QUIZZIPEDIA

team404swe@gmail.com

Piano di qualifica 1.0

Informazioni sul documento	
Nome Documento	Piano di qualifica
Versione	1.0
Uso	Esterno
Data Creazione	21 dicembre 2015
Data Ultima Modifica	16 marzo 2016
Redazione	Andrea Multineddu
Verifica	Marco Crivellaro - Luca Alessio
Approvazione	Davide Bortot
Committente	Zucchetti SPA
Lista di distribuzione	Prof. Vardanega Tullio
	TEAM404

Registro delle modifiche

Versione	Autore	Data	Descrizione
1.0.0	D. Bortot (Responsabile)	16/03/2016	Approvazione documento
0.6.3	A. Multineddu (Verificatore)	11/03/2016	Correzione errori di impaginazione e tipografici
0.6.2	M. Crivellaro (Verificatore)	10/03/2016	Verifica documento completo
0.6.1	A. Multineddu (Verificatore)	19/01/2016	Correzione errori nella sezione Misure e metriche
0.6.0	A. Multineddu (Verificatore)	18/01/2016	Aggiunta metriche e tabella resoconto misure
0.5.0	M. Crivellaro (Verificatore)	16/01/2016	Modifica layout e aggiunta file template.tex
0.4.0	A. Multineddu (Verificatore)	07/01/2016	Redazione sezione Misure e metriche
0.3.0	A. Multineddu (Verificatore)	03/01/2016	Redazione sezione Risorse per la verifica
0.2.0	A. Multineddu (Verificatore)	22/12/2015	Redazione sezione Strategia generale di qualifica
0.1.0	A. Multineddu (Verificatore)	21/12/2015	Prima stesura del documento.

Indice

Elenco delle tabelle

Elenco delle figure

Sommario

Il presente documento contiene il *piano di qualifica* del capitolato **Quizzipedia**. Vengono rese note le strategie di verifica qualitativa dei processi e del prodotto utilizzate dal gruppo **Team404**. Strategie che coinvolgono l'attuazione di modelli standard e l'utilizzo di metriche e misure necessarie alla valutazione oggettiva del processo o prodotto in analisi.

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di esporre le strategie che il gruppo **Team404** intende adottare per assicurare la qualità del prodotto software **Quizzipedia** e dei processi coinvolti per il suo sviluppo.

1.2 Scopo del prodotto

Il progetto **Quizzipedia** ha come obiettivo lo sviluppo di un sistema software basato su tecnologie Web (Javascript$_G$, Node.js$_G$, HTML5$_G$, CSS3$_G$) che permetta la creazione, gestione e fruizione di questionari. Il sistema dovrà quindi poter archiviare i questionari suddivisi per argomento, le cui domande dovranno essere raccolte attraverso uno specifico linguaggio di markup (Quiz Markup Language) d'ora in poi denominato QML$_G$. In un caso d'uso a titolo esemplificativo, un esaminatore dovrà poter costruire il proprio questionario scegliendo tra le domande archiviate, ed il questionario così composto sarà presentato e fruibile all' esaminando, traducendo l'oggetto QML in una pagina HTML$_G$, tramite un'apposita interfaccia web. Il sistema presentato dovrà inoltre poter proporre questionari preconfezionati e valutare le risposte fornite dall'utente finale.
Per un'analisi più precisa ed approfondita del progetto si rimanda al documento *analisi_dei_requisiti_1.0.pdf* .

1.3 Glossario

Viene allegato un glossario nel file "*glossario_1.0.pdf*" nel quale viene data una definizione a tutti i termini che in questo documento appaiono con il simbolo '$_G$' a pedice.

1.4 Riferimenti

1.4.1 Normativi

- Capitolato d'appalto Quizzipedia:
  http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C5.pdf
- Norme di Progetto: *norme_di_progetto_1.0.pdf*

1.4.2 Informativi

- Corso di Ingegneria del Software anno 2015/2016:
  http://www.math.unipd.it/~tullio/IS-1/2015/
- Regole del progetto didattico:
  http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/PD01.pdf http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/
  http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/PD01b.html
- Metriche di progetto:
  https://it.wikipedia.org/wiki/Metriche_di_progetto
- Metriche per il software
  http://www.verifysoft.com/en_software_complexity_metrics.pdf
- Complexity-report:
  https://github.com/jared-stilwell/complexity-report

