Università degli Studi di Padova

KaleidosCode

SWEDesigner

Software per diagrammi UML

Piano di qualifica v1.0.0

Informazioni sul documento

Versione	2.0.0
Data Redazione	09/03/2017
Redazione	Bonato Enrico
	Bonolo Marco
	Pace Giulio
	Sovilla Matteo
Verifica	Pezzuto Francesco
Approvazione	Sanna Giovanni
Uso	Esterno
Distribuzione	*Prof. Vardanega Tullio*
	Prof. Cardin Riccardo
	Zucchetti s.p.a.

kaleidos.codec6@gmail.com

Diario delle Modifiche

Versione	Data	Autore	Descrizione
1.0.4	20/04/2017	Pezzuto Francesco	Preso atto degli errori segnalati nella valutazione di RR; riorganizzata struttura del documento; sistemate/ampliate sezioni: Obiettivi di qualità, Organizzazione, Scadenze temporali, Misure e metriche, Resoconto attività di verifica (Periodo Analisi e Analisi di dettaglio); inserite appendici Capability Maturity Model, Ciclo di Deming, ISO/IEC 9126
1.0.0	02/04/2017	Sanna Giovanni	Approvazione del documento
0.2.0	31/03/2017	Pezzuto Francesco	Verifica del documento
0.1.1	28/03/2017	Sovilla Matteo	Correzione e integrazione come indicato da verifica
0.1.0	25/03/2017	Pezzuto Francesco	Verifica del documento
0.0.5	23/03/2017	Sanna Giovanni	Stesura parte capitolo 2
0.0.4	23/03/2017	Sovilla Matteo	Stesura parte capitolo 3
0.0.3	23/03/2017	Bonato Enrico	Stesura parte capitolo 3
0.0.2	22/03/2017	Pace Giulio	Stesura sezione Obiettivi di qualità
0.0.1	09/03/2017	Bonolo Marco	Creazione scheletro del documento e stesura della sezione Introduzione

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Questo documento definisce gli obiettivi e le metodologie che ogni membro del gruppo *KaleidosCode* adotterà per garantire un determinato livello di qualità del prodotto.
A tal proposito ogni membro del gruppo è tenuto a leggere, perseguire e raggiungere gli obiettivi definiti in esso.

1.2 Scopo del prodotto

Lo scopo del progetto è la realizzazione di un software di costruzione di diagrammi UML_{G} con la relativa generazione di codice Java_{G} e $\text{Javascript}_{\text{G}}$ utilizzando tecnologie web. Il prodotto deve essere conforme ai vincoli qualitativi richiesti dal committente.

1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite sono riportate nel documento *Glossario v1.0.0*.
La prima occorrenza di ciascuno di questi vocaboli è marcata da una "G" maiuscola in pedice.

1.4 Riferimenti utili

1.4.1 Riferimenti normativi

- **$\text{Capitolato}_{\text{G}}$ d'appalto**:
  http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/C6.pdf (09/03/2017);
- **Norme di progetto**: *Norme di progetto v1.0.0*.

1.4.2 Riferimenti informativi

- **Slide dell'insegnamento di Ingegneria del Software 1° semestre**:
  - Qualità del software:
    http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L10.pdf (02/04/2017);
  - Qualità di Processo:
    http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L11.pdf (02/04/2017).
- **Slide dell'insegnamento di Ingegneria del Software 2° semestre**:
  - Metodi e obiettivi di quantificazione:
    http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L15.pdf (02/04/2017).
- **ISO_{G} 9001**:
  https://it.wikipedia.org/wiki/Norme_della_serie_ISO_9000#ISO_9001 (02/04/2017);

- **ISO 9126**: https://it.wikipedia.org/wiki/ISO/IEC_9126 (02/04/2017);
- **ISO 12207**: https://en.wikipedia.org/wiki/ISO/IEC_12207 (02/04/2017);
- **Indice Gulpease$_G$**:
  https://it.wikipedia.org/wiki/Indice_Gulpease (02/04/2017);
- **Complessità ciclomatica**:
  https://en.wikipedia.org/wiki/Cyclomatic_complexity (02/04/2017);
- **Capability Maturity Model (CMM$_G$)**:
  https://en.wikipedia.org/wiki/Capability_Maturity_Model (02/04/2017);
- **Analisi dei requisiti**: *Analisi dei requisiti v1.0.0*;
- **Piano di progetto**: *Piano di progetto v1.0.0*;
- **Glossario**: *Glossario v1.0.0*.

