8Lab Solutions - Progetto "Soldino"

Piano di Qualifica

Versione	1.0.0
Approvazione	Paolo Pozzan
Redazione	Federico Bicciato
	Sara Feltrin
	Paolo Pozzan
Verifica	Samuele Giuliano Piazzetta
	Mattia Bolzanella
Stato	Approvato
Uso	Esterno
Destinato a	8Lab Solutions
	Red Babel
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin

Descrizione

Questo documento descrive le operazioni di verifica e validazione seguite durante lo svolgimento del progetto *Soldino*.

8labsolutions@gmail.com

Diario delle modifiche

Versione	Data	Nominativo	Ruolo	Descrizione
1.0.0	2019-10-12	Paolo Pozzan	*Responsabile di Progetto*	Approvazione del documento per RR.
0.2.0	2019-10-10	Samuele Giuliano Piazzetta	*Verificatore*	Verifica del documento.
0.1.2	2019-01-03	Federico Bicciato	*Progettista*	Stesura Appendice B.
0.1.1	2019-01-02	Sara Feltrin	*Progettista*	Stesura §4.
0.1.0	2018-10-30	Mattia Bolzonella	*Verificatore*	Verifica del documento.
0.0.6	2018-12-30	Sara Feltrin	*Progettista*	Stesura Appendice C.
0.0.5	2018-12-29	Paolo Pozzan	*Progettista*	Stesura Appendice A.
0.0.4	2018-12-28	Federico Bicciato	*Progettista*	Stesura §2.
0.0.3	2018-12-28	Paolo Pozzan	*Progettista*	Sistemata struttura del documento e inizia stesura §3.
0.0.2	2018-12-27	Sara Feltrin	*Progettista*	Sistemata struttura del documento e inizia stesura §4.
0.0.1	2018-12-26	Paolo Pozzan	*Progettista*	Creato documento LaTeX e stesura §1.

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Premessa

Il Piano di Qualifica è un documento su cui si prevede di lavorare l'intera durata del progetto. Molti dei contenuti del documento sono di natura instabile. Ad esempio, molte delle metriche scelte non sono applicabili nella fase iniziale, e solo con il loro utilizzo pratico si può valutarne l'effettiva utilità. Anche i processi selezionati possono essere soggetti a cambiamenti, rivelandosi insufficienti o inadeguati agli scopi del progetto e al modo di lavorare del team.
Alcune parti del documento sono prodotte in fasi temporali successive, come l'appendice sul resoconto delle verifiche.
Per tutte queste ragioni, il documento è prodotto in maniera incrementale, e i suoi contenuti iniziali sono da considerarsi incompleti: subiranno significative aggiunte e modifiche nel tempo.

1.2 Scopo del documento

Questo documento ha lo scopo di mostrare le strategie di verifica e validazione adottate al fine di garantire la qualità di prodotto e di processo. Per raggiungere questo obiettivo viene applicato un sistema di verifica continua sui processi in corso e sulle attività svolte. In questo modo è quindi possibile rilevare e correggere all'istante eventuali anomalie, riducendo al minimo lo spreco delle risorse.

1.3 Scopo del prodotto

Lo scopo del prodotto è quello di realizzare un software, in particolare un sito internet, che consenta il monitoraggio automatico dell'IVA, ovvero assiste il governo e gli utenti nell'esecuzione di operazioni come liquidazione, versamento e rimborso, e permetta l'acquisto di oggetti tramite una valuta denominata "Cubit"$_G$.

1.4 Ambiguità

All'interno dei documenti alcuni termini presentano significati ambigui a seconda del contesto, fraintendibili, o che necessitano di una descrizione più approfondita. Per evitare questa ambiguità è stato creato il documento "Glossario" volto a fare chiarezza, ponendo a fianco di ogni termine il suo preciso significato. Questi termini sono pertanto marchiati con una "G" a pedice per ogni loro occorrenza all'interno di tutti i documenti.

1.5 Riferimenti

1.5.1 Riferimenti normativi

- **Capitolato d'appalto C6 - Soldino, piattaforma Ethereum per pagamenti IVA**: https://www.math.unipd.it/~tullio/IS-1/2018/Progetto/C6.pdf

1.5.2 Riferimenti informativi

- **ISO/IEC 9126**: https://en.wikipedia.org/wiki/ISO/IEC_9126

- **ISO/IEC 12207**:
  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
- **Indice di Gulpease**:
  https://it.wikipedia.org/wiki/Indice_Gulpease
- **Schedule Variance e metriche correlate**:
  https://www.smartsheet.com/hacking-pmp-how-calculate-schedule-variance

2 Qualità di processo

Per ricercare qualità nello svolgimento del progetto si adoperano dei processi. Inizialmente, tali processi sono stati scelti tra quelli proposti nello standard ISO/IEC/IEEE 12207:1995. Successivamente sono stati semplificati o adattati secondo le esigenze: in alcuni casi (analisi dei requisiti, progettazione dell'architettura, progettazione di dettaglio, pianificazione) le attività sono trattate alla pari di processi, vista la loro importanza nell'esito del progetto.
Il risultato sono i processi esposti a seguito.