2 Strategia generale di qualifica

2.1 Definizione obiettivi

Il gruppo di lavoro, per garantire la qualità del prodotto software **Quizzipedia** e dei processi coinvolti al suo sviluppo, intende adottare metodologie standard ed il più possibile automatizzabili per ridurre al minimo l'attività umana e favorire processi automatici laddove sia possibile. Vengono di seguito quindi elencati gli obiettivi di qualità che si vogliono raggiungere per quanto riguarda il prodotto finale e i processi coinvolti per lo sviluppo dello stesso, e le strategie necessarie all'attuazione della corretta verifica e validazione del prodotto.
Lo scopo principale è quello di rendere definibili e misurabili tutti i processi di sviluppo e verifica per poterne controllare l'andamento e la produttività.

2.2 Qualità di processo

Per quanto riguarda la valutazione qualitativa dei processi si fa riferimento allo standard ISO/IEC 15504:1998 denominato SPICE$_6$ (Software Process Improvement Capability dEtermination) che definisce un modello per la valutazione del livello di "maturità" dei processi per identificare quali azioni possono essere necessarie per migliorare un processo specifico.

2.3 Qualità di prodotto

Come punto di riferimento per la qualità del prodotto software si fa riferimento allo standard ISO/IEC 9126:2001.
L'immagine seguente elenca le principali caratteristiche a cui un prodotto software di qualità deve aspirare:

Figura 1: Riepilogo caratteristiche ISO/IEC 9126:2001

2.4 Pianificazione attività di sviluppo

La suddivisione temporale delle diverse fasi di sviluppo segue il cosiddetto modello a V ed è descritta nel documento allegato *Piano-di-Progetto-1.0.pdf*. Questo modello prevede un lavoro di controllo e verifica corrispondente ad ogni attività per evitare l'accumulo di errori difficilmente gestibili in seguito.
La divisione in diverse fasi, e l'attività di verifica abbinata ad ogni fase e processo, hanno come scopo quello di facilitare l'integrazione e il corretto funzionamento delle parti che comporranno il sistema finale. Il ramo discendente descritto nella precedente figura rappresenta la successione delle fasi di sviluppo: ciascuna fase è accompagnata da una costante attività di verifica in modo da poter permettere il passaggio alla fase successiva esclusivamente quando si è sicuri che non ci siano errori. Ciò è essenziale durante la delicata attività di raccolta e documentazione dei requisiti, nella progettazione architetturale ad alto livello ed in seguito nella progettazione di dettaglio.
Del ramo ascendente che attraversa le attività di testing è importante sottolineare l'utilizzo

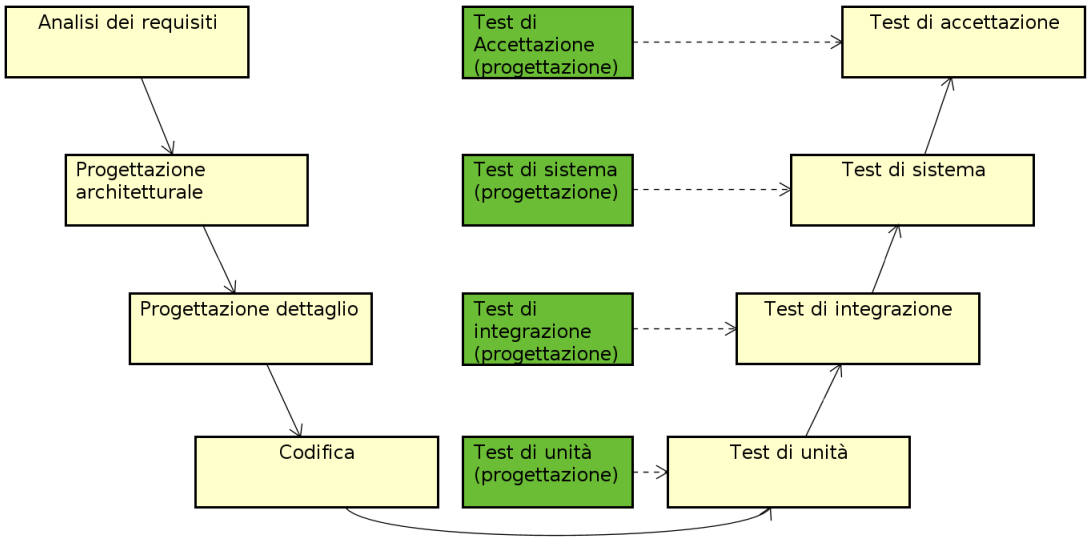

Figura 2: Fasi di sviluppo

dell'approccio bottom-up. Dalla fase di codifica quindi, per mezzo di test specifici sulle singole componenti (test di unità), si è in grado di garantire la correttezza del lavoro svolto prima che le singole componenti vengano integrate tra loro. Ovviamente l'esito positivo di ogni test su ogni unità non garantisce la correttezza dell'integrazione tra le stesse.
Nelle sezioni successive verranno elencate le varie fasi del ramo discendente e ne verranno descritte le strategie di verifica qualitativa inerenti.