2 Visione generale della strategia

Per garantire la qualità dei prodotti realizzati durante lo sviluppo del progetto è indispensabile definire e perseguire strategie che assicurino la qualità dei processi adottati nonché il loro continuo miglioramento; inoltre, è necessario definire metriche e pianificare attività che valutino in modo preciso la qualità dei prodotti ottenuti e dei processi adottati. A tal scopo verranno adottate le seguenti strategie:

- Definizione accurata di norme che regolamentano e standardizzano i processi coinvolti nel progetto in termini di:
  - Processo di fornitura;
  - Processo di sviluppo;
  - Processi di supporto;
  - Processi organizzativi.
- Descrizione dettagliata delle strategie di pianificazione adottate per lo sviluppo del progetto in termini di:
  - Modello di sviluppo adottato;
  - Analisi dei rischi che si possono incontrare;
  - Pianificazione delle attività e dei tempi;
  - Stima preventiva delle risorse che saranno impiegate;
  - Assegnazione delle risorse al fine di portare a termine le attività pianificate nei tempi previsti;
  - Consuntivo, durante lo sviluppo del progetto, delle risorse impiegate.
- Ad ogni processo coinvolto nello sviluppo del progetto è stato scelto di applicare il ciclo di $PDCA_G$ affiancato dal CMM.[1] Essi infatti permettono il controllo, la valutazione e il miglioramento continuo dei processi nonché la determinazione del livello di maturità dell'organizzazione nel gestirli.

2.1 Obiettivi di qualità di processo

Prendendo come riferimento la normativa [ISO/IEC 12207], il gruppo *KaleidosCode* ha definito i seguenti obiettivi di qualità di processo che si impegna a perseguire.

2.1.1 Rispetto della pianificazione concordata

È necessario pianificare le attività per la realizzazione del progetto ed è quindi fondamentale rispettare tale pianificazione per garantire la consegna del prodotto secondo le tempistiche concordate.
Misura e metrica adottate sono descritte nella sezione 3.2.1.

[1] Per maggiori informazioni, consultare le appendici A e B

2.1.2 Rispetto del budget concordato

È necessario stabilire un budget per il costo della realizzazione del prodotto ed è quindi fondamentale far rientrare i costi per le risorse nella spesa prevista.
Misura e metrica adottate sono descritte nella sezione 3.2.1.

2.1.3 Leggibilità della documentazione prodotta

Durante la realizzazione del prodotto è necessario redigere la documentazione delle attività di pianificazione, gestione, sviluppo, verifica e validazione oltre che i loro prodotti.
È quindi fondamentale che i documenti prodotti siano, per quanto tecnici, facilmente leggibili.
Misura e metrica adottate sono descritte nella sezione 3.2.2.

2.1.4 Rispetto delle norme di progettazione

In progettazione si definiranno i moduli e le componenti del prodotto. Sarà necessario rispettare le norme di progettazione definite nel documento *Norme di progetto* mantenendone un valore ideale di 0 violazioni.
Inoltre, sarà utilizzata una misura e metrica, descritte nella sezione 3.2.3, per valutarne la qualità.

2.1.5 Rispetto delle norme di codifica

Nel periodo di codifica, bisognerà sviluppare le unità software ideate in progettazione. Per questo motivo, sarà necessario rispettare le norme di codifica concordate nel documento *Norme di progetto* mantenendone un valore ideale di 0 violazioni.

2.1.6 Corretto funzionamento delle componenti del sistema

Sarà necessario sottoporre le unità software sviluppate ad una serie di test[2] per valutarne il corretto funzionamento; l'obiettivo è di effettuare il 100% dei test indicati per ogni tipologia.

2.2 Obiettivi di qualità di prodotto

Prendendo come riferimento la normativa [ISO/IEC 9001] ed in particolare [ISO/IEC 9126][3], il gruppo *KaleidosCode* ha definito i seguenti obiettivi di qualità che si impegna a far raggiungere al prodotto *SWEDesigner*.

