086/", defaultDB, influx)).to.throw(Error, "invalid defaultDB parameter");
</pre>
</td></tr>
<tr><td>TU-147</td><td>

Progettazione: verifica il funzionamento del costruttore di InfluxWriteClient() con influx non definito;
Caso:

Input: influx non definito;
Risultato atteso: il sistema lancia un'eccezione di tipo Error mostrando il messaggio "invalid influx parameter".

Procedura:

<pre>
let unInflux: InfluxDB;
expect(() => new InfluxWriteClient("http://localhost:8086/", "prova", unInflux)).to.throw(Error, "invalid influx parameter");
</pre>
</td></tr>
<tr><td>Scope</td><td>InfluxWriteClient::getAddress</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>TU-148</td><td>

- **Progettazione**: verifica il funzionamento di *getAddress()*;
- **Caso**:
  - **Input**: oggetto *InfluxWriteClient*;
  - **Risultato atteso**: "http://localhost:8086/".
- **Procedura**:

```
expect(new InfluxWriteClient("http://localhost:8086/", "prova", influx).getAddress()).to.equal("http://localhost:8086/");
```

</td></tr>
<tr><td>Scope</td><td>InfluxWriteClient::getDefaultDB</td></tr>
<tr><td>TU-149</td><td>

- **Progettazione**: verifica il funzionamento di *getDefaultDB()*;
- **Caso**:
  - **Input**: oggetto *InfluxWriteClient*;
  - **Risultato atteso**: "prova".
- **Procedura**:

```
expect(new InfluxWriteClient("http://localhost:8086/", "prova", influx).getDefaultDB()).to.equal("prova");
```

</td></tr>
<tr><td>Scope</td><td>InfluxWriteClient::writeBatchData</td></tr>
<tr><td>TU-150</td><td>

- **Progettazione**: verifica il funzionamento di *writeBatchData()*;
- **Caso**:
  - **Input**: oggetto *InfluxWriteClient*;
  - **Risultato atteso**: TRUE.
- **Procedura**:

```
const influxWriter: InfluxWriteClient = new InfluxWriteClient("http://localhost:8086/", "prova", influx);
let calcArray: Array<CalcResult> = new Array<CalcResult>();
let calcItemArray: Array<CalcResultItem> = new Array<CalcResultItem>();
calcItemArray.push(new CalcResultItem("field1", 0.5));
calcArray.push(new CalcResult("burglary", calcItemArray));
let toBeWritten: CalcResultAggregate = new CalcResultAggregate(calcArray);
influxWriter.writeBatchData(toBeWritten).then(function(){
   expect(true).to.equal(true);
});
```

</td></tr>
</table>

<table>
<tr><th>Test</th><th>Specifica</th></tr>
<tr><td>Scope</td><td>InfluxWriteClient::writePointData</td></tr>
<tr><td>TU-151</td><td>

- **Progettazione**: verifica il funzionamento di *writePointData()*;
- **Caso**:
  - **Input**: oggetto *InfluxWriteClient*;
  - **Risultato atteso**: TRUE.
- **Procedura**:

```
const influxWriter: InfluxWriteClient = new InfluxWriteClient("http://localhost:8086/", "prova", influx);
let calcItemArray: Array<CalcResultItem> = new Array<CalcResultItem>();
calcItemArray.push(new CalcResultItem("field1", 0.5));
influxWriter.writePointData(new CalcResult("burglary", calcItemArray)).then(function(){
   expect(true).to.equal(true);
});
```

</td></tr>
<tr><td>Scope</td><td>WriteClientFactory::constructor</td></tr>
<tr><td>TU-152</td><td>

- **Progettazione**: verifica il funzionamento del costruttore di *WriteClientFactory()* con *host* non definito;
- **Caso**:
  - **Input**: *host* non definito;
  - **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Error: invalid host parameter"*.
- **Procedura**:

```
let host: string;
new ConcreteWriteClientFactory().makeInfluxWriteClient(
host, "something", "something else", ["admin", "password"]
).then(function(){}).catch(function(e){
expect(<Error> e.toString()).to.equal("Error: invalid host parameter");
});
```

</td></tr>
</table>

Test	Specifica

TU-153	• **Progettazione**: verifica il funzionamento del costruttore di *WriteClientFactory()* con *port* non definito;
	• **Caso**:
	– **Input**: *port* non definito;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Error: invalid port parameter"*.
	• **Procedura**:
	`let port: string;`
	`new ConcreteWriteClientFactory().makeInfluxWriteClient("something", port, "something else", ["admin", "password"]).then(function(){}).catch(function(e){`
	`expect(<Error> e.toString()).to.equal("Error: invalid port parameter");`
	`});`

TU-154	• **Progettazione**: verifica il funzionamento del costruttore di *WriteClientFactory()* con *defaultDB* non definito;
	• **Caso**:
	– **Input**: *defaultDB* non definito;
	– **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Error: invalid defaultDB parameter"*.
	• **Procedura**:
	`let defaultDB: string;`
	`new ConcreteWriteClientFactory().makeInfluxWriteClient("something", "something else", defaultDB, ["admin", "password"]).then(function(){}).catch(function(e){`
	`expect(<Error> e.toString()).to.equal("Error: invalid defaultDB parameter");`
	`});`

Test	Specifica
TU-155	• **Progettazione**: verifica il funzionamento del costruttore di *WriteClientFactory()* con *host* vuoto; • **Caso**: – **Input**: *host* vuoto; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Error: invalid host parameter"*. • **Procedura**: `new ConcreteWriteClientFactory().makeInfluxWriteClient("", "something", "something else", ["admin", "password"]).then(function(){}).catch(function(e){ expect(<Error> e.toString()).to.equal("Error: invalid host parameter"); });`
TU-156	• **Progettazione**: verifica il funzionamento del costruttore di *WriteClientFactory()* con *port* vuoto; • **Caso**: – **Input**: *port* vuoto; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Error: invalid port parameter"*. • **Procedura**: `new ConcreteWriteClientFactory().makeInfluxWriteClient("something", "", "something else", ["admin", "password"]).then(function(){}).catch(function(e){ expect(<Error> e.toString()).to.equal("Error: invalid port parameter"); });`

Test	Specifica
TU-157	• **Progettazione**: verifica il funzionamento del costruttore di *WriteClientFactory()* con *deafultDB* vuoto; • **Caso**: – **Input**: *deafultDB* vuoto; – **Risultato atteso**: il sistema lancia un'eccezione di tipo *Error* mostrando il messaggio *"Error: invalid deafultDB parameter"*. • **Procedura**: `new ConcreteWriteClientFactory().makeInfluxWriteClient("something", "something else", "", ["admin", "password"]).then(function(){}).catch(function(e){ expect(<Error> e.toString()).to.equal("Error: invalid deafultDB parameter"); });`

Tabella 16: Specifica test di unità