Piano di Qualifica

Gruppo SWEet BIT — Progetto SWEDesigner

Informazioni sul documento	
Versione	1.2.0
Redazione	Salmistraro Gianmarco
	Santimaria Davide
	Pilò Salvatore
Verifica	Bodian Malick
	Massignan Fabio
Approvazione	Pilò Salvatore
Uso	Esterno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Zucchetti S.p.A.

Descrizione

Questo documento descrive le strategie di verifica, adottate dal gruppo SWEet BIT, atte a garantire gli obiettivi qualitativi riguardanti il prodotto SWEDesigner.

Indice

Elenco delle figure

Elenco delle tabelle

Versioni del documento

Versione	Data	Persone coinvolte	Descrizione
1.2.0	2017/04/01	Pilò Salvatore	Approvazione documento
1.1.1	2017/03/27	Massignan Fabio	Verifica documento
1.1.0	2017/03/26	Bodian Malick	Verifica documento
1.0.4	2017/03/22	Massignan Fabio	Stesura Capitolo Resoconto delle attività di verifica
1.0.3	2017/03/15	Pilò Salvatore	Stesura Capitolo Standard e metodi per la gestione della Qualità
1.0.2	2017/03/09	Salmistraro Gianmarco	Stesura Capitolo Strategia di verifica nel dettaglio
1.0.1	2017/03/03	Salmistraro Gianmarco	Stesura Capitoli: Introduzione e Visione generale delle strategie di verifica
1.0.0	2017/03/02	Santimaria Davide	Creazione Struttura del documento

1 Introduzione

1.1 Scopo del documento

Questo documento descrive le strategie adottate dal gruppo SWEet BIT per il conseguimento degli obiettivi di qualità riguardanti il prodotto. Il raggiungimento di tali obiettivi è possibile solo tramite una precisa e continua verifica delle attività svolte; così facendo è possibile rilevare e correggere tempestivamente eventuali anomalie, minimizzando lo spreco di risorse.

1.2 Scopo del Prodotto

Lo scopo del progetto è la realizzazione di una *Web App*$_G$ che fornisca all'*Utente*$_G$ un *UML*$_G$ *Designer*$_G$ con il quale riuscire a disegnare correttamente *Diagrammi delle classi*$_G$ e descrivere il comportamento dei *Metodi*$_G$ interni alle stesse attraverso l'utilizzo del *Diagramma delle attività*$_G$. La *Web App*$_G$ permetterà all'*Utente*$_G$ di generare *Codice*$_G$ *Java*$_G$ dal *Diagramma*$_G$ disegnato ed eventualmente andare a ritoccarne il risultato al fine di ottenere un *Codice*$_G$ eseguibile, funzionante e funzionale.

1.3 Glossario

Con lo scopo di evitare ambiguità di linguaggio e di massimizzare la comprensione dei documenti, il gruppo ha steso un documento interno che è il *Glossario 1.2.0*. In esso saranno definiti, in modo chiaro e conciso, i termini che possono causare ambiguità o incomprensione del testo.

1.4 Riferimenti

1.4.1 Informativi

- **Analisi dei requisiti:** 1.2.0
- **Piano di progetto:** 1.4.0
- **Lucidi dell'insegnamento di Ingegneria del *Software*:**
  http://www.math.unipd.it/~tullio/IS-1/2016/
- **Libro SWEBOK v3.0: *Chapter 10: Software Quality*:**
  https://www.computer.org/web/swebok/
- ***ISO*$_G$*/IEC*$_G$ 9126 *Software engineering — Product quality*:**
  https://en.wikipedia.org/wiki/ISO/IEC_9126

- ***ISO*$_G$/*IEC*$_G$ 15504 *Information technology – Process assessment*:** https://en.wikipedia.org/wiki/ISO/IEC_15504

1.4.2 Normativi

- **Norme di progetto:** 1.2.0
- **Capitolato di appalto SWE*Designer*$_G$ (C6):** http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/C6.pdf

2 Visione generale delle strategie di verifica

2.1 Definizione obiettivi di qualità

Questa sezione si presta a descrivere sia gli obiettivi di qualità riguardanti il prodotto che quelli relativi ai processi necessari alla sua produzione.