2.1 Processi Primari

2.2 Processo di Sviluppo

Il processo contiene tutte le attività tipiche dello sviluppo software: alcune di esse sono molto importanti per cui vengono approfondite: si tratta dell'analisi dei requisiti, la progettazione architetturale, la progettazione di dettaglio.

2.2.1 Analisi dei Requisiti

Durante l'analisi le informazioni ottenute dalle varie fonti sono trasformate in forma di casi d'uso e requisiti. Questa forma fornisce una descrizione dettagliata del sistema e definisce il funzionamento e le caratteristiche di ogni sua parte.

2.2.1.1 Obiettivi

- formulare la definizione di casi d'uso e requisiti;
- ottenere la loro approvazione;
- tracciare il loro cambiamento nel tempo.

2.2.1.2 Strategia

- considerare lo scopo del progetto e le richieste degli stakeholder;
- esprimere ciò in forma di requisiti, classificati in obbligatori, desiderabili e opzionali;
- valutare il corpo dei requisiti e negoziare cambiamenti se necessario;
- ottenere la loro approvazione da parte del proponente.

Disporre del tracciamento dei requisiti del sistema.

2.2.1.3 Metriche

Percentuale di requisiti obbligatori soddisfatti - PROS Indica appunto la percentuale di requisiti obbligatori soddisfatti.

- misurazione: valore percentuale: $PROS = \frac{requisiti\ obbligatori\ soddisfatti}{requisiti\ obbligatori\ totali}$;
- valore preferibile: 100%;
- valore accettabile: 100%.

2.2.2 Progettazione dell'Architettura

Progettare l'architettura significa tradurre i requisiti in un modello architetturale del sistema. Il modello così ottenuto è ad alto livello di dettaglio: esso dà una visione del sistema come composizione di parti, utili a capire il funzionamento del sistema, ma a grossa granularità quindi non ancora realizzabili nella pratica.
Nell'attività vanno considerati in particolare il tipo di software da produrre, le caratteristiche desiderate e i suoi requisiti non funzionali per scegliere l'architettura più adatta.

2.2.2.1 Obiettivi

- agevolare la realizzazione fornendo un'architettura adeguata;
- perseguire la correttezza per costruzione;
- comprendere meglio il sistema scomponendolo in parti;
- progettare adoperando stili architetturali e design pattern.

2.2.2.2 Strategia

- valutare il tipo di software da produrre, le caratteristiche desiderate, i casi d'uso e i requisiti da soddisfare;
- valutare i modelli architetturali secondo il punto precedente e scegliere il più adeguato;
- individuare nel modello i macro-componenti del sistema;
- comprendere e definire le relazioni tra i componenti.

2.2.2.3 Metriche

Structural Fan-in SFIN Indica quante componenti utilizzano un dato modulo. Un alto valore indica un alto riuso della componente.

- **Misurazione**: valore intero: conteggio delle componenti;
- **Valore preferibile**: ≥ 1;
- **Valore accettabile**: ≥ 0.

Structural Fan-out SFOUT Indica quante componenti vengono utilizzate dalla componente in esame. Un alto valore indica un alto accoppiamento della componente.

- **Misurazione**: valore intero: conteggio delle componenti;
- **Valore preferibile**: $= 0$;
- **Valore accettabile**: ≤ 6.

2.2.3 Progettazione di Dettaglio

Nota: in questa sezione si parla di modulo, unità e componente riferendosi allo stesso oggetto, ma ponendo l'attenzione sulla proprietà (rispettivamente modularità, unità e composizione) più rilevante nel contesto d'uso.
La progettazione di design segue la progettazione dell'architettura e prevede la scomposizione delle macro-componenti di cui il modello architetturale è fatto in componenti più piccole, che sono:

- facilmente comprensibili;
- strettamente collegati ai requisiti funzionali;
- implementabili da un singolo programmatore.

2.2.3.1 Obiettivi

- tradurre i requisiti in unità di codice, i moduli;
- favorire il lavoro dei programmatori con compiti individuali relativi ai singoli moduli;
- produrre un sistema software da raffinare ma già eseguibile;
- mantenere il tracciamento tra requisiti e componenti.

2.2.3.2 Strategia

- scomporre le componenti architetturali in componenti piccole;
- implementare le micro-componenti individuate.

2.2.3.3 Metriche

Accoppiamento tra le classi di oggetti - CBO Una classe è accoppiata a una seconda se usa metodi o variabili definiti nella seconda.

