

SWEENEYTHREADS

ACTORBASE

A NoSQL DB BASED ON THE ACTOR MODEL

Piano di qualifica

Redattori:

Bonato PAOLO
Bortolazzo MATTEO
Maino ELIA
Nicoletti LUCA
Padovan TOMMASO

Approvazione:

Padovan TOMMASO

Verifica:

Biggeri MATTIA
Tommasin DAVIDE



Versione 1.0.2

21 gennaio 2016

Indice

Diario delle modifiche

Versione	Data	Autore	Descrizione
1.0.2	2016-01-17	<i>Amministratore</i> Tommaso Padovan	Aggiunta qualità di prodotto, Analisi e Metriche
1.0.1	2016-01-17	<i>Amministratori</i> Nicoletti Luca + Tommaso Padovan	Visione generale della strategia di verifica e Gestione amministrativa della revisione
1.0.0	2016-01-17	<i>Amministratore</i> Nicoletti Luca	Scrittura scheletro logico del docu- mento

Tabella 1: Diario delle modifiche

Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è di descrivere 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

Lo scopo del progetto è 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 v1.0.0*. La prima occorrenza di ogni termine contenuto nel *Glossario* e presente in questo documento verrà marcato con una "G" maiuscola in pedice.

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 Formativi

- **Piano di progetto:**
Piano di progetto v1.0.4;

- **Slide del corso:**
[http://www.math.unipd.it/~tullio/IS-1/2015/;](http://www.math.unipd.it/~tullio/IS-1/2015/)
- **SWEBOK - Version 3:**
<http://www.computer.org/web/swebok/v3>
- ...

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.

2.1.1 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 livelli di maturità del processo:

0 - Incompetete

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 si 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.

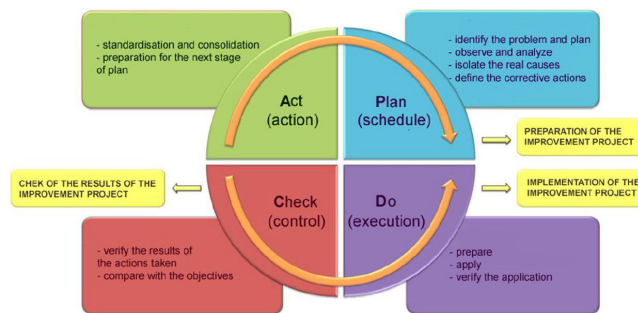


Figura 2.1: Fasi del principio PDCA.

2.1.2 Qualità di prodotto

Al fine di aumentare il valore del prodotto e di migliorarne il funzionamento è necessario fissare degli obiettivi qualitativi e garantire che saranno raggiunti. Questi obiettivi 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 prefissate all'utente;
 - **Accuratezza:** capacità del prodotto software di fornire risultati esatti o concordati con in 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 al 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.

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.

2.2 Organizzazione

Ognuna delle fasi descritte nel Piano di progetto necessita di differenti attività di verifica, a causa della diversità degli output attesi.

- **Analisi:** Si devono seguire le attività di verifica descritte nella sezione 2.9 di questo documento.

2.3 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.

2.4 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*;⁷
- 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.

2.5 Risorse

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

2.5.1 Risorse umane

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

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

2.5.2 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.

2.5.3 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.

2.6 Strumenti

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

- ***TexMaker***: un ambiente grafico Open-Source per L^AT_EX cross-platform, permette la compilazione rapida e la visualizzazione del PDF generato;
- ***Scalastyle***: analizzatore statico che rileva potenziali problemi nel codice (<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* (cloc.sourceforge.net);
- ***ScalaTest***: framework per i test su Scala (<http://www.scalatest.org/>).

2.7 Analisi

2.7.1 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.

Walkthrough

Si svolge effettuando una lettura critica a pettine. Questa tecnica viene utilizzata prevalentemente nelle prime fasi del progetto, in cui non si ha né una adeguata esperienza, né 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.

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*:

- **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.
- **L^AT_EX:**
 - 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.

2.7.2 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 essere 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.

2.8 Misure e metriche

Il processo di verifica, per avere un valore informativo, deve essere quantificabile; le misure rilevate, inoltre, devono essere basate su metriche stabilite a priori. Complice la scarsa esperienza del gruppo alcune metriche stabilite all'inizio potrebbe risultare approssimative, il modello incrementale esposto nel Piano di progetto permetterà però di migliorarne la precisione e l'accuratezza in maniera incrementale.

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.

2.8.1 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):** È 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.

2.8.2 Metriche per i documenti

Come metrica di riferimento per i documenti prodotti si è scelto di utilizzare un *indice di leggibilità*. Tra i molti disponibili si è scelto di utilizzare l'indice Gulpease, 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:

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

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]

2.8.3 Metriche per il software

2.9 Metodi

2.9.1 Analisi dei processi

2.9.2 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 \LaTeX utilizzato. 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.
- **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 siano 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 dell'indice di leggibilità:** I Verificatori dovranno calcolare tramite strumenti automatici l'indice Gulease di leggibilità del testo, nel caso risultasse troppo basso sarà necessario procedere ad un walkthrough, finalizzato alla individuazione di frasi troppo lunghe e complesse.
- **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.

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:** è garantita dall'applicazione del principio *PDCA*.

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.

Resoconto delle attività di verifica

4.1 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.

4.1.1 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.

4.1.2 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 capitolo 2.3.1

- Uniformità dei casi d'uso rispetto al capitolato
- Rispetto della struttura definita nelle norme di progetto v1.1.1 capitolo 2.3.1

I Casi d'uso sono stati successivamente ricontrollati all'interno del documento di Analisi Dei Requisiti come descritto in 4.1.

4.1.3 Requisiti

4.2 Tracciamento componenti - requisiti

4.3 Dettaglio delle verifiche tramite analisi

4.3.1 Processi

4.3.2 Documenti

4.4 Dettaglio dell'esito delle revisioni