2.1.1 Qualità di prodotto

Il gruppo SWEet BIT si è impegnato ad aderire allo standard ISO_G/IEC_G 9126, in modo garantire le seguenti caratteristiche di qualità per il prodotto:

- **Funzionalità**: il prodotto deve soddisfare i requisiti indicati nel documento *Analisi dei Requisiti 1.2.0*;
- **Affidabilità**: il software prodotto deve essere robusto e deve essere prevista una completa gestione degli errori ed eccezioni, in modo da prevenire perdite di dati e facilitarne il ripristino;
- **Usabilità**: il software prodotto deve risultare semplice e di facile utilizzo da parte dell'$Utente_G$. Esso deve essere di facile apprendimento e l'interfaccia $Utente_G$ deve risultare familiare e intuitiva. Il prodotto deve soddisfare al meglio le esigenze dell'$Utente_G$ finale.
  La documentazione per l'utilizzo del prodotto deve essere esaustiva e di facile comprensione;
- **Efficienza**: Il software deve essere capace di fornire tutti i servizi attesi con il minimo dispendio di risorse;
- **Manutenibilità**: Il software prodotto deve essere facilmente modificabile, in caso di correzione di errori e migliorie da apportare. Esso deve essere facilmente adattabile in caso di cambio di ambiente operativo e/o cambio di requisiti;
- **Portabilità**: Il software prodotto deve, per la parte $front\ end_G$, offrire gli stessi servizi nella maggior parte dei $browser_G$. La parte di $back\ end_G$ deve poter funzionare su vari sistemi operativi.

Per ognuna delle caratteristiche principali appena elencate esistono diverse sotto-caratteristiche, le quali sono approfondite nell'Appendice *A - Standard e Metodi$_G$ per la gestione della Qualità* di questo documento, dove è descritto lo standard ISO_G/IEC_G 9126.

2.1.2 Qualità di processo

La qualità dei processi che portano allo sviluppo di un software hanno un ruolo fondamentale sulla qualità del prodotto. Il gruppo SWEet BIT ha deciso di adottare il *Ciclo*

di Deming, o *Ciclo PDCA*$_G$, come modello per il continuo miglioramento dei processi produttivi. Come complemento al ciclo di Deming è stato scelto il modello descritto dallo standard *ISO*$_G$/*IEC*$_G$ 15504 detto *SPICE*$_G$, il quale fornisce gli strumenti per la valutazione dei processi produttivi. Sia lo standard *ISO*$_G$/*IEC*$_G$ 15504 che il *Ciclo di Deming* sono descritti nell'Appendice *A - Standard e Metodi*$_G$ *per la gestione della Qualità* del presente documento.

2.2 Organizzazione

Per ognuno dei periodi descritto dal 1.4.0 è necessario seguire una verifica mirata al tipo di processo e relativo prodotto:

- **Analisi:** in questo periodo vengono controllati i processi e la documentazione prodotta. La verifica dovrà essere eseguita secondo i metodi descritti nel documento *Norme di Progetto 1.2.0*;
- **Consolidamento dei requisiti:** durante questo periodo sono verificati i processi che portano all'incremento dei documenti, e i documenti stessi, prodotti durante il periodo di analisi. Le procedure di verifica sono descritte nel documento *Norme di Progetto 1.2.0*;
- **Progettazione Architetturale:** in questo periodo vengono verificati i processi, e relativi prodotti, che portano all'incremento dei documenti redatti nel precedente periodo di *Consolidamento dei Requisiti*; sempre in questa fase sono verificati i processi e prodotti riguardanti la *Progettazione Architetturale*. Le procedure di verifica sono descritte in dettaglio nel documento *Norme di Progetto 1.2.0*;
- **Progettazione di Dettaglio e Codifica:** anche in questo periodo vengono verificati i processi, e relativi prodotti, che portano all'incremento dei documenti del periodo di *Progettazione Architetturale*. Vengono verificati processi e prodotti della sotto-fase di codifica. Le procedure di verifica sono descritte nel documento *Norme di Progetto 1.2.0*.
- **Verifica e Validazione:** sono verificati i processi che portano all'aggiornamento della documentazione e i documenti stessi. In questo periodo vengono effettuati i test per la validazione del prodotto finale.

2.3 Pianificazione strategica e temporale

Intendendo rispettare le scadenze specificate nel *Piano di Progetto 1.4.0*, è necessario un approccio organico e ben pianificato alle attività di verifica. Deve essere pianificato ogni incremento su *Codice*$_G$ e documentazione e di conseguenza lo dovranno essere tutte le attività di verifica, in modo da avere il tempo di effettuare le dovute correzioni. Ogni processo di verifica dovrà essere automatizzato il più possibile per poter, dove possibile,

risparmiare su risorse umane. L'utilizzo di software apposito, e dei metodi per la verifica, sono descritti nel documento *Norme di Progetto 1.2.0*.