- misurazione: valore intero: CBO;
- valore preferibile: $0 \leq CBO \leq 1$;
- valore accettabile: $0 \leq CBO \leq 6$.

2.3 Processi di Supporto

2.3.1 Pianificazione

La pianificazione è un'attività significativa del gestione di progetto. Consiste governare le risorse a disposizione, ovvero tempi, costi e ruoli, monitorarle nel tempo e reagire efficacemente ai cambiamenti. La pianificazione si esprime specificatamente all'interno del Piano di Progetto

2.3.1.1 Obiettivi

- avere a disposizione piani e obiettivi ben definiti;
- aver definito ruoli, responsabilità, obblighi e autorità a cui rispondere;
- aver allocato le risorse e i beni necessari;
- attivare il piano per sostenere il progetto.

2.3.1.2 Strategia

- produrre la pianificazione delle attività;
- mantenerla aggiornata mentre esse vengono svolte;
- usarla come riferimento e supporto.

2.3.1.3 Metriche

BAC: Budget at Completion Budget totale allocato per il progetto

- misurazione: numero intero;
- valore preferibile: *pari al preventivo*;
- valore accettabile: *di poco maggiore o pari al preventivo*.

AC: Actual Cost Il denaro speso fino al momento del calcolo.

- misurazione: numero intero;
- valore preferibile: $0 \leq AC < PV$;
- valore accettabile: $0 \leq AC \leq$ *budget totale* .

EV - Earned Value Metrica di utilità per il calcolo di SV e CV. Si tratta del valore del lavoro fatto fino al momento del calcolo; corrisponde al denaro guadagnato fino a quel momento.

- misurazione: $BAC \cdot \%$ *di lavoro completato* ;
- valore preferibile: ≥ 0;
- valore accettabile: ≥ 0.

PV: Planned Value Metrica di utilità per il calcolo di SV e CV. Si tratta del valore del lavoro pianificato al momento del calcolo: corrisponde al denaro che si dovrebbe aver guadagnato in quel momento.

- misurazione: $BAC \cdot \%$ *di lavoro pianificato* ;
- valore preferibile: ≥ 0;
- valore accettabile: ≥ 0.

SV: Schedule Variance Dice se si è avanti o indietro nello svolgimento del progetto rispetto alla pianificazione.

- misurazione: $SV = EV - PV$
- valore preferibile: > 0;
- valore accettabile: 0.

Cost Variance Differenza tra il costo del lavoro effettivamente completato e quello pianificato. Una CV positiva indica che si sta rispettando il budget.

- misurazione: $CV = EV - AC$;
- valore preferibile: > 0;
- valore accettabile: ≥ 0.

2.3.2 Verifica

Il processo consiste nella ricerca e correzione di anomalie e difetti nei processi e prodotti del progetto, mediante tecniche predefinite e se possibile automatiche.

2.3.2.1 Obiettivi

- individuare e correggere le anomalie;
- provare che il sistema soddisfi i requisiti.

2.3.2.2 Strategia

- individuare tecniche e strumenti di verifica;
- applicarli;
- affinarli con l'esperienza;

2.3.2.3 Metriche

Code Coverage - CC Indica il numero di righe di codice percorse dai test durante la loro esecuzione. Per linee di codice totali si intende tutte quelle appartenenti all'unità in fase di test.

- misurazione: valore percentuale: $CC = \frac{linee\ di\ codice\ percorse}{linee\ di\ codice\ totali}$;
- valore preferibile: 100%;
- valore accettabile: 75%.

2.3.3 Documentazione

La documentazione consiste nella produzione di informazioni, la cui forma tipica è documentale, e nella loro gestione. I documenti sono prodotti a supporto di tutte le attività di progetto.

2.3.3.1 Obiettivi Si costruisce la documentazione affinché costituisca un body of knowledge$_G$ che raccoglie la conoscenza in modo:

- completo;
- non ambiguo;
- modulare e fatto di parti coese;
- trasparente, adatto alla trasmissione delle informazioni;
- disponibile esternamente.

2.3.3.2 Strategia I documenti sono:

- prodotti in concomitanza con tutte le attività di sviluppo;
- prodotti in modo collaborativo;
- supportati da un glossario;
- supportati dalle Norme di Progetto;
- prodotti con strumenti software adatti alla collaborazione e alla modularità in LaTeX;
- ospitati in una repository$_G$ pubblica su Github.

2.3.3.3 Metriche

Indice Gunning fog Indice della leggibilità del testo. Stima gli anni di educazione scolastica richiesti per comprendere il testo a una prima lettura.
Ha dei limiti: uno di questi è la sua definizione di *parola complessa*, intesa come una parola di tre o più sillabe eccetto alcuni suffissi comuni.