2.2.1 Funzionalità

Si garantisce che *SWEDesigner* abbia tutte le funzionalità definite e concordate con il *Zucchetti s.p.a.* nel documento *Analisi dei requisiti v1.0.0*. L'implementazione di ogni requisito deve essere quanto più completa ed economica.

[2] Definiti nella sezione 4
[3] Per maggiori informazioni, consultare l'appendice C

- **Misura**: si è deciso di utilizzare il numero totale di funzionalità del prodotto che soddisfano i requisiti definiti.
- **Metrica**: la sufficienza è raggiunta quando vengono soddisfatti almeno tutti i requisiti obbligatori.

2.2.2 Affidabilità

Il sistema deve funzionare nella sua completezza.

- **Misura**: sono stati definiti dei test a cui sottoporre il prodotto realizzato[4]; l'unità di misura sarà quindi il numero di test superati dal sistema.
- **Metrica**: la sufficienza è raggiunta quando il sistema supera almeno l'80% dei test.

2.2.3 Efficienza

Il sistema deve minimizzare l'utilizzo delle risorse impiegate e fornire le funzionalità richieste nel minor tempo possibile.
Misure e metriche adottate sono descritte nella sezione 3.2.4.

2.2.4 Manutenibilità

Il codice prodotto per realizzare il sistema deve essere comprensibile ed estensibile.
Misure e metriche adottate sono descritte nella sezione 3.2.4.

2.2.5 Portabilità

Il sistema è un applicativo web. Per questo motivo, si garantisce che il front-end$_G$ sarà completamente funzionante ed utilizzabile almeno dal browser$_G$ Google Chrome, dove verrà testato il prodotto durante lo sviluppo. Inoltre, si perseguirà l'obiettivo di garantire la completa funzionalità del prodotto anche su altri browser, in particolare: Mozilla Firefox e Microsoft Edge.
Misure e metriche adottate sono descritte nella sezione 3.2.4.

2.3 Organizzazione

La gestione della strategia di verifica si basa sull'attuazione delle relative attività descritte nelle *Norme di progetto v1.0.0*. Tali attività vengono eseguite per ogni processo attuato allo scopo di verifica della qualità del processo stesso e dell'eventuale prodotto ottenuto facendo riferimento anche alle metriche definite nella sezione 3.2.
In ogni documento è presente un diario delle modifiche che permette di concentrare l'attività di verifica solo nelle parti modificate dopo l'ultima eseguita.
Data la diversa natura dei prodotti ottenuti dai periodi del progetto si applicherà, per ognuno di essi, una diversa procedura di verifica:

- **Analisi e Analisi di dettaglio**: si effettuerà una prima stesura dei documenti illustrati nel *Piano di progetto v1.0.0*;

[4]Consultabili nella sezione 4

  - Verrà controllata la correttezza ortografica con *LanguageTool 3.6*, opportunamente integrato in *TexStudio*;
  - Verrà controllata la correttezza lessicale con un'attenta ed accurata rilettura affiancata dal controllo di *LanguageTool 3.6*;
  - Verrà controllata la correttezza dei contenuti rispetto alle aspettative del documento con un'attenta rilettura;
  - Verrà controllato il corretto tracciamento e la corrispondenza di ciascun requisito con un caso d'uso, mediante l'utilizzo dell'applicativo web creato appositamente[5];
  - Verrà controllato che la stesura di ciascun documento rispetti le norme definite in *Norme di progetto v1.0.0*;
  - Verranno controllate le rappresentazioni grafiche, figure e tabelle assicurandosi che per ciascuna di esse sia presente un'opportuna didascalia e un relativo indice nel corrispondente documento;

- **Progettazione architetturale**: verrà controllato che tutti i requisiti corrispondano ad un componente individuato in questo periodo e se ne assicurerà la tracciabilità;
- **Progettazione di dettaglio e Codifica**: durante ciascuna delle iterazioni in questo periodo i *Programmatori* svolgeranno l'attività di codifica e di esecuzione dei test previsti per la verifica del codice prodotto. Tali attività avverranno nel modo più automatizzato possibile seguendo le norme descritte in *Norme di progetto v1.0.0*. I *Verificatori* avranno il compito di supervisionare le attività controllando la presenza di eventuali errori;
- **Validazione e verifica**: verrà effettuato il collaudo del prodotto, in modo da assicurarne il corretto funzionamento al momento della consegna.

