404NotFound

Premi: better than Prezi.

Piano di Qualifica

Informazioni sul documento

Versione	4.0
Redazione	Gobbo Ismaele Vegro Federico
Verifica	Rettore Andrea
Responsabile	Camborata Marco
Uso	Esterno
Ultima modifica	06 Settembre 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
4.0	Camborata Marco	2015-09-06	Approvazione documento
3.6	Rettore Andrea	2015-09-05	Verifica finale del documento
3.5	Gobbo Ismaele	2015-09-05	Stesura appendice C7 Revisione di Accettazione
3.4	Vegro Federico	2015-09-05	Aggiunte sezioni B5 B6, Copertura dei test e risultati delle misurazioni sul codice
3.3	Gobbo Ismaele	2015-08-29	Incrementata la sezione 2.7.3 sulle Metriche, aggiunto indice LINT
3.2	Gobbo Ismaele	2015-08-29	Spostamento Capitolo 2 in appendice A e correzione dei relativi riferimenti nell'intero documento
3.1	Vegro Federico	2015-08-28	Revisione capito 3.1 sugli obiettivi di qualità, aggiunta quantificazione degli obiettivi.
3.0	Cossu Mattia	2015-08-20	Approvazione documento
2.11	Gobbo Ismaele	2015-08-18	Verifica finale del documento
2.10	De Lazzari Enrico	2015-08-13	incremento appendice B con sezione B6 Revisione di Qualifica
2.9	Camborata Marco	2015-08-05	Fine stesura appendice A3 Test di unità, tracciamento dei test di unità con le componenti
2.8	Camborata Marco	2015-07-22	Inizio stesura appendice A3 Test di unità, tracciamento dei test di unità con le componenti
2.7	Manuto Monica	2015-07-15	Realizzazione e aggiunta dei diagrammi di flusso al capitolo 3.4, fine incremento del capitolo 3.4
2.6	De Lazzari Enrico	2015-07-07	Correzione riferimenti informativi del documento con aggiunta in ogni riferimento della versione del documento o della data.
2.5	Camborata Marco	2015-07-02	Correzione errori grammaticali e sintattici nei capitoli 3.1 e 3.7.2, glossarizzazione delle nuove parti del documento
2.4	Gobbo Ismaele	2015-06-23	Verifica del documento allo stato attuale
2.3	Camborata Marco	2015-06-17	Correzione della sezione Misure e Metriche 3.7.2, aggiunta degli indici di accettabilità e ottimità per l'interpretazione delle metriche

2.2	Manuto Monica	2015-06-12	Stesura sezione 3.2 Controllo di Qualità, inizio incremento del capitolo 3.4
2.1	Manuto Monica	2015-06-10	Apportata correzione all'ordine dei capitoli 2 e 3, ora invertiti stesura sezione 3.1 introduttiva alla strategia di verifica.
2.0	Gobbo Ismaele	2015-05-22	Approvazione documento
1.7	Camborata Marco	2015-05-18	Verifica finale del documento
1.6	Vegro Federico	2015-05-14	Correzioni ortografiche e stesura resoconto revisione di progettazione
1.5	De Lazzari Enrico	2015-04-22	Stesura sezione Test di Validazione e tracciamento dei test con i requisiti
1.4	Vegro Federico	2015-04-12	Stesura sezione Test di Sistema e Test di Integrazione
1.3	Camborata Marco	2015-04-02	Verifica delle correzioni richieste dal committente
1.2	Vegro Federico	2015-03-22	Correzione sezione Responsabilità rimandando alla relativa sezione del Piano di Progetto
1.1	De Lazzari Enrico	2015-03-09	Spostamento Sezione Risorse nelle Norme di Progetto
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_v4.0.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.4.0.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 Definizione obiettivi

2.1.1 Qualità di Processo

Per garantire la qualità del prodotto è necessario perseguire la qualità dei processi che lo definiscono. Per fare questo si è deciso di adottare lo standard ISO /IEC$_G$ 15504 denominato SPICE il quale fornisce gli strumenti necessari a valutare l'idoneità di questi ultimi. Per applicare correttamente questo modello si deve utilizzare il ciclo di Deming (ciclo 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.

2.1.2 Qualità di Prodotto

Al fine di aumentare il valore commerciale di un prodotto software e di garantirne il corretto funzionamento è necessario fissare degli obiettivi qualitativi e di garantire che questi vengano effettivamente rispettati. Lo standard ISO/IEC$_G$ 9126 è stato redatto con lo scopo di descrivere questi obiettivi e delineare delle metriche capaci di misurare il raggiungimento di tali obiettivi.

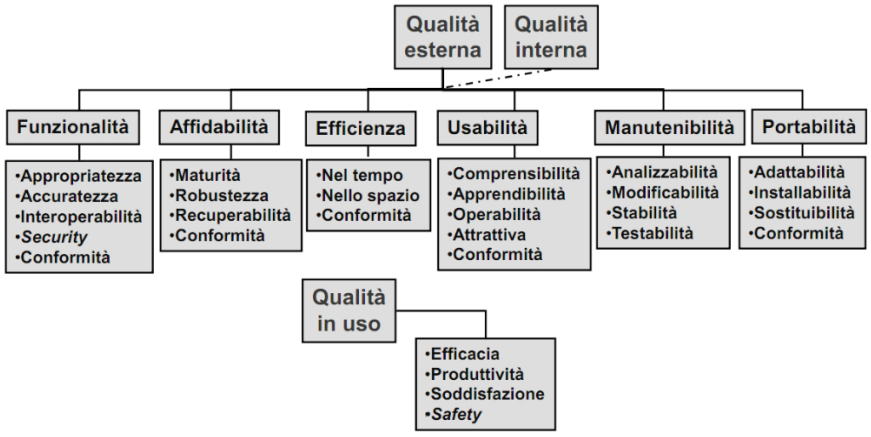

Figura 1: Rappresentazione del modello ISO/IEC 9126

2.1.2.1 Funzionalità. E' un requisito funzionale che indica la capacità del software di soddisfare le esigenze esposte dal capitolato ed individuate durante l'analisi dei requisiti, per valutare questa caratteristica si considerano l'appropriatezza e l'accuratezza delle funzioni offerte. Si sarà ottenuto un buon risultato in questo settore quando il software avrà superato in maniera positiva tutti i test e assicurerà copertura pari al 100% dei requisiti obbligatori.

2.1.2.2 Affidabilità. E' un requisito non funzionale che indica la capacità del software di svolgere correttamente il suo compito, mantenendo delle buone prestazioni anche al variare dell'ambiente nel tempo, per questo vengono considerate la sua tolleranza agli errori, la capacità di evitare fallimenti nell'esecuzione a seguito

di malfunzionamenti, detta maturità, e la recuperabilità dei dati e delle prestazioni nell'eventualità di un malfunzionamento inevitabile. Il prodotto può considerarsi affidabile se il numero di esecuzioni andate a buon fine è sufficientemente grande rispetto al numero di esecuzioni totali.

- Range ottimale : N. esecuzioni a buon fine / N. tot. di esecuzioni = [97% - 100%]
- Range accettabile : N. esecuzioni a buon fine / N. tot. di esecuzioni = [90% - 97%]

2.1.2.3 Efficienza. E' un requisito non funzionale che indica il rapporto tra le prestazioni e le risorse disponibili, si valuta cioè se il software utilizza al meglio le risorse a sua disposizione per fornire le funzionalità richieste, considerando il suo comportamento rispetto al tempo, ossia la velocità di risposta e di elaborazione in determinate condizioni, e rispetto all'uso delle risorse, ossia la sua capacità di utilizzare la quantità adeguata di risorse per eseguire le funzioni richieste. Un modo per valutare l'efficienza di un software è calcolarne i tempi di attesa in seguito all'esecuzione di un comando. Nello specifico del software Premi si intende misurare la velocità di risposta del browser e il tempo necessario al caricamento delle pagine web, misurando il TTI (time to interact) cioè in sostanza il tempo che la pagina web impiega per rendere i suoi contenuti fruibili all'utente.

- Valore ottimale : TTI inferiore a 0.8 secondi
- Valore accettabile : TTI compreso tra 0.8 e 1.5 secondi

2.1.2.4 Usabilità. E' un requisito non funzionale che indica la capacità del software di essere compreso, appreso ed usato con soddisfazione dall'utente, per far ciò il prodotto deve soddisfare condizioni di comprensibilità, apprendibilità ed operabilità, deve inoltre avere una certa attrattiva nei confronti dell'utente allo scopo di rendergliene piacevole l'utilizzo. Questa caratteristica non è facilmente misurabile in quanto non esistono metriche per quantificarla, perciò si farà affidamento sul buon senso e sul giudizio dei proponenti che sono a conoscenza delle capacità e delle necessità del bacino di utenza a cui il prodotto si rivolge.

2.1.2.5 Manutenibilità. E' un requisito non funzionale che indica la capacità del software di essere corretto, migliorato o adattato con un impegno contenuto, a tale scopo esso deve essere facilmente analizzabile e modificabile, deve garantire stabilità a seguito di modifiche e testabilità di tali modifiche. Per misurare questa caratteristica esistono una serie di metriche descritte nella sezione 2.7.3, molte delle quali quantificabili tramite il plugin Plato. L'obiettivo si riterrà raggiunto se la manutenubilità calcolata risulterà essere superiore a 70 (valore accettabile) o 80 (valore ottimale) punti su 100 nella scala di misurazione.

2.1.2.6 Portabilità. E' un requisito non funzionale che indica la capacità del software di essere trasferito da un ambiente operativo ad un altro, il prodotto deve essere in grado di adattarsi al nuovo sistema limitando la necessità di apportare cambiamenti,

non deve richiedere troppo impegno per essere installato in uno specificato ambiente e deve garantire sostituibilità con versioni precedenti o software simili. Nello specifico caso del software Premi è richiesto come requisito obbligatorio che esso funzioni correttamente e sia pienamente compatibile con il browser Chrome. Per garantire la potabilità del software è richiesto che tutti i test vengano eseguiti e il software sia testato sul browser sia in ambiente Windows 8.0 o superiore, MacOSX, Linux.

2.2 Controllo di qualità

2.2.1 Procedure di controllo di qualità di processo

La qualità dei processi viene monitorata mediante l'analisi costante della qualità del prodotto. Un prodotto di bassa qualità è indice di carenza di qualità anche nei processi coinvolti alla realizzazione di quel prodotto, e quindi indica un processo da migliorare. Per quantificare la qualità dei processi il team si impegna ad utilizzare le metriche descritte nella sezione 2.7.3.
Per avere controllo dei processi, e conseguentemente qualità, è necessario che:

- Vi sia un sufficiente livello dettaglio nella pianificazione dei processi;
- Le risorse vengano ripartite in modo chiaro nella pianificazione;

La qualità dei processi verrà inoltre garantita dall'applicazione del principio PDCA, descritto approfonditamente nella sezione 4.3. Grazie a tale principio, sarà possibile garantire un miglioramento continuo della qualità di tutti i processi, inclusa la verifica, e come diretta conseguenza si otterrà il miglioramento dei prodotti risultanti.

2.2.2 Procedure di controllo di qualità di prodotto

Il controllo di qualità del prodotto verrà garantito da tre diverse attività:

- **Quality Assurance**: insieme di attività realizzate per garantire il raggiungimento degli obbiettivi di qualità. Prevede l'attuazione di tecniche di analisi statica e dinamica, descritte nella sezione 2.7.2;
- **Verifica**: processo che determina se l'output di una fase è consistente, completo e corretto. La verifica andrà eseguita costantemente durante l'intera durata del progetto. I risultati delle attività di verifica eseguiti nelle varie fasi del progetto sono riportati nell'appendice B;
- **Validazione**: conferma in modo oggettivo che il sistema risponda ai requisiti.

2.3 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_v2.0.pdf*), si intende svolgere una costante attività di verifica trasversale a tutte le fasi di sviluppo del progetto. Si è scelto e adottato il metodo "Broken Window Theory" secondo il quale, non appena un errore viene rilevato, questo dovrà essere segnalato e corretto il prima possibile onde evitarne la propagazione.

Per poter svolgere 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_v4.0.pdf*, sezione 6.6) e una fase di verifica finale è necessaria affinché 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.4.0.pdf* sezione 5.4)

Per quanto riguarda il codice è molto importante che i processi di verifica siano resi il più automatizzati possibile per garantire così la loro ripetibilità e per elevarne l'efficienza. Le attività di verifica avvengono allo scopo di trovare e rimuovere eventuali problemi presenti. Un problema può verificarsi a vari livelli, e per ogni livello assume un nome diverso:

- **Fault**: è il *difetto* che sta all'origine del problema, ciò da cui scaturisce un malfunzionamento;
- **Error**: è lo stato di *errore* per il quale il software si trova in un punto sbagliato del flusso di esecuzione o con valori sbagliati rispetto a quanto previsto dalla specifica;
- **Failure**: è un *Fallimento* o *guasto*, cioè un comportamento difforme dalla specifica, la manifestazione dell'errore del software all'utente.

Esiste una relazione di causa-effetto fra questi tre termini:

DIFETTO → ERRORE → FALLIMENTO

Non sempre un errore dà origine ad un fallimento: ad esempio potrebbero esserci alcune variabili che si trovano in stato erroneo ma non vengono lette, o non viene percorso il ramo di codice che le contiene. E' compito dei *Verificatori* prestare particolare attenzione a questo tipo di errori (detti anche quiescenti).

2.4 Pianificazione Strategica e Temporale

Avendo l'obiettivo di rispettare le scadenze fissate nel *PianodiProgettov4.0.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.

Essendo il ciclo di vita scelto per lo sviluppo del progetto un ciclo di vita incrementale (vedi documento allegato *PianoDiProgetto_v4.0.pdf*), 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.

2.4.1 Analisi dei Requisiti

Tutta la documentazione relativa alla RR, una volta completata, entrerà nella dedicata fase di revisione. La verifica inizia quando i documenti:

- Piano di Progetto
- Norme di Progetto
- Analisi dei Requisiti
- Studio di Fattibilità
- Piano di Qualifica

hanno una struttura definita e sufficienti contenuti su cui poter verificare. La verifica avviene mediante controllo ortografico, grammaticale e concettuale dei documenti da consegnare alla prima revisione. Di seguito i parametri di controllo:

- 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;

- 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;

- Correttezza logica e formale dei requisiti, della loro tracciabilità, non ambiguità e conformità con quanto richiesto dal *Proponente*.
- Ogni requisito deve possedere un codice identificativo univoco e un titolo non ambiguo e sufficientemente descrittivo.
- Corrispondenza tra ogni requisito e caso d'uso corrispondente;
- La 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* al redattore, il quale dovrà apportare le modifiche richieste e ripresentare il documento stesso per un'ulteriore attività di verifica ed eventuale validazione.

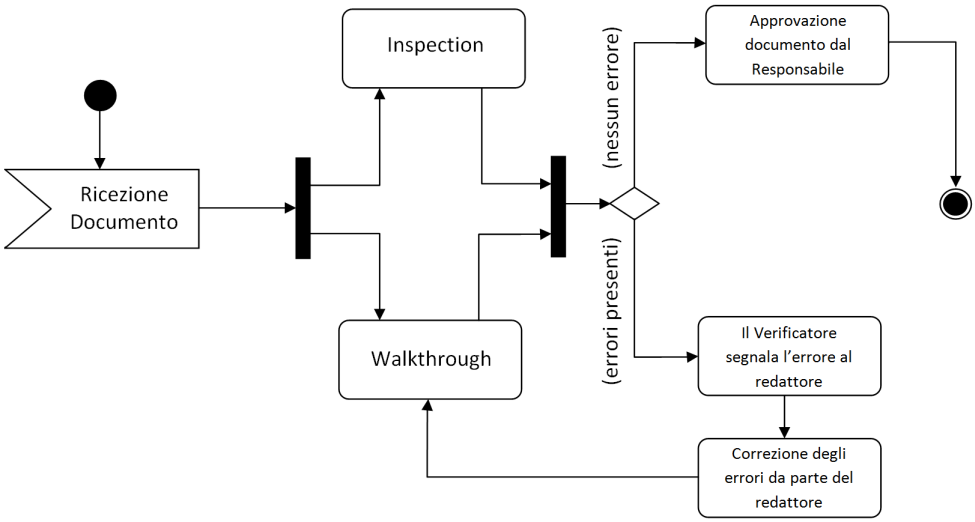

Figura 2: Verifica dei documenti

2.4.2 Progettazione Architetturale

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. Nello svolgere questa attività il gruppo di lavoro dovrà porsi come obiettivo il conseguimento delle seguenti proprietà:

- **Semplicità**: i componenti devono contenere solo quello che è necessario al loro funzionamento;
- **Coesione**: misura quanto sono collegati tra di loro le componenti di un modulo. Se i metodi di una classe svolgono compiti simili, il grado di coesione di tale classe è alto. L'obiettivo sarà quello di massimizzare questo aspetto;

- **Incapsulamento**: ll funzionamento interno di una classe viene nascosto all'esterno, proteggendo così gli utenti di quella classe da eventuali modifiche;
- **Accoppiamento**: indica quanto una componente fa affidamento sull'altra, dipendendo da essa. Un basso grado di accoppiamento favorisce la manutenibilità del software.

Gli obiettivi di qualità vengono descritti nell'appendice A. il lavoro del *Verificatore* è quello di effettuare una verifica incrociata tra i requisiti riportati nel documento *AnalisiDeiRequisiti_v4.0.pdf* e le componenti descritte nel documento *SpecificaTecnica_v3.0.pdf*, accertandosi i requisiti utente siano soddisfatti dalle componenti architetturali prodotte durante la progettazione ad alto livello e dai metodi e dagli attributi prodotti nella progettazione di dettaglio. Le attività di tracciamento, controllo diagrammi UML e verifica della progettazione vengono eseguite in parallelo assicurando il soddisfacimento delle quattro proprietà descritte sopra. Inoltre, i *Verificatori* devono controllare che:

- non vi sia eccessiva complessità nelle componenti del sistema;
- ciascuna parte individuata non sia ulteriormente divisibile;
- i pattern utilizzati siano descritti e motivati, specificando i vantaggi e le implicazioni derivanti dalla loro adozione;
- i pattern siano usati correttamente nell'architettura del sistema;

I diagrammi UML devono essere conformi a quanto segue:

- essere conformi allo standard 2.0;
- non possedere associazioni scorrette;
- avere una complessità non eccessiva: la complessità è troppo elevata se è possibile individuare dei gruppi logici indipendenti in uno stesso diagramma. Ciò è causa di errori, diminuisce la comprensibilità e riduce il tempo di vita del diagramma stesso, in quanto una delle componenti rappresentate potrebbe essere soggetta a cambiamenti;
- non possedere relazioni di dipendenza scorrette;
- le classi devono essere posizionate correttamente (al posto giusto e nel package giusto).

2.4.3 Programmazione di dettaglio e codifica

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. Devono essere effettuati test di unità, scritti anch'essi dai programmatori, per verificare che il comportamento del codice che hanno scritto corrisponda a quanto ci si aspetta. Un test su un'unità è composto da:

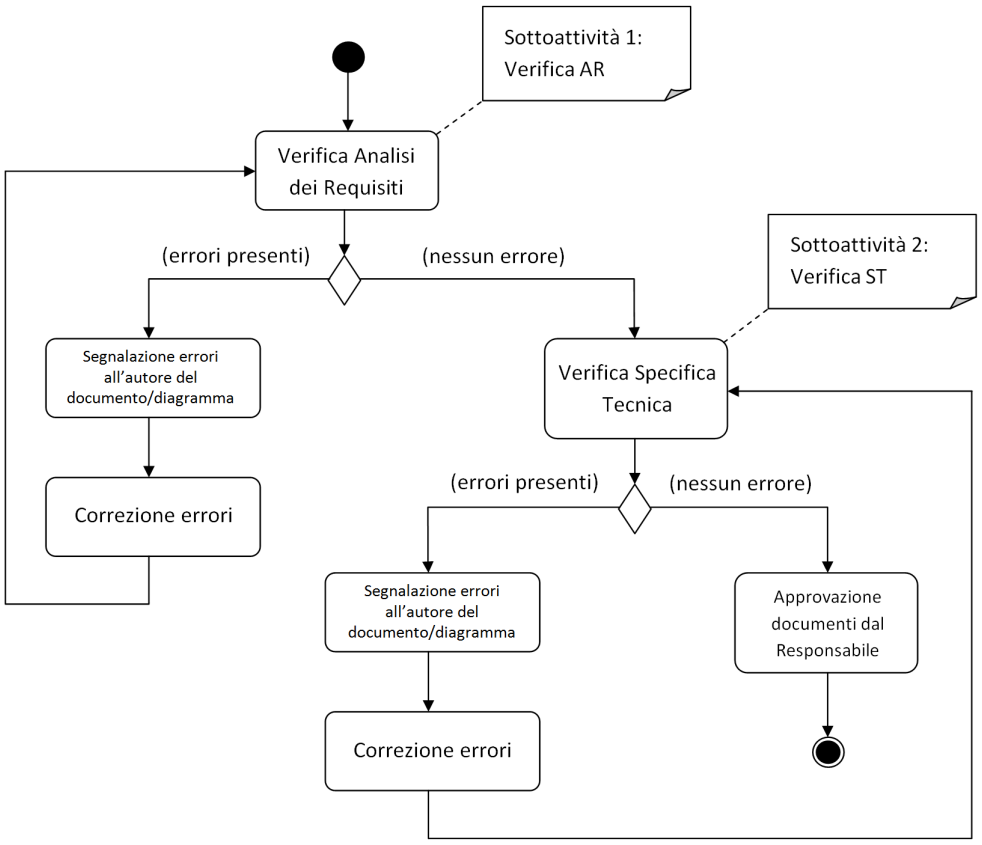

Figura 3: Verifica della progettazione

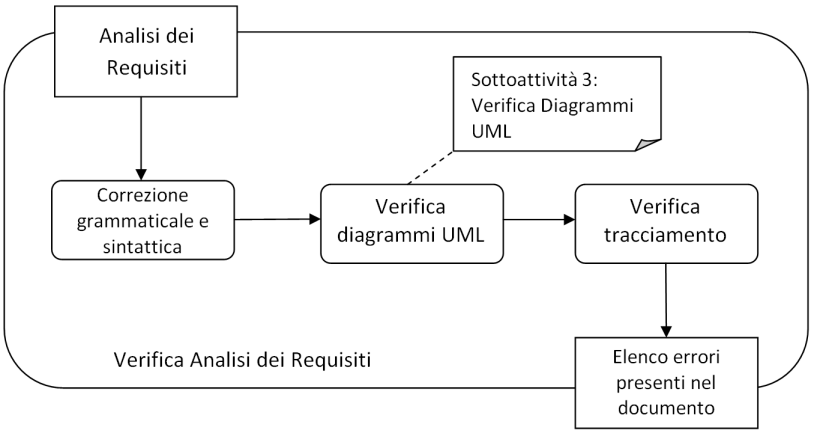

Figura 4: Sottoattività 1: Verifica dell'Analisi dei Requisiti

- l'oggetto su cui viene eseguito il test;
- la strategia utilizzata per effettuare la prova;
- il piano di esecuzione del test stesso, che comprende gli ingressi del test e deve prevederne le uscite attese.

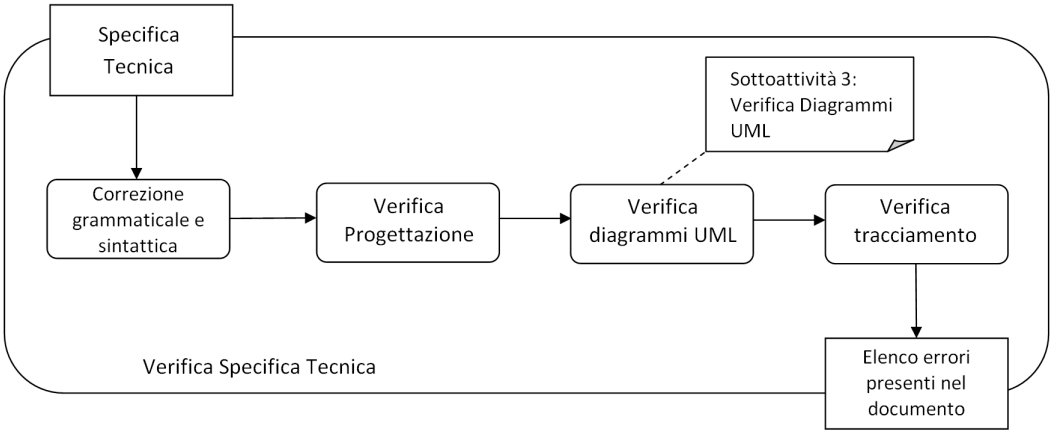

Figura 5: Sottoattività 2: Verifica della Specifica Tecnica

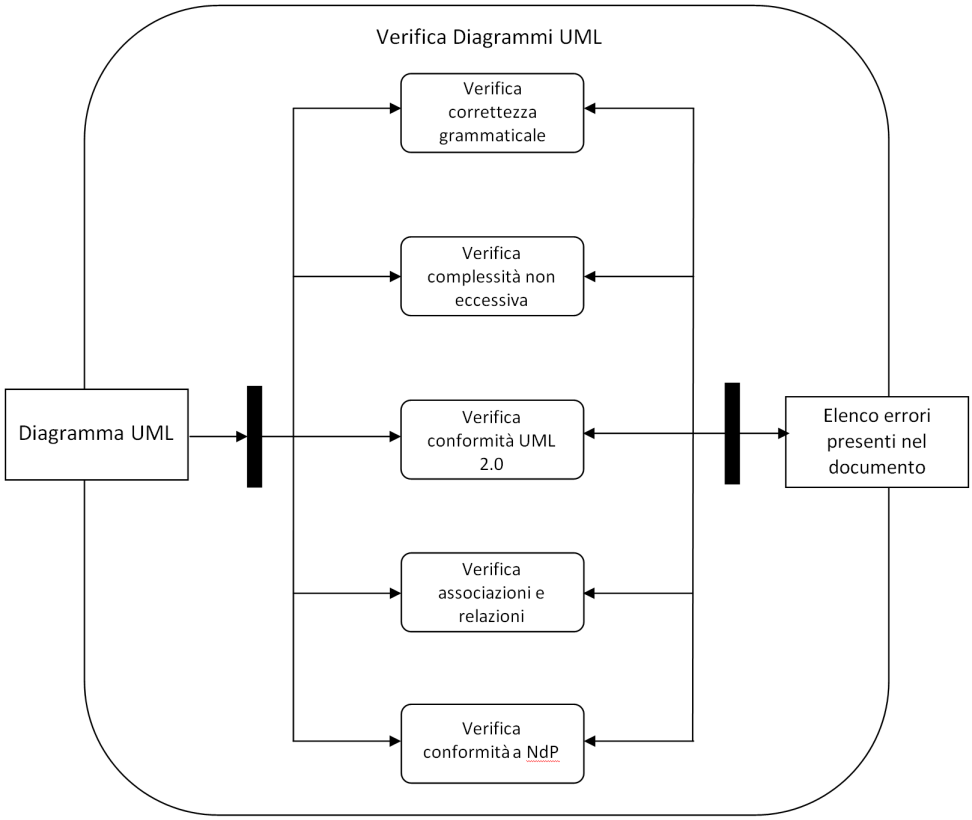

Figura 6: Sottoattività 3: Verifica dei diagrammi UML

Perchè sia possibile individuare il maggior numero di errori possibile è necessario che i test vengano pianificati assicurando:

- **Statement Coverage**: il test deve coprire tutte le linee di codice del modulo in esame;
- **Branch Coverage**: il test deve coprire tutti i rami del flusso di controllo almeno una volta.

Questi test apparterranno alla categoria white box: eseguiti cioè conoscendo in dettaglio il codice sorgente delle singole unita‘ e analizzando come, a partire dagli input, sono prodotti gli output. In questo modo si garantisce la correttezza logica di ogni metodo. I verificatori dovranno poi eseguire un'attività di inspection (vedi sezione 2.7.2) su quanto scritto dai programmatori secondo la seguente lista di controllo:

- le variabili globali non sono ammesse;
- le variabili vanno inizializzate al momento della dichiarazione;
- non sono ammesse variabili e metodi non utilizzati;
- minimizzare, se non eliminare, la presenza di alert$_G$ o altre forme di warning$_G$.
- l'uso del break$_G$ non ‘e ammesso se non all'interno dei costrutti switch$_G$, che devono comunque essere presenti il più esiguamente possibile nel codice in quanto aumentano la complessità ciclomatica di 1 per ogni ramo case;
- le variabili che controllano l'uscita dai cicli non devono poter essere modificate dall'esterno del ciclo;
- deve essere favorita la lazy evaluation$_G$ delle condizioni booleane;
- il codice deve essere il più possibile comprensibile; qualora risulti di difficile comprensione il programmatore deve inserire dei commenti. I commenti non vanno tuttavia inseriti per descrivere righe il cui significato è ovvio;
- i commenti al codice devono essere scritti in italiano;
- l'indentazione deve essere consistente e favorire la lettura del codice.

Vanno infine garantite le caratteristiche di qualità descritte dallo standard ISO/IEC 91264 e riportate in nella sezione 2.2 del presente documento e il rispetto dei valori per le metriche stabiliti nella sezione 2.7.3.

2.4.4 Test e 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. I test devono essere contraddistinti da un ID, devono elencare le componenti in esame e le modalità di testing. Nell'appendice A vengono elencati i test svolti dal gruppo per lo sviluppo dell'applicazione Premi.

Quando tutte le componenti sono state testate e integrate si procede con il test di sistema, che controlla che il prodotto svolga correttamente i suoi compiti e rispetti i requisiti fissati in sede di analisi. Il controllo di sistema si divide in due categorie:

- **Alfa-test**: consiste nell'utilizzo del software, effettuato all'interno del gruppo, andando a testare tutte le funzionalità e inserendo istruzioni di controllo a tempo di esecuzione, segnalando eventuali malfunzionamenti;

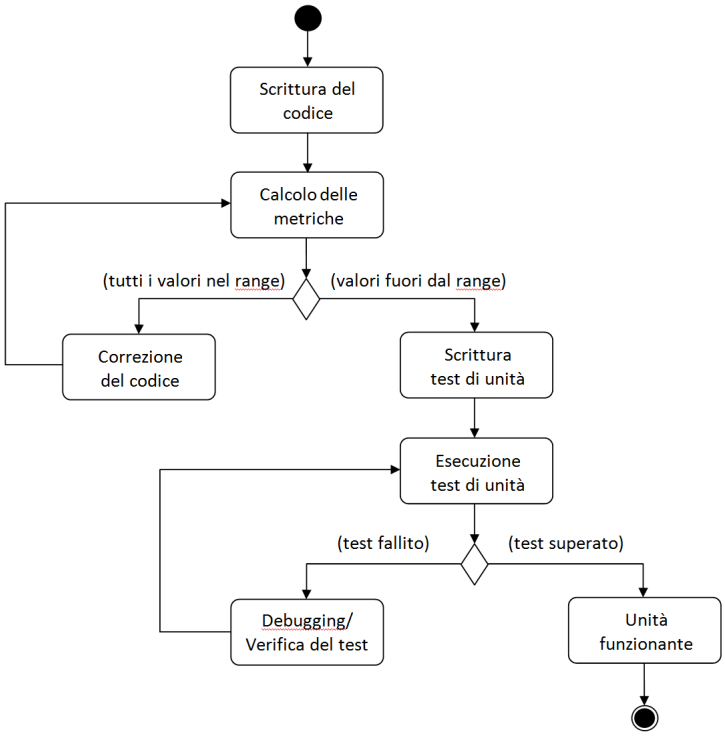

Figura 7: Verifica della codifica

- **Beta-test**: consiste in un utilizzo normale del prodotto, effettuato possibilmente da membri estranei al team di sviluppo, alla ricerca degli ultimi problemi residui.

2.5 Responsabilità

Per garantire che il processo di verifica sia efficace e sistematico vengono attribuite delle responsabilità ad ogni specifico ruolo del progetto. I ruoli che detengono le responsabilità del processo di verifica sono il *Responsabile del Progetto* ed i *Verificatori*. La suddivisione dei compiti e le modalità di attuazione degli stessi sono definiti nel documento*NormeDiProgetto_v4.0.pdf*.

2.6 Risorse

Per assicurare che gli obiettivi qualitativi vengano raggiunti e` necessario l'utilizzo di risorse sia umane che tecnologiche. Coloro che detengono la responsabilità maggiore per l'attività di verifica e validazione sono il *Responsabile del Progetto* e il *Verificatore*. Per una dettagliata descrizione dei ruoli e delle loro responsabilità fare riferimento alle *NormeDiProgetto_v4.0.pdf*. Per risorse tecniche e tecnologiche sono da intendersi tutti gli strumenti software e hardware che il gruppo intende utilizzare per attuare le attività di verifica su processi e prodotti.

2.7 Strumenti, Tecniche e Metodi

2.7.1 Strumenti

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

- **Correttore automatico di TeXMaker**$_G$: come segnalato nelle *NormeDiProgetto_v4.0.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$_G$;
- **Plato** (https://github.com/jsoverson/plato): visualizzatore del codice sorgente, strumento per l'analisi statica e calcolo della complessità.
- **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.

2.7.2 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, poiché 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. Perché 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.

2.7.3 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.

Essendo la natura delle metriche molto variabile, vi possono essere due tipologie di range:

- **Accettazione**: valori da rispettare affinché il prodotto possa essere accettato;
- **Ottimale**: valori entro cui dovrebbe collocarsi la misurazione. Tale range non e` vincolante, ma fortemente consigliato.

2.7.3.1 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. Verrà adottata la seguente tabella di riferimento per la complessità:

Complessità ciclomatica	Tipo di procedura	Rischio
1-4	Semplice	Basso
5-10	Ben strutturata e stabile	Medio Basso
11-20	Complessa	Moderato
21-50	Molto complessa, preoccupante	Alto
>50	Soggetta a errori, problematiche, instabile	Molto alto

Figura 8: Stima della Complessità Ciclomatica

Parametri utilizzati:

- Range-accettazione: $[1 - 20]$;
- Range-ottimale: $[1 - 10]$.

Il valore 10 come massimo di complessità ciclomatica fu raccomandato da T. J. McCabe, l'inventore di tale metrica.

2.7.3.2 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; Il valore e` calcolato come:

$$(\text{lunghezza funzione})^2 \times \text{Fan-In} \times \text{Fan-Out}$$

- **Livelli di annidamento**: Rappresenta il numero di livelli di annidamento dei metodi, cioè l'inserimento di una struttura di controllo all'interno di un'altra. Un valore elevato di tale indice implica un'alta complessità ed un basso livello di astrazione del codice.

  Parametri utilizzati:

  - Range-accettazione: $[1 - 6]$;
  - Range-ottimale: $[1 - 3]$.

- **Profondità di annidamento dei costrutti condizionali**: costrutti IF profondamente annidati sono più soggetti a errori e risultano più difficili da comprendere e da correggere.

  Parametri utilizzati:

  - Range-accettazione: $[0 - 3]$;
  - Range-ottimale: $[0 - 1]$.

2.7.3.3 Misure sul codice

- **SLOC (Source Lines Of Code)**: indice della lunghezza complessiva del codice, misura la dimensione del prodotto in termini di numero di linee di codice, al netto delle linee vuote o commentate.
- **MLOC (Method Lines Of Code)**: numero di linee di codice di un metodo, al netto delle linee vuote o commentate; Generalmente, maggiore è la dimensione di un metodo o componente, maggiore è la probabilità che esso contenga degli errori;
- **MIL (Medium Identificators Length)**: misura la lunghezza media degli identificatori (variabili, classi, metodi, etc.). Una lunghezza media elevata è indice di un elevato grado di complessità;

Anche se si ritiene ragionevole porsi l'obiettivo di mantenere la lunghezza dei moduli al di sotto delle 400 righe per agevolarne la manutenibilità, le metriche dimensionali, riguardanti cioè il numero di linee di codice, non devono essere viste troppo rigidamente. Nella maggior parte dei casi la lunghezza non dovrebbe comunque superare le poche decine di righe.

Parametri utilizzati:

- Range-accettazione: $[0 - 400]$;
- Range-ottimale: $[0 - 50]$.

- **NOF (Number Of Fields)**: è il numero totale dei campi dati per classe;

  Parametri utilizzati:

  - Range-accettazione: $[0 - 30]$;
  - Range-ottimale: $[5 - 20]$.

- **NOP (Number Of Parameters)**: 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.

  Parametri utilizzati:

  - Range-accettazione: $[0 - 6]$;
  - Range-ottimale: $[0 - 4]$.

- **NOM (Number of Methods)**: è il numero totale di metodi in un particolare scope_G;

  Parametri utilizzati:

  - Range-accettazione: $[0 - 20]$;
  - Range-ottimale: $[0 - 7]$.

- **PAR (Number of Parameters)**: è il numero totale di parametri in un particolare scope_G.

  Parametri utilizzati:

  - Range-accettazione: $[0 - 9]$;
  - Range-ottimale: $[0 - 4]$.

- **LINT Errors**: indice che misura il numero di discrepanze rilevate confrontando il codice analizzato con i costrutti noti del linguaggio Javascript, e le sue best practice.

  Parametri utilizzati:

  - Range-accettazione: $[0 - 10]$;
  - Range-ottimale: $[0 - 5]$.

- **Est. Errors**: Numero di errori stimati nel codice calcolato sulla base della complessità misurata.

  Parametri utilizzati:

  - Range-accettazione: $[0 - 15]$;
  - Range-ottimale: $[0 - 3]$.

3 Gestione amministrativa della revisione

3.1 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_v4.0.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_v4.0.pdf*.

3.2 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.

3.3 Procedure di controllo di qualità di processo

Come accennato nella sezione 2.2.1, l'organizzazione dei processi fa riferimento al ciclo di Deming, che ha come scopo il miglioramento continuo, secondo il principio del PDCA. 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 perché 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. Tale metodo è suddiviso in quattro fasi:

1. **Plan - Pianificare**:
   (a) Definire il problema/impostare il progetto;
   (b) Documentare la situazione di partenza;
   (c) Analizzare il problema;
   (d) Pianificare le azioni da realizzare.
2. **Do - Realizzare**:
   (a) Addestrare le persone incaricate della realizzazione;
   (b) Realizzare le azioni che sono state pianificate.
3. **Check -Verificare**:
   (a) Verificare i risultati e confrontarli con gli obiettivi;
   (b) Se si 'e raggiunto l'obiettivo: passare alla lettera a della fase Act;
   (c) Se non si è raggiunto l'obiettivo: passare alla lettera b della fase Act.
4. **Act - Mantenere o Migliorare**:
   (a) Obiettivo raggiunto:
      i. Standardizzare, consolidare e addestrare gli operatori;
      ii. Procedere a un nuovo PDCA per un ulteriore miglioramento sul tema.
   (b) Obiettivo non raggiunto:
      i. Ripetere il ciclo PDCA sullo stesso problema, analizzando criticamente le varie fasi del ciclo precedente per individuare le cause del non raggiungimento dell'obiettivo.

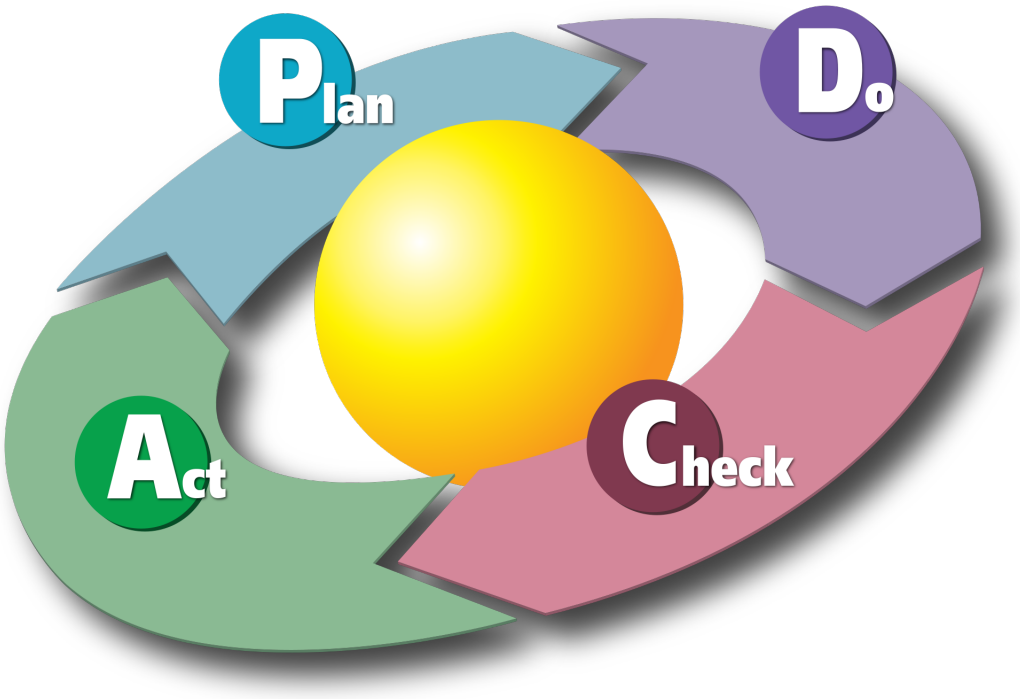

Figura 9: Modello PDCA

I parametri che permetteranno di valutare la qualità del processo saranno principalmente:

- il tempo impiegato per essere portato a termine;
- la quantità di risorse impiegate;
- il numero di iterazioni che è stato fatto;
- il numero di difetti trovati durante la fase di testing.

Tali parametri devono essere quantificati sia durante che al termine del processo, al fine di individuare eventuali problemi e capire, attraverso un'analisi condivisa dai membri del gruppo, in quali aree c'è bisogno di un miglioramento. Alla successiva istanziazione del processo, i dati raccolti la volta precedente vanno capiti e migliorati in modo da rendere più efficiente ed efficace il processo stesso.

Il gruppo 404NotFound adotta come riferimento lo standard ISO/IEC$_G$ 14598 che descrive il processo di valutazione della qualità del software, in accordo con la norma ISO/IEC 9126 descritta nell'appendice A. Esse verranno utilizzate per valutare il software durante tutto il suo ciclo di vita. La figura seguente schematizza il processo di valutazione:

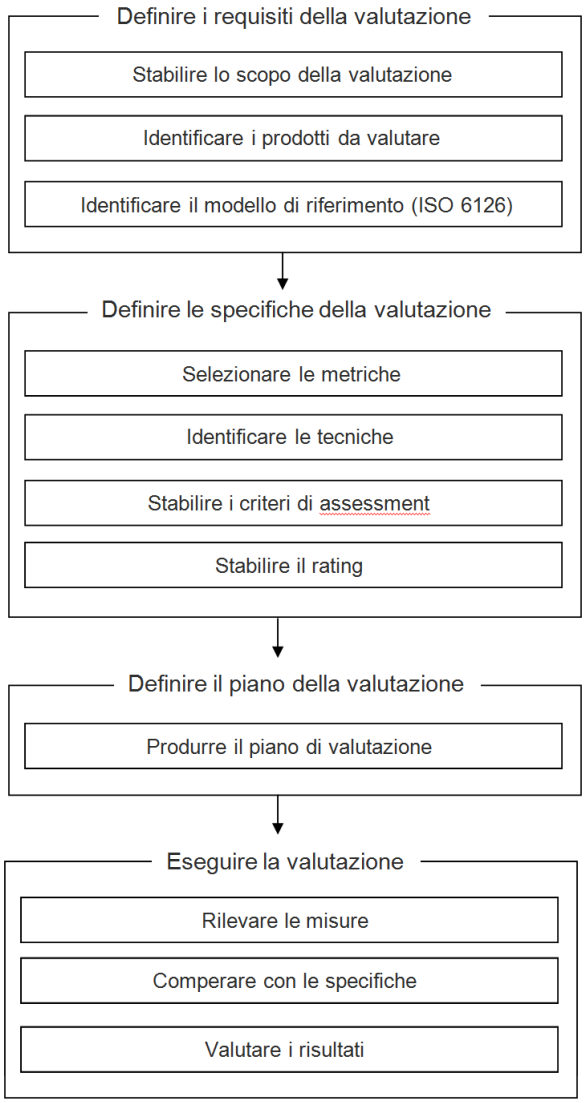

Figura 10: Il Processo di Valutazione

A 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.

A.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).

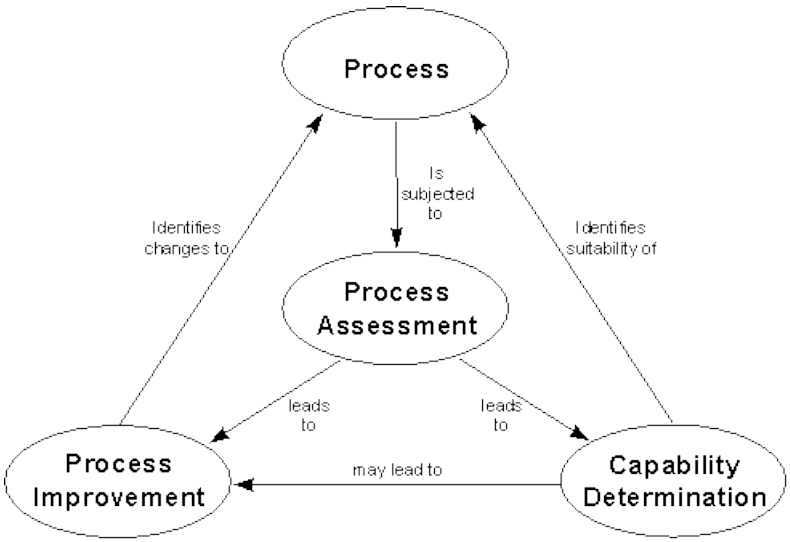

Figura 11: Modello SPY

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$_G$.

A.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.

A.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.

A.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.

A.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.

A.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.

A.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 2.7.3.

A.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$_G$.

B Pianificazione dei test

Si descrivono di seguito tutti i test di validazione, sistema ed integrazione previsti, prevedendo un aggiornamento futuro per i test di unita`. Per le tempistiche di esecuzione dei test si faccia riferimento al *PianoDiProgetto_v4.0.pdf* Nelle tabelle sottostanti lo stato dei test "Pianificato" e` da intendersi come non applicato in quanto tali test saranno applicati successivamente, come descritto nel *Piano di Progetto*.

B.1 Test di sistema

In questa sezione vengono descritti i test di sistema che permettono di verificare il comportamento dinamico del sistema completo rispetto ai requisiti descritti nell'*Analisi dei Requisiti v2.0*. I test di sistema riportati sono quelli relativi ai requisiti software individuati e pertanto meritevoli di un test.

B.1.1 Descrizione dei test di sistema

Requisito	Stato	Descrizione	Test
FOb1	Success	Si verifica che il sistema permetta la creazione di una presentazione e che questa venga effettivamente creata e salvata sul database	TS1
FOb3	Success	Si testa l'esecuzione di una presentazione, verificando che il percorso seguito sia quello scelto dall'utente ed il corretto avanzamento delle slides	TS3
FOb3.5	Success	Si verifica il funzionamento dei checkpoint$_G$, la possibilità per l'utente di impostare un checkpoint$_G$ e il corretto funzionamento dei percorsi di approfondimento	TS3.5
FOb4	Success	Si verifica che le modifiche apportate dall'utente ad una presentazione, quali ad esempio inserimento di oggetti grafici, testi o nuovi frame$_G$ vengano correttamente applicate e salvate dal sistema	TS4
FOb5	Success	Viene verificato che il salvataggio della presentazione avvenga con successo e nel formato atteso	TS5

FOb6	Success	Si verifica che l'azione di eliminazione da parte di un'utente di una presentazione cancelli effettivamente tutti i dati relativi a quella presentazione dal sistema	TS6
FOb6.1	Success	Si testa l'annullamento del comando di eliminazione e si verifica che non vi sia alcuna modifica alla presentazione fintanto che l'azione non viene confermata al sistema dall'utente.	TS6.1
FOb7	Success	Si verifica che il sistema permetta la pubblicazione di una presentazione e che questa vada a buon fine	TS7
FOb8	Success	Il sistema deve poter generare un link per una presentazione live	TS8
FOp9	Success	l'utente deve poter rendere privata una presentazione pubblica	TS9
FOb11.1	Success	Viene verificato che il sistema permetta di esportare la presentazione in formato poster controllando che il file di output corrisponda alla presentazione esportata	TS11.1
FOb11.2	Success	Si verifica che il formato con cui viene esportata la presentazione sia portabile	TS11.2
FOb13	Success	Si verifica che il sistema permetta all'utente di registrarsi simulando i passi della procedura di registrazione e successivamente verificando il salvataggio dei dati inseriti dall'utente	TS13
FOb14	Success	Si verifica che il sistema permetta all'utente di autenticarsi	TS14

FOb15	Success	Viene verificato il corretto funzionamento di tutti i messaggi di errore da parte del sistema, l'utente deve sapere quando ha commesso un errore	TS15
FOb16	Success	Viene verificata e provata la procedura di cambio password, assicurandosi che il sistema salvi correttamente le modifiche dell'utente	TS16
VOb17	Success	Viene testato l'avvio del sistema su browser$_G$ Chrome da versione 40+	TS17

Tabella 2: Tabella di tracciamento test di sistema / requisiti

B.2 Test di integrazione

In questa sezione vengono descritti i test di integrazione per i vari componenti descritti nella progettazione ad alto livello, che permettono di verificare la corretta integrazione ed il corretto flusso dei dati all'interno del sistema. Si e` scelta una strategia di integrazione incrementale, il cui principale vantaggio è quello di poter sviluppare e verificare le componenti in parallelo. Con l'approccio incrementale, infatti, i difetti rilevati da un test sono da attribuirsi, con maggior probabilità, all'ultima componente aggiunta ed essendo ogni passo di integrazione reversibile è possibile retrocedere ed effettuare un rollback ad uno stato noto e sicuro. Per i test di integrazione è stato utilizzato il metodo bottom-up, questa scelta risulta automatica avendola già adottata nella progettazione delle componenti con l'obiettivo quello di ridurre le dipendenze funzionali di ogni singola componente. Il diagramma seguente non rispetta il formalismo UMLG 2.x ed e` utilizzato per semplificare l'illustrazione della strategia di integrazione. Si possono notare, lungo l'albero, dei macro componenti che possono integrare al loro interno due o più componenti di maggior dettaglio.

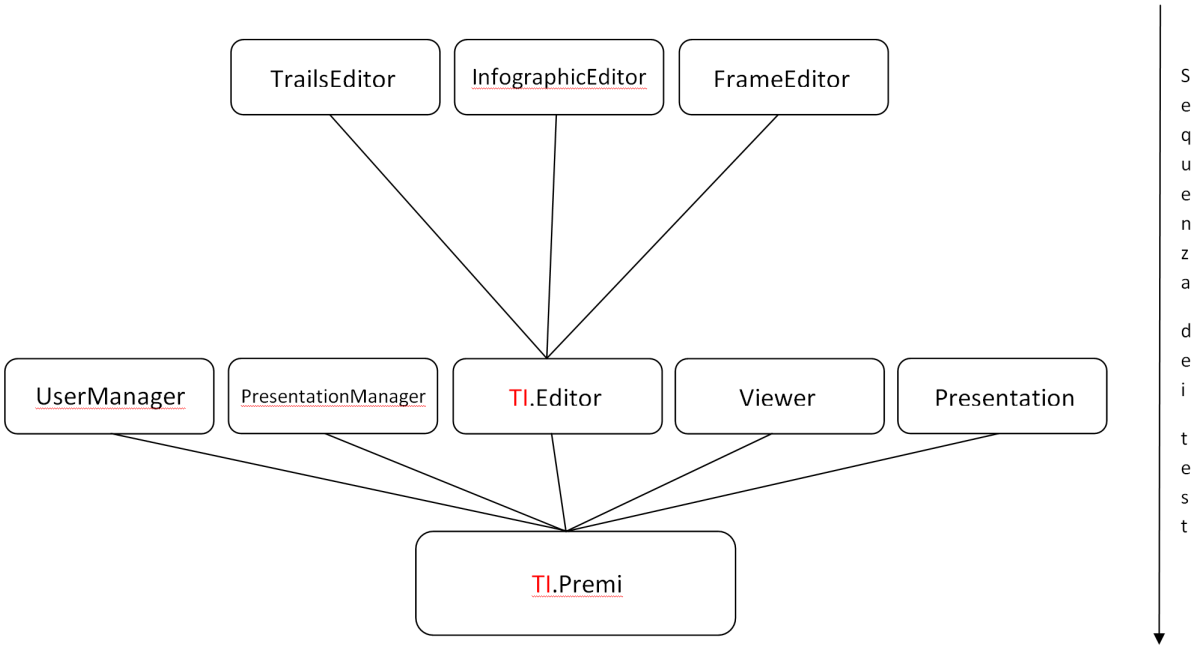

Figura 12: Diagramma informale della strategia di integrazione

B.2.1 Descrizione dei test di integrazione

Test	Descrizione	Stato
TI.Premi	Test di integrazione finale per le componenti del modulo Premi	Success
TI.UserManager	**Gestione dell'Utente**: Verifica il corretto funzionamento delle operazioni di registrazione, autenticazione e cambio password	Success
TI.PresentationManager	**Gestione della presentazione**: Verifica il corretto funzionamento del sistema di gestione della presentazione, testa la correttezza delle operazioni di creazione, modifica eliminazione, pubblicazione, esportazione e portabilità di una presentazione	Success
TI.Editor	Test di integrazione finale per le componenti del modulo Premi.Editor	Success
TI.TrailsEditor	**Edit dei percorsi**: Verifica il corretto funzionamento dei percorsi, e le procedure di creazione, modifica, clonazione ed eliminazione di un persorso	Success
TI.InfographicEditor	**Edit dell'Infografica**: Verifica il corretto funzionamento delle infografiche, nello specifico l'aggiunta di frame$_G$, immagini, testi e shape$_G$ all'infografica e la modifica dello stile	Success
TI.FrameEditor	**Edit dei Frame**: Verifica il corretto funzionamento dei frame$_G$, nello specifico l'aggiunta di immagini, testi e shape$_G$ al frame$_G$ e la modifica dello stile	Success
TI.Viewer	**Visualizzazione della presentazione**: Verifica la corretta riproduzione della presentazione	Success

Tabella 3: Tabella test di integrazione

B.2.2 Tracciamento componenti – test di integrazione

Nella tabella seguente è riportato il tracciamento delle singole componenti del sistema con il relativo test di integrazione. Si noti che l'integrazione di Views e Controllers sono garantite dall'architettura del sistema.

Componente	**Test**
Premi	TI.Premi
Premi.UserManager	TI.UserManager
Premi.UserManager.Views	Architettura del Sistema
Premi.UserManager.Controllers	Architettura del Sistema
Premi.PresentationManager	TI.PresentationManager
Premi.PresentationManager.Views	Architettura del Sistema
Premi.PresentationManager.Controllers	Architettura del Sistema
Premi.Editor	TI.Editor
Premi.Editor.Views	Architettura del Sistema
Premi.Editor.Controllers	Architettura del Sistema
Premi.Editor.TrailsEditor	TI.TrailsEditor
Premi.Editor.TrailsEditor.Views	Architettura del Sistema
Premi.Editor.TrailsEditor.Controllers	Architettura del Sistema
Premi.Editor.InfographicEditor	TI.InfographicEditor
Premi.Editor.InfographicEditor.Views	Architettura del Sistema
Premi.Editor.InfographicEditor.Controllers	Architettura del Sistema
Premi.Editor.FrameEditor	TI.FrameEditor
Premi.Editor.FrameEditor.Views	Architettura del Sistema
Premi.Editor.FrameEditor.Controllers	Architettura del Sistema
Premi.Viewer	TI.Viewer
Premi.Viewer.Views	Architettura del Sistema
Premi.Viewer.Controllers	Architettura del Sistema

Tabella 4: Tabella tracciamento componente - test di integrazione

B.3 Test di unità

Di seguito vengono riportati i test di unità previsti per l'applicazione. La tabella riporta anche il tracciamento dei test con le componenti. Si è scelto di scrivere dei test di sistema solo per i metodi e le componenti principali dell'applicazione Premi, tralasciando tutti i metodi il cui test sarebbe risultato ovvio o banale come ad esempio i metodi *get* e *set* di ogni classe.

Test	**Componente**	**Descrizione**	**Metodi**	**Stato**
TU1	Premi.Editor.graphicObject	Si verifica che l'inizializzazione dei campi dati di default avvenga con i valori attesi.	initByDefault()	Success

TU2	Premi.Editor.graphicObject	Si verifica che inizializzando un oggetto grafico a partire da un oggetto JSON il nuovo oggetto creato sia effettivamente un riferimento all'oggetto JSON passato.	initByJSON()	Success
TU3	Premi.Editor.graphicObject	Si verifica che ritorni loggetto JSON	getJSON()	Success
TU4	Premi.Editor.graphicObject	Si verifica che date due dimensioni valide l'oggetto venga ridimensionato correttamente.	resize()	Success
TU5	Premi.Editor.graphicObject	Dati due valori di posizionamento, si verifica che l'oggetto si trovi nella posizione attesa.	drag()	Success
TU6	Premi.Editor.Image	Si verifica che l'inizializzazione dei campi dati di default avvenga con i valori attesi.	initByDefault()	Success
TU7	Premi.Editor.Image	Si verifica che inizializzando un oggetto grafico a partire da un oggetto JSON il nuovo oggetto creato sia effettivamente un riferimento all'oggetto JSON passato.	initByJSON()	Success
TU8	Premi.Editor.Shape	Si verifica che l'inizializzazione dei campi dati di default avvenga con i valori attesi.	initByDefault()	Success

TU9	Premi.Editor.Shape	Si verifica che inizializzando un oggetto grafico a partire da un oggetto JSON il nuovo oggetto creato sia effettivamente un riferimento all'oggetto JSON passato.	initByJSON()	Success
TU10	Premi.Editor.Text	Si verifica che l'inizializzazione dei campi dati di default avvenga con i valori attesi.	initByDefault()	Success
TU11	Premi.Editor.Text	Si verifica che inizializzando un oggetto grafico a partire da un oggetto JSON il nuovo oggetto creato sia effettivamente un riferimento all'oggetto JSON passato.	initByJSON()	Success
TU12	Premi.Editor.Frame	Si verifica che l'inizializzazione dei campi dati di default avvenga con i valori attesi.	initByDefault()	Success
TU13	Premi.Editor.Frame	Si verifica che inizializzando un oggetto grafico a partire da un oggetto JSON il nuovo oggetto creato sia effettivamente un riferimento all'oggetto JSON passato.	initByJSON()	Success
TU14	Premi.Editor.Frame	Si verifica che valorizzi correttamente il compo selectedGO dell'oggetto in questione.	selectGO() deselectGO()	Success
TU15	Premi.Editor.Frame	Si verifica che l'oggetto venga aggiunto correttamente.	addGo()	Success

TU16	Premi.Editor.Frame	Si verifica che l'oggetto venga rimosso correttamente.	removeSelectedGO()	Success
TU17	Premi.Editor.Frame	Si verifica che date due dimensioni valide l'oggetto venga ridimensionato correttamente.	resizeGO()	Success
TU18	Premi.Editor.Frame	Dati due valori di posizionamento, si verifica che l'oggetto si trovi nella posizione attesa.	dragGO()	Success
TU19	Premi.Editor.Infographic	Si verifica che l'inizializzazione dei campi dati di default avvenga con i valori attesi.	initByDefault()	Success
TU20	Premi.Editor.Infographic	Si verifica che inizializzando un oggetto grafico a partire da un oggetto JSON il nuovo oggetto creato sia effettivamente un riferimento all'oggetto JSON passato.	initByJSON()	Success
TU21	Premi.Editor.Infographic	Si verifica che valorizzi correttamente il compo selectedGO dell'oggetto in questione.	selectGO() deselectGO()	Success
TU22	Premi.Editor.Infographic	Si verifica che l'oggetto venga aggiunto correttamente.	addGo()	Success
TU23	Premi.Editor.Infographic	Si verifica che l'oggetto venga rimosso correttamente.	removeSelectedGO()	Success
TU24	Premi.Editor.Infographic	Si verifica che date due dimensioni valide l'oggetto venga ridimensionato correttamente.	resizeGO()	Success

TU25	Premi.Editor.Infographic	Dati due valori di posizionamento, si verifica che l'oggetto si trovi nella posizione attesa.	dragGO()	Success
TU26	Premi.Editor.-FrameEditorController	Si controlla che a seconda che dell'oggetto passato alla funzione venga effettivamente aggiunto un oggetto di quel tipo	addGObject()	Success
TU27	Premi.Editor.-FrameEditorController	Si controlla che a seconda che dell'oggetto passato alla funzione venga effettivamente aggiunto un oggetto di quel tipo	addGObject()	Success
TU28	Premi.Editor.- InfographicEditorController	Si controlla che l'immagine venga effettivamente inserita nell'inforgrafica.	changeImage()	Success
TU29	Premi.Editor.- InfographicEditorController	Si verifica che l'infografica corrente venga salvata correttamente all'interno del database.	saveInfographic()	Success
TU30	Premi.Trail	Si verifica che il campo currentStep, che serve ad undentificare la posizione corrente all'interno del percorso di specializzazione, venga incrementato/decrementato.	nextSlide() prevSlide()	Success
TU31	Premi.Trail	Si verifica che l'oggetto restituito si riferisca effettivamente ad un checkpoint.	enterInCheckPoint() returnToCheckPoint()	Success

TU32	Premi.Trail	Si verifica che l'oggetto restituito si riferisca alla slide attesa (al checkpoint o alla slide con lo stesso id fornito in input al metodo)	goToSlide() returnToCheckPoint()	Success
TU33	Premi.Trail	Si verifica che il campo currentStep venga valorizzato correttamente, in modo tale da posizionare la slide in questione appena prima (o dopo) quella corrente all'interno dell'array.	insertSlideAfterCurrent() insertSlideBeforeCurrent()	Success
TU34	Premi.Trail	Si verifica che la slide corrente venga rimossa correttamente	removeCurrentSlide()	Success
TU35	Premi.UserManager()	Si controlla che i dati inseriti vengano effettivamente salvati nel database.	signUp()	Success
TU36	Premi.UserManager()	Si verifica che i dati inseriti vengano confrontati correttamente con quelli salvati nel database.	signIn()	Success
TU37	Premi.UserManager()	Si controlla che l'utente venga effettivamente reindirizzato alla pagina principale.	signOut()	Success
TU38	Premi.UserManager()	Si verifca che la password venga modificata correttamente	changePassword()	Success

Tabella 5: Tabella test di unità

B.4 Test di validazione

In questa sezione vengono descritti i test di validazione che servono per accertarsi che il prodotto realizzato sia conforme alle attese. Per ogni test vengono descritti i vari passi che un utente deve eseguire per testare i requisiti ad esso associati.

B.4.1 Test TV1

L'utente vuole verificare che ci si possa registrare al sistema Premi.
All'utente e` richiesto di:

- Aprire il sistema
- Navigare nell'area di registrazione
- Inserire un indirizzo email.
  All'utente è chiesto di:
  - Verificare che l'inserimento di un indirizzo email non valido generi un avviso da parte del sistema
  - Inserire un indirizzo email valido
- Inserire una password
- Reinserire la password per conferma
- Verificare che il completamento della registrazione al sistema vada a buon fine

B.4.2 Test TV2

L'utente vuole verificare il corretto funzionamento dell'autenticazione al sistema Premi
All'utente e` richiesto di:

- Eseguire la registrazione al sistema Premi eseguendo **TV1**.
- Accedere alla sezione di Login del sistema Premi
- Inserire un indirizzo email
  All'utente è chiesto di:
  - Verificare che l'inserimento di un indirizzo email non valido generi un avviso da parte del sistema
  - Provare l'inserimento di un indirizzo email diverso da quello fornito in fase di registrazione
- Inserire una password
  All'utente è chiesto di:
  - Provare l'inserimento di una password diversa da quella fornita in fase di registrazione
  - Testare la procedura guidata di cambio password (**TV2.1**)

- Verificare che in caso di indirizzo mail o password diversi da quelli forniti in fase di registrazione generi un messaggio di errore non facendo terminare a buon fine l'autenticazione.
- Verificare che l'autenticazione vada a buon fine e il sistema renda esplicito il successo di questa azione.

B.4.3 Test TV3

L'utente vuole testare la creazione di una nuova presentazione e successivamente la sua eliminazione
All'utente e` richiesto di:

- Autenticarsi al sistema Premi eseguendo **TV2**
- Creare una nuova presentazione
- Scegliere ed inserire un titolo per la nuova presentazione (**TV.3.1**)
- Scrivere una descrizione di almeno di almeno 15 parole per la presentazione (**TV.3.2**)
- Completare la procedura di creazione confermando i dati inseriti
- Constatare l'avvenuta creazione della presentazione
- Selezionare la presentazione
- Eliminare la presentazione selezionata
- Annullare l'operazione di eliminazione prima che questa avvenga effettivamente (**TV.3.3**)

B.4.4 Test TV4

L'utente vuole testare l'esecuzione di una presentazione
All'utente e` richiesto di:

- Autenticarsi al sistema Premi eseguendo **TV2**
- Selezionare una presentazione dall'elenco delle presentazioni (**TV4.1**)
- Scegliere un percorso per la presentazione tra quelli disponibili (**TV4.2**)
- Navigare nella presentazione:
  All'Utente è chiesto di:
  - Avanzare nel percorso presentativo un passo alla volta (**TV4.3**)
  - Retrocedere nel percorso presentativo un passo alla volta(**TV4.4**)
  - Seguire un percorso di approfondimento a partire da un checkpoint$_G$ (**TV4.5**)
  - Tornare ad un checkpoint$_G$ dopo aver completato un percorso di approfondimento (**TV.4.6**)
- Interrompere l'esecuzione della presentazione (**TV4.7**)

B.4.5 Test TV5

L'utente vuole testare la modifica di una presentazione e il salvataggio delle modifiche
All'utente e` richiesto di:

- Autenticarsi al sistema Premi eseguendo **TV2**
- Selezionare una presentazione dall'elenco delle presentazioni (**TV4.1**)
- Entrare nell'editor
- Inserire nuovi oggetti grafici nella presentazione (**TV5.1**)
  All'utente è chiesto di:
  - Provare l'inserimento di un'area di testo (**TV5.2**)
    All'utente è chiesto di:
    * Inserire un testo di almeno 10 parole (**TV5.2.1**)
    * Scegliere un font per il testo (**TV5.2.2**)
    * Scegliere un colore per il testo tra quelli disponibili (**TV5.2.3**)
    * Scegliere una dimensione diversa da quella di default per il testo (**TV5.2.4**)
  - Provare l'inserimento di un frame$_G$ (**TV5.3**) Fob4.1.2
    All'utente è chiesto di:
    * Scegliere una forma tra quelle disponibili per il frame$_G$ (**TV5.3.1**)
  - Provare l'inserimento di un'immagine (**TV5.4**)
    All'utente è chiesto di:
    * Scegliere un file immagine da filesystem (**TV5.4.1**)
  - Provare l'inserimento di uno shape$_G$ (**TV5.5**)
    All'utente è chiesto di:
    * Scegliere una forma tra quelle disponibili per lo shape$_G$ (**TV5.5.1**)
- Selezionare un oggetto grafico tra quelli precedentemente inseriti (**TV5.6**)
- Provare a modificare un oggetto grafico selezionato (**TV5.7**)
  All'utente è chiesto di:
  - Provare a modificare un frame$_G$ (**TV5.8**)
    All'utente è chiesto di:
    * Ridimensionare il frame$_G$ (**TV5.8.1**)
    * Riposizionare il frame$_G$ (**TV5.8.2**)
    * Modificare lo stile del frame$_G$ (**TV5.8.3**)
  - Provare a modificare un'area di testo (**TV5.9**)
    All'utente è chiesto di:
    * Ridimensionare l'area di testo (**TV5.9.1**)
    * Riposizionare l'area di testo (**TV5.9.2**)
    * Modificare lo stile dell'area di testo (**TV5.9.3**)
    * Modificare il contenuto dell'area di testo (**TV5.9.4**)

    - Cambiare il livello dell'area di testo (**TV5.9.5**)
  - Provare a modificare uno shape (**TV5.10**)
    All'utente è chiesto di:
    - Riposizionare lo shape$_G$ (**TV5.10.1**)
    - Ridimensionare lo shape$_G$ (**TV5.10.2**)
    - Modificare lo stile dello shape$_G$ (**TV5.10.3**)
    - Cambiare il livello dello shape$_G$ (**TV5.10.4**)
  - Provare a modificare un'immagine (**TV5.11**)
    All'utente è chiesto di:
    - Riposizionare l'immagine (**TV5.11.1**)
    - Ridimensionare l'immagine (**TV5.11.2**)
    - Cambiare il livello dell'immagine (**TV5.11.3**)
- Eliminare un oggetto grafico selezionato (**TV5.12**)
- Creare un nuovo percorso per la presentazione (**TV5.13**)
  All'utente è chiesto di:
  - Scegliere un titolo per il percorso creato (**TV5.14**)
  - Clonare un percorso esistente (**TV5.15**)
- Selezionare un percorso tra quelli disponibili (**TV5.16**)
- Modificare il percorso selezionato (**TV5.17**)
  All'utente è chiesto di:
  - Cambiare il titolo del percorso (**TV5.18**)
  - Aggiungere un passo al percorso (**TV5.19**)
  - Modificare l'ordine dei frame$_G$ del percorso(**TV5.20**)
  - Impostare un frame$_G$ come checkpoint$_G$ (**TV5.21**)
  - Rimuovere la marcatura a checkpoint$_G$ da un frame$_G$ (**TV5.22**)
    All'utente è chiesto di:
    - Verificare che si debba confermare l'azione intrapresa (**TV5.22.1**)
    - Verificare che si possa annullare l'azione intrapresa (**TV5.22.2**)
  - Selezionare un frame$_G$ del percorso del percorso (**TV5.23**)
- Eliminare il percorso selezionato (**TV5.24**)
- Modificare il titolo di una presentazione (**TV5.25**)
- Modificare la descrizione di una presentazione (**TV5.26**)

B.4.6 Test TV6

L'utente vuole verificare il corretto funzionamento dell'esportazione
All'utente e` richiesto di:

- Autenticarsi al sistema Premi eseguendo **TV2**
- Selezionare una presentazione dall'elenco delle presentazioni (**TV4.1**)
- Esportare la presentazione come poster
  All'utente è chiesto di:
  - Scegliere uno dei formati proposti dal sistema per l'esportazione (**TV6.1**)
- Esportare una presentazione in formato portabile
  All'utente è chiesto di:
  - Verificare che il formato prodotto dal sistema sia portabile e quindi eseguibile offline (**TV6.2**)

B.4.7 Test TV7

L'utente vuole verificare il corretto funzionamento della pubblicazione di una presentazione
All'utente e` richiesto di:

- Autenticarsi al sistema Premi eseguendo **TV2**
- Selezionare una presentazione dall'elenco delle presentazioni (**TV4.1**)
- Pubblicare la presentazione
- Verificare che il sistema generi un link alla presentazione e lo renda noto all'utente (**TV7.1**)

B.4.8 Tracciamento Test di Validazione - Requisiti

Requisito	Test di Validazione
Fob13	TV1
Fob14	TV2
FOb16	TV2.1
FOb1	TV3
FOb1.1	TV3.1
FOb1.2	TV3.2
FOb6	TV3
FOb6.1	TV3.3
FOb3	TV4
FOb2	TV4.1
FDe3.1	TV4.2
FOb3.2	TV4.3
FOb3.3	TV4.4
FDe3.4	TV4.5
FOb3.5	TV4.6
FOb3.6	TV4.7
FOb4	TV5
FOb4.1	TV5.1
FOb4.1.1	TV5.2
FOb4.1.1.1	TV5.2.1
FOb4.1.1.2	TV5.2.2
FOb4.1.1.3	TV5.2.3
FOb4.1.1.4	TV5.2.4
FOb4.1.2	TV5.3
FOb4.1.2.1	TV5.3.1
FOb4.1.3	TV5.4
FOb4.1.3.1	TV5.4.1
FOb4.1.4	TV5.5
FOb4.1.4.1	TV5.5.1
FOb4.2	TV5.6
FOb4.3	TV5.7
FOb4.3.1	TV5.8
FOb4.3.1.1	TV5.8.1
FOb4.3.1.2	TV5.8.2
FOb4.3.1.3	TV5.8.3
FOb4.3.3	TV5.9
FOb4.3.3.1	TV5.9.1
FOb4.3.3.2	TV5.9.2
FOb4.3.3.3	TV5.9.3
FOb4.3.3.4	TV5.9.4
FOb4.3.3.5	TV5.9.5
FOb4.3.4	TV5.10
FOb4.3.4.1	TV5.10.1

FOb4.3.4.2	TV5.10.2
FOb4.3.4.3	TV5.10.3
FOb4.3.4.4	TV5.10.4
FOb4.3.2	TV5.11
FOb4.3.2.1	TV5.11.1
FOb4.3.2.2	TV5.11.2
FOb4.3.2.3	TV5.11.3
FOb4.4	TV5.12
FDe4.5	TV5.13
FDe4.5.1	TV5.14
FDe4.5.2	TV5.15
FDe4.6	TV5.16
FOb4.7	TV5.17
FDe4.7.1	TV5.18
FOb4.7.2	TV5.19
FOb4.7.3	TV5.20
FDe4.7.4	TV5.21
FDe4.7.5	TV5.22
FDe4.7.5.1	TV5.22.1
FDe4.7.5.2	TV5.22.2
FOb4.7.6	TV5.23
FDe4.9	TV5.24
FOb4.8	TV5.25
FOb4.10	TV5.26
FOb11	TV6
FOb11.1	TV6.1
FOb11.2	TV6.2
FOb7	TV7
FOb8	TV7.1

B.5 Copertura dei test

La copertura del codice relativa ai test effettuati è pari al 73.3% del codice prodotto, come premesso all'inizio del capitolo il team 404NotFound ha scelto di implementare i test solo per tutti i metodi ritenuti fondamentali alla logica del programma e non banali.

B.6 Analisi delle misurazioni sul codice

Di seguito vengono riportati i risultati delle misurazioni effettuate tramite il plugin Plato$_G$. In particolare le misurazioni riguardano la manutenibilità del codice prodotto.

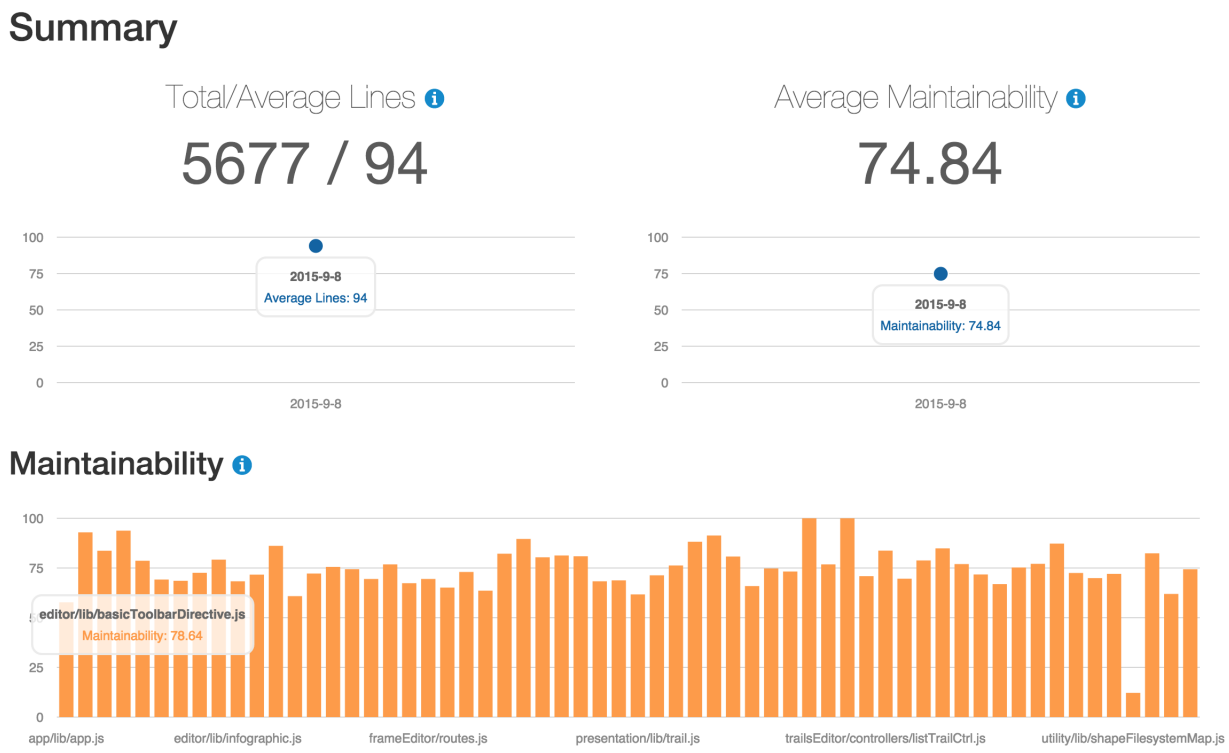

Figura 13: Stima della Manutenibilità

Si nota come la manutenibilità risulti essere al di sopra del valore atteso di 70 punti su 100. Di seguito sono riportati i risultati nel dettaglio degli indici SLOC (Source Line Of Codes), Complessità ciclomatica e indice LINT che misura quanto il codice analizzato si discosta dai costrutti noti e dalle best practice del linguaggio Javascript, oltre agli errori stimati essere possibilmente presenti nel codice sulla base della complessità calcolata.

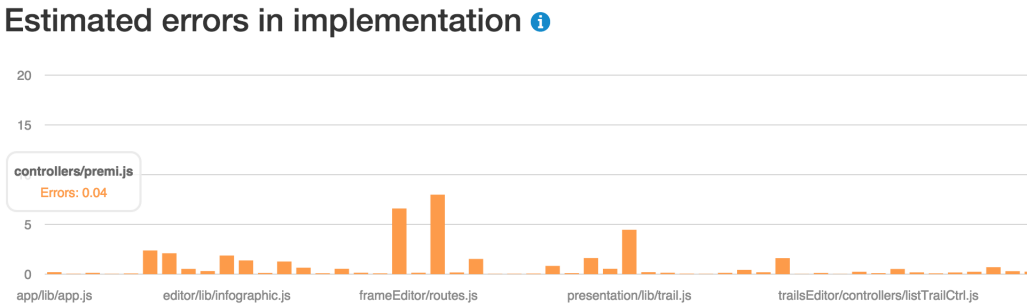

Figura 14: Errori possibilmente stimati nel codice

Anche in questo caso i valori rilevati restano al di sotto di 10 in linea con il range di accettabilità.

trailMap/controller/trailMapCtrl.js	complexity 13 sloc 191 est errors 1.61 lint errors 2
presentation/lib/order.js	complexity 12 sloc 146 est errors 1.62 lint errors 6
editor/lib/interactInit.js	complexity 10 sloc 165 est errors 1.38 lint errors 3
lib/fixMaterilizeModal.js	complexity 10 sloc 182 est errors 1.53 lint errors 0
editor/lib/saver.js	complexity 9 sloc 134 est errors 1.27 lint errors 12
presentation/lib/databaseAPI.js	complexity 9 sloc 110 est errors 0.83 lint errors 5
editor/lib/graphicObject.js	complexity 7 sloc 100 est errors 0.53 lint errors 0
userManager/controllers/signupCtrl.js	complexity 6 sloc 53 est errors 0.37 lint errors 0
editor/lib/shape.js	complexity 5 sloc 86 est errors 0.65 lint errors 3
trailsEditor/controllers/trailsEditorCtrl.js	complexity 5 sloc 43 est errors 0.24 lint errors 1
ager/controllers/changePasswordCtrl.js	complexity 5 sloc 47 est errors 0.30 lint errors 0
editor/lib/text.js	complexity 4 sloc 75 est errors 0.54 lint errors 4
presentation/lib/signalCtrl.js	complexity 4 sloc 75 est errors 0.54 lint errors 4
userManager/controllers/signinCtrl.js	complexity 4 sloc 45 est errors 0.25 lint errors 0
editor/controllers/basicToolbarCtrl.js	complexity 3 sloc 18 est errors 0.13 lint errors 1
presentation/lib/filter.js	complexity 3 sloc 13 est errors 0.10 lint errors 0
ager/controllers/editPresentationCtrl.js	complexity 3 sloc 39 est errors 0.20 lint errors 2
routes.js	complexity 3 sloc 33 est errors 0.19 lint errors 1

trailsEditor/controllers/editTrailCtrl.js	complexity 3 sloc 28 est errors 0.24 lint errors 3
viewer/routes.js	complexity 3 sloc 62 est errors 0.48 lint errors 2
editor/lib/image.js	complexity 2 sloc 48 est errors 0.31 lint errors 2
ager/controllers/newPresentationCtrl.js	complexity 2 sloc 32 est errors 0.14 lint errors 3
r/controllers/removePresentationCtrl.js	complexity 2 sloc 26 est errors 0.13 lint errors 2
trailsEditor/controllers/modTrailCtrl.js	complexity 2 sloc 47 est errors 0.52 lint errors 1
trailsEditor/controllers/newTrailCtrl.js	complexity 2 sloc 22 est errors 0.18 lint errors 3
ailsEditor/controllers/removeTrailCtrl.js	complexity 2 sloc 19 est errors 0.17 lint errors 2
app/lib/app.js	complexity 1 sloc 64 est errors 0.20 lint errors 0
controllers/premi.js	complexity 1 sloc 8 est errors 0.04 lint errors 0
editor/controllers/editorCtrl.js	complexity 1 sloc 5 est errors 0.03 lint errors 0
editor/lib/basicToolbarDirective.js	complexity 1 sloc 13 est errors 0.07 lint errors 0
editor/lib/observer.js	complexity 1 sloc 24 est errors 0.12 lint errors 2
editor/lib/shapeMenuDirective.js	complexity 1 sloc 17 est errors 0.09 lint errors 0
editor/routes.js	complexity 1 sloc 31 est errors 0.14 lint errors 0
allback/lib/fallbackMessageDirective.js	complexity 1 sloc 14 est errors 0.08 lint errors 0
frameEditor/routes.js	complexity 1 sloc 27 est errors 0.14 lint errors 0
infographicEditor/routes.js	complexity 1 sloc 34 est errors 0.17 lint errors 0
lib/standardFrameResolution.js	complexity 1 sloc 9 est errors 0.03 lint errors 0

Figura 15: Dettaglio della stima di Manutenibiltà

Tutti i valori calcolati rientrano nell'intervallo di valori accettabili imposti dalle metriche, sebbene la complessità cicolmatica in qualche caso sfugga al range ottimale superando il limite 10 essa rimane sempre all'interno del range di accettabilità preposto dal gruppo 404NotFound.

C Resoconto dell'Attività di Verifica

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

C.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 frecce scorretta;
- controllo ortografico.

C.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.

C.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.

C.4 Revisione dei Requisiti

Nel periodo antecedente la consegna di tale revisione sono stati verificati i documenti ed i processi. L'analisi statica e` stata applicata secondo i criteri e le modalità indicate nella sezione 2.5.2. Effettuando walkthrough sono stati riscontrati degli errori. Sono state quindi avviate le procedure per la segnalazione e la correzione, descritte nell'apposita sezione delle *NormeDiProgetto_v4.0.pdf*. Noti gli errori, si e` provveduto a:

- Correggere le imperfezioni rilevate;
- Segnalare gli errori più frequenti.Si e` quindi applicato il ciclo PDCA per rendere più efficiente ed efficace il processo di verifica.

E` stata in seguito applicata l'inspection utilizzando la lista di controllo stilata durante la verifica dei documenti precedentemente verificati, ponendo particolare attenzione ai grafici dei casi d'uso. Il tracciamento (requisiti - fonti, use-case - requisiti) e` stato effettuato tramite l'applicativo 404TrackerDB.

C.5 Revisione di Progettazione

In seguito alle osservazioni effettuate dal committente in sede di RR, il gruppo 404NotFound si è riunito per analizzare l'esito e correggere gli errori rilevati con lo scopo di presentare una versione migliorativa dei vari documenti.

Le principali e più importanti correzioni svolte sono:

- **Analisi dei requisiti**:
  - Ristesura dei requisiti secondo le correzioni del committente
  - Ristesura dei casi d'uso secondo le correzioni del committente
  - Tracciamento ex novo dei Requisiti
- **Piano di Qualifica**:
  - Rimossa sezione risorse e spostata nel documento *NormeDiProgetto_v4.0.pdf*
  - Correzione della sezione 2.3 e introduzione del riferimento alle *Norme di Progetto*
  - Stesura appendice A: Pianificazione dei Test
  - Descrizione Test di Sistema, Test di Integrazione e Test di Validazione
  - Resoconto Revisione di Progettazione
- **Norme di Progetto**:
  - Riordinate le sezioni per ambito
  - Aggiunte sezioni derivate da altri documenti
- **Piano di Progetto**:
  - Cambiate le date di consegna
  - Adattati i periodi di durata delle attività
  - Ricalcolo dei consuntivi
  - Spostate sezioni indicate nelle *Norme di Progetto*
- **Specifica Tecnica**:
  - Stesura Integrale del documento
- **Glossario**:
  - Cambiamento della struttura del documento
  - Aggiunta dell'indice
  - Rimozione delle lettere senza definizioni

C.6 Revisione di Qualifica

In seguito alle osservazioni effettuate dal committente in sede di RP, il gruppo 404NotFound si è riunito per analizzare gli esiti e correggere gli errori rilevati con lo scopo di presentare una versione migliorativa dei vari documenti. Le correzioni e le aggiunte svolte sono:

- **Analisi dei requisiti**:
  - Modificati alcuni casi d'uso: spostati sottocasi nel caso principale
- **Piano di Qualifica**:
  - Aggiunta sezione introduttiva alla strategia di verifica
  - Riorganizzato per struttura e contenuti l'intero documento, in particolare il capitolo 3.
  - Ampliati sottocapitoli della Pianificazione Strategica e Temporale, aggiunta dei diagrammi di flusso.
  - Approfondito l'uso delle metriche adottate per rendere quantitativi gli obiettivi di qualità descritti.
  - Aggiunti in appendice i test di unità.
- **Norme di Progetto**:
  - Aggiunta sottosezione Gestione di Progetto
  - Aggiunta sottosezione Progettazione di Dettaglio
  - Aggiunta sottosezione Codifica
  - Tolta la sezione degli strumenti di lavoro, e spostati gli strumenti nelle sezioni delle attività in cui vengono usati
- **Piano di Progetto**:
  - Modificate le tabelle delle attività delle macro-fasi di Progettazione di Dettaglio e Codifica, e Verifica e Validazione, per adeguarsi alle nuove date di consegna delle Revisioni
  - Aggiunto il Consuntivo e il Preventivo a Finire della macro-fase di Progettazione di Dettaglio e Codifica
- **Specifica Tecnica**:
  - Aggiunte alcune descrizioni sulle tecnologie utilizzate che prima erano solamente citate
  - Modificati quasi interamente i package per rispecchiare le modifiche apportate all'architettura del sistema
  - Rinominati alcuni componenti che avevano lo stesso nome
  - Apportate alcune correzioni ai Diagrammi di Attività
- **Definizione di Prodotto**:

  - Stesura integrale del documento.

- **Manuale Utente**:

  - Stesura integrale del documento.

- **Glossario**:

  - Aggiunte voci mancanti e nuove voci.

C.7 Revisione di Accettazione

In seguito alle osservazioni effettuate dal committente in sede di RPQ il gruppo 404NotFound si è riunito per analizzare gli esiti e correggere gli errori rilevati con lo scopo di presentare una versione migliorativa dei vari documenti. Le correzioni e le aggiunte svolte sono:

- **Analisi dei requisiti**:

  -

- **Piano di Qualifica**:

  - Mantenuto il capitolo 2 nella forma attuale ma spostato come suggerito in appendice A al documento.
  - Rivisto e corretto capito 3.1 (ora 2.1) per struttura e contenuti, incrementato anche il capitolo Misure e Metriche.
  - Ampliata appendice B con copertura dei test effettuati e resoconto delle misurazioni effettuate sul codice.

- **Norme di Progetto**:

  -

- **Piano di Progetto**:

  - Aggiunto il Consuntivo e il Preventivo a Finire della macro-fase di Verifica Finale e Validazione

- **Specifica Tecnica**:

  - Revisione del tracciamento Requisiti/Componenti.
  - Apportate correzioni ai package.

- **Definizione di Prodotto**:

  - Apportate correzioni ai package.

- **Manuale Utente**:

  -

- **Manuale di installazione**:
  - Stesura Integrale del documento.
- **Glossario**:
  - Aggiunte voci mancanti e nuove voci.