- misurazione: valore intero: $I_{GF} = 0.4 \cdot (\frac{numero\ di\ parole}{numero\ di\ frasi} + 100 \cdot \frac{numero\ di\ parole\ complesse}{numero\ di\ frasi})$;
- valore preferibile: ≤ 12;
- valore accettabile: ≤ 16.

Indice di Gulpease Indice della leggibilità del testo. Valuta la lunghezza delle parole e delle frasi rispetto al numero totale di lettere.

- misurazione: valore intero da 0 a 100:
  $I_G = 89 + \frac{(300 \cdot numero\ di\ frasi - 10 \cdot numero\ di\ lettere)}{numero\ di\ parole}$;
- valore preferibile: $80 < I_G < 100$;
- valore accettabile: $40 < I_G < 100$.

Correttezza ortografica Tutti i documenti non devono assolutamente contenere errori grammaticali o errori ortografici.

- misurazione: valore intero: numero di errori grammaticali o ortografici per documento;
- valore preferibile: 0;
- valore accettabile: 0.

2.4 Processi Organizzativi

2.4.1 Gestione della Qualità

Tale processo ha per scopo il raggiungimento di un grado soddisfacente di qualità nel progetto. Fornisce:

- obiettivi da perseguire;
- strumenti tecnici, come procedure e metriche.

2.4.1.1 Obiettivi Garantire che i prodotti e processi rispettino gli standard di qualità richiesti.

2.4.1.2 Strategia

- pianificare le proprie azioni perseguendo gli obiettivi;
- utilizzare gli strumenti forniti per misurare e monitorare i risultati;
- reagire ai risultati aggiornando obiettivi, strategia e strumenti.

2.4.1.3 Metriche

Percentuale di metriche soddisfatte - PMS La percentuale di metriche soddisfatte valuta quante metriche raggiungono soglie accettabili sul numero totale delle metriche calcolate. Una bassa percentuale di soddisfazione può indicare poca qualità, metriche inadeguate o mancata correttezza nel calcolo.

- misurazione: $\frac{numero\ di\ metriche\ soddisfatte}{numero\ di\ metriche\ totali}$;
- valore preferibile: $\geq 80\%$;
- valore accettabile: $\geq 60\%$.

3 Qualità di prodotto

Per valutare la qualità del prodotto il gruppo ha deciso di far riferimento allo standard ISO/IEC 9126_G che definisce le caratteristiche di cui tener conto per produrre un prodotto di buona qualità. Le caratteristiche sono descritte attraverso dei parametri, che ne quantificano il grado di raggiungimento. Di seguito vengono citate le voci che il gruppo ha ritenuto rilevanti nel contesto del progetto.

3.1 Funzionalità

Capacità del prodotto di fornire funzioni che riescano a soddisfare tutti i requisiti, sia espliciti che impliciti, presenti nell'Analisi dei Requisiti.

3.1.1 Obiettivi

- **Appropriatezza**: il prodotto deve mettere a disposizione un insieme di funzioni conformi agli obiettivi richiesti;
- **Accuratezza**: il prodotto deve fornire risultati attesti con il grado di precisione richiesto;
- **Conformità**: il prodotto deve aderire a determinati standard.

3.1.2 Metriche

3.1.2.1 Completezza dell'implementazione La completezza del prodotto e il rispetto dei requisiti viene indicato da una percentuale.

- **Misurazione**: Si calcola con la seguente formula:
$$C = (1 - \frac{N_{FNI}}{N_{FI}}) \cdot 100$$
dove N_{FNI} indica il numero di funzionalità non implementate e N_{FI} indica il numero di funzionalità individuate dall'analisi;
- **Valore preferibile**: 100%;
- **Valore accettabile**: 100%.

3.2 Affidabilità

Capacità del prodotto di mantenere prestazioni elevate anche in caso di situazioni anomale o critiche.

3.2.1 Obiettivi

- **Maturità**: il prodotto deve evitare che si verifichino errori e malfunzionamenti;
- **Tolleranza agli errori**: il prodotto mantiene alte prestazioni anche in caso di malfunzionamenti o di un uso scorretto.

3.2.2 Metriche

3.2.2.1 Densità errori L'abilità del prodotto di resistere a malfunzionamenti viene indicata con una percentuale.

- **Misurazione**: Si calcola con la seguente formula:
$$M = \frac{N_{ER}}{N_{TE}} \cdot 100$$
dove N_{ER} indica il numero di errori rilevati durante il testing e N_{TE} indica il numero di test eseguiti;
- **Valore preferibile**: 0%;
- **Valore accettabile**: $\leq$ 10%.

3.3 Usabilità

Capacità del prodotto di essere di facile comprensione e utilizzo da parte degli utenti.