2.5 Gestione amministrativa della revisione

Ogni processo coinvolto nell'attività di sviluppo ha bisogno di una costante attività di verifica di supporto in grado di identificare possibili miglioramenti o peggioramenti ed apportare eventuali correzioni.
Il modello che il gruppo intende quindi seguire è il cosiddetto PDCA (o Ciclo di Deming), modello volto al miglioramento continuo che definisce quattro fasi cicliche:

- **P**lan: definizione del problema, cosa deve essere realizzato e come andrà controllato per la verifica qualitativa;
- **D**o: eseguire le attività secondo i piani;
- **C**heck: verifica nel tempo dei risultati conseguiti in seguito a modifiche e migliorie, confronto tra risultati attesi e risultati effettivi;
- **A**ct: applicazione di soluzioni correttive atte al miglioramento.

Ogni processo deve essere sottoposto a verifica in modo da poterlo migliorare all'iterazione successiva. Le due fasi chiave del modello PDCA sono la prima fase di definizione (P) del problema e la fase di verifica dei risultati attesi (C). Il concetto generale fa riferimento anche al metodo di *Correctness by construction*, che ha come principio base quello di fare in modo di non introdurre errori già dall'inizio e/o di correggere tali errori nel **momento più vicino possibile a quando sono stati introdotti**.
L'attività costante di verifica, quindi, deve servire proprio a far sì che gli errori e le divergenze vengano individuate il prima possibile.

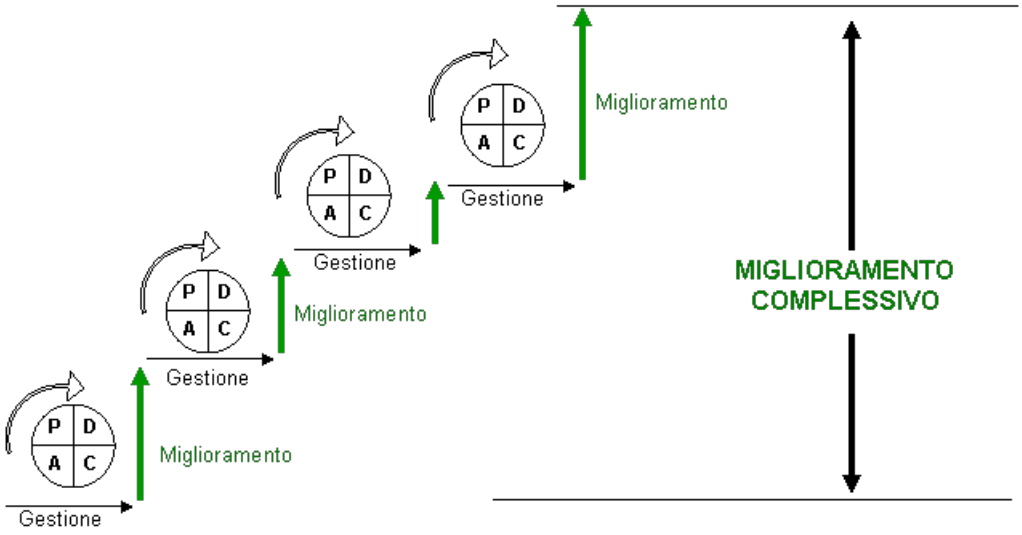

Figura 3: PDCA e miglioramento complessivo

Piccoli errori iniziali, se non gestiti, possono portare all'assoluta ingestibilità dell'attività di verifica e quindi compromettere pesantemente il successo finale del progetto. Per evitare ciò, un processo dipendente da un altro non può avere inizio finché il precedente non sia stato verificato e ne sia stata accertata la correttezza.