2.4 Gestione delle responsabilità

Il *Responsabile di Progetto* e i *Verificatori* hanno la responsabilità delle attività di verifica. I compiti di queste figure sono argomento trattato nel documento *Piano di Progetto 1.4.0*.

2.5 Risorse

Si possono identificare due distinti tipi di risorse necessari a garantire la qualità dei prodotti e processi e sono:

- **Risorse umane:** riguardano i ruoli rivestiti dai vari componenti del gruppo SWEet BIT:
  - Responsabile di progetto;
  - Amministratore;
  - Analista;
  - Progettista;
  - Programmatore;
  - Verificatore.

  La responsabilità maggiore per l'attività di verifica e validazione ricade sul Responsabile di Progetto e sul Verificatore. Per una descrizione dettagliata di tutti gli altri ruoli e delle loro responsabilità fare riferimento al *Piano di Progetto 1.4.0*.
- **Risorse tecnologiche:** riguardano tutti i software utilizzati per l'automazione delle attività di verifica su prodotti e processi, e l'hardware necessario per eseguire i suddetti software. Nel documento *Norme di Progetto 1.2.0* sono descritti nel dettaglio tutti gli strumenti software utilizzati.

2.6 Tecniche di analisi

2.6.1 Analisi statica

L'analisi statica è il processo di verifica del sistema o di un suo componente, senza che esso debba necessariamente poter essere eseguito. Questa tecnica permette di verificare sia la documentazione che il $Codice_G$, individuandone errori ed anomalie. Questa tecnica di analisi può essere svolta in due modi distinti e complementari.

2.6.1.1 Walkthrough Si svolge effettuando una lettura critica e a largo spettro del documento. È una tecnica utilizzata soprattutto nelle prime attività del progetto, quando ancora non è presente un'esperienza tale da permettere una verifica più mirata e precisa. Il verificatore genererà, infatti, una lista di controllo con gli errori più frequenti in modo da favorire il miglioramento della verifica nei periodi successivi. Il walkthrough è un'attività onerosa e collaborativa che richiede l'intervento di più persone per essere efficace. Questa tecnica si svolge in tre fasi principali:

- **Fase uno:** lettura dei documenti ed individuazione degli errori;
- **Fase due:** discussione dei errori riscontrati e delle correzioni da applicare;
- **Fase tre:** correzione degli errori rilevati, e stesura di un rapporto che ne elenchi le modifiche effettuate

La lista di controllo risultante, contenente le tipologie di errori più frequenti è in appendice alle *Norme di Progetto 1.2.0*

2.6.1.2 Inspection Consiste nell'analisi mirata di alcune parti del documenti o del codice ritenute fonti maggiori di errore. Deve essere seguita una lista di controllo per svolgere efficacemente questa attività; tale lista deve essere redatta anticipatamente ed è sostanzialmente frutto dell'esperienza maturata dai membri del team con tecniche di walkthrough. L'inspection è dunque più rapida del walkthrough, in quanto il documento viene analizzato solo in alcune sue parti e con una lista di controllo ben precisa. In questa attività sono coinvolte solo i verificatori che, dopo aver individuato gli errori, procedono alla loro correzione e alla redazione di un rapporto che tenga traccia del lavoro svolto.

2.6.2 Analisi dinamica

L'analisi dinamica è il processo che verifica il prodotto software mentre esso è in esecuzione. Viene effettuata tramite test che devono essere coerentemente pianificati in modo da evitare spreco di risorse. L'obiettivo dei test sul software è la realizzazione di un prodotto il più possibile esente da errori. Il principale ostacolo alla fase di test è sintetizzato nella tesi di Dijkstra, che afferma che il test può indicare la presenza di errori, ma non ne può garantire l'assenza. Affinché tale attività sia utile e generi risultati attendibili è necessario che i test effettuati siano ripetibili: dato un certo input deve essere prodotto sempre uno stesso output in uno specifico ambiente. Di conseguenza, i tre elementi fondamentali di un test sono:

- **Ambiente:**sistema hardware e software sui quali è stato pianificato l'utilizzo del prodotto software sviluppato. Su di essi deve essere specificato uno stato iniziale dal quale poter eseguire il test;
- **Specifica:**definizione di quali input sono necessari per l'esecuzione del test e quali output sono attesi;

- **Procedure:**definizione di come devono essere svolti i test, in che ordine devono essere eseguiti e come devono essere analizzati i risultati.

