404NotFound

Premi: better than Prezi.

Piano di Qualifica

Informazioni sul documento

Versione	1.0
Redazione	Camborata Marco Rettore Andrea
Verifica	De Lazzari Enrico
Responsabile	Vegro Federico
Uso	Esterno
Ultima modifica	22 Gennaio 2015
Lista di distribuzione	404NotFound

Descrizione

Documento riguardante le strategie di verifica adottati dal gruppo 404NotFound e gli obiettivi qualitativi del il progetto Premi.

Registro delle modifiche

Versione	Autore	Data	Descrizione
1.0	Vegro Federico	2015-01-22	Approvazione documento
0.7	De Lazzari Enrico	2015-01-22	Verifica finale documento
0.6	Camborata Marco	2015-01-21	Correzione errori ortografici
0.5	De Lazzari Enrico	2015-01-19	Verifica prima stesura documento
0.4	Rettore Andrea	2015-01-15	Stesura gestione amministrativa della revisione e resoconto attività di verifica
0.3	Camborata Marco	2015-01-07	Stesura Versione generale della strategia di verifica e obiettivi di qualità
0.2	Camborata Marco	2015-01-05	Stesura sezione risorse
0.1	Camborata Marco	2015-01-05	Stesura introduzione e scheletro documento

Tabella 1: Storico versioni del documento.

Indice

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di illustrare le strategie adottate per implementare i processi di verifica e validazione del lavoro svolto da 404NotFound per assicurare la qualità del progetto Premi e dei processi coinvolti nel suo sviluppo. Per raggiungere gli obiettivi di qualità è necessario un processo di verifica continua sulle attività svolte, per questo motivo il presente documento potrà essere aggiornato in seguito a scelte progettuali del gruppo e/o variazione dei requisiti da parte del Proponente.

1.2 Scopo del Prodotto

Lo scopo del progetto è la realizzazione di un software di presentazione di slide non basato sul modello di PowerPoint$_G$, sviluppato in tecnologia HTML5$_G$ e che funzioni sia su desktop che su dispositivo mobile. Il software dovrà permettere la creazione da parte dell'autore e la successiva presentazione del lavoro, fornendo effetti grafici di supporto allo storytelling e alla creazione di mappe mentali.

1.3 Glossario

Al fine di evitare ogni ambiguità relativa al linguaggio e ai termini utilizzati nei documenti formali tutti i termini e gli acronimi presenti nel seguente documento che necessitano di definizione saranno seguiti da una "G" in pedice e saranno riportati in un documento esterno denominato Glossario_v1.0.pdf. Tale documento accompagna e completa il presente e consiste in un listato ordinato di termini e acronimi con le rispettive definizioni e spiegazioni.

1.4 Riferimenti

1.4.1 Normativi

- **Norme di Progetto:** *NormeDiProgetto_v1.0.pdf*;
- **Capitolato d'appalto C4:** Premi: Software di presentazione "better than Prezi" - http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C4.pdf.

1.4.2 Informativi

- **Piano di Progetto**: *PianoDiProgetto_v.1.0.pdf*;
- **Slide dell'insegnamento Ingegneria del Software modulo A**: http://www.math.unipd.it/~tullio/IS-1/2014/;
- **Ingegneria del software - Ian Sommerville - 8a Edizione (2007)**:
  - Capitolo 27 - Gestione della qualità;
  - Capitolo 28 - Miglioramento dei processi.
- **Complessità ciclomatica**: http://it.wikipedia.org/wiki/Complessità_ciclomatica;

- **ISO/IEC$_G$ 9126:2001** (inglobato da ISO/IEC$_G$ 25010:2011): http://www2.cnipa.gov.it/site/_contentfiles/01379900/1379951_ISO 209126.pdf:
  - Systems and software engineering;
  - Systems and software Quality Requirements and Evaluation (SQuaRE);
  - System and software quality models.
- **ISO/IEC$_G$ 15504:1998**: Information Tecnology - Process Assessment, conosciuto come SPICE (Software Process Improvement and Capability Determination): http://www2.cnipa.gov.it/site/_contentfiles/00310300/310320_15504.pdf.

2 Visione Generale della Strategia di Verifica

2.1 Organizzazione

Per garantire la qualità del prodotto in tutte le sue fasi di realizzazione, accertandone la conformità rispetto a quanto emerso durante la fase di Analisi dei Requisiti (vedi allegato *AnalisiDeiRequisiti_v1:0:pdf*), si intende svolgere una costante attività di verifica trasversale a tutte le fasi di sviluppo del progetto.

Per poter effettuare un corretto processo di verifica si è scelto di effettuare le dovute operazioni di controllo ogni volta che il prodotto in esame avrà maturato sostanziali modifiche rispetto alla sua precedente versione. Per quanto riguarda la documentazione questa maturazione si rispecchia nel variare dell'indice di versione dei documenti stessi (vedi documento interno *NormeDiProgetto_v1.0.pdf*, sezione 6.6) e una fase di verifica finale è necessaria affinchè un qualsiasi documento possa passare alla fase di approvazione da parte del *Responsabile del Progetto*. E' auspicabile che siano svolte verifiche sui documenti non solo prima dell'approvazione ma anche in fasi intermedie nelle quali il documento può non essere ancora stato completato. Ogni svolgimento di una fase di verifica globale sarà riportata nell'apposito registro delle modifiche. Per assicurare il massimo livello di controllo, tuttavia, un primo controllo sommario sui nuovi contenuti viene svolto dal *Verificatore* ad ogni modifica del documento (per approfondimento vedi documento interno *NormeDiProgetto_v.1.0.pdf* sezione 5.4)

Si è scelto e adottato il metodo "Broken Window Theory" secondo il quale, non appena un errore viene rilevato, questo andrà segnalato e corretto il prima possibile onde evitarne la propagazione.