Per ogni periodo a partire dalla progettazione architetturale verranno inoltre effettuati tutti i controlli opportuni descritti al primo punto di questo paragrafo nei nuovi documenti redatti e in presenza di modifiche o integrazioni ai documenti precedentemente stesi.

2.4 Scadenze temporali

Dato l'obiettivo di rispettare le scadenze fissate nel *Piano di progetto v1.0.0*, è indispensabile pianificare anche l'attività di verifica della documentazione e del codice prodotto in modo che risulti sistematica e organizzata. Grazie all'applicazione di tale strategia l'individuazione e la correzione degli errori avverrà il prima possibile, impedendo la loro rapida diffusione e mitigando la possibilità che gli stessi si ripresentino in futuro, diminuendo così il rischio di ritardi. Tale pianificazione è documentata nel *Piano di progetto v1.0.0* il quale contiene, nella sottosezione "Scadenze", le scadenze temporali che il gruppo *KaleidosCode* si impegna a rispettare.

[5] Consultare *Norme di progetto v1.0.0* per maggiori informazioni sullo strumento di tracciamento utilizzato.

3 La strategia di gestione della qualità nel dettaglio

3.1 Risorse

3.1.1 Necessarie

Per la realizzazione del prodotto sono necessarie le risorse umane e tecnologiche elencate di seguito.

- **Risorse umane**: sono descritte dettagliatamente nel *Piano di progetto v1.0.0*:
  - *Responsabile di progetto*;
  - *Amministratore*;
  - *Analista*;
  - *Progettista*;
  - *Programmatore*;
  - *Verificatore*.
- **Risorse software**: sono descritte dettagliatamente nelle *Norme di progetto v1.0.0*. Si tratta di software che permettono:
  - la comunicazione e la condivisione del lavoro tra gli elementi del team;
  - la stesura della documentazione in formato LaTeX$_G$;
  - la creazione di diagrammi UML;
  - la codifica nei linguaggi di programmazione scelti;
  - la semplificazione delle attività di verifica;
  - la gestione dei test sul codice.
- **Risorse hardware**: ciascun componente del gruppo deve avere un computer con tutti i software necessari descritti nelle *Norme di progetto v1.0.0*. È necessario avere a disposizione almeno un luogo dove poter effettuare le riunioni interne.

3.1.2 Disponibili

Ogni membro del team ha a disposizione uno o più computer personali dotati degli strumenti necessari.
Le riunioni interne si svolgono presso le aule del dipartimento di Matematica dell'Università degli Studi di Padova.

3.2 Misure e metriche

Il processo di verifica deve essere quantificabile per fornire informazioni utili. Bisogna quindi stabilire le metriche da adottare per le misurazioni durante i processi di verifica. Si definiranno due intervalli di misure (range$_G$):

- **Range di accettazione**: intervallo di valori vincolante per l'accettazione del prodotto;

- **Range ottimale**: intervallo di valori entro cui è consigliabile rientri la misurazione. Il mancato rispetto di questa condizione non pregiudica l'accettazione del prodotto, ma richiede verifiche più approfondite in merito.

3.2.1 Metriche per i processi

Per poter tenere continuamente sotto controllo il progresso e lo stato del progetto, si è deciso di utilizzare le seguenti metriche.

Schedule Variance$_G$
È una metrica di progetto standard, indica se si è in linea, in anticipo o in ritardo rispetto alla schedulazione pianificata delle attività di progetto. È pari alla differenza tra il valore delle attività realizzate ed il valore delle attività pianificate alla data corrente; più precisamente:

$$Schedule\ Variance\ (SV) = Earned\ Value\ (BCWP) - Planned\ Value\ (BCWS)$$

Dove:

- Earned Value o BCWP (Budgeted Cost of Work Performed) è il valore (denaro) delle attività realizzate alla data corrente;
- Planned Value o BCWS (Budgeted Cost of Work Scheduled) è il costo pianificato (denaro) per realizzare le attività di progetto alla data corrente.

