MaaP: MongoDB as an admin Platform

Norme di Progetto

Versione	3.2.0
Data creazione	2013-11-20
Data ultima modifica	2014-01-24
Stato del Documento	Formale
Uso del Documento	Interno
Redazione	Alberto Garbui
Verifica	Alessandro Benetti
	Fabio Miotto
Approvazione	Fabio Miotto
Distribuzione	Aperture Software

Sommario

Questo documento si propone di presentare le norme che il gruppo Aperture Software ha stabilito per la realizzazione del prodotto MaaP: MongoDB as an admin Platform.

Diario delle modifiche

Versione	Data	Autore	Modifiche effettuate
3.2.0	2014-01-24	Fabio Miotto (RE)	Approvazione documento.
3.1.1	2014-01-23	Alessandro Benetti (VE)	Verifica documento.
3.1.0	2014-01-21	Fabio Miotto (VR)	Verifica documento.
3.0.2	2014-01-15	Alberto Garbui (AM)	Incremento documento.
3.0.1	2014-01-13	Alberto Garbui (AM)	Effettuate correzioni segnalate dal Committente.
2.2.0	2014-01-04	Alberto Garbui (RE)	Approvazione documento.
2.1.0	2014-01-03	Giacomo Pinato (VR)	Verifica documento.
2.0.2	2013-12-27	Fabio Miotto (AM)	Sezione Progettazione, Riorganizzazione.
2.0.1	2013-12-23	Fabio Miotto (AM)	Modifica sezione Comunicazioni.
1.2.0	2013-11-27	Andrea Perin (RE)	Approvazione documento.
1.1.0	2013-11-26	Alessandro Benetti (VR)	Verifica documento.
1.0.2	2013-11-25	Giacomo Pinato (RE)	Ampliamento documento.
1.0.1	2013-11-20	Andrea Perin (RE)	Creazione documento.

Tabella 1: Registro delle modifiche

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento è volto a definire le norme che dovranno essere osservate da tutti i componenti del team per l'intera durata del progetto. Tali norme sono volte a garantire la qualità finale del prodotto attraverso la rigida regolamentazione dei processi e delle strategie di produzione. In queste pagine vengono delineate le linee guida e le norme per tutte le *attività*$_G$ che concorreranno allo sviluppo del *software*$_G$, della documentazione e del progetto in generale.

1.2 Scopo del prodotto

Lo scopo del prodotto è produrre un *framework*$_G$ per generare interfacce web di amministrazione dei dati di business basato sullo stack *Node.js*$_G$ e *MongoDB*$_G$.
L'obiettivo è quello di semplificare il lavoro allo *sviluppatore*$_G$ che dovrà rispondere in modo rapido e standard alle richieste degli esperti di *business*$_G$.

1.3 Glossario

Al fine di evitare ogni ambiguità nella comprensione del linguaggio utilizzato nel presente documento e, in generale, nella documentazione fornita dal gruppo Aperture Software, ogni termine tecnico, di difficile comprensione o di necessario approfondimento verrà inserito nel documento *Glossario_v3.2.0.pdf*.
Saranno in esso definiti e descritti tutti i termini in corsivo e allo stesso tempo marcati da una lettera "G" maiuscola in pedice nella documentazione fornita.

1.4 Riferimenti

1.4.1 Informativi

- **Slide dell'insegnamento Ingegneria del Software modulo A**:
  Ingegneria dei requisiti: http://www.math.unipd.it/~tullio/IS-1/2013/;
- **Software Engineering - Ian Sommerville - 9th Edition (2010)**;
- **Standard ISO 8601** - Data elements and interchange formats: http://it.wikipedia.org/wiki/ISO_8601
- **Jenkins:** http://en.wikipedia.org/wiki/Jenkins_(software);
- **Piano di Qualifica**: *Piano_di_Qualifica_v3.2.0*;
- **Piano di Progetto**: *Piano_di_Progetto_v3.2.0*.

2 Comunicazioni

2.1 Comunicazioni interne

Le comunicazioni interne sono divise in comunicazioni formali e informali, in base alla natura della comunicazione stessa.

2.1.1 Comunicazioni interne formali

Le comunicazioni interne formali appartengono all'area di organizzazione generale del gruppo. A questa categoria appartengono tutti gli avvisi di impegni collettivi del team, la segnalazione di imprevisti organizzativi o la convocazione di riunioni.