Il ciclo di vita scelto per lo sviluppo del progetto è un ciclo di vita incrementale (vedi documento allegato *PianoDiProgetto_v1.0.pdf*) e di conseguenza le operazioni di verifica verranno realizzate in modo tale da intervenire in maniera coerente nelle varie fasi del progetto come illustrato di seguito:

2.1.1 Analisi dei requisiti

Tutta la documentazione relativa alla RR, una volta completata, entrerà nella dedicata fase di revisione. Di seguito i parametri di controllo:

- La presenza di eventuali errori lessico/grammaticali e la generale correttezza dei contenuti esposti. Nel dettaglio, il controllo ortografico verrà effettuato con gli strumenti messi a disposizione da TexMaker$_G$, mentre il controllo lessicale, grammaticale e sintattico da un'accurata rilettura del testo;
- Il controllo dei contenuti con l'obiettivo di verificare la copertura delle richieste del proponente e questo tramite un'accurata rilettura e confronto con il capitolato d'appalto;
- Corrispondenza tra ogni requisito e caso d'uso corrispondente;
- Verifica dei contenuti grafici e tabellari e conformità dei documenti alle *Norme di Progetto* stabilite.

Se durante la verifica saranno state rilevate irregolarità queste verranno segnalate tramite un apposito ticket dal *Verificatore* e corrette dal redattore.

2.1.2 Progettazione

Il processo di verifica in fase di Progettazione consisterà nel verificare che tutti i requisiti descritti durante la fase di Analisi dei Requisiti siano tracciabili nei componenti individuati e viceversa che ogni componente soddisfi o sia associato ad almeno un requisito. Qualora dalla verifica sorgano incongruenze o mancanze, queste verranno segnalate tramite ticket e successivamente risolte.

2.1.3 Realizzazione

La verifica in questa fase verrà effettuata da parte dei programmatori stessi utilizzando appositi e specifici strumenti di verifica automatizzata del codice. La presenza di errori verrà segnalata da un apposito ticket che verrà preso in carico dai programmatori e chiuso una volta risolto il problema.

2.1.4 Validazione

Il team 404NotFound si impegna a garantire il corretto funzionamento del prodotto Premi e a fornire al collaudo una versione funzionante e possibilmente completa del prodotto. Nel caso in cui vengano riscontrati malfunzionamenti o discrepanze tra le caratteristiche del prodotto e le richieste del cliente sarà cura del fornitore eliminare tali difetti, interamente a proprio carico.

2.2 Pianificazione Strategica e Temporale

Avendo l'obiettivo di rispettare le scadenze fissate nel *PianodiProgettov1.0.pdf*, è necessario che l'attività di verifica della documentazione e del codice sia sistematica e ben organizzata. Ogni fase di redazione dei documenti e di codifica deve essere preceduta da una fase di studio preliminare per eliminare all'origine possibili imprecisioni di natura concettuale e/o tecnica.

Il processo di verifica viene strutturato in tre fasi:

1. **Pre-Verifica:** Si tratta della pianificazione e la preparazione delle attività di verifica. Consiste nella scelta delle persone che si occuperanno di questa attività e nella distribuzione dei documenti o componenti software da controllare;

2. **Verifica effettiva:** I *Verifcatori* lavorano indipendentemente per trovare errori, omissioni e scostamenti rispetto agli standard, durante questa fase, un autore del documento o componente software attende il responso del *Verificatore*. Deve stillato un elenco delle azioni correttive da intraprendere;

3. **Post-Verifica:** Dopo che le correzioni sono state apportate al componente in esame il *Verificatore* usando come checklist l'elenco delle correzioni da lui redatto nella fase precedente, potrà constatare l'avvenuta correzione.

Durante le attività di verifica è inevitabile che gli errori commessi dagli individui vengano esposti a tutto il gruppo. E' quindi molto importante che si incoraggi nel team una mentalità per la quale la segnalazione degli errori non diventi motivo per screditare il lavoro di un singolo, ma occasione di crescita per la persona e per l'intero gruppo di lavoro.

2.3 Responsabilità

La responsabilità dell'attività di verifica viene affidata ai seguenti ruoli:

- **Responsabile di Progetto:** Macroscopicamente ha il compito di controllare che l'evoluzione del progetto rispetti le tempistiche prefissate, è garante della qualità dei processi interni e della conformità dei prodotti a quanto pianificato e progettato ponendosi come garante nei confronti del *Committente*. In particolare in questo contesto ha il compito di assicurarsi che le attività di verifica vengano svolte sistematicamente e non vi siano conflitti di interesse tra redattori e verificatori. Egli è l'unico a poter decidere l'approvazione di un documento e a sancirne la distribuzione;
- ***Verificatore*:** Ha il compito di coordinare e definire le attività volte alla verifica del materiale prodotto, sia esso software, documenti o materiale d'altro genere. La responsabilità del verificatore è quella di respingere o validare ogni nuovo documento o modifica di esso e di segnalare formalmente gli errori riscontrati.

