SWEeneyThreads

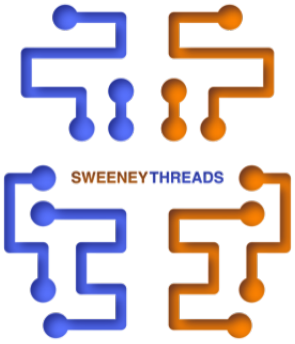
ActorBase

a NoSQL DB based on the Actor model

Piano di qualifica

Redattori:
Biggeri Mattia

Approvazione:
Verifica:
Bonato Paolo



Versione 1.1.3

18 marzo 2016

Indice

Diario delle modifiche

Versione	Data	Autore	Descrizione
1.1.3	2016-03-09	*Progettista* Padovan Tommaso	Aggiunte le seguenti metriche: Numero di errori ortografici, Livelli di annidamento dei requisiti, Complessità ciclomatica, SLOC, Righe per ogni metodo. Corretto la forma tipografica di alcuni range di accettazione che non rispettavano lo standard.
1.1.2	2016-03-09	*Amministratore* Biggeri Mattia	Correzioni post verifica ed aggiunta immagini
1.1.1	2016-03-05	*Amministratore* Biggeri Mattia	Spostamento sezioni qualità di processo, qualità di prodotto e resoconto attività di verifica in appendice, sezioni misure e metriche e strumenti spostate sotto definizione obiettivi
1.1.0	2016-01-17	*Verificatore* Biggeri Mattia	Verificato il documento e comunicato errori
1.0.4	2016-01-17	*Amministratori* Nicoletti Luca + Tommaso Padovan	Miglioramento sezione Metriche e Analisi
1.0.2	2016-01-17	*Amministratore* Tommaso Padovan	Aggiunta qualità di prodotto, Analisi e Metriche
1.0.1	2016-01-17	*Amministratori* Nicoletti Luca + Tommaso Padovan	Visione generale della strategia di verifica e Gestione amministrativa della revisione
1.0.0	2016-01-17	*Amministratore* Nicoletti Luca	Scrittura scheletro logico del documento

Tabella 1: Diario delle modifiche

1 Introduzione

1.1 Scopo del documento

Questo documento descrive le scelte effettuate in merito alle strategie che il gruppo ha deciso di adottare per raggiungere obiettivi qualitativi e misurabili da applicare al proprio prodotto. Per soddisfare questi obiettivi sarà necessario attuare un processo di verifica continuo sulle attività svolte in modo da poter rilevare ed eventualmente correggere anomalie e incongruenze in modo tempestivo per evitare danni e sprechi di risorse.

1.2 Scopo del prodotto

Il progetto consiste nella la realizzazione di un DataBase NoSQL key-value basato sul modello ad Attori$_G$ con l'obiettivo di fornire una tecnologia adatta allo sviluppo di moderne applicazioni che richiedono brevissimi tempi di risposta e che elaborano enormi quantità di dati. Lo sviluppo porterà al rilascio del software sotto licenza MIT.

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 v1.0.3*. 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 Normativi

- **Norme di progetto:**
  Norme di progetto v1.1.1;
- **Capitolato d'appalto:**
  `http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C1p.pdf`.

1.4.2 Informativi

- **Piano di progetto:**
  Piano di progetto v1.0.4;
- **Slide del corso:**
  `http://www.math.unipd.it/~tullio/IS-1/2015/`;
- **SWEBOK - Version 3:**
  `http://www.computer.org/web/swebok/v3`;
- ***TexMaker*:**
  `http://www.xm1math.net/texmaker/`;
- ***Scalastyle*:**
  `https://github.com/scalastyle/scalastyle`;

- ***Scapegoat*:**
  `https://github.com/sksamuel/scapegoat`;
- ***CLOC (Count Lines Of Code)*:**
  `cloc.sourceforge.net`;
- ***ScalaTest*:**
  `http://www.scalatest.org/`.

2 Visione generale della strategia di verifica

2.1 Definizione obiettivi

In questa sezione verranno descritti gli obiettivi di qualità relativi al prodotto che il gruppo ha deciso di raggiungere e gli obiettivi relativi ai processi che saranno svolti per il completamento del progetto. Per la stesura di questo documento e per la definizione delle metriche e degli strumenti atti a garantire la qualità del prodotto abbiamo seguito quanto definito in Appendice 4.5.1 e 4.5.2.

2.2 Misure e metriche

Il processo di verifica, per avere un valore informativo, deve essere quantificabile e le misure rilevate devono essere basate su metriche stabilite a priori. A causa della scarsa esperienza del gruppo, alcune metriche stabilite all'inizio potrebbero risultare approssimative ma seguendo il modello incrementale esposto nel *Piano di progetto* si potrà migliorarne la precisione e l'accuratezza.

Le tipologie di range ammesse sono due:

- **Accettabile:** Superiore al minimo valore richiesto affinché il prodotto sia accettato.
- **Ottimale:** Valori entro cui dovrebbe collocarsi la misurazione. Non sono vincolanti, ma fortemente consigliati. Misurazioni al di fuori di questi valori necessitano una verifica approfondita e nel caso non si trovi una maniera immediata per farli rientrare le cause di tale scostamento dovranno essere discusse nella successiva riunione.