2.1.1.1 Mailing List

È stata creata una mailing list all'indirizzo aperture-team@googlegroups.com. Ogni *email*$_G$ inviata a tale indirizzo verrà inoltrata alla email personale di ogni *componente*$_G$ nel gruppo. La mailing list è collegata ad un gruppo su *Google*$_G$ Groups, che permette di velocizzare le comunicazioni implementando una *simil-chat*$_G$ tramite email distribuite, inoltre mantiene uno storico di qualsiasi comunicazione. Va usato quindi come strumento per discussioni e comunicazioni ufficiali. Una mail usata per la comunicazione interna deve avere la seguente struttura:

- **Destinatario:** l'unico indirizzo possibile è aperture-team@googlegroups.com;
- **Mittente:** è l'indirizzo email personale di chi scrive il messaggio;
- **Oggetto:** deve essere una sintesi del contenuto della email;
- **Corpo:** contiene una descrizione completa di tutte le informazioni che si vogliono comunicare; esse devono essere facilmente capibili e possibilmente strutturate in maniera che siano leggibili; inoltre le frasi devono essere corte e contenere termini di linguaggio comune;
- **Allegati:** si possono, qualora se ne ritenga necessario, usare degli allegati, per completare l'informazione che si vuole spedire. Un esempio utile potrebbe essere quello di allegare un verbale o un resoconto di incontri con il Proponente o con il Committente.

2.1.2 Comunicazioni interne informali

Le comunicazioni riguardanti dubbi o chiarimenti marginali o comunque non inerenti all'organizzazione del gruppo, vengono definiti informali. Per loro natura, queste comunicazioni devono essere veloci e non complesse, quindi abbiamo deciso di utilizzare strumenti immediati di comunicazione, quali le *chat*$_G$. In questo caso, un membro del gruppo esprime il dubbio come messaggio istantaneo, il quale sarà visibile da ogni altro componente.

2.1.2.1 *Skype*$_G$

Per facilitare le comunicazioni istantanee è stata creata su Skype una chat di gruppo dove sono presenti tutti i membri del team. Tale chat deve essere usata solo per conversazioni non ufficiali o per discussioni non importanti come lo scambio di articoli, consigli o comunicazioni non rilevanti.

2.1.2.2 Gruppo *Facebook*$_G$

Data la grande diffusione di questo *social network*$_G$ e l'aderenza di tutti i membri del gruppo ad esso, abbiamo deciso di creare un gruppo apposito su questa piattaforma. Facebook fornisce un accesso ancora più immediato di Skype, data la sua diffusione su dispositivi mobile.

2.2 Comunicazioni esterne

2.2.1 Email

La email ufficiale del gruppo è ApertureSWE@gmail.com.
Tale email può essere usata soltanto dal Responsabile di Progetto e verrà utilizzata per tutte le comunicazioni che il gruppo terrà con l'*ambiente*$_G$ esterno. Una mail usata per la comunicazione interna deve avere la seguente struttura:

- **Destinatario:** l'indirizzo email può essere quello del Proponente oppure quello del Committente, che può essere o il Prof. Vardanega Tullio o il Prof. Cardin Riccardo;
- **Mittente:** l'unico indirizzo possibile è ApertureSWE@gmail.com e deve essere usato solo dal Responsabile di Progetto;
- **Oggetto:** deve essere una sintesi della contenuto della email;
- **Corpo:** contiene una descrizione completa di tutte le informazioni che si vogliono comunicare; esse devono essere facilmente capibili e possibilmente strutturate in maniera che siano leggibili; inoltre le frasi devono essere corte e contenere termini specifici dell'ambiente *business*$_G$ inerenti al progetto;
- **Allegati:** si possono, qualora se ne ritenga necessario, usare degli allegati, per completare l'informazione che si vuole spedire. Un esempio utile potrebbe essere quello di allegare un verbale o un resoconto di incontri con il Proponente o con il Committente.

2.3 Riunioni

2.3.1 Richiesta

Ogni membro del team può richiedere che venga organizzata una riunione al Responsabile che, una volta verificata la motivazione della richiesta, accoglie o meno la stessa. Il Responsabile di Progetto ha la facoltà di indire riunioni qualora sentisse la necessità di farlo. La riunione deve essere segnalata sul calendario di gruppo con almeno due giorni di anticipo e non deve andare a sovrapporsi con impegni precedenti di altri componenti, a meno che la loro presenza non possa essere esclusa. Deve inoltre essere inviata una email di promemoria a tutti i membri del gruppo indicante giorno, ora e motivazione della riunione.

2.3.2 Svolgimento

Alle riunioni è gradita, ma non richiesta, la partecipazione di tutti i membri del gruppo. È giustificata l'assenza nel caso in cui la riunione riguardi temi non inerenti al ruolo di progetto che si sta ricoprendo o nel caso di impegni validi, giustificabili e improrogabili.