2.4 Strumenti, Tecniche e Metodi

2.4.1 Strumenti

Per lo svolgimento del processo di verifica faremo uso dei seguenti strumenti:

- **Correttore automatico di TeXMaker$_G$**: come segnalato nelle *NormeDiProgetto_v1.0.pdf* per la scrittura di documenti si è scelto di utilizzare l'ambiente grafico TeXMaker$_G$. Tale strumento integra i dizionari di OpenOffice.org e segnala i potenziali errori ortografici presenti nel testo;
- **404TrackerDB**: Strumento software realizzato dal gruppo 404NotFound che contiene ed associa:
  - Requisiti individuati durante l'analisi;
  - Fonti di requisiti individuate, inclusi anche i casi d'uso.

  Permette inoltre di esportare automaticamente:
  - Codice LaTeX per la descrizione dei casi d'uso;
  - Tabella in LaTeX per il tracciamento fonti-requisiti.
- Strumenti W3C$_G$ (www.w3.org) per la validazione:
  - **validatore HTML5$_G$** (http://validator.w3.org);
  - **validatore CSS$_G$** (http://jigsaw.w3.org/css-validator/).

- Strumenti per debugging$_G$ HTML$_G$, CSS$_G$ e JavaScript$_G$ messi a disposizione dai vari browser$_G$:
  - **Chrome Developer Tools** (https://developers.google.com/chrome-developer-tools);
  - **Firebug** (http://getfirebug.com/).

  ;
- **JSLint** Ambiente di test (http://www.junit.org): tool per la validazione di codice JavaScript$_G$;
- **JUnit** (http://www.junit.org): semplice framework per eseguire test ripetibili;
- **BrowserStack** (http://www.browserstack.com/): per eseguire il test comparato sui vari browser$_G$;
- **WebStorm** (https://www.jetbrains.com/webstorm/): IDE JavaScript$_G$ scelto come ambiente di sviluppo.

2.4.2 Tecniche di Analisi

Anamilisi Statica:

Consiste nell'analisi della documentazione e dei prodotti software senza effettuare l'esecuzione. Viene svolta mediante due tecniche complementari:

- **Walkthrough**: È una tecnica che viene utilizzata soprattutto nelle prime fasi del progetto, quando ancora non è stata maturata una adeguata esperienza da parte dei membri del gruppo, che permetta di attuare una verifica più mirata e precisa. Consiste nella rilettura completa e metodica da parte dell'autore stesso o da parte del *Verificatore* allo scopo di trovare errori. Con l'utilizzo di questa tecnica, il Verificatore sarà in grado di stilare una lista di controllo con gli errori più frequenti in modo da favorire il miglioramento di tale attività nelle fasi future. Questa è un'attività onerosa e collaborativa che richiede l'intervento di più persone per essere efficace ed efficiente. Segue una fase di discussione con la finalità di esaminare i difetti riscontrati e di proporre le dovute correzioni. L'ultima fase consiste nel correggere gli errori rilevati;
- **Inspection**: Questa tecnica consiste nell'analisi di alcune parti del documento o del codice alla ricerca di errori solo in parti ritenute critiche in base all'esperienza derivata dalle revisioni precedenti. La lista di controllo o checklist, che deve essere seguita per svolgere efficacemente questo processo, deve essere redatta anticipatamente ed è frutto del lavoro svolto dai verificatori con la tecnica di walkthrough. L'Inspection è da preferire al Walkthrough, poichè non necessità della lettura integrale dei documenti in oggetto, ma richiede un sufficiente livello di dettaglio nella lista di controllo.

Metodi di Analisi Statica:

- **Analisi del flusso di controllo**: si controlla che il codice sia correttamente strutturato e che segua il flusso aspettato, che non vi siano parti del programma che possano non terminare e che non esistano porzioni di codice non raggiungibile;
- **Analisi del flusso dei dati**: si accerta che il software non acceda mai a variabili non inizializzate o non modifichi più volte di seguito una variabile senza leggerne il valore tra una modifica e l'altra;
- **Analisi del flusso di informazione**: si verifica che le uniche dipendenze tra gli input e gli output di ogni unità di codice o di più unità siano quelle previste in fase di progettazione.

Analisi Dinamica:

Consiste nella verifica dei componenti del software o del sistema in generale e richiede l'esecuzione del programma per eseguire il test. Perchè tale attività sia utile e generi risultati attendibili è necessario che i test effettuati siano ripetibili: cioè dati uno stesso input e uno stesso ambiente di esecuzione deve fornire gli stessi risultati quando vengono effettuate più prove. Questi risultati saranno utili solo se porteranno alla luce errori permettendo di correggerli, nel caso non vengano riscontrate anomalie, ciò non costituisce una prova dell'assenza di errori.

Metodi di Analisi Dinamica:

- **Test di unità**: Viene verificata ogni unità software che deve soddisfare i requisiti per essa richiesti ed è necessario testare tutte le possibili esecuzioni del codice che la compongono. Per unità si intende la più piccola quantità di software che è utile verificare singolarmente e che viene prodotta da un singolo *Programmatore*;
- **Test di integrazione**: I moduli che hanno superato il test di unità possono venire integrati tra di loro. Il test di integrazione ha lo scopo di individuare errori residui nella realizzazione dei moduli e problemi nell'integrazione con altre componenti fornite da terze parti che non si conoscono a fondo;
- **Test di sistema e collaudo**: Serve a verificare il completo soddisfacimento dei requisiti software stabiliti in fase di Analisi. Consiste nella validazione del prodotto software nel momento in cui vengono aggiunti tutti i componenti. Il test potrà riguardare, in una fase iniziale, solamente alcune delle componenti del prodotto finale, per poi interessare il sistema nella sua interezza;
- **Test di regressione**: Consiste nell'eseguire nuovamente i test di unità e di integrazione su componenti software alle quali sono stati apportati cambiamenti. Serve a controllare che le modifiche apportate non provochino malfunzionamenti alla componente stessa o ad altre che dipendono da essa;
- **Test di accettazione**: È il test di collaudo del prodotto software che viene eseguito in presenza del Committente. Se questa fase finale di test viene superata positivamente si può procedere al rilascio ufficiale del prodotto sviluppato.

2.4.3 Misure e Metriche

Le misure e le metriche che il team adotterà si ispireranno alle indicazioni dello standard ISO/IEC$_G$-14598. Tale norma descrive il processo di valutazione della qualità del software. Vengono di seguito descritte le metriche sulle quali 404NotFound intende basarsi nei processi di verifica, sia in fase di progettazione che di codifica, che potranno essere integrate e stabilite con maggiore precisione durante l'avanzamento del progetto.

- **Complessità ciclomatica**: La complessità ciclomatica di un programma è il numero di cammini linearmente indipendenti attraverso il codice sorgente. Per esempio, se il codice sorgente non contiene IF o FOR, allora il livello di complessità sarà 1, poiché esiste un solo cammino, se è presente un IF la complessità diventa di 2. Un programma si può quindi rappresentare come un albero nel quale i nodi sono i blocchi del programma e gli archi sono il passaggio del controllo da un blocco all'altro. La complessità è quindi definita come:

  $$C = e - n + 2p$$

  dove e è il numero di archi, n il numero di nodi e p il numero delle componenti connesse. Il valore massimo di complessità ciclomatica accettabile per il gruppo di lavoro 404NotFound è 10;

- **Misure nella progettazione**:
  - **Complessità di flusso**: misura la quantità di informazioni in entrata ed uscita da una funzione (fan in e fan out). Fan-in è una misura del numero di metodi che invocano una determinata procedura. Un alto valore per fan-in significa che cambiamenti a quella procedura potrebbero avere effetti a catena sulle altre. Fan-out indica quanto una procedura ne richiama delle altre, dando una valutazione del grado di dipendenza di quella procedura dalle altre;
  - **Profondità di annidamento dei costrutti condizionali**: costrutti IF profondamente annidati sono più soggetti a errori e risultano più difficili da comprendere e da correggere.

- **Misure sul codice**:
  - **Lunghezza del codice**: misura la dimensione di un componente in termini di numero di linee di codice. Generalmente, maggiore è la dimensione di un componente, maggiore è la probabilità che esso contenga degli errori;
  - **Lunghezza degli identificatori**: misura la lunghezza media degli identificatori (variabili, classi, metodi, etc.). Una lunghezza media elevata è indice di un elevato grado di complessità;
  - **Numero dei parametri**: indica il numero di parametri formali nei metodi. Quando questo indice è troppo elevato, è opportuno pensare di semplificare i metodi suddividendoli in metodi più semplici. Il numero massimo di parametri per metodo stabilito dal team di sviluppo è 6.

3 Risorse

La gestione della qualità prevede l'utilizzo di alcune risorse suddivisibili in categorie.

3.1 Risorse Necessarie

3.1.1 Risorse Umane

I ruoli necessari per garantire un'adeguata qualità sono i seguenti:

- ***Responsabile del Progetto***: E' responsabile nei confronti del committente della corretta realizzazione del prodotto;
- ***Verificatore***: Coordina e svolge le attività di verifica vere e proprie;
- ***Programmatore***: Esegue attività di debugging sul codice.

Per una descrizione dettagliata delle figure elencate e di tutti gli altri ruoli specifici si rimanda al documento allegato *PianoDiPrgetto_v1.0.pdf*.

3.1.2 Risorse Software

Durante la fase di realizzazione del progetto saranno necessari:

- Software per la gestione di documenti in LaTeX;
- Piattaforma di testing sui vari browser$_G$ dell'applicazione da sviluppare;
- Piattaforma di versionamento per la creazione e la gestione di ticket$_G$;
- Software per la creazione dei diagrammi in UML$_G$;
- Ambiente per lo sviluppo del codice nel linguaggio di programmazione scelto;
- Strumenti di validazione del codice prodotto.