2.2.1 Metriche per i processi

Gli indici scelti per la quantificazione dei processi prendono in considerazione principalmente costi e tempi; hanno lo scopo di mantenere il controllo sui processi e che il progetto segua quanto descritto nel *Piano di progetto*.

Gli indici scelti sono:

- **SV (Schedule Variance):** É un indicatore di efficacia, mostra se si è o meno in linea con la pianificazione temporale rispetto alle attività nella baseline. Una schedule variance positiva indica che il gruppo è in anticipo rispetto al Piano di progetto, che è in ritardo altrimenti.
- **BV (Budget Variance):** Indica se la spesa sostenuta alla data corrente è superiore o inferiore a quella preventivata. Una budget variance positiva indica che si è speso meno di quanto inizialmente previsto, viceversa altrimenti.

2.2.2 Metriche per i documenti

Per i documenti abbiamo scelto di adottare le seguenti metriche:

- **Gulpease:** Lo abbiamo scelto come *indice di leggibilità* in quanto:
  - È l'unico tarato appositamente per la lingua italiana.
  - Utilizza la lunghezza delle parole in lettere anziché in sillabe, quindi è più semplice da automatizzare
  - Considera la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere
  - Permette di misurare la complessità dello stile di un documento

  La formula per il suo calcolo è la seguente:

  $$89 + \frac{300 \cdot (numero\ delle\ frasi) - 10 \cdot (numero\ delle\ lettere)}{numero\ delle\ parole} \quad (1)$$

  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

  Il gruppo si atterrà ai seguenti parametri:
  - Range di accettazione: [35|100]
  - Range di ottimale: [45|100]
- **Numero pagine e parole:** Questa metrica servirà principalmente per misurare l'incremento del documento durante il completamento delle varie fasi.
- **Numero figure e tabelle su numero pagine:** Questa metrica verrà valutata solamente su i documenti esterni per verificarne la difficoltà di lettura. Il gruppo si atterrà ai seguenti parametri:
  - Range di accettazione: [0,25|2]
  - Range di ottimale: [0,5|2]
- **Media parole per section:** Servirà ad evitare la costruzione di sezioni troppo grandi, in modo che tutti gli argomenti siano facilmente trovabili sull'indice.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere
- **Numero di errori ortografici:** Garantisce l'assenza di errori ortografici rilevati dal correttore automatico, a meno di falsi positivi.

  - Range di accettazione: [0|0]
  - Range di ottimale: [0|0]

- **Livelli di annidamento dei requisiti:** Misura i livelli degli alberi dei requisiti. In questo modo si vuole dare una stima di quanto essi siano organizzati gerarchicamente.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere

2.2.3 Metriche per il software

Questa sezione è da intendere come una dichiarazione di intenti e probabilmente verrà rivista quando inizieremo realmente la fase di programmazione.

Al fine di perseguire gli obbiettivi sulla qualità del software, applicheremo le seguenti metriche:

- **Attributi per classe:** Un grande numero di attributi interni ad una classe mostra probabilmente la necessità di suddividere la classe in più classi relazionate tra di se.
  - Valori accettati: [0|18]
  - Valori ottimali: [2|9]
- **Numero livelli annidamento:** Mostra il livello di annidamento dei metodi, un numero alto implica una bassa astrazione del codice ed un' elevata complessità.
  - Valori accettati: [1|8]
  - Valori ottimali: [1|4]
- **Numero parametri per metodo:** Un valore elevato indica che probabilmente il metodo ha un sovraccarico di funzionalità
  - Valori accettati: [0|8]
  - Valori ottimali: [0|5]
- **Linee di codice per linee di commento:** È la percentuale di righe di commento rispetto al totale delle righe di codice. Utile per la manutenibilità.
  - Valori accettati: [0,2|0,7]
  - Valori ottimali: [0,3|0,5]
- **Accoppiamento afferente:** Indica il numero di classi esterne ad un package che dipendono da esso, un grande valore indica una forte dipendenza del software per il package in questione, un valore basso invece indica una bassa utilità del package per il resto del software.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere

- **Accoppiamento efferente:** Il numero di classi di un package che dipendono da package esterni, un valore basso indica che il package ha numerose funzionalità indipendenti dal resto del software.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere
- **Linee di codice per metodo:** Da una misura della leggibilità del codice: valori troppo elevanti indicano che il corpo di un metodo potrebbe essere scarsamente comprensibile.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere
- **Source Line Of Code (SLOC):** Il numero di istruzioni presenti nel codice. Questa metrica fornisce una stima della complessità del programma. È utile anche per dare una stima di quanto il codice incrementerà nel tempo, semplificando così la pianificazione.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere
- **Complessità Ciclomatica:** Una metrica sviluppata da Thomas J. McCabe che consente di stimare la complessità di un programma misurando misurando il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso.
  - Valori accettati: da decidere
  - Valori ottimali: da decidere

2.3 Strumenti e metodi per l'applicazione delle metriche

Le risorse software che si utilizzeranno durante il processo di verifica sono:

- ***TexMaker*:** un ambiente grafico Open-Source per LaTeXcross-platform, permette la compilazione rapida e la visualizzazione del PDF generato, sarà anche usato per il controllo ortografico;
- ***Scalastyle*:** analizzatore statico che rileva potenziali problemi nel codice, da utilizzare sia durante la produzione del codice, sia durante la sua verifica (`https://github.com/scalastyle/scalastyle`);
- ***Scapegoat*:** un altro analizzatore statico che si concentra maggiormente sugli standard di stile e di coding (`https://github.com/sksamuel/scapegoat`);
- ***CLOC (Count Lines Of Code)*:** misura alcune metriche riguardanti il codice sorgente in vari linguaggi, tra cui *Scala*, servirà a calcolare le linee di codice per linee di commento (`cloc.sourceforge.net`);
- ***ScalaTest*:** framework per i test su Scala (`http://www.scalatest.org/`).