3.3.1 Obiettivi

- **Comprensibilità**: l'utente deve essere in grado di comprendere le funzionalità offerte dal prodotto e ad utilizzarle;
- **Apprendibilità**: l'utente deve poter imparare facilmente ad utilizzare il prodotto;
- **Attrattività**: il prodotto deve essere piacevole da usare.

3.3.2 Metriche

3.3.2.1 Facilità di utilizzo La facilità con cui l'utente raggiunge ciò che vuole viene rappresentata tramite il numero di click necessari per arrivare al contenuto desiderato.

- **Misurazione**: Click per raggiungere la schermata di checkout;
- **Valore preferibile**: 10;
- **Valore accettabile**: $\leq$ 15.

3.3.2.2 Facilità di apprendimento La facilità con cui l'utente riesce ad imparare ad usare le funzionalità del prodotto viene rappresentata tramite il tempo medio che serve per comprenderle.

- **Misurazione**: Minuti per raggiungere pagina di checkout;
- **Valore preferibile** 3;
- **Valore accettabile** $\leq$ 5.

3.3.2.3 Profondità della gerarchia La profondità del sito. Un sito per essere facile da utilizzare non deve essere troppo profondo.

- **Misurazione**: Livello di profondità della pagina di un oggetto;
- **Valore preferibile**: 4;
- **Valore accettabile**: $\leq$ 7.

3.4 Manutenibilità

Capacità del prodotto di essere modificato, includendo correzioni, miglioramenti o adattamenti.

3.4.1 Obiettivi

- **Analizzabilità**: facilità con la quale è possibile analizzare il codice per localizzare un errore;
- **Modificabilità**: capacità del prodotto di permettere l'implementazione di una modifica.

3.4.2 Metriche

3.4.2.1 Facilità di comprensione La facilità con cui è possibile comprendere cosa fa il codice può rappresentata dal numero di linee di commento nel codice.

- **Misurazione**: Si può calcolare con la seguente formula:
$$\mathrm{R} = \frac{N_{\mathrm{LCOM}}}{N_{\mathrm{LCOD}}}$$
dove $\mathrm{N_{LCOM}}$ indica le linee di commento e $\mathrm{N_{LCOD}}$ indica le linee di codice;
- **Valore preferibile**: ≥ 0.20;
- **Valore accettabile**: ≥ 0.10.

3.4.2.2 Semplicità delle funzioni La facilità di un metodo può essere rappresentata dal numero di parametri per metodo: meno parametri ha una funzione più è semplice e intuitiva.

- **Misurazione**: Numero di parametri per metodo;
- **Valore preferibile** ≤ 3;
- **Valore accettabile** ≤ 6.

3.4.2.3 Semplicità delle classi La facilità di una classe può essere rappresentata dal numero di metodi per classe: una classe con pochi metodi ha uno scopo ben preciso e facilmente comprensibile.

- **Misurazione**: Numero di metodi per classe;
- **Valore preferibile** ≤ 8;
- **Valore accettabile** ≤ 15.

4 Specifica dei test

Per assicurare la qualità del software prodotto, il gruppo *8Lab Solutions* adotta come modello di sviluppo del software il **Modello a V**$_G$, il quale prevede lo sviluppo dei test in parallelo alle attività di analisi e progettazione. In questo modo i test permetteranno di verificare sia la correttezza delle parti di programma sviluppati, sia che tutti gli aspetti del progetto siano implementati e corretti. Segue quindi il tracciamento dei test e il loro esito per mezzo di tabelle che ne semplificheranno la consultazione e che potranno fornire una precisa indicazione degli output prodotti, specificando se il risultato ottenuto sia quello atteso, errato oppure non coerente a quanto fissato in precedenza. Per definire lo stato dei test, vengono utilizzate le seguenti sigle:

- **I**: per indicare che il test è stato implementato;
- **NI**: per indicare che il test non è stato implementato.

Inoltre per lo stato dei test si usano le seguenti abbreviazioni:

- **S**: per indicare che il test ha soddisfatto la richiesta;
- **NS**: per indicare che il test non ha soddisfatto la richiesta.

4.1 Tipi di test

Vengono individuate quattro tipologie di test:

- **Test di Accettazione[TA]**: essi vengono fatti per verificare che il prodotto soddisfi quanto richiesto dal proponente;
- **Test di Sistema [TS]**: questi test vengono utilizzati quando il sistema viene installato su una piattaforma e verificano che esso raggiunga gli obiettivi fissati e soddisfi le richieste formulate in partenza;
- **Test di Integrazione [TI]**: lo scopo di questi test è quello di testare come un gruppo i singoli moduli$_G$ del software. Essi vengono svolti dopo i Test di Unità e prima dei Test di Sistema;
- **Test di Unità [TU]**: questi test hanno il compito di verificare le singole unità del software, ovvero le minime componenti del programma che hanno un funzionamento autonomo. Il successo da parte di questi test non implica il corretto funzionamento da parte del software.