3.1.3 Risorse Hardware

- Computer dotati di tutti gli strumenti software descritti nel Piano di Qualifica e nelle Norme di Progetto;
- Luogo fisico in cui incontrarsi per lo sviluppo del progetto, possibilmente con una connessione ad Internet.

3.2 Risorse Disponibili

3.2.1 Risorse Software

Vengono di seguito elencate le risorse software disponibili. Per una descrizione più dettagliata si rimanda alla sottosezione Strumenti 2.4.1 del presente documento.

- TeXMaker per l'editing dei documenti in LaTeX;
- BrowserStack per il testing sui vari browser$_G$;

- GitHub per il versionamento e la gestione dei ticket$_G$;
- Astah per i diagrammi UML$_G$;
- WebStorm come ambiente di sviluppo;
- Strumenti di validazione online del W3C$_G$.

3.2.2 Risorse Hardware

- Computer personali (portatili o fissi) dei membri del gruppo;
- Computer messi a disposizione nei laboratori informatici del Dipartimento di Matematica Pura ed Applicata dell'Università di Padova;
- Aule studio del Dipartimento di Matematica Pura ed Applicata dell'Università di Padova.

4 Obiettivi di Qualità

Il gruppo 404NotFound ha ritenuto importante fissare alcuni obbiettivi di qualità da perseguire nel prodotto finale e nei processi di realizzazione, questo per garantire una migliore e più efficace soddisfazione dei requisiti richiesti nel capitolato d'appalto.

4.1 Qualità dei Processi

Per garantire la qualità del prodotto è necessario perseguire la qualità dei processi che lo definiscono. Per fare questo il team 404NotFound ha deciso di adottare lo standard ISO/IEC$_G$ 15504 denominato SPICE$_G$ (Software Process Improvement Capability Determination), il quale definisce il modello denominato SPY$_G$ (SW Process Assessment & Improvement, vedi Figura 1), per la valutazione dei processi in un'organizzazione del settore IT (Information Technology).

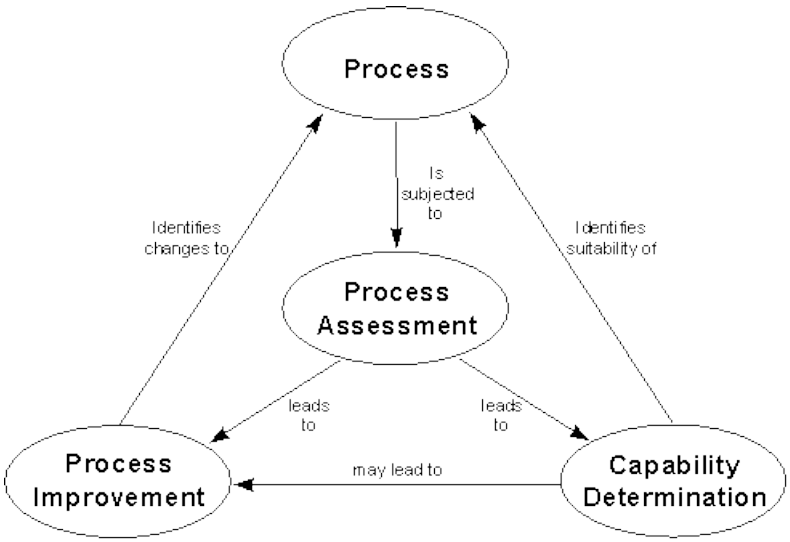

Figura 1: Modello SPY

Lo standard identifica e definisce nove attributi di qualità:

1. **Process performance attribute**: un processo raggiunge i suoi obiettivi, trasformando input identificabili in output identificabili;
2. **Performance management attribute**: l'attuazione di un processo è pianificata e controllata al fine di produrre risultati che rispondano agli obiettivi attesi;
3. **Work product management attribute**: capacità del processo di elaborare un prodotto documentato, controllato e verificato;
4. **Process definition attribute**: l'esecuzione del processo si basa su standard di processo per raggiungere i propri obiettivi;
5. **Process resource attribute**: capacità del processo di attingere a risorse tecniche e umane appropriate per essere attuato efficacemente;

6. **Process measurement attribute**: i risultati raggiunti e le misure rilevate durante l'attuazione di un processo sono stati usati per assicurarsi che l'attuazione di tale processo supporti efficacemente il raggiungimento di specifici obiettivi;
7. **Process control attribute**: il processo viene controllato attraverso la raccolta, analisi ed utilizzo delle misure di prodotto e di processo, al fine di correggere, se necessario, le sue modalità di attuazione;
8. **Process change attribute**: i cambiamenti strutturali, di gestione e di esecuzione vengono gestiti in modo controllato;
9. **Continuous improvement attribute**: le modifiche al processo sono identificate e implementate per garantire il miglioramento continuo nella realizzazione degli obiettivi di business dell'organizzazione.