2.4 Metodi

2.4.1 Analisi dei processi

I processi compiuti durante una fase verranno analizzati secondo il seguente protocollo:

- **Controllo delle metriche:** Alla conclusione di ogni fase del progetto si calcolano gli indici definiti nella sezione 2.2.1 confrontando le macro-attività preventivate nel *Piano di progetto* con i dati effettivi riscontrati dal sistema di ticketing.
- **Analisi PDCA:** Secondo il ciclo PDCA una fase di progetto ha inizio con la pianificazione di come possono essere migliorati i processi. Durante la fase vengono attuati i cambiamenti prefissati e alla fine viene effettuato un controllo. Se un processo risulta essere migliore viene adottato in modo definitivo, se invece risulta inalterato o addirittura peggiore i cambiamenti vengono scartati.

2.4.2 Analisi dei documenti

Ogni documento redatto è verificato mediante il seguente protocollo:

- **Controllo sintattico:** Il testo deve venire sottoposto a controllo dell'ortografia con il tool fornito dall'ambiente di sviluppo LaTeXutilizzato. Alcuni errori non possono essere comunque rilevati da meccanismi automatici, quindi al fine di ottenere correttezza sintattica e semantica i Verificatori effettueranno un walkthrough al fine di ricercare errori sfuggiti al correttore ortografico.
- **Controllo semantico:** I Verificatori sono tenuti a leggere il documento, controllando che le frasi lette abbiano senso compiuto e che siano pertinenti all'argomento trattato nella sezione. Errori di questo tipo possono essere dovuti a sviste o errori di copia nella stesura.
- **Rispetto delle Norme di progetto:** I Verificatori sono tenuti a verificare che il documento rispetti tutte le norme tipografiche e di struttura del documento riportate nelle *Norme di progetto*. Porzioni di questa verifica sono automatizzabili, i Verificatori dovranno quindi usare tool ove possibile.
- **Inspection secondo checklist:** I Verificatori dovranno scorrere la lista di controllo e verificare che non sia presenti gli errori comuni lì riportati.
- **Verifica Glossario:** I Verificatori dovranno controllare che tutti i termini contenuti nel Glossario siano indicati all'interno dei documenti con la G a pedice.
- **Calcolo delle metriche:** I Verificatori dovranno calcolare le seguenti metriche per ogni documento:
  - **Gulpease:**I verificatori calcoleranno l'indice Gulpease del documento tramite strumenti automatici forniti su `http://sweeneytreadaas.altervista.org/menuPrincipale/documentation_tools/`, in caso

i risultati non siano soddisfacenti presenterà al redattore le sezioni peggiori;

- **Numero figure e tabelle su numero pagine:**Tramite gli elenchi delle figure e delle tabelle sarà calcolato il rapporto tra la somma delle stesse e il numero di pagine, e se risulterà scadente il verificatore lo segnalerà al redattore del documento.
- **Media parole per section:**La metrica sarà calcolata tramite strumenti automatici e in caso di valori non accettabili le sezioni peggiori saranno comunicate al redattore.

- **Miglioramento:** Se nello svolgimento di uno qualunque dei punti precedenti un Verificatore notasse nuove possibilità di automatizzazione dovrà segnalarle e si dovrà cercare o costruire uno strumento per rendere effettiva tale automatizzazione. Inoltre durante le esecuzioni dei walkthrough, i verificatori sono tenuti ad annotare gli errori più frequenti o potenzialmente dannosi rilevati. Tali errori andranno poi ad aggiornare la lista di controllo che verrà utilizzata per le successive inspection.

2.5 Organizzazione

Il processo di verifica inizia fin dal primo rilascio ufficiale nella repository di un documento vX.0.0, il risultato del processo di verifica cambia la versione del documento, incrementandone la versione al secondo livello di profondità, il documento quindi passa a vX.1.0. Ogni modifica apportata al documento ne aumenta la versione all'ultimo livello di profondità. Una volta completate le modifiche e le verifiche continue sui documenti, esso passa nella fase di validazione, a carico del responsabile, che ne fa cambiare la versione al secondo livello di priorità.

Come descritto nelle *Norme di progetto*, il primo valore in una versione verrà cambiato ad ogni consegna.

Grazie al diario delle modifiche è più facile individuare dove concentrare l'attenzione nelle verifiche successive alla prima, in quanto sono segnate le sezioni che sono state aggiunte o che hanno subito dei cambiamenti dall'ultima verifica, evitando cosi di controllare sempre tutto il documento ad ogni rilascio di versione, per cui ottimizzando il tempo necessario al controllo.