4.1.1 Test di Accettazione

I test di accettazione hanno lo scopo di dimostrare che il software sviluppato soddisfi i requisiti presentati nel capitolato e concordati con il proponente, essi vengono eseguiti durante il collaudo finale. Tali test verranno indicati nel seguente modo:

TA[Tipo][Importanza][Codice]

dove:

- **Tipo**: indica di che tipo si tratta il requisito. Può appartenere a una delle seguenti categorie indicate con:
  - **O** per i requisiti obbligatori;
  - **D** per i requisiti desiderabili;

  - **F** per i requisiti facoltativi.

- **Importanza**: indica l'importanza del requisito. L'importanza che può assumere un requisito viene indicata con:
  - **F** per indicare un requisito funzionale;
  - **V** per indicare un requisito di vincolo;
  - **Q** per indicare un requisito di qualità;
  - **P** per indicare un requisito prestazionale.
- **Codice**: rappresenta il codice identificativo crescente del componente da verificare.

Tabella 4.1.1: Riepilogo Test di Accettazione

Requisito	Classificazione	Descrizione	Fonti
Prov	Provaaaa	Prova	Prova
R1	Cl	Desc	In
R1	Cl	Desc	In

4.1.2 Test di Sistema

I test di sistema sono impiegati per garantire il corretto funzionamento delle componenti dell'intero sistema. Tali test verranno indicati nel seguente modo:

TS[id]

dove *id* rappresenta il codice identificativo crescente del componente da verificare.
Tale tipologia di test verrà sviluppata in un immediato futuro, in seguito alla richiesta della sua istanziazione.

4.1.3 Test di Integrazione

I test di integrazione sono usati per verificare il corretto funzionamento tra le varie unità dell'architettura. Tali test verranno indicati nel seguente modo:

TI[id]

dove *id* rappresenta il codice identificativo crescente del componente da verificare.
Tale tipologia di test verrà sviluppata in un immediato futuro, in seguito alla richiesta della sua istanziazione.

4.1.4 Test di Unità

I test di unità hanno l'obiettivo di verificare il corretto funzionamento della parte più piccola autonoma del lavoro realizzato. Tali test verranno indicati nel seguente modo:

TU[id]

dove *id* rappresenta il codice identificativo crescente dell'unità da verificare.

Tale tipologia di test verrà sviluppata in un immediato futuro, in seguito alla richiesta della sua istanziazione.

A Standard di qualità

A.1 ISO/IEC 9126

ISO/IEC 9126 è uno standard internazionale per valutare la qualità del software.
Questo standard è diviso in quattro parti che vengono riportate di seguito.

A.1.1 Metriche per la qualità interna

Metriche che si applicano al software non eseguibile durante le fasi di progettazione e codifica. Permettono di individuare eventuali problemi che potrebbero influire sulla qualità finale del prodotto prima che venga realizzato un eseguibile. Grazie alle misure effettuate tramite le metriche interne è possibile prevedere il livello di qualità esterna e di qualità in uso del prodotto finale, poiché entrambe vengono influenzate dalla qualità interna.
Viene rilevata tramite analisi statica. Idealmente la qualità interna determina la qualità esterna;

A.1.2 Metriche per la qualità esterna

Metriche applicabili al software in esecuzione che ne misurano il comportamento attraverso dei test, in funzione degli obiettivi stabiliti.
Viene rilevata tramite analisi dinamica. Idealmente la qualità esterna determina la qualità in uso;

A.1.3 Metriche per la qualità in uso

Metriche applicabili solo al prodotto finito ed in uso in condizioni reali.
La qualità in uso viene raggiunta solo se è stato raggiunto il livello di qualità interna e di qualità esterna;

A.1.4 Modello della qualità del software

Il modello di qualità del software suddivide la qualità in sei caratteristiche generali e varie sotto caratteristiche, misurabili attraverso delle metriche, utilizzate per fornire una scala ed un metodo per la misurazione. Di seguito sono riportate queste caratteristiche.

A.1.4.1 Funzionalità Capacità del software di soddisfare i requisiti, descritti nell'Analisi dei Requisiti, in un determinato contesto.
Nello specifico il software deve soddisfare le seguenti caratteristiche:

- **Appropriatezza**: capacità di fornire funzioni appropriate per attività specifiche, che permettano di raggiungere gli obiettivi prefissati;
- **Accuratezza**: capacità di fornire i risultati concordati o la precisione richiesta;
- **Interoperabilità**: capacità di interagire ed operare con uno o più sistemi specificati;
- **Conformità**: capacità di aderire a standard;
- **Sicurezza**: capacità di proteggere informazioni e dati.