Organizzazione delle attività di test:

- **Test di unità**: l'obiettivo di questo test è verificare che ogni unità del $Codice_G$ si comporti esattamente come previsto. Un unità di $Codice_G$ è la più piccola parte di software che è utile verificare singolarmente e che viene prodotta da un singolo programmatore. Attraverso tali test si vuole verificare il corretto funzionamento dei moduli che compongono l'intero sistema, in modo da esaminare possibili errori di implementazione da parte dei programmatori.
- **Test di integrazione**: l'obiettivo di questo test è verificare che le componenti del sistema vengano aggiunte incrementalmente al prodotto e analizzare che la combinazione di due o più unità software funzioni come previsto. Una componente è l'aggregazione di più unità software. Questo tipo di test serve ad individuare errori residui nella realizzazione dei singoli moduli, modifiche delle interfacce e comportamenti inaspettati di componenti software preesistenti forniti da terze parti che non si conoscono a fondo. er effettuare tali test devono essere aggiunte delle componenti fittizie al posto di quelle che non sono ancora state sviluppate, in modo da non influenzare negativamente l'esito dell'analisi.
- **Test di sistema**: l'obiettivo di questo test è verificare il comportamento del prodotto software finale. Lo scopo è quello di capire se esso rispetta i requisiti individuati nella fase di Analisi.
- **Test di regressione**: l'obiettivo di questo test è stabilire se le modifiche apportate al software hanno compromesso componenti software precedentemente funzionanti. Questi test vengono eseguiti ogni volta che viene apportata una modifica al software. Tale operazione è aiutata dal tracciamento, che permette di individuare e ripetere facilmente i test di unità, integrazione ed eventualmente sistema che sono stati potenzialmente influenzati dalla modifica;
- **Test di accettazione**: l'obiettivo di questo test è la realizzazione del collaudo del prodotto software eseguito in presenza del proponente. Se l'esito risulta positivo, si può procedere al rilascio ufficiale del prodotto.

2.7 Misure e metriche

Il processo di verifica, per essere informativo, deve esse quantificabile. Le misure rilevate dal processo di verifica devono quindi essere basate su metriche stabilite a priori. Per automatizzare il più possibile il lavoro di misurazione saranno utilizzati degli strumenti, adeguatamente configurati, definiti nelle *Norme di Progetto 1.2.0*, con lo scopo di avere un resoconto affidabile e quantitativo che permetta di assicurare il grado di qualità voluto. Le metriche, essendo di natura molto variabile, vi possono essere due tipologie di range:

- **Accettazione**: l'intervallo di valori entro il quale deve trovarsi il risultato della misurazione di un processo, o prodotto, per essere ritenuto accettabile;
- **Ottimale**: l'intervallo di valori in cui si deve trovare il risultato della misurazione di un processo, o prodotto, per essere considerato ottimale. Questo range è l'obiettivo del gruppo SWEet BIT.

2.7.1 Metriche per i processi

Le seguenti metriche rappresentano un indicatore volto a monitorare i tempi e i costi associati al progetto. Sono metriche che danno un riscontro immediato sullo stato attuale del progetto, mantenendo il controllo sui processi durante il loro svolgimento. Per ogni metrica indicata è stato definito nelle *Norme di Progetto 1.2.0* come si calcola, e con quale strumento. In *Appendice B* si può visionare il resoconto di tali metriche.

2.7.1.1 Schedule Variance (SV): è un indice che dà informazioni necessarie a determinare se ci si trova in anticipo, in ritardo o in linea alle tempistiche delle attività di progetto. Si tratta di un indicatore di efficacia nei confronti del cliente. Se il risultato di tale metrica risulta positivo significa che il progetto sta procedendo con maggior velocità rispetto a quanto pianificato, viceversa se negativo. Alla fine del progetto questo indice assumerà il valore 0, perchè in quel momento tutte le attività saranno state realizzate.

2.7.1.2 Budget Variance (BV): è un indice che dà informazioni necessarie a determinare se vi sono state più o meno spese rispetto al previsto. Si tratta di un indicatore che ha un valore contabile e finanziario. Se il risultato di tale metrica risulta positivo significa che l'attuazione del progetto sta consumando il proprio budget con minor velocità rispetto a quanto pianificato, viceversa se negativo.

2.7.2 Metriche per i documenti

La seguente metrica rappresenta un indicatore volto a misurare la complessità con la quale vengono costruite le frasi all'interno dei documenti. È stato definito nelle *Norme di Progetto 1.2.0* come si calcola, e con quale strumento. In *Appendice B* si può visionare il resoconto di tale metrica.

2.7.2.1 Indice Gulpease: è un indice di leggibilità di un testo per la lingua italiana. Rispetto ad altri ha il vantaggio di utilizzare la lunghezza delle parole in lettere anziché in sillabe, semplificandone il calcolo automatico. Permette di misurare la complessità dello stile di un documento.

2.7.3 Metriche per il software

Le presenti metriche rappresentano gli obbiettivi di qualità per il prodotto software da perseguire. Per ogni metrica indicata è stato definito nelle *Norme di Progetto 1.2.0* come si calcola, e con quale strumento. In *Appendice B* si può visionare il resoconto di tali metriche.

2.7.3.1 Dimensione del prodotto software: Rappresenta le dimensioni del prodotto software, è misurata in termini di migliaia di linee di codice (KLOC s, Thousands Line Of Code), ma da alcuni anni è stata introdotto una nuova misura, legata al numero di funzionalità offerte, e quindi dal valore che il prodotto ha per l'utente.

2.7.3.2 Complessità ciclomatica: è una metrica per la misura della complessità di funzioni, moduli, metodi o classi di un programma; la quale è calcolata mediate il grafo di controllo di flusso relativo al $Codice_G$. I nodi del grafo sono gruppi di istruzioni indivisibili. Se nel $Codice_G$ non sono presenti punti decisionali o cicli, allora la complessità ciclomatica sarà pari a 1. Alti valori di complessità ciclomatica implicano una ridotta manutenibilità del codice. Valori bassi di complessità ciclomatica potrebbero però determinare scarsa efficienza dei metodi.

2.7.3.3 Numero di file: rappresenta la media di occorrenze di metodi per file: un file, infatti, non dovrebbe contenere un numero eccessivo di metodi. Valori troppo elevati indicano la necessità di una migliore decomposizione del file.

2.7.3.4 Variabili non utilizzate e/o non definite: rappresenta il numero di variabili che vengono definite, ma non utilizzate, o viceversa. Questo viene considerato pollution, e pertanto considerato inaccettabile. La misurazione avviene mediante un'analisi dell'AST (Abstract Syntax Tree).

2.7.3.5 Numero di argomenti per funzione: rappresenta il numero di argomenti di una funzione. Una funzione con troppi argomenti risulta complessa e poco mantenibile; pertanto è necessario che tale numero sia contenuto.

2.7.3.6 Linee di codice per linee di commento: rappresenta il rapporto tra le linee di codice e linee di commento, utile per stimare la manutenibilità del codice.

2.7.3.7 Copertura del codice: rappresenta la percentuale di istruzioni che sono eseguite durante i test. Maggiore è la percentuale di istruzioni coperte dai test eseguiti, maggiore sarà la probabilità che le componenti testate abbiano una ridotta quantità di

errori. Il valore di tale indice può essere abbassato da metodi molto semplici che non richiedono testing.

A Standard e Metodi per la gestione della Qualità

A.1 ISO/IEC 9126 *Software engineering — Product quality*

L'obiettivo di questo standard è quello di definire un modello per poter valutare la qualità del software.
Lo standard si divide in quattro parti:

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

A.1.1 Modello della qualità del software

Il modello della qualità del software presentato nella prima parte dello standard (ISO_G/IEC_G 9126-1) identifica sei caratteristiche generali e varie sottocaratteristiche:

1. **Funzionalità:**
   - **Adeguatezza**: la capacità del software di offrire un appropriato insieme di funzioni per determinati compiti;
   - **Accuratezza**: la capacità del prodotto software di rendere risultati concordati o i precisi effetti aspettati;
   - **Interoperabilità**: capacità del prodotto software di operare e interagire con sistemi uno o più sistemi specificati;
   - **Sicurezza**: la capacità di proteggere dati e informazioni, impedendo a persone e sistemi non autorizzati di accedervi e garantendo sempre l'accesso ai dati a persone e sistemi autorizzati;
   - **Conformità funzionale**: capacità del prodotto software di aderire a standard, convenzioni e regolamentazioni riguardanti il settore operativo a cui vengono applicate.
2. **Affidabilità:**
   - **Maturità**: capacità del prodotto software di evitare il verificarsi di errori, malfunzionamenti o che siano prodotti risultati errati;
   - **Tolleranza agli errori**: è la capacità del prodotto software di mantenere livelli predeterminati di prestazioni anche in presenza di malfunzionamenti o usi scorretti del prodotto;

- **Recuperabilità**: capacità di ripristinare il livello appropriato di prestazioni e di recupero delle informazioni rilevanti, in seguito a un malfunzionamento. A seguito di un errore, il software può risultare non accessibile per un determinato periodo di tempo e questo arco di tempo è valutato proprio dalla caratteristica di recuperabilità;
- **Aderenza**: è la capacità di aderire a standard, regole e convenzioni inerenti l'affidabilità.

3. **Usabilità:**
   - **Comprensibilità**: è la capacità del prodotto software di mettere l'*Utente*$_G$ in grado di comprendere se il software è appropriato e come esso possa essere usato per scopi e condizioni d'uso particolari;
   - **Apprendibilità**: è la capacità di ridurre l'impegno richiesto agli utenti per imparare ad usare la sua *Applicazione*$_G$;
   - **Operabilità**: è la capacità del prodotto software di rendere l'*Utente*$_G$ in grado di operarlo e controllarlo;
   - **Attrattiva**: è la capacità del software di essere piacevole per l'*Utente*$_G$;
   - **Conformità**: è la capacità del prodotto software di aderire a standard o convenzioni relativi all'usabilità.
4. **Efficienza:**
   - **Comportamento rispetto al tempo**: è la capacità di fornire adeguati tempi di risposta, elaborazione e velocità di attraversamento, sotto determinate condizioni;
   - **Utilizzo delle risorse**: capacità del prodotto software di usare un adeguato quantitativo e tipo di risorse quando il software esegue le sue funzioni in determinate condizioni;
   - **Conformità**: è la capacità di aderire a standard e specifiche sull'efficienza.
5. **Manutenibilità:**
   - **Analizzabilità**: è la capacità del prodotto software di essere facilmente controllato per la ricerca di errori o di facilitare l'identificazione delle parti che devono essere modificate;
   - **Modificabilità**: capacità del prodotto software di rendere possibili eventuali implementazioni di modifiche;
   - **Stabilità**: è la capacità del prodotto software di evitare effetti indesiderati dovuti alle modifiche del software stesso;

- **Testabilità**: è la capacità del prodotto software di poter validare le modifiche ad esso apportate;
- **Conformità di manutenibilità**: è la capacità di aderire a standard e specifiche riguardanti la manutenibilità.

6. **Portabilità:**
   - **Adattabilità**: è la capacità del software di essere adattato per differenti ambienti operativi senza dover applicare modifiche diverse da quelle fornite per il software considerato;
   - **Installabilità**: è la capacità del software di essere installato in uno specifico ambiente;
   - **Sostituibilità**: è la capacità di essere utilizzato al posto di un altro software per svolgere gli stessi compiti nello stesso ambiente;
   - **Conformità**: è la capacità del prodotto software di aderire a standard e convenzioni relative alla portabilità.

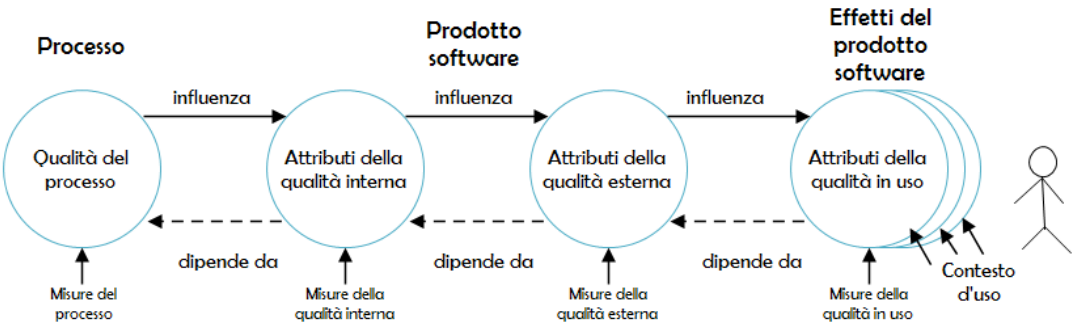

Figura 1: Ciclo di qualità del software

A.1.2 Metriche per la qualità interna

Le metriche per la qualità interna o metriche interne, descritte nel *technical report* ISO_G/IEC_G 9126-3, sono delle metriche che si applicano al software non eseguibile durante i periodi di progettazione e codifica. Le metriche interne permettono di individuare eventuali problemi che potrebbero influire sulla qualità finale del prodotto prima che sia realizzato il software eseguibile. Le misure effettuate permettono di prevedere il livello di qualità esterna ed in uso del prodotto finale poiché gli attributi interni influiscono su quelli esterni e quelli in uso.