Nella sezione Misure e metriche vengono elencate due metriche (**Schedule variance** e **Budget variance**) necessarie al controllo e alla valutazione dei processi in termini di tempo (schedule) e in termini di costo e risorse impiegate (budget). Tali misurazioni, da effettuare alla fine di ogni fase, si basano sui valori dei consuntivi di ogni fase (presenti nel documento *piano_di_progetto_1.0.pdf*). La valutazione di queste metriche è essenziale per verificare l'andamento di ogni processo misurato e valutarne quindi il miglioramento o il peggioramento complessivo.
Per il coordinamento delle attività giornaliere ci si affida ad un sistema di **ticketing**$_G$ fornito dalla piattaforma online **Trello**$_G$ (vedere documento *norme_di_progetto_1.0.pdf* per una spiegazione dettagliata del suo utilizzo). Tale sistema permette di tenere costantemente sotto controllo l'andamento dei compiti (**Task**$_G$) e la loro realizzazione.

3 Risorse per la verifica

3.1 Risorse umane

Le risorse attivamente coinvolte nel processo di verifica sono le seguenti:

- **Responsabile**: Ha il compito di coordinare le attività di verifica e il controllo di qualità dei processi interni. Distribuisce le risorse sulle attività e ne monitora il corretto svolgimento.
- **Verificatore**: svolge l'effettiva attività di verifica su ogni prodotto. I risultati di tali compiti, che saranno gli esiti delle attività di misurazione, vengono presentati al Responsabile di progetto per la gestione della loro risoluzione.

Per una più completa ed approfondita descrizione di ogni ruolo si rimanda ai documenti *Piano-di-progetto-1.0.pdf* e *Norme-di-Progetto.pdf* (sezione 4.2).

3.2 Risorse tecnologiche

Le risorse tecnologiche coinvolte nel processo di verifica sono principalmente le seguenti:

- **Complexity-report**: strumento software per l'analisi statica del codice JavaScript;
- **MySpell**: correttore ortografico integrato all'interno di TexMaker;
- **Trello**: piattaforma online per l'organizzazione e la gestione di progetti.

Si rimanda alla sezione Misure e Metriche del presente documento per una spiegazione dettagliata delle metriche utilizzate da Complexity-report, e al documento Norme-di-progetto.pdf per quanto riguarda le modalità di utilizzo dei sopracitati software.

4 Misure e Metriche

4.1 Di progetto

4.1.1 Schedule Variance

Il valore SV (schedule variance) indica se si è in linea, in anticipo o in ritardo rispetto alla schedulazione delle attività di progetto pianificate nella baseline.
Se SV > 0 significa che il progetto sta producendo (ossia rilasciando deliverable) con maggior velocità a quanto pianificato, viceversa se negativo. Formula:

$$\mathbf{SV = BCWP - BCWS}$$

Dati:

- **BCWP** (Budgeted Cost of Work Performed): Valore delle attività realizzate alla data corrente.
- **BCWS** (Budgeted Cost of Work Scheduled): Costo pianificato per realizzare le attività di progetto alla data corrente.
- Range-accettazione: $[\geq -(PreventivoFase * 5\%)]$
- Range-ottimale: $[\geq 0]$

4.1.2 Budget variance

Indica se alla data corrente si è speso di più o di meno rispetto a quanto previsto a budget alla data corrente.
Formula:

$$\mathbf{BV = BCWS - ACWP}$$

Dati:

- **ACWP** (Actual Cost of Work Performed): Costo effettivamente sostenuto alla data corrente.

Se BV > 0 significa che il progetto sta spendendo il proprio budget con minor velocità di quanto pianificato, viceversa se negativo. Il fatto di spendere più velocemente il budget non ha nulla a che fare con il risparmio che se ne può avere, rappresentato invece da CV.

Parametri utilizzati:

- Range-accettazione: $[\geq -(PreventivoFase * 10\%)]$
- Range-ottimale: $[\geq 0]$

4.2 Per i documenti

4.2.1 Indice Gulpease

Indice di leggibilità di un testo tarato sulla lingua italiana.
Questo indice considera due variabili linguistiche: la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere. La formula per il suo calcolo è la seguente:

$$89 - \frac{(300 * NumeroFrasi) - (10 * NumeroLettere)}{NumeroParole}$$

I risultati sono compresi tra 0 e 100, dove il valore 100 indica la leggibilità più alta e 0 la leggibilità più bassa.
In generale risulta che testi con un indice:

- inferiore a 80 sono difficili da leggere per chi ha la licenza elementare;
- inferiore a 60 sono difficili da leggere per chi ha la licenza media;
- inferiore a 40 sono difficili da leggere per chi ha un diploma superiore.

- Range di accettazione: $[50 - 100]$
- Range ottimale: $[60 - 100]$

4.3 Per il software

In questa sezione vengono elencate le metriche utilizzate dal software di analisi Complexity-report come **Maintainability Index**$_G$, **Metriche di Halstead**$_G$ e **Complessità Ciclomatica**$_G$) e altri indicatori come la percentuale di *Copertura dei test* e la validazione W3C$_G$ dei file HTML$_G$.
Il software complexity-report offre un resoconto delle misurazioni effettuate sull'intero codice, a livello di file, e a livello di funzione presente in ogni file. Di seguito vengono presentate le metriche utilizzate per ogni livello.

4.3.1 Parameter count

Numero di parametri per funzione. Valori bassi sono da preferire. Parametri utilizzati:

- Range di accettazione: $[0 - 7]$
- Range ottimale: $[0 - 5]$

4.3.2 Cyclomatic complexity

Metrica software usata per indicare la complessità ciclomatica di un programma. Rappresenta una misura quantitativa del numero di cammini linearmente indipendenti che si possono percorrere nel codice sorgente. La complessità ciclomatica può essere misurata per funzioni individuali, metodi e classi all'interno di un programma.

- Range di accettazione: $[0 - 15]$
- Range ottimale: $[0 - 10]$

4.3.3 Metriche di Halstead

Queste metriche sono state concepite per identificare proprietà misurabili del codice e le relazioni tra di esse. Questi numeri sono staticamente calcolati dal codice sorgente.
Dati:

- n_1 = numero di operatori distinti;
- n_2 = numero di operandi distinti;

- N_1 = numero totale degli operatori;
- N_2 = numero totale degli operandi.

Da questi numeri si possono calcolare le seguenti misure:

- Program Vocabulary: $n = n_1 + n_2$
- Program Lenght: $N = N_1 + N_2$
- **Volume**: $V = Nlog_2 n$
  - Range di accettazione: $[20 - 1500]$
  - Range ottimale: $[20 - 1000]$
- **Difficulty**: $D = \frac{n_1}{2} * \frac{N_1}{n_2}$
  - Range di accettazione: $[0 - 30]$
  - Range ottimale: $[0 - 15]$
- **Effort**: $E = D * V$
  - Range di accettazione: $[0 - 400]$
  - Range ottimale: $[0 - 300]$

4.3.4 Dependency count

Conteggio del numero di chiamate di tipo **require**$_G$ di ogni metodo. Valori bassi sono da preferire.

4.3.5 Maintainability Index

Rappresenta l'indice principale dei risultati dell'analisi effettuata da Plato sull'intero codice. Compreso tra 0 e 100, questo indice rappresenta la relativa facilità di manutenzione del codice analizzato. Valori alti rappresentano miglior manutenibilità.
Questo indice viene calcolato utilizzando la seguente formula:

$$MI = MAX\left[0, 100\frac{171 - 5.2\ln V - 0{,}23G - 16{,}2\ln L}{171}\right]$$

dove:

- **MI** = Maintainability Index
- **V** = Halstead Volume
- **L** = Source Lines of Code (SLOC)
- **G** = Complessità Ciclomatica totale

Parametri utilizzati:

- Range di accettazione: $[20 - 100]$
- Range ottimale: $[70 - 100]$

4.3.6 First-order density

Percentuale di tutte le possibili dipendenze interne tra i moduli del progetto.

- Range di accettazione: $[\leq 20\%]$
- Range ottimale: $[\leq 15\%]$

4.3.7 Change cost

Percentuale dei moduli affetti da cambiamento quando un modulo all'interno del progetto viene modificato.

- Range di accettazione: $[\leq 50\%]$
- Range ottimale: $[\leq 40\%]$

4.3.8 Core size

La percentuale dei moduli che hanno molte dipendenze verso (e da) altri moduli.

- Range di accettazione: $[\leq 30\%]$
- Range ottimale: $[\leq 25\%]$

4.3.9 Copertura dei test

Indica la percentuale dei casi testati rispetto alla totalità dei casi da testare. Una percentuale del 100% può essere auspicabile solo se sono stati ben definiti i casi che necessitano realmente di essere testati.