Per ognuna delle 5 fasi del progetto descritte nel *Piano di Progetto v 1.0.5*, sono necessarie diverse attività di verifica a causa dei diversi output ottenuti:

- **Analisi:** Si devono seguire le attività di verifica descritte nella sezione 2.9.2 e 2.9.1 sui processi e sui documenti attuati.

Come detto in precedenza conclusa la verifica, ha inizio il processo di approvazione. Durante questo processo è compito del Responsabile di Progetto accertarsi che i prodotti ottenuti siano conformi a quanto pianificato e progettato.

2.6 Pianificazione strategica e temporale

Avendo lo scopo di rispettare le scadenze riportate nel *Piano di progetto*, è necessario che l'attività di verifica, sia del codice che della documentazione,

sia sistematica e ben organizzata; in questo modo l'individuazione, e quindi la correzione degli errori avverrà il prima possibile, limitando la diffusione degli stessi.
Per cercare di ridurre il numero degli errori, e quindi semplificare l'attività di verifica, ogni fase di codifica o documentazione sarà preceduta da una fase di studio preliminare. Evitando le imprecisioni di natura concettuale si ridurranno le correzioni necessarie.

Di seguito vengono riportate le scadenze previste:

- **Revisione dei requisiti:** 2016-01-22;
- **Revisione di progettazione:** 2016-04-18;
- **Revisione di qualifica:** 2016-05-23;
- **Revisione di accettazione:** 2016-06-17.

2.7 Responsabilità

Il *Responsabile di progetto* ha il compito di:

- Accertarsi che le attività di verifica vengano svolte sistematicamente secondo quanto riportato nelle *Norme di progetto*;
- Accertarsi che vengano rispettati ruoli e competenze assegnate nel *Piano di progetto*;
- Verificare che non ci siano conflitti di interesse tra redattori e *Verificatori*;
- Aprire ed assegnare i ticket principali e le task-list;
- Approvare un documento e sancirne la distribuzione.

I *Verificatori* hanno il compito di:

- Effettuare la verifica dei documenti con strumenti e metodi proposti nel *Piano di Qualifica*;
- Attenersi rigidamente a quanto sancito nelle *Norme di progetto*;
- Segnalare tempestivamente un errore, qualora riscontrato;
- Sottoporre i documenti all'approvazione del *Responsabile*, una volta giunti ad uno stadio finale.

2.8 Risorse

Per la realizzazione del progetto sono necessarie risorse sia umane che tecnologiche.

2.8.1 Risorse umane

Vengono descritte nel dettaglio nel *Piano di progetto* e sono:

- *Responsabile di progetto*;
- *Amministratore*;
- *Analista*;
- *Progettista*;
- *Programmatore*;
- *Verificatore*.

2.8.2 Risorse software

Sono necessari tutti i software utili

- alla gestione di documentazione in LaTeX;
- alla creazione di diagrammi UML;
- allo sviluppo di codice *Scala*;
- a semplificare ed automatizzare la verifica;
- a semplificare ed automatizzare la pianificazione e la documentazione della stessa;
- a semplificare ed automatizzare la comunicazione interna tra i membri del gruppo;
- a gestire test ed analisi sul codice.

2.8.3 Risorse hardware

- computer dotati di tutti i software descritti nel *Piano di qualifica* e nelle *Norme di progetto*;
- luoghi dove effettuare le riunioni del gruppo.

Tutti i membri del gruppo hanno a disposizione almeno un computer personale dotato di tutti gli strumenti necessari per il progetto; tutte le macchine in questione sono portatili. Inoltre in caso di rottura o guasto è messo a disposizione un computer di riserva. Sono a disposizione quattro appartamenti a Padova dove effettuare le riunioni. La scelta di quale viene presa di volta in volta a seconda delle disponibilità. Tutti gli appartamenti sono dotati di connessione internet a banda larga.

2.9 Analisi

2.9.1 Tecniche per l'analisi statica

L'analisi statica non richiede l'esecuzione del codice in oggetto, ed è quindi applicabile sia alla documentazione che al codice. Permette di individuare errori ed anomalie al più presto possibile, scongiurandone la diffusione.
Essa può essere svolta in due modi distinti.

2.9.2 Walkthrough

Si svolge effettuando una lettura critica a pettine. Questa tecnica viene utilizzata prevalentemente nelle prime fasi del progetto, in cui non si ha ne una adeguata esperienza, ne uno storico degli errori più comuni che permetta una indagine più mirata. I Verificatori, tramite questa tecnica, saranno in grado di stilare una lista di errori più frequenti, potendo così applicare successivamente la tecnica *Inspection*. Il *Walkthrough* è una tecnica onerosa e richiede l'intervento di più persone. Dopo una fase iniziale in cui i Verificatori leggono il documento ed individuano potenziali errori essi devono essere discussi in una riunione con altri componenti del gruppo per accertare che non siano dei falsi positivi.