La norma definisce poi quattro livelli di possesso di ciascun attributo:

- **N** - Non posseduto (0%-15% di possesso): non c'è evidenza oppure ce n'è poca del possesso di un attributo;
- **P** - Parzialmente posseduto (16%-50% di possesso): vi è evidenza di approccio sistematico al raggiungimento del possesso di un attributo, ma alcuni aspetti del possesso possono essere non prevedibili;
- **L** - Largamente posseduto (51%-85% di possesso): vi è evidenza di approccio sistematico al raggiungimento di un significativo livello di possesso di un attributo, ma l'attuazione del processo può variare nelle diverse unità operative della organizzazione;
- **F** - (Fully) Pienamente posseduto (86%-100% di possesso): vi è evidenza di un approccio completo e sistematico e di un pieno raggiungimento del possesso dell'attributo, non esistono significative differenze nel modo di attuare il processo tra le diverse unità operative.

Vi sono infine cinque livelli di maturità di processi:

- **Livello 0 - Processo incompleto**: il processo non è implementato o non raggiunge gli obiettivi. Non vi è evidenza di approcci sistematici agli attributi definiti;
- **Livello 1 - Processo semplicemente attuato**: il processo viene messo in atto e raggiunge i suoi obiettivi. Non vi è evidenza di un approccio sistematico ad alcuno degli attributi definiti. Il raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di "process performance";
- **Livello 2 - Processo gestito**: il processo è attuato, ma anche pianificato, tracciato, verificato ed aggiustato se necessario, sulla base di obiettivi ben definiti. Il raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di "Performance management" e "Work product management";

- **Livello 3 - Processo definito**: il processo è attuato, pianificato e controllato sulla base di procedure ben definite, basate sui principi del software engineering. Il raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di "Process definition" e "Process resource";
- **Livello 4 - Processo predicibile**: il processo è stabilizzato ed è attuato all'interno di definiti limiti riguardo i risultati attesi, le performance, le risorse impiegate ecc. Il raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di "Process measurement" e "Process control";
- **Livello 5 - Processo ottimizzante**: il processo è predicibile ed in grado di adattarsi per raggiungere obiettivi specifici e rilevanti per l'organizzazione. Il raggiungimento di questo livello è dimostrato attraverso il possesso degli attributi di "Process change" e "Continuous integration".

Adottando lo standard ISO/IEC$_G$ 15504 gli sviluppatori software del gruppo 404NotFound possono e intendono ottimizzare l'uso delle risorse e contenere così i costi con una migliore stima dei rischi e la possibilità di confrontarsi con delle best practice$_G$.

4.2 Qualità del Prodotto

Per quanto concerne la qualità del prodotto si è scelto di seguire alcune linee guida dettate dallo standard ISO/IEC$_G$ 9126, che definisce la qualità del prodotto software come l'insieme delle caratteristiche che incidono sulla capacità del prodotto di soddisfare requisiti espliciti o impliciti. Tale standard individua sei caratteristiche indicatrici di qualità del prodotto software, ciascuna delle quali suddivisa in sotto-caratteristiche.

4.2.1 Funzionalità

Il prodotto software realizzato deve offrire apposite funzionalità che siano in grado di soddisfare requisiti funzionali espliciti o impliciti. Le sue sotto-categorie sono:

- **Appropiatezza**: capacità di offrire un insieme di funzioni appropriate per i compiti e gli obbiettivi prefissati all'utente;
- **Accuratezza**: capacità del software di fornire i risultati concordati o i precisi effetti richiesti;
- **Interoperabilità**: capacità di interagire ed operare con altri sistemi;
- **Conformità**: capacità di aderire a standard, convenzioni e regolamentazioni rilevanti al settore operativo in cui viene applicato;
- **Sicurezza**: capacità di proteggere informazioni e dati impedendo gli accessi e le modifiche non autorizzati, mentre garantendo queste operazioni a utenti o sistemi autorizzati.

Per misurare il raggiungimento di questo obiettivo si verificherà la quantità di requisiti soddisfatti che avranno un riscontro in elementi funzionanti nell'applicazione prodotta. La soglia di sufficienza è il soddisfacimento di tutti i requisiti obbligatori previsti dal capitolato d'appalto.

4.2.2 Affidabilità

L'affidabilità misura la capacità di un prodotto software di mantenere un determinato livello di prestazioni se usato in determinate condizioni e per un certo periodo.

- **Maturità**: capacità di evitare che si verifichino fallimenti o malfunzionamenti a causa di errori nel software;
- **Tolleranza agli errori**: capacità di mantenere determinati livelli di prestazioni nonostante l'insorgere di errori, malfunzionamenti o un uso scorretto del prodotto;
- **Recuperabilità**: capacità di ripristinare il livello appropriato di performance e di recuperare le informazioni o dati rilevanti in seguito all'insorgere di un'anomalia;
- **Aderenza**: capacità di aderire a standard, convenzioni e regolamentazioni inerenti l'affidabilità.

Per misurare il raggiungimento di questo obiettivo si calcolerà il numero di esecuzioni totale confrontandolo con quelle andate a buon fine e che hanno mantenuto un livello di prestazioni tali da poter permettere l'utilizzo previsto del prodotto.

4.2.3 Efficienza

L'efficienza si misura mettendo in relazione la capacità di fornire prestazioni adeguate con la quantità di risorse impiegate.

- **Comportamento rispetto al tempo**: capacità di fornire tempi di risposta e di elaborazione adeguati per le funzioni richieste, sotto condizioni determinate;
- **Utilizzo delle risorse**: capacità di utilizzare in maniera adeguata la giusta quantità e tipologia di risorse.

Il raggiungimento di questo obiettivo sarà misurato dal tempo necessario per ottenere una risposta dal servizio (risposta dell'applicazione più il tempo necessario alla connessione) in condizioni normali e in condizioni di sovraccarico.

4.2.4 Usabilità

L'usabilità di un prodotto software si determina in base alla sua capacità di essere capito, appreso e usato dall'utente.

- **Comprensibilità**: costituisce la facilità di comprensione dei concetti del prodotto, permettendo all'utente quindi di comprendere se il programma è appropriato e come può essere utilizzato per compiti specifici;
- **Apprendibilità**: capacità di diminuire l'impegno richiesto agli utenti per imparare ad utilizzare l'applicazione;
- **Operabilità**: capacità di porre gli utenti in condizioni tali da utilizzare il prodotto e controllarne l'uso;

- **Attrattivà**: capacità di essere piacevole e di creare interesse nell'utente;
- **Conformità**: capacità di adesione a standard o convenzioni relativi all'usabilità.

Il raggiungimento di questo obiettivo sarà misurato in base alla capacità dell'applicativo di adattarsi ai vari tipi di ambienti in cui esso verrà eseguito (ambienti desktop o dispositivi mobile). L'usabilità sarà poi ritenuta raggiunta fornendo un'interfaccia il più possibile chiara, semplice ed intuitiva per l'utente.

4.2.5 Manutenibilità

La manutenibilità rappresenta la capacità del software di subire modifiche di natura correttiva, miglioramenti o adattamenti, con un impegno contenuto.

- **Analizzabilità**: capacità di facilitare l'analisi del codice e limitare l'impegno richiesto per localizzare un eventuale errore;
- **Modificabilità**: capacità del prodotto di permettere l'implementazione di una specificata modifica;
- **Stabilità**: capacità di evitare effetti inaspettati a seguito delle modifiche apportate;
- **Testabilità**: capacità di essere facilmente testato per validare le modifiche apportate.

La misurazione del raggiungimento di questo obiettivo sarà legata al rispetto delle misure e metriche descritte nel capitolo 2.4.3.

4.2.6 Portabilità

La portabilità è la capacità di un software d'essere trasferito da un ambiente di lavoro ad un altro.

- **Adattabilità**: capacità di essere adattato per differenti ambienti operativi eliminando o limitando la necessità di applicare modifiche;
- **Installabilità**: capacità di richiedere il minor impegno possibile per essere installato in uno specifico ambiente;
- **Coesistenza**: capacità di coesistere condividendo risorse con altri software nel medesimo ambiente;
- **Sostituibilità**: capacità di essere utilizzato al posto di un altro software per svolgere gli stessi compiti nello stesso ambiente.

L'obiettivo dovrà essere raggiunto ottenendo la compatibilità con i principali browser$_G$ (Google Chrome, FireFox e Internet Explorer) e validando il codice secondo gli standard del W3C$_G$.

5 Gestione amministrativa della revisione

5.1 Comunicazione e risoluzione di anomalie

Un'anomalia consiste in un comportamento non coerente con le aspettative. Un'esempio di anomalie che possono essere riscontrate sono:

- Violazione delle norme tipografiche da parte di un documento;
- Incongruenza del prodotto con funzionalità indicate nell'analisi dei requisiti.

Lo strumento scelto per la gestione delle anomalie è la sezione "Issue" messa a disposizione da Github$_G$. Coerentemente con l'organizzazione generale delle strategie di verifica, nuove anomalie potranno essere scoperte in due modi:

- In ogni fase di verifica, il *Verificatore* avrà il compito di cercare eventuali anomalie;
- Grazie all'approccio "Broken Window Theory" (vedi sezione 2.1), chiunque in qualunque momento è incentivato alla ricerca di possibili anomalie.

Nel caso in cui un *Verificatore* o un membro del gruppo individui un anomalia dovrà segnalarlo aprendo un ticket$_G$ (vedi documento allegato *NormeDiProgetto_v1.0.pdf* sezione 5.2). Un *Verificatore* ha il compito di controllare le pull request quindi nel caso trovasse un anomalia deve impedire la pull request con le modalità descritte nella sezione 5.4 delle *NormeDiProgetto_v1.0.pdf*.

5.2 Trattamento delle discrepanze

Una discrepanza è un discostamento dai requisiti attesi del capitolato o una violazione delle Norme di Progetto. Il trattamento delle discrepanze avviene come la gestione delle anomalie. Quando un membro del gruppo o il *Verificatore* ne individuasse una segnalerà il problema aprendo un ticket$_G$ oppure un *Verificatore* può bloccare la pull specificando il motivo al richiedente come per il trattamento delle anomalie.

5.3 Procedure di controllo di qualità di processo

Le Procedure di controllo di qualità di processo si basano sul ciclo di Deming o PDCA$_G$. Questo garantisce un miglioramento continuo di tutti i processi e delle attività di verifica che si realizza con comunicazioni attive delle componenti del gruppo e con la connessione delle fasi di analisi, progettazione, verifica e collaudo. La qualità dei processi viene monitorata anche grazie alla qualità di prodotto perchè un prodotto di bassa qualità può indicare che uno o più processi vadano migliorati. Per questo motivo si presta attenzione a monitorare i singoli processi ed è necessario quindi che i processi vengano pianificati nel dettaglio, le risorse vengano ripartite nella pianificazione in modo chiaro e ci sia un adeguato controllo sui processi.

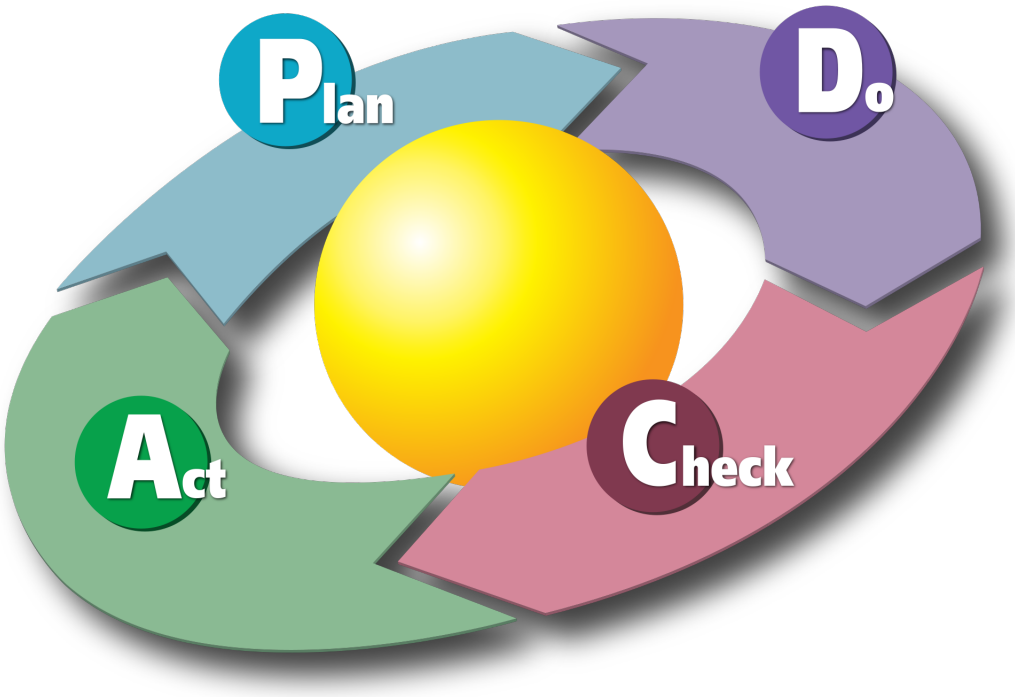

Figura 2: Modello PDCA

6 Resoconto dell' Attività di Verifica

In questa sezione vengono descritte le procedure adottate durante il processo di verifica e i risultati ottenuti.

6.1 Revisione della Documentazione

Riguardo all'attività di verifica della documentazione, la checklist stilata dai verificatori durante i controlli sui documenti tramite Inspection è la seguente:

Per i documenti in LaTeX:

- assenza di doppi spazi;
- uso corretto delle lettere maiuscole e minuscole negli elenchi puntati e all'inizio di ogni frase;
- assenza di errori ortografici di battitura;
- presenza dello spazio dopo il segno di punteggiatura;
- assenza di parti mancanti nei documenti;
- mancanza nel glossario della spiegazione di termini segnati $_G$ nei documenti;
- evitare di scrivere frasi troppo lunghe;
- evitare l'inserimento di spazi nei tag LaTeX;

- assenza di spazi all'apertura e alla chiusura delle parentesi tonde o quadre;
- presenza dello spazio dopo i segni di punteggiatura;
- verifica del funzionamento dei link dei documenti.

Per i diagrammi UML$_G$:

- il sistema non deve essere un attore;
- direzione delle freccie scorretta;
- controllo ortografico.

6.2 Tracciamento requisiti

Il tracciamento dei requisiti viene eseguito tramite il software 404TrackerDB descritto nella sezione 2.4.1. Grazie anche allo strumento TexMaker$_G$ descritto nella sezione 2.4.1 si sono potuti individuare errori ortografici mentre la parte di controllo grammaticale è avvenuta mediante la rilettura da parte dei verificatori dei documenti. I verificatori nel segnalare gli errori hanno emesso i ticket ai redattori che sono stati poi risolti dagli stessi. Ciò che si è controllato viene descritto dalla lista seguente:

- ad ogni use case$_G$ deve corrispondere un requisito;
- ad ogni requisito deve corrispondere la sua fonte;
- i requisiti devono coprire l'intero capitolato;
- Ogni requisito deve avere un codice univoco;
- i codici dei casi d'uso nei diagrammi devono corrispondere;
- la numerazione dei casi d'uso non deve contenere salti.

I requisiti presenti nel documento *AnalisiDeiRequisiti_v1.0.pdf* sono:

- **Totali**: 45;
- **Funzionali utente**: 42;
- **di Vincolo**: 3.

Gli use case$_G$ individuati sono 93 tutti lato utente.

6.3 Dettaglio delle verifiche tramite analisi

La verifiche tramite analisi statica avvengono con le modalità walkthrough e inspect spiegate sezione 2.4.2 e permettono di controllare l'andamento e la qualità del lavoro svolto.