*Copertura dei test = Numero funzioni testate * 100/Numero funzioni da testare*

Parametri utilizzati:

- Range-accettazione: $[70 - 100]$
- Range-ottimale: $[80 - 100]$

4.3.10 W3C – Markup Validation Service

Per la validazione delle pagine HTML e i file CSS sviluppati si intende affidarsi allo strumento online W3C Markup Validation Service (https://validator.w3.org/).

5 Riepilogo parametri di tolleranza

5.1 Processi

Metrica	Accettazione	Ottimale
Schedule Variance	$\geq -(P * 5\%)$	≥ 0
Budget Variance	$\geq -(P * 10\%)$	≥ 0

P = PreventivoFase

5.2 Documenti

Metrica	Accettazione	Ottimale
Gulpease Index	$[50 - 100]$	$[60 - 100]$

5.3 Software

Metrica	Accettazione	Ottimale
Parameters count	$[0 - 7]$	$[0 - 5]$
Dependency count	≤ 5	$= 0$
Halstead's Volume	$[20 - 1500]$	$[20 - 1000]$
Halstead's Difficulty	$[0 - 30]$	$[0 - 15]$
Cyclomatic complexity	$[0 - 15]$	$[0 - 10]$
First-order density	$\leq 20\%$	$\leq 15\%$
Change cost	$\leq 50\%$	$\leq 40\%$
Core size	$\leq 30\%$	$\leq 25\%$
Maintainability Index	$[20 - 100]$	$[70 - 100]$
Copertura dei test	$[70 - 100]$	$[80 - 100]$

6 Resoconto attività di verifica

Di seguito viene fornito il resoconto dell'attività di verifica inerente ad ogni fase di sviluppo.

6.1 Analisi

Riassunto attività di verifica per quello che concerne la fase di analisi:

- **Organizzazione interna**:
  - **Assegnazione dei ruoli** secondo la pianificazione specificata nel documento *Piano-di-Progetto-1.0.pdf* redatto dal Responsabile di progetto;
  - **Corretta pianificazione di sviluppo** secondo quanto stilato sempre nel documento *Piano-di-Progetto-1.0.pdf*. In particolare viene verificata la corretta divisione temporale delle diverse fasi di progetto (vedere sezione precedente) e la correttezza del preventivo in base al budget disponibile;
  - **Disponibilità delle risorse umane e tecnologiche** e dell'ambiente di lavoro;
  - **Lettura e sottoscrizione delle Norme di Progetto**. Il responsabile deve accertarsi che ogni membro del gruppo abbia letto e compreso il documento *Norme-di-progetto-1.0*. Il documento deve essere quindi preventivamente redatto.
- **Documentazione**:
  - Verifica della presenza di tutti i documenti necessari;
  - Attinenza di ogni documento alle specifiche di stile e formattazione presenti nel documento *Norme-di-progetto-1.0*;
  - Verifica della presenza e del corretto utilizzo del registro delle modifiche interno ad ogni documento;
  - Verifica iniziale dei documenti tramite la tecnica di analisi statica **walkthrough**;
  - Verifica dei documenti tramite tecnica di analisi statica di tipo **inspection**. In questo caso l'attenzione del verificatore si focalizza solo su particolari aspetti del testo (per esempio accenti, sillabazione, maiuscole, ecc);
  - Monitoraggio da parte dei verificatori del registro delle modifiche in modo da poter attuare la verifica solo alla parte inerente alla sezione modificata;
  - **Coerenza** di informazioni tra i diversi documenti;
  - Verifica di assenza di **ridondanza** di determinati contenuti tra i diversi documenti;
  - Calcolo dell'indice di leggibilità **Gulpease**.
- **Requisiti e Use Case**:
  - Correttezza dei diagrammi UML per quanto riguarda l'adeguata rappresentazione grafica degli Use Case. Verifica, leggibilità e adeguato livello di granularità;
  - Verifica della correttezza degli Use Case (testuali e grafici) secondo lo standard UML2.0;
  - Verifica delle precondizioni e delle postcondizioni di ogni Use Case;
  - Completezza dei requisiti in base alle esigenze di capitolato;
  - Tracciabilità dei requisiti tramite tabella di mappatura requisiti - casi d'uso;
  - Controllo eventuali conflitti o imprecisioni nella codifica dei requisiti e dei casi d'uso.

7 Esito dei test

In questa sezione verranno presentati gli esiti dei test effettuati per ogni fase di sviluppo.