2.9.3 Inspection

È una tecnica molto meno onerosa. Consiste nel controllare alcune parti dei documenti che si sono rivelate maggiormente prone ad errori. Per ottenere questo risultato è necessario avere una lista di controllo che indichi quali sono le parti da controllare in maniera mirata. Essa viene stilata durante le fasi di *Walkthrough*. Un altro motivo per cui la *Inspection* è preferibile è il fatto che essa richiede l'intervento dei soli verificatori, che poi possono procedere alla correzione della maggior parte degli errori, oppure ad aprire un ticket riguardante quelli che non sono di immediata risoluzione.

Durante l'applicazione del *Walkthrough* ai documenti sono state riportate le tipologie di errori più frequenti, esse costituiscono quindi la lista di controllo per le verifiche ad *Inspection*, l'attuale lista si trova in appendice sezione 4.6 e una versione sempre aggiornata della stessa è presente sul drive del gruppo.

2.9.4 Tecniche per l'analisi dinamica

Questo tipo di analisi richiede una esecuzione di parte del programma, quindi ovviamente non applica ai documenti ma solo al codice. Il suo obiettivo è rilevare errori o difetti di implementazione mediante l'uso di test che devono essere necessariamente ripetibili: solo un test che produca lo stesso output partendo dallo stesso ambiente e lo stesso input può è capace di riscontrare problemi. L'attore che esegue un test deve definire a priori ed avere il pieno controllo su:

- **Ambiente:** insieme di hardware a software come sistema operativo e altri programmi o processi in esecuzione;
- **Specifiche:** definizione degli input e dei relativi output attesi, che sono ripetibili in quanto si postula di essere in un ambiente deterministico;
- **Procedure:** descrizione delle azioni compiute dall'attore (umano o computer che sia) per arrivare allo stato iniziale, far partire l'esecuzione, inserire gli input specificati e verificare che l'output sia uguale a quello atteso.

Sono definiti 5 tipi di test:

- **Test di unità:** Una unità viene definita come la più piccola quantità di software che conviene testare singolarmente. Il fine di questi test è cercare di individuare eventuali errori presenti nelle singole unità che compongono

l'intero sistema. Essi vengono testati attraverso l'uso di stub, driver e logger. Queste verifiche sono spesso le più onerose, ma anche quelle che portano alla luce il maggior numero di errori, quindi quelle che producono il maggior valore.

- **Test di integrazione:** Consiste nella verifica di componenti del sistema che vengono aggiunti incrementalmente, è necessario dunque analizzare combinazioni di due o più unità di software. Hanno lo scopo di 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. Per la loro realizzazione è necessario usare spesso componenti fittizie non ancora sviluppate, ma che emulano il comportamento atteso.

- **Test di Sistema:** Consiste nella validazione del prodotto software una volta che siano stati aggiunti tutti i componenti e lo si ritiene giunto ad una versione definitiva. Lo scopo principale è verificare che ci sia totale copertura dei requisiti stabiliti nella fase di Analisi di dettaglio. È obiettivo fondamentale della qualità del processo fare in modo che giunti a questo punto l'esito del test sia comunque positivo, in quanto garantito dal tracciamento dei requisiti.

- **Test di regressione:** Consiste nell'eseguire nuovamente i test di unità e integrazione in porzioni di software che hanno subito modifiche in maniera da accertare che questi cambiamenti non pregiudichino il funzionamento dei componenti non toccati da questa modifica.

- **Test di accettazione:** Consiste nel collaudo del prodotto che viene eseguito in presenza del proponente. Un esito positivo di questo test permette il rilascio ufficiale del software.

3 Gestione amministrativa della revisione

3.1 Comunicazione e risoluzione di anomalie

Una anomalia è una violazione da parte di un documento, o unità di codice, di una o più delle seguenti condizioni:

- Conformità alla norme tipografiche o di codifica;
- Appartenenza al range di accettabilità per tutte le metriche descritte nella *Sezione 2*;
- Congruenza del prodotto con funzionalità indicate nell'*analisi dei requisiti*;
- Congruenza del codice con il design del prodotto.

Se un *Verificatore* dovesse trovare una anomalia egli è tenuto ad aprire un sotto-ticket all'interno della task-list a lui assegnata. Nel caso la risoluzione del ticket avesse la necessità di essere strutturato in sotto-attività sarà compito del *Responsabile* aprire una nuova task-list ed assegnarla alle figure coinvolte.

3.2 Procedure di controllo qualità per i processi

La qualità del processo viene garantita da:

- **Pianificazione:** i processi devono essere pianificati nel dettaglio, in maniera da determinare i punti e le tempistiche in cui effettuare controlli;
- **Controllo:** i controlli pianificati devono essere eseguiti in maniera oggettiva e neutrale, quindi con strumenti automatici ovunque possibile;
- **Miglioramento continuo:** l'adozione del principio di *PDCA* aiuterà a migliorare i processi durante l'intera durata del progetto..

3.3 Procedure di controllo qualità per il prodotto

La qualità del prodotto viene garantita da:

- **Comprensione ed analisi del dominio**;
- **Verifica:** determina che l'output di una fase sia consistente, completo e corretto. Deve essere eseguita costantemente per tutta la durata del progetto, ma cercando di essere minimamente invasiva;
- **Validazione:** conferma oggettivamente che il prodotto sia conforme alle aspettative;
- **Quality Assurance:** garantisce il raggiungimento degli obiettivi di qualità, in maniera preventiva. In questo modo si riduce drasticamente il ricorso a tecniche retrospettive, e con esse si riducono le iterazioni.