A.1.3 Metriche per la qualità esterna

Le metriche per la qualità esterna o metriche esterne, descritte nel *technical report* ISO_G/IEC_G 9126-2, sono delle metriche adatte alla misurazione dei comportamenti del software sulla base di misure ottenute da test, operando e osservando il software eseguibile o il sistema stesso.

A.1.4 Metriche per la qualità in uso

Le metriche per la qualità in uso misurano il grado con cui il prodotto software permette agli utenti di svolgere le proprie attività con efficacia, produttività, sicurezza e soddisfazione nel contesto operativo previsto. Questa valutazione è quindi fatta in specifici scenari d'uso.
La qualità deriva anche dall'effetto combinato di più caratteristiche interne ed esterne di qualità. Il *technical report* ISO_G/IEC_G 9126-4 fornisce alcune metriche per la misurazione della qualità in uso.

A.2 ISO_G/IEC_G 15504 *Information technology – Process assessment*

ISO_G/IEC_G 15504, anche conosciuta come SPICE (Software Process Improvement and Capability Determination), è un insieme di nove documenti di standard tecnici relativi ai processi di sviluppo del software e relative funzioni di business e, in particolare, alla loro valutazione.

A.2.1 Modello di riferimento

ISO_G/IEC_G 15504 presenta un modello di riferimento che definisce una dimensione del processo e una della capacità.
La dimensione del processo definisce processi divisi in cinque categorie di processo:

- *Customer/Suplier*;
- *Engineering*;
- *Supporting*;
- *Management*;
- *Organization*.

La dimensione della capacità definisce una scala di maturità a cinque livelli (più il livello base, detto livello 0) così definiti:

- *Level 5: pptimizing process*;
- *Level 4: predictable process*;
- *Level 3: established process*;
- *Level 2: managed process*;
- *Level 1: performed process*;
- *Level 0: incomplete process.*

La capacità dei processi è misurata tramite gli attributi definiti a livello internazionale e che sono:

- *1.1 process performance*;
- *2.1 performance management*;
- *2.2 work product management*;
- *3.1 process definition*;
- *3.2 process deployment*;
- *4.1 process measurement*;
- *4.2 process control*;
- *5.1 process innovation*;
- *5.2 process optimization.*

Ogni attributo del processo consiste di una o più pratiche generiche, le quali sono ulteriormente elaborate in *Indicatori della pratica* che aiutano nella fase di valutazione delle prestazioni.
Ciascun attributo del processo è valutato seconod una scala di quattro valori (N-P-L-F):

- *Not achieved* (0 - 15%);
- *Partially achieved* (>15% - 50%);
- *Largely achieved* (>50% - 85%);
- *Fully achieved* (>85% - 100%).

La valutazione si basa su prove raccolte fronte di ciascun indicatore di processo durante la fase di valutazione.

A.2.2 Valutazione

ISO_G/IEC_G 15504 fornisce una guida per effettuare una verifica dei processi:

- il processo di valutazione (*the assessment process*);
- il modello di valutazione (*the model for the assessment*);
- tutti strumenti per effettuare la valutazione (*any tools used in the assessment*).

A.2.3 Processo di valutazione

Eseguire la valutazione dei processi è oggetto della parte 2 e 3 del ISO_G/IEC_G 15504. Il processo di valutazione può essere generalizzato nei seguenti passi:

- Inizio dell'assessment;
- Selezione del valutatore e del team di valutazione;
- Pianificazione dell'assessment, inclusa la definizione dei processi e dell'organizzazione da valutare;
- Riunione preliminare;
- Raccolta dei dati;
- Validazione dei dati raccolti;
- Valutazione del processo;
- Rapporto sul risultato della valutazione.

A.3 Ciclo di Deming o Ciclo di Shewhart

Il ciclo di Deming (o ciclo di Shewhard) detto anche $PDCA_G$ (o $PDSA_G$) si compone di 4 fasi:

- **P - Plan**: questa fase stabilisce gli obiettivi e processi necessari all'ottenimento dei risultati coerenti ai risultati attesi. Lo stabilire le aspettative di risultati, la completezza e accuratezza delle specifice scelte sono parte del miglioramento mirato. Iniziare su piccola scala, quando possibile, per poter verificare possibili effetti;
- **D - Do**: Implementazione della fase *Plan*, esecuzione del processo, creazione del prodotto. Raccogliere i dati per la creazione di grafici e analisi da destinare alla fase di *Check* e *Act*;
- **C - Check(or S - Study)**: Controllo e studio dei risultati raccolti nella fase *Do*. Confrontare i risultati con quelli attesi, stabiliti nella fase *Plan*;
- **A - Act**: Azione per rendere definitivo e/o migliorare il processo. L'informazione ottenuta nella fase *Check* è utile per poter individuare i possibili punti di un processo dove apportare modifiche.

B Resoconto delle attività di verifica

In questa sezione vengono riportati i risultati delle attività di verifica svolte.

B.1 Revisione dei requisiti

Prima della consegna relativa alla revisione dei requisiti, sono stati verificati i processi che hanno portato alla stesura dei documenti ed essi stessi.
Per i documenti è stata effettuata una verifica di tipo manuale, la quale consiste nella rilettura dei documenti per l'individuazione di errori di forma ed eventuali inconsistenze; e una verifica automatizzata tramite gli strumenti definiti nel documento *Norme di Progetto 1.2.0*, di seguito citati per completezza:

- $Script_G$ $Perl_G$ per il calcolo dell'indice Gulpease
- $Script_G$ $Perl_G$ per la glossarizzazione dei termini
- Aspel

B.2 Dettaglio delle verifiche

B.2.1 Analisi

B.2.1.1 Processi Nella seguente tabella sono riportati i valori per la Schedule Variance(SV) e la Budget Variance(BV) riguardanti i processi del periodo di **Analisi**.

- **Schedule Variance:** I range stabiliti sono:
  - Range di accettabilità = [≥-(*141.25*)];
  - Range di ottimalità = [≥0].
- **Budget Variance:** I range stabiliti sono:
  - Range di accettabilità = [≥-(*282.5*)];
  - Range di ottimalità = [≥0].

Attività	**SV**(Euro)	**BV**(Euro)
Analisi dei requisiti	25	-45
Piano di progetto	35	0
Piano di qualifica	0	-20
Norme di progetto	0	0
Studio di fattibilità	0	10
Glossario	0	0
Totale	60	-55

Tabella 2: Indici SV e BV - Periodo di Analisi

Stando a quanto preventivato dal prospetto economico del *Piano di Progetto 1.4.0*, il valore di SV rientra nel range di accettabilità stabilito (SV≥-141.25 Euro) che in quello di ottimalità; anche il valore di BV rientra nel range di accettabilità stabilito (BV≥-282.5 Euro).

B.2.1.2 Documenti Nella tabella seguente sono riportati i valori dell'indice Gulpease per ogni documento prodotto nel periodo di Analisi.

- **Indice Gulpease:** I range stabiliti sono:
  - Range di accettabilità = [40-100];
  - Range di ottimalità = [50-100].

Documento	**Indice Gulpease**	**Esito**
Analisi dei requisiti	74.3	Superato
Piano di progetto	51.1	Superato
Piano di qualifica	51.3	Superato
Norme di progetto	50.7	Superato
Studio di fattibilità	49.5	Superato
Glossario	51.6	Superato
Verbali esterni	51.1	Superato

Tabella 3: Indici Gulpease per i documenti - Periodo di Analisi

B.2.1.3 Software

- **Dimensione del prodotto software:** ???
- **Complessità ciclomatica:** I range stabiliti sono:
  - Range di accettabilità = [0-15];
  - Range di ottimalità = [0-10].

Per il periodo di **Analisi** non è stata applicata questa metrica.

- **Numero di metodi per file:** I range stabiliti sono:
  - Range di accettabilità = [3-10];
  - Range di ottimalità = [0-7].

  Per il periodo di **Analisi** non è stata applicata questa metrica.
- **Variabili non utilizzate e/o non definite:** I range stabiliti sono:
  - Range di accettabilità = [0-0];
  - Range di ottimalità = [0-0].

  Per il periodo di **Analisi** non è stata applicata questa metrica.
- **Numero di argomenti per funzione:** I range stabiliti sono:
  - Range di accettabilità = [0-6];
  - Range di ottimalità = [0-4].

  Per il periodo di **Analisi** non è stata applicata questa metrica.
- **Linee di codice per linee di commento:** I range stabiliti sono:
  - Range di accettabilità = [>0.25];
  - Range di ottimalità = [>0.30].

  Per il periodo di **Analisi** non è stata applicata questa metrica.
- **Copertura del codice:** I range stabiliti sono:
  - Range di accettabilità = [70-100];
  - Range di ottimalità = [80-100].