A.1.4.2 Affidabilità Capacità del software di mantenere uno specifico livello di prestazioni quando usato in condizioni specificate.
Nello specifico il software deve soddisfare le seguenti caratteristiche:

- **Maturità**: capacità di evitare il verificarsi di errori, malfunzionamenti o risultati non corretti;
- **Tolleranza agli errori**: capacità di mantenere livelli prefissati di prestazioni anche in presenza di malfunzionamenti o usi scorretti del prodotto finale;
- **Recuperabilità**: capacità di ripristinare un livello appropriato di prestazioni o di recupero di informazioni rilevanti a seguito di un malfunzionamento;
- **Aderenza**: capacità di aderire a standard, regole e convenzioni che riguardano l'affidabilità.

A.1.4.3 Efficienza Capacità del prodotto software di eseguire le proprie funzioni minimizzando il tempo necessario e sfruttando al meglio le risorse che necessita.
Nello specifico il software deve soddisfare le seguenti caratteristiche:

- **Nel tempo**: capacità di fornire adeguati tempi di risposta, elaborazione e velocità di attraversamento in determinate condizioni;
- **Nello spazio**: capacità di utilizzo di quantità e tipo di risorse in maniera adeguata;

A.1.4.4 Usabilità Capacità del prodotto software di essere compreso, appreso, usato e accettato dall'utente, quando usato sotto determinate condizioni.
Nello specifico il software deve soddisfare le seguenti caratteristiche:

- **Comprensibilità**: capacità di essere chiaro riguardo le proprie funzionalità e il proprio utilizzo;
- **Apprendibilità**: capacità di essere facilmente apprendibile dagli utenti;
- **Operabilità**: capacità di permettere all'utente di eseguire i suoi scopi e controllarne l'uso;
- **Attrattività**: capacità di essere piacevole all'utente che l'utilizza.

A.1.4.5 Manutenibilità Capacità del software di essere modificato, al fine di aggiungere correzioni, miglioramenti o adattamenti.
Nello specifico il software deve soddisfare le seguenti caratteristiche:

- **Analizzabilità**: capacità di essere facilmente analizzato al fine di localizzare un errore;
- **Modificabilità**: capacità di poter essere agevolmente modificato nel codice, nella progettazione o nella documentazione;
- **Stabilità**: capacità di evitare effetti indesiderati a seguito di una modifica;
- **Testabilità**: capacità di essere facilmente testato per validare le modifiche apportate.

A.1.4.6 Portabilità Capacità del software di essere trasportato da un ambiente di lavoro ad un altro, sia esso hardware che software.
Nello specifico il software deve soddisfare le seguenti caratteristiche:

- **Adattabilità** Capacità di essere facilmente adattato a differenti ambienti operativi, senza applicare modifiche;
- **Installabilità**: capacità di poter essere installato in un determinato ambiente;
- **Conformità**: capacità di coesistere con altre applicazioni e di condividere risorse;
- **Sostituibilità**: capacità di essere utilizzato al posto di un altro software per svolgere gli stessi compiti, nello stesso ambiente.

B Resoconto attività di verifica

In questa sezione vengono descritti e analizzati gli esiti delle attività di verifica svolte su tutti i documenti che vengono consegnati nelle varie revisioni di avanzamento del progetto.

B.1 Revisione dei Requisiti

B.1.1 Tracciamento dei casi d'uso e dei requisiti

Per facilitare il tracciamento delle relazioni fra casi d'uso e requisiti che fra requisiti e fonti, il gruppo ha deciso di utilizzare il software ???.

B.1.2 Analisi statica dei documenti

L'analisi dei documenti mediante *Walkthrough*$_G$ ha portato all'individuazione di alcuni errori frequenti a partire dai quali è stata stilata una lista di controllo interna. In questo modo sarà possibile applicare l'*Inspection*$_G$ per le future attività di verifica.

B.1.3 Esiti verifiche automatizzate

Nella tabella seguente vengono riportati gli indici Gulpease$_G$ di tutti i documenti prodotti finora.

Tabella B.1.1: Esiti verifiche automatizzate - Revisione dei Requisiti

Documento	Indice Gulpease	Esito
Analisi dei Requisiti v1.0.0	85	Superato
Glossario v1.0.0	77	Superato
Norme di Progetto v1.0.0	76	Superato
Piano di Progetto v1.0.0	72	Superato
Piano di Qualifica v1.0.0	82	Superato
Studio di Fattibilità v1.0.0	71	Superato
Verbale Interno 2018-12-04 v1.0.0	80	Superato
Verbale Interno 2018-12-10 v1.0.0	79	Superato
Verbale Interno 2018-12-20 v1.0.0	85	Superato
Verbale Interno 2018-12-24 v1.0.0	74	Superato
Verbale Esterno 2018-12-07 v1.0.0	85	Superato
Verbale Esterno 2018-12-21 v1.0.0	75	Superato
Verbale Esterno 2019-01-04 v1.0.0	73	Superato