4 Appendice

4.1 Qualità

Riportiamo gli standard di riferimento per sviluppare le metriche e i metodi atti a garantire la qualità del prodotto.

Figura 1: Influenze e dipendenze delle varie misure di qualità.

4.1.1 Qualità di processo

Per garantire la qualità del prodotto è necessario garantire anche quella dei processi necessari al suo completamento. A questo scopo si è deciso di adottare lo standard ISO/IEC 15504 denominato SPICE.

Questo modello descrive come ogni processo debba essere controllato costantemente in maniera da rilevare possibili errori o debolezze e correggerli prima che essi si diffondano, facendo aumentare esponenzialmente il carico di lavoro. Affinché le singole valutazioni contribuiscano all'effettivo miglioramento dei processi devono essere sempre ripetibili, oggettivi e comparabili. SPICE definisce 6 livelli di maturità del processo:

0 - Incomplete

1 - Performed

2 - Managed

3 - Established

4 - Predictable

5 - Optimizing

Al fine di applicare correttamente questo modello è evidentemente indispensabile adottare il principio PDCA il quale definisce una metodologia di controllo dei processi durante il loro ciclo di vita che consente di migliorarne in modo continuativo la qualità.
Esso di compone di 4 fasi:

- **Plan:** definire dettagliatamente cosa deve essere realizzato rispetto agli obiettivi di miglioramento, e come questi controlli saranno effettuati;
- **Do:** fase di esecuzione delle attività pianificate;
- **Check:** vengono confrontati i dati in uscita dalla fase *Do* con quelli pianificati nella fase *Plan*, per intervenire in tempo e migliorare i risultati;
- **Act:** fase in cui si mette in pratica il miglioramento continuo dei processi utilizzando i risultati della verifica per modificare gli aspetti critici dei processi in esame.

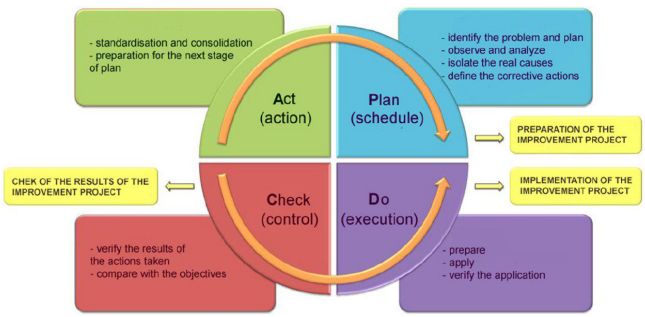


Figura 2: Fasi del principio PDCA.

4.1.2 Qualità di prodotto

Al fine di aumentare il valore del prodotto e di migliorarne il funzionamento è necessario fissare degli obbiettivi qualitativi e garantire che saranno raggiunti. Questi obbiettivi sono descritti nell'ISO/IEC 9126 dove sono anche descritte le metriche per misurare gli stessi.
I criteri valutativi sono suddivisi in 3 aree:

- **Qualità esterna:** Le metriche esterne, specificate nella norma ISO/IEC 9126-2, valutano i comportamenti del prodotto sulla base di prove, dall'operatività e dall'osservazione durante la sua esecuzione, in funzione degli obiettivi stabiliti.
- **Qualità interna:** È specificata nella norma ISO/IEC 9126-3 e si applica al software non eseguibile durante la progettazione e la codifica dello stesso, le misure effettuate consentono di prevedere il livello di qualità esterna ed interna, in quanto gli attributi interni influenzano quelli esterni e di uso
- **Qualità d'uso:** Rappresenta la qualità dal punto di vista dell'utente finale, viene raggiunto quando sono raggiunte la qualità esterna e quella interna, le metriche di valutazione sono fornite nella norma ISO/IEC 9126-4.

lo standard ISO/IEC 9126 prevede di suddividere la qualità esterna ed interna in 6 caratteristiche principali tra le quali la funzionalità è l'unico "requisito funzionale" mentre le altre 5 sono "requisiti di qualità", ciascuna caratteristica si suddivide in altre sotto caratteristiche che possono essere misurate qualitativamente:

- **Funzionalità:** capacità del prodotto software di fornire funzioni che rispondano a esigenze stabilite
  - **Idoneità:** capacità del prodotto software di fornire un insieme di funzioni per attività specifiche già conosciute dall'utente;
  - **Accuratezza:** capacità del prodotto software di fornire risultati esatti o concordi al grado di precisione necessario;
  - **Interoperabilità:** capacità del prodotto software di interagire con uno o più sistemi precedentemente specificati;
  - **Sicurezza:** capacità del prodotto software di proteggere dati e informazioni;
  - **Conformità funzionale:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni e prescrizioni in materia di funzionalità.
- **Affidabilità:** capacità del prodotto software di mantenere uno specifico livello di prestazioni quando usato
  - **Maturità:** capacità del prodotto software di non fallire a causa di errori nel software;

  - **Tolleranza agli errori:** capacità del prodotto software di mantenere un adeguato livello di prestazioni e funzioni in caso di errori software o di violazioni;
  - **Capacità di recupero:** capacità del prodotto software di ristabilire un adeguato livello di performance e di recuperare i dati in caso di errori;
  - **Conformità di affidabilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di affidabilità.

- **Usabilità:** capacità del software di essere capito, imparato, usato e apprezzato dall'utente quando usato
  - **Comprensibilità:** capacità del prodotto software di far comprendere all'utente se il prodotto è adatto ad uno specifico scopo;
  - **Apprendibilità:** capacità del prodotto software di ridurre all'utente il tempo necessario per apprendere le sue funzioni;
  - **Operabilità:** capacità del prodotto software di essere utilizzato dall'utente in modo controllato
  - **Attrattiva:** capacità del prodotto software di creare interesse nell'utente;
  - **Conformità di usabilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di usabilità.

- **Efficienza:** capacità del software di fornire prestazioni appropriate in relazione alla quantità di risorse in utilizzo
  - **Comportamento temporale:** capacità del software di fornire tempi di risposta e di elaborazione adeguati sotto condizioni determinate;
  - **Utilizzo di risorse:** capacità del prodotto software di utilizzare quantità e tipo di risorse adeguate durante la sua esecuzione;
  - **Conformità di efficienza:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di efficienza.

- **Manutenibilità:** capacità del prodotto software di essere modificato e ampliato.
  - **Analizzabilità:** rappresenta la facilità con la quale è possibile analizzare il software alla ricerca di carenze e difetti;
  - **Modificabilità:** capacità del prodotto software di permettere l'implementazione di una specifica modifica o di un aggiornamento;
  - **Stabilità:** capacità del prodotto software di evitare effetti indesiderati causati da uno o più aggiornamenti o modifiche;
  - **Testabilità:** capacità del prodotto software di consentire una facile validazione di una versione modificata del software;
  - **Conformità di manutenibilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di manutenibilità.

- **Portabilità:** capacità del prodotto software di poter essere trasferito da un ambiente di lavoro ad un altro sia dal punto di vista hardware che per quanto riguarda il sistema operativo
  - **Adattabilità:** capacità del prodotto software di essere adattato a diversi ambienti di lavoro senza la necessità di effettuare modifiche aggiuntive;
  - **Installabilità:** capacità del prodotto software di poter essere installato in specifici ambienti;
  - **Coesistenza:** capacità del prodotto software di coesistere in ambienti comuni con altri software indipendenti condividendo risorse comuni;
  - **Sostituibilità:** capacità del prodotto software di poter sostituire un software analogo o simile nello stesso ambiente;
  - **Conformità di portabilità:** capacità del prodotto software di aderire a standard, convenzioni o regolamentazioni in materia di portabilità.

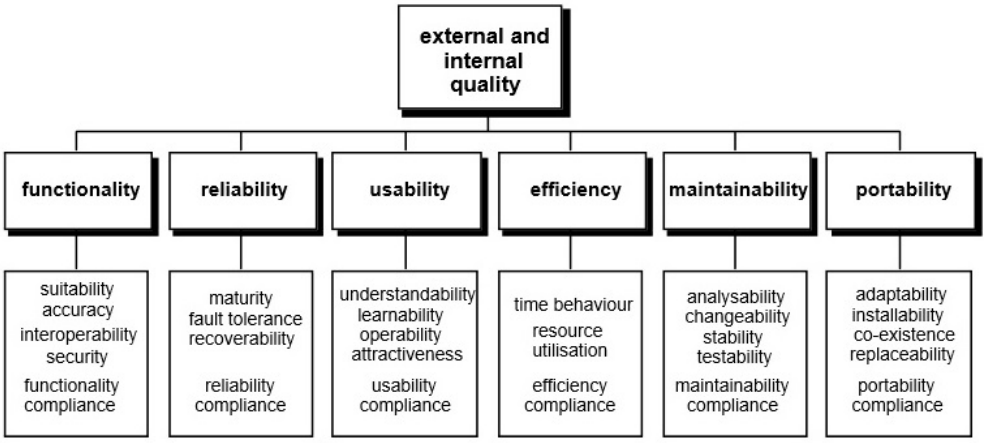


Figura 3: Caratteristiche della qualità esterna e interna.

Per la qualità d'uso invece vengono invece definite le seguenti caratteristiche:

- **Efficacia:** la capacità del prodotto di consentire agli utenti di raggiungere gli obiettivi specificati con precisione sufficiente e completezza.
- **Produttività:** la capacità di consentire agli utenti di utilizzare una quantità di risorse appropriate in relazione all'efficacia ottenuta in contesto d'uso definito.
- **Soddisfazione:** è la capacità del prodotto di soddisfare gli utenti.
- **Sicurezza:** rappresenta la capacità del prodotto di avere accettabili livelli di rischio per quanto riguarda i danni alle persone, al software, ad apparecchiature o all'ambiente operativo d'uso.

Figura 4: Caratteristiche della qualità d'uso.

4.2 Resoconto delle attività di verifica

4.2.1 Riassunto delle attività di verifica

Durante la stesura della documentazione sono stati verificati ad ogni modifica sostanziale la documentazione stessa, i casi d'uso e i requisiti; è stato in oltre verificato se i processi si sono svolti in maniera corretta.

4.2.2 Documentazione

I documenti sono stati verificati tramite walkthrough da due persone differenti seguendo il seguente protocollo:

- Verifica della sintassi e della correzione ortografica
- Verifica della chiarezza espositiva
- Verifica del rispetto delle *Norme Di Progetto v1.1.1* capitolo 3.1
- Verifica dell'uniformità dei termini rispetto allo stesso documento
- Verifica dell'uniformità dei termini rispetto agli altri documenti
- Verifica che i termini che lo necessitano siano stati inseriti nel glossario
- Produzione dei un file .txt contenente tutti gli errori al fine di mostrare le correzioni necessarie e facilitare la stesura della checklist degli errori più frequenti

In seguito i documenti sono stati nuovamente verificati tramite inspection utilizzando la checklist precedentemente stilata.

4.2.3 Casi d'uso

I casi d'uso sono stati verificati sempre tramite walkthrough ponendo l'attenzione sui seguenti punti:

- Uniformità dei termini usati tra i vari UC
- Uniformità tra l'immagine dei diagrammi e la spiegazione della suddetta

- Correttezza del codice utilizzato rispetto a quanto definito nelle *Norme Di Progetto v1.1.1* sezione 2.1.3
- Uniformità dei casi d'uso rispetto al capitolato
- Rispetto della struttura definita nelle *Norme Di Progetto v1.1.1* sezione 2.3.1

I Casi d'uso sono stati successivamente ricontrollati all'interno del documento di *Analisi Dei Requisiti* come descritto nella sezione 4.2.2

4.2.4 Requisiti

I requisiti sono stati controllati tramite walkthrough seguendo il seguente protocollo:

- Verifica che la fonte del requisito sia corretta.
- Nel caso in cui la fonte sia un UC, verifica che sia l'UC corretto e valutazione sulla possibilità di dedurre altri requisiti dallo stesso.
- Verifica uniformità dei termini tra i requisiti.
- Rispetto del codice e della struttura definiti nelle *Norme di Progetto v1.1.1* sezione 2.1.2.
- I Casi d'uso sono stati successivamente ricontrollati all'interno del documento di *Analisi Dei Requisiti* come descritto in sezione 4.1.

4.2.5 Processi

I processi sono stati verificati dal responsabile grazie all'utilizzo di teamwork controllando le seguenti condizioni:

- Che la verifica non sia stata effettuata dallo stesso che ha prodotto il materiale da verificare.
- Che i documenti siano stati prodotti nell'ordine e nei tempi corretti.
- Che le ore di verifica siano almeno 30% delle ore totali.
- Che le riunioni e i brainstorming si siano svolti come definito nelle *Norme di Progetto v1.1.1* sezione 4.2.3.

4.3 Tracciamento componenti - requisiti

4.4 Dettaglio delle verifiche tramite analisi

4.4.1 Processi

Vengono qui di seguito riportate le macro-attività della fase di analisi con associati i relativi valori di SV e BV.

Macro-attività	SV	BV
Norme di progetto	ore 1	€ 30
Studio di fattibilità	ore 2	€ 50
Analisi dei requisiti	ore -2	€ -20
Piano di progetto	ore -1	€ -22
Piano di qualifica	ore 0	€ 0

Tabella 2: Tabella delle attività con SV e BV

Da questi dati si può dedurre che:

- I tempi impiegati hanno subito delle leggere variazioni e grazie agli slack inseriti nel *Piano di progetto* non ci sono stati ritardi significativi.
- Il costo effettivo è molto vicino a quanto preventivato nel *Piano di progetto* anche se alcuni picchi, specie nello studio di fattibilità, significano che è possibile migliorare la pianificazione dei processi.

Durante la prima fase di analisi le problematiche riscontrate sui processi sono dovute per la maggior parte all'inesperienza del gruppo. A questa fase seguirà una pianificazione su come migliorare i processi che sono risultati più critici.

4.4.2 Documenti

4.5 Dettaglio dell'esito delle revisioni

Il progetto prevede quattro revisioni formali a cui sottoporsi, per mezzo delle quali il Committente potrà segnalare le eventuali problematiche riscontrate. Ogni revisione prevede una valutazione globale del progetto ed una valutazione dettagliata per ciascun documento prodotto. Grazie alle segnalazioni ricevute sarà possibile individuare le criticità e gli errori che verranno corretti per poter avanzare alla fase successiva con una baseline verificata.

4.6 Lista errori frequenti

- **Norme stilistiche:**
  - Nome del documento: non viene utilizzata la macro predisposta;
  - Versione del documento in prima pagina errata;
  - Immagini mancanti;
  - Spazi lasciati vuoti per aggiunte successive e non rimossi;
  - Mancanza di uniformità delle espressioni all'interno dello stesso documento;
  - Mancanze nella sezione dei riferimenti.
- **Italiano:**
  - Doppie;
  - Accenti.
- **LaTeX:**

  - mancanza dell'indice delle immagini e delle tabelle.

- **UML:**
  - incongruenze tra l'immagine contenente i diagrammi e la descrizione testuale della stessa;
  - errori nel testo delle immagini dovute a copia-incolla.

Elenco delle figure

Elenco delle tabelle