Per avere una migliore visione sull'utilizzo del tempo a disposizione, verrà calcolato anche lo Schedule Performance Indicator ed il To Complete Schedule Performance Indicator[6]. Lo Schedule Performance Indicator,

$$SPI = \frac{BCWP}{BCWS}$$

mostra l'efficienza del team nell'utilizzo del tempo allocato per il progetto; un SPI > 1 indica che il team di lavoro è molto efficiente, viceversa, un SPI < 1 indica che non lo è. Il To Complete Schedule Performance Indicator,

$$TSPI = \frac{(Budget\ Totale - BCWP)}{(Budget\ Totale - BCWS)}$$

mostra quanto il team dovrà essere efficiente nel tempo rimasto per la realizzazione del progetto; un TSPI < 1 indica che il team può rilassare l'attività lavorativa nel tempo rimasto; viceversa, un TSPI > 1 indica che il team ha bisogno di lavorare più duramente.

Parametri utilizzati

- **Range di accettazione**: $\geq -(preventivo * 5\%)$;
- **Range ottimale**: ≥ 0.

[6] I due indici verranno calcolati periodicamente durante lo sviluppo, ma non saranno presenti nei resoconti di verifica.

Budget Variance$_G$
È una metrica di progetto standard, indica se si è speso di più o di meno rispetto a quanto preventivato alla data corrente. È pari alla differenza tra costo pianificato e costo effettivamente sostenuto alla data corrente; più precisamente:

$$Budget\ Variance\ (BV) = Planned\ Value\ (BCWS) - Actual\ Cost\ (ACWP)$$

Dove l'Actual Cost o ACWP (Actual Cost of Work Performed) è il costo effettivamente sostenuto (denaro) alla data corrente.

Parametri utilizzati

- **Range di accettazione**: $\geq -(preventivo * 10\%)$;
- **Range ottimale**: ≥ 0.

3.2.2 Metriche per i documenti

Per poter misurare la documentazione realizzata, è stato scelto di adottare il calcolo di un indice di leggibilità per la lingua italiana.

Indice Gulpease
Definito nel 1988 all'Università degli Studi di Roma "La Sapienza" per valutare la leggibilità di un documento redatto in lingua italiana, l'indice Gulpease si basa sul calcolo del numero di caratteri contenuto in una parola rapportato con altri fattori quali il numero di parole e di frasi. La formula per il calcolo dell'indice Gulpease è la seguente:

$$89 + \frac{300\,(numero\ di\ frasi) - 10\,(numero\ di\ lettere)}{numero\ di\ parole}$$

Il risultato indica quindi la complessità del documento con un valore compreso tra 0 e 100, dove 100 indica la più alta leggibilità. Attraverso gli studi condotti, 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 licenza media;
- **inferiore a 40** sono difficili da leggere per chi ha un diploma superiore.

Tale indice, però, non indica la comprensibilità del testo. Il documento potrebbe contenere frasi incomprensibili ed avere comunque un alto indice Gulpease. Per la tipologia dei documenti redatti, la formalità nella scrittura e gli argomenti trattati risulta difficile adeguare la stesura del testo ad un indice Gulpease ottimale. Ogni documento sarà quindi controllato anche da un essere umano che avrà il compito di valutare se parti di testo dovranno essere semplificate o meno. Inoltre, i limiti imposti da tale indice saranno sufficientemente rilassati per accettare anche frasi poco più complesse.

Parametri utilizzati

- **Range di accettazione**: 40 - 100;
- **Range ottimale**: 50 - 100.

3.2.3 Metriche progettuali

Grado di accoppiamento
Viene calcolato in base a due indici:

- Accoppiamento afferente: numero di classi esterne al package$_G$ che dipendono da sue classi interne. Un numero alto indica che troppe classi dipendono da tale package, quindi eventuali modifiche provocherebbero forti ripercussioni sull'esterno. Se il numero è basso il package risulta poco utile;
- Accoppiamento efferente: numero di classi interne al package dipendenti da classi esterne. Un numero alto può essere sintomo di una scarsa progettazione.

Parametri utilizzati (valgono per entrambe le tipologie di grado di accoppiamento)

- **Range di accettazione**: 0 - 7;
- **Range ottimale**: 0 - 5.

3.2.4 Metriche per il codice

Rapporto linee di commento su linee di codice
Indica il rapporto tra linee di commento e linee di codice in un file (linee vuote escluse). Ritenendo importante la rapidità di comprensione del codice, questa metrica è utile per stimare la manutenibilità.

Parametri utilizzati

- **Range di accettazione**: ≥ 0.25;
- **Range ottimale**: ≥ 0.30.

Numero di parametri
Indica il numero di parametri formali di un metodo. Più alto è il numero dei parametri formali, più aumenta la quantità di memoria occupata nella pila dei processi.

Parametri utilizzati

- **Range di accettazione**: 0 - 8;
- **Range ottimale**: 0 - 5.

Numero di campi dati

Indica il numero di campi dati interni ad una classe. Un numero elevato può rendere difficile la manutenibilità del codice della classe oltre ad essere indice di cattiva programmazione.
È possibile ridurre il numero di campi dati attraverso l'incapsulamento di ulteriori classi.

Parametri utilizzati

- **Range di accettazione**: 0 - 16;
- **Range ottimale**: 0 - 10.

Complessità ciclomatica

Indica il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso del metodo/funzione: i nodi del grafo corrispondono a gruppi indivisibili di istruzioni mentre gli archi connettono due nodi se il secondo gruppo può essere eseguito immediatamente dopo il primo.
È possibile ridurre l'indice di complessità attraverso la suddivisione del metodo/funzione in più parti.

Parametri utilizzati

- **Range di accettazione**: 0 - 10;
- **Range ottimale**: 0 - 6.

È accettato anche un valore più elevato, qualora dovesse influire positivamente sulla velocità di esecuzione.

Livello di annidamento

Indica quante strutture di controllo sono inserite l'una all'interno dell'altra. Un alto livello di annidamento può portare ad una complessità maggiore del codice causando difficoltà nella verifica, comprensione e modifica dello stesso.

Parametri utilizzati

- **Range di accettazione**: 0 - 6;
- **Range ottimale**: 0 - 4.

Chiamate innestate di metodi

Indica quante chiamate innestate di metodi sono inserite l'una all'interno dell'altra. Un alto valore può portare a una saturazione dello stack_{G}.

Parametri utilizzati

- **Range di accettazione**: 0 - 6;
- **Range ottimale**: 0 - 4.

Copertura del codice
Rappresenta la percentuale di codice eseguita durante i test. Maggiore è questo valore, più esaurienti saranno i test e maggiori sono le probabilità di individuare gli eventuali errori.

Parametri utilizzati

- **Range di accettazione**: 80% - 100%;
- **Range ottimale**: 90% - 100%.

Numero di linee per metodo
Indica il numero di statement$_G$ che compongono un metodo. Se un metodo risulta troppo lungo il suo funzionamento risulterà più complicato da comprendere, quindi può essere opportuno dividerlo in più sotto-funzioni. Un metodo troppo lungo potrebbe addirittura essere sintomo di cattiva progettazione della classe.

Parametri utilizzati

- **Range di accettazione**: ≤ 60;
- **Range ottimale**: ≤ 40.

Validazione W3C$_G$
L'applicativo web deve superare il test di validazione offerto da W3C con 0 errori gravi. Sono accettati avvisi ed inesattezze che non compromettano le funzionalità del sito fino a un massimo di 10 per pagina.

Parametri utilizzati

- **Range di accettazione**: 0 - 10 (per pagina);
- **Range ottimale**: 0 - 0 (per pagina).

3.2.5 Tabella riepilogativa

Metriche	Range di accettazione	Range ottimale
Metriche per i processi		
Schedule Variance	$\geq -(preventivo * 5\%)$	≥ 0
Budget Variance	$\geq -(preventivo * 10\%)$	≥ 0
Metriche per i documenti		
Indice Gulpease	40 - 100	50 - 100
Metriche progettuali		
Grado di accoppiamento afferente	0 - 7	0 - 5
Grado di accoppiamento efferente	0 - 7	0 - 5
Metriche per il codice		
Linee di commento su linee di codice	≥ 0.25	≥ 0.30
Numero di parametri	0 - 8	0 - 5
Numero di campi dati	0 - 16	0 - 10
Complessità ciclomatica	0 - 10	0 - 6
Livello di annidamento	0 - 6	0 - 4
Chiamate innestate di metodi	0 - 6	0 - 4
Copertura del codice	80% - 100%	90% - 100%
Numero di linee per metodo	≤ 60	≤ 40
Validazione W3C	0 - 10 (per pagina)	0 - 0 (per pagina)

Tabella 2: Riepilogo misure e metriche adottate

4 Test

Sono state individuate quattro tipologie di test:

- **Test di unità**: servono alla verifica della correttezza degli algoritmi;
- **Test di integrazione**: servono alla verifica della correttezza delle componenti individuate;
- **Test di sistema**: servono alla verifica del corretto funzionamento dell'architettura e della soddisfazione dei requisiti descritti nell'*Analisi dei requisiti*;
- **Test di validazione**: servono per accertarsi che il prodotto sia conforme con quanto concordato con il Proponente.

La classificazione ed il tracciamento dei test è definito nelle *Norme di progetto v1.0.0*.

4.1 Test di validazione

I test di validazione vengono effettuati con il Proponente. Per ogni test è descritta una serie di passi che l'utente deve seguire in modo tale da effettuarlo correttamente.

4.2 Test di sistema

I test di sistema vengono eseguiti sul prodotto finito, cioè quando tutte le sue componenti sono integrate.

4.3 Test di integrazione

5 Resoconto delle attività di verifica

5.1 Periodo di Analisi e Analisi di dettaglio

5.1.1 Indici di Gulpease

Documento	Valutazione	Esito
Analisi dei requisiti v1.0.0	41.08	Accettabile
Glossario v1.0.0	45.95	Accettabile
Norme di progetto v1.0.0	46.95	Accettabile
Piano di progetto v1.0.0	48.23	Accettabile
Piano di qualifica v1.0.0	53.99	Ottimale
Studio di fattibilità v1.0.0	46.73	Accettabile

Tabella 3: RR - Indici di Gulpease calcolati sulla documentazione prodotta

5.2 Periodo di Progettazione architetturale

A Capability Maturity Model

Il Capability Maturity Model (CMM), divenuto CMMI (I per Integration), è un modello per il miglioramento dei processi di sviluppo del software il cui obiettivo è di aiutare un'organizzazione a migliorare le sue prestazioni in termini di qualità del software prodotto, produttività dell'organizzazione e riduzione dei tempi di sviluppo.
Le lettere dell'acronimo indicano:

- **Capability**: si indica la misura di quanto un singolo processo è adeguato allo scopo per il quale è stato definito e determina il range del risultato raggiungibile utilizzando quel processo in termini di efficienza ed efficacia;
- **Maturity**: si indica la misura che comunica quanto è governato l'insieme di processi aziendali ed è influenzata dalle capability dei processi coinvolti;
- **Model**: si indica l'insieme dei requisiti che vanno ad essere sempre più stringenti per valutare il miglioramento dei processi aziendali;
- **Integration**: si indica l'architettura di integrazione delle diverse discipline (system, hardware, software) e tipologie di attività delle aziende.

A.1 Struttura

Il modello coinvolge cinque aspetti:

- **Livelli di maturità**: un processo di continua maturità a cinque livelli dove il più alto indica lo stato ideale dove i processi sono sistematicamente gestiti da una combinazione di ottimizzazione e continuo miglioramento di processi;
- **Aree chiave di processo**: un'area chiave di processo identifica un insieme di attività che raggiungono obiettivi ritenuti importanti quando svolte assieme;
- **Obiettivi**: gli obiettivi di un'area chiave di processo indicano lo scopo, i confini e le intenzioni di quest'ultima;
- **Caratteristiche comuni**: includono pratiche che implementano e istituzionalizzano un'area chiave di processo; sono suddivise in cinque tipi: impegno nell'operare, abilità nell'operare, attività svolte, misurazioni ed analisi, verifica dell'implementazione;
- **Pratiche chiave**: descrivono gli elementi di infrastruttura e pratiche che contribuiscono maggiormente nell'implementazione ed istituzionalizzazione dell'area.

A.2 Livelli

I livelli di maturità sopra citati sono i seguenti:

1. **Iniziale**: i processi sono imprevedibili, insufficientemente controllati e reattivi; risultano non documentati e perciò non sono ripetibili;
2. **Ripetibile**: i processi sono sufficientemente documentati tanto da renderli ripetibili;

3. **Definito**: i processi sono definiti, documentati e ripetibili;
4. **Gestito**: i processi sono controllati e gestiti attraverso analisi e utilizzo di metriche concordate;
5. **Efficiente**: la gestione dei processi punta al loro miglioramento/ottimizzazione.

B Ciclo di Deming

Il ciclo di Deming o ciclo PDCA (Plan-Do-Check-Act) è un metodo di gestione iterativo focalizzato sul miglioramento continuo dei processi; l'acronimo "PDCA" definisce i passi in cui è diviso:

- Pianificare (**P**lan): definizione delle attività e dei processi da migliorare secondo misurazioni effettuate nonché scadenze e risorse utili al raggiungimento del miglioramento;
- Fare (**D**o): attuazione delle azioni pianificate al passo precedente con conseguente misurazione e raccolta di dati utili ai passi successivi;
- Verificare (**C**heck): studio dei risultati misurati e raccolti nel passo precedente e confronto con i risultati attesi;
- Agire (**A**ct): standardizzazione dei cambiamenti apportati nell'esecuzione di processo (solamente se l'esito del passo precedente è positivo).

Per poter applicare il ciclo PDCA è necessario che i processi siano documentati, analizzabili e ripetibili per poter individuare gli eventuali errori da correggere.

C ISO/IEC 9126

La normativa [ISO/IEC 9126] si suddivide in:

- Modello della qualità del software;
- Metriche per la qualità esterna;
- Metriche per la qualità interna;
- Metriche per la qualità in uso.

C.1 Modello della qualità

C.1.1 Qualità esterna ed interna

Il modello di qualità stabilito dallo standard è classificato da sei caratteristiche generali:

- **Funzionalità**: capacità del prodotto di fornire l'insieme di funzioni per soddisfare le richieste e gli obiettivi dell'utente;

- **Affidabilità**: capacità del prodotto di mantenere un certo livello di prestazioni quando utilizzato in particolari condizioni in un periodo temporale definito;
- **Efficienza**: capacità del prodotto di fornire determinate prestazioni in relazione alla quantità di risorse utilizzate;
- **Usabilità**: capacità del prodotto di essere capito e usato dall'utente in specifiche condizioni;
- **Manutenibilità**: capacità del software di poter essere modificabile, correggendolo, migliorandolo o adattandolo;
- **Portabilità**: capacità del software di essere trasportato da un ambiente di lavoro ad un altro.

Sono presenti anche varie sotto-caratteristiche misurabili attraverso metriche.

C.1.2 Qualità in uso

La qualità in uso rappresenta la qualità del prodotto software dal punto di vista dell'utente ed è classificata da quattro caratteristiche:

- **Efficacia**: capacità del software di mettere in grado gli utenti di raggiungere i loro obiettivi con accuratezza e completezza;
- **Produttività**: capacità di mettere in grado gli utenti di utilizzare una quantità di risorse relativamente all'efficacia ottenuta in uno specifico contesto d'uso;
- **Soddisfazione**: capacità del prodotto di soddisfare gli utenti;
- **Sicurezza**: capacità del prodotto di raggiungere accettabili livelli di rischio di danni a persone, software, strumenti o all'ambiente operativo.

C.2 Metriche per la qualità

C.2.1 Esterna

Le metriche esterne misurano i comportamenti del software rilevati da test, operatività e osservazione durante la sua esecuzione in un contesto tecnico rilevante.

C.2.2 Interna

Le metriche interne sono applicate a software non eseguibile (codice sorgente) e documentazione durante la progettazione e la codifica. Le misure effettuate permettono di prevedere il livello di qualità esterna ed in uso del prodotto finale data l'influenza degli attributi interni su quelli esterni e quelli in uso. Le metriche interne permettono di individuare eventuali problemi che potrebbero influire sulla qualità finale del prodotto prima che sia realizzato effettivamente il codice eseguibile.