C Valutazioni per il miglioramento

In questa sezione viene riportata la valutazione fatta dal gruppo riguardo il lavoro svolto finora. Lo scopo di questa scelta è trattare i problemi sorti e procedere alla loro più efficiente risoluzione in modo tale che non si verifichino in futuro.
Verano dunque tracciati problemi riguardanti i seguenti ambiti:

- **Organizzazione**: in cui vengono analizzati i problemi riguardanti l'organizzazione e la comunicazione all'interno del gruppo;
- **Ruoli**: in cui vengono analizzati i problemi riguardanti il corretto svolgimento di un ruolo;
- **Strumenti di lavoro**: in cui vengono analizzati i problemi riguardanti l'uso degli strumenti scelti.

Ogni problema viene sollevato sulla base dell'autovalutazione dei membri del gruppo, poiché non vi è una persona esterna che possa dare una valutazione oggettiva. Nonostante possa sembrare un sistema poco efficace, il gruppo ha beneficiato di questa scelta dal punto di vista della comunicazione e della produzione, migliorando progressivamente la qualità del lavoro. Questa sezione verrà aggiornata con l'avanzamento del lavoro riportando nuove problematiche, nel caso in cui queste dovessero verificarsi. Per rendere le valutazioni più leggibili e consultabili, si è deciso di organizzarle in forma tabellare, la cui struttura è consultabile nel documento *Norme di Progetto v1.0.0*.

C.1 Valutazioni sull'organizzazione

Tabella C.1.1: Tabella delle problematiche relative all'organizzazione

Problema	Descrizione	Gravità	Soluzione
Incontro di gruppo	Si è riscontrata una certa difficoltà nell'organizzare gli incontri in modo tale che fossero presenti tutti i componenti.	2	Si è deciso di utilizzare un calendario condiviso per scegliere il giorno in cui tutto il team potesse essere presente.
Incontro con il proponente	Poiché l'azienda proponente ha sede all'estero, gli incontri sono stati fatti via skype o hangouts e c'è stata un'iniziale difficoltà nel trovare una sede in cui trovarsi con il gruppo per effettuare la videochiamata senza essere disturbati.	1	Il Dipartimento di Matematica concede la possibilità di prenotare delle aule per necessità come la nostra, quindi, venuti a conoscenza di ciò, il problema è stato risolto in breve tempo.

C.2 Valutazione sui ruoli

Tabella C.2.1: Tabella delle problematiche relative ai ruoli

Problema	Descrizione	Gravità	Soluzione
Rivestire il ruolo di *Responsabile*	A causa dell'inesperienza, chi ha lavorato come *Responsabile* ha avuto discrete difficoltà nella suddivisione bilanciata delle ore tra i membri provocando diverse ridistribuzioni delle ore.	3	Per evitare eventuali ritardi nelle consegne, il gruppo ha deciso di dedicare del tempo per analizzare meglio la mole di lavoro e compiere così una più accurata distribuzione delle ore.
Rivestire il ruolo di *Analista*	La scelta e la configurazione del software per il tracciamento dei requisiti ha richiesto più ore del previsto, come si può vedere anche dal documento *Piano di Progetto v1.0.0*.	2	Per evitare ritardi sul lavoro, chi ha svolto il ruolo di *Verificatore* e ha avanzato ore, è stato affiancato agli *Analisti* per completare i loro compiti.

C.3 Valutazioni sugli strumenti di lavoro

Tabella C.3.1: Tabella delle problematiche relative agli strumenti di lavoro

Problema	Descrizione	Gravità	Soluzione
GitHub	Si sono riscontrati in più occasioni conflitti sui file in cui si stava lavorando e il tempo utilizzato per risolverli è stato sottratto dal tempo di lavoro.	2	Il gruppo è stato istruito sull'uso di specifici rami di lavoro in modo tale che la modifiche di tutti i componenti si potessero integrare con il proprio lavoro senza che quest'ultimo potesse soffrire di conflitti. Da questa scelta, il team ha potuto beneficiare nell'avere nel ramo *master* la versione stabile del proprio lavoro e *branch* dedicati su cui lavorare.
LaTeX	A causa dell'inesperienza di alcuni membri del gruppo all'utilizzo di questo strumento, si sono riscontrare diverse difficoltà sopratutto nell'inserimento di figure e nella costruzione di tabelle.	1	Per risolvere in breve tempo questa problematica, si è deciso di affiancare ai membri meno esperti chi sapeva già utilizzare i comandi di LaTeX dando così la possibilità ai primi di imparare e permettendo ai secondi di non subire grossi rallentamenti nel lavoro.