2.3.3 Verbale

Ad ogni riunione verrà eletto un segretario che avrà il compito di annotare gli argomenti di discussione e le decisioni prese durante la stessa. Dovrà inoltre redigere un verbale che verrà ufficialmente raccolto e archiviato come documentazione interna entro tre giorni dalla data della riunione. La struttura del Verbale deve contenere i seguenti elementi:

- Data;
- Luogo;
- Ora;
- Durata;
- Partecipanti interni;
- Partecipanti esterni;
- Motivazione della riunione/incontro;
- Argomenti trattati.

2.4 Comunicazioni con i Committenti e Proponenti

2.4.1 Prerequisiti

Prima di richiedere un colloquio personale con il Proponente e/o Committente il team deve preparare un documento che riassume i punti che verranno discussi contenente argomenti, dubbi e domande da porre.

2.4.2 Richiesta di colloquio

I colloqui con i committenti di progetto possono essere richieste dal Responsabile mediante la email ufficiale del team. Tutti i membri devono essere avvisati prima di richiedere un incontro con il Committente.

2.4.3 Svolgimento del colloquio

Al fine di ottimizzare l'estrapolazione delle informazioni derivanti dal colloquio, durante tutto lo svolgimento dello stesso dovrà essere registrata una traccia audio come supporto alla stesura del verbale.

2.4.4 Verbale

Alla fine del colloquio verrà redatto un verbale, vedi sezione 2.3.3 di questo documento, che riassume gli argomenti trattati, le conclusioni, nonché le strategie che si sono delineate in concordanza col Committente/Proponente.

3 Documentazione

3.1 Template

Viene fornito un *template*$_G$ in LaTeX per la Realizzazione della documentazione, sia interna che esterna, che i membri del gruppo dovranno seguire nella stesura dei documenti.

3.2 Struttura del documento

3.2.1 Header

Ogni documento ha un *header*$_G$ presente in ogni pagina che riporta logo e nome del gruppo sulla sinistra.

3.2.2 Footer

Ogni documento ha un footer presente in ogni pagina che riporta il nome e la *versione*$_G$ del documento a sinistra e il numero della pagina a destra.

3.2.3 Prima pagina

La prima pagina di ogni documento deve riportare nel seguente ordine:

- Il logo esteso del gruppo;
- Il nome del progetto;
- Il nome del documento;
- Informazioni sul documento (versione, data creazione, data ultima modifica, stato del documento, uso del documento, i redattori del documento, i Verificatori, chi ha approvato il documento e la *distribuzione*$_G$ del documento);
- Breve sommario del documento.

3.2.4 Seconda pagina

La seconda pagina deve riportare il diario delle modifiche apportate al documento dalla sua creazione fino alla versione corrente, strutturato nella seguente maniera:

Versione	Data	Autore	Modifiche effettuate

Le modifiche dovranno essere ordinate cronologicamente dalla più alla meno recente.

3.2.5 Terza pagina

La terza pagina deve riportare l'indice del documento.

3.3 Tabelle

Ogni tabella inserita deve essere descritta da una didascalia, in cui compare un numero identificativo incrementale per la tracciabilità.

3.4 Immagini

Ogni immagine inserita deve essere descritta da una didascalia, in cui compare un numero identificativo incrementale per la tracciabilità.

3.5 Norme tipografiche e stili di testo

Nella stesura della documentazione si dovranno seguire le seguenti indicazioni:

- I documenti dovranno essere grammaticalmente e sintatticamente corretti e scritti in modo fluido;
- Elenco puntato termina con il punto e virgola oppure con il punto se è l'ultimo elemento;
- Un carattere di punteggiatura non deve mai seguire un carattere di spaziatura;
- Il testo racchiuso tra parentesi non deve aprirsi o chiudersi con un carattere di spaziatura e non deve terminare con un carattere di punteggiatura;
- Le lettere maiuscole vanno poste solo dopo il punto, il punto di domanda, il punto esclamativo e all'inizio di ogni elemento di un elenco puntato, oltre che dove previsto dalla lingua italiana. È inoltre utilizzata l'iniziale maiuscola nel nome del team, del progetto, dei documenti, dei ruoli di progetto, delle fasi di lavoro e nelle parole Proponente e Committente;
- Nel caso in cui ci si riferisca ad un documento, il titolo di quest'ultimo dovrà essere scritto in corsivo e si dovrà riportare la versione riferita;
- I ruoli di progetto (es. Analista) dovranno essere scritti con la lettera iniziale maiuscola;
- Gli acronimi dovranno essere scritti in lettere maiuscole;
- È preferibile usare la forma attiva a quella passiva;
- Quando possibile usare elenchi puntati invece che frasi;
- Usare termini specifici e segnare i termini del glossario in corsivo e con la G in pedice;
- Dividere i documenti in sezioni e sottosezioni titolate;
- Le date dovranno essere espresse nella forma AAAA-MM-GG secondo lo standard ISO_G 8601:2004;
- Le attività (es. Verifica) vanno scritte con la lettera iniziale maiuscola;
- Gli elenchi puntati con un primo livello di profondità sono formati da dei pallini neri pieni, tranne quando si deve elencare una sequenza numerata di istruzioni da fare in un ordine stabilito, allora in quel caso si una un elenco numerato;
- Gli elenchi puntati con un secondo livello di profondità hanno un trattino.

3.5.1 Sigle e abbreviazioni

Le sigle e le abbreviazioni dovranno essere utilizzate solo in contesti in cui lo spazio è limitato come tabelle e diagrammi. Sono previste le seguenti abbreviazioni:

- AdR = Analisi dei Requisiti;
- GL = Glossario;
- NdP = Norme di Progetto;
- PdP = Piano di Progetto;
- PdQ = Piano di Qualifica;
- SdF = Studio di Fattibilità;
- ST = Specifica Tecnica;
- RA = Revisione di Accettazione;
- RP = Revisione di Progettazione;
- RQ = Revisione di Qualifica;
- RR = Revisione dei Requisiti;
- AS = Aperture Software;
- DP = *Design Pattern*$_G$.

Per i ruoli si useranno le seguenti abbreviazioni:

- AN=Analista;
- VR=Verificatore;
- RE=Responsabile;
- AM=Amministratore;
- PT=Progettista;
- PR=Programmatore.

3.6 Versionamento

Il *versionamento*$_G$ verrà effettuato sia sul *codice*$_G$ che sui documenti prodotti dal gruppo al fine di differenziare e rendere immediatamente riconoscibile la fase di sviluppo in cui ci si trova attualmente, mantenendo uno storico organizzato dei cambiamenti effettuati.

3.6.1 Regole generali

- Il numero di versionamento deve essere nella forma X.Y.Z, con X,Y e Z numeri interi non negativi. Tutti gli elementi devono salire numericamente di una unità alla volta;
- Ogni qualvolta che una versione viene rilasciata non può più essere effettuato alcun cambiamento ad essa. Ogni modifica sarà inserita nella versione successiva.

3.6.2 Versionamento del software

Nel versionare il software, le tre cifre di versionamento verranno modificate in base ai seguenti parametri:

- La prima cifra decimale verrà aumentata nel caso in cui vengano introdotti cambiamenti non retro compatibili al *framework*$_G$. Possono essere inclusi cambiamenti minori. Nel caso in cui la prima cifra venga aumentata la seconda e la terza ripartono da 0;
- La seconda cifra decimale verrà aumentata nel caso in cui vengano rilasciate nuove funzionalità retro compatibili. Deve necessariamente essere aumentata se una qualsiasi funzionalità pubblica del framework viene marcata deprecata. È possibile aumentare la seconda cifra anche in caso di rilascio di nuove funzionalità o miglioramenti sostanziali. Al cambiamento della seconda cifra la terza deve ripartire da 0;
- La terza cifra decimale verrà aumentata nel caso di correzione di errori o altri piccoli cambiamenti retro compatibili;
- La cifra X a 0 è riservata per lo sviluppo iniziale. Le versioni con X a 0 non sono da considerarsi stabili;
- La versione 1.0.0 definisce la prima versione stabile del framework.

3.6.3 Versionamento dei documenti

Nel versionare i documenti, le tre cifre di versionamento verranno modificate in base alle seguenti regole:

- X: indica il numero di uscite formali del documento, diviso come segue:
  1. Fase di Analisi, si estende fino alla Revisione dei Requisiti;
  2. Fase di Analisi in Dettaglio, si estende fino all'ingresso nella fase di Progettazione;
  3. Fase di Progettazione Architetturale, si estende fino alla Revisione di Progettazione;
  4. Fase di Progettazione di Dettaglio e Codifica,si estende fino alla Revisione di Qualifica;
  5. Fase di Verifica e Validazione, si estende fino alla Revisione di Accettazione e alla fine del progetto.
- Y: indica la fase di sviluppo del documento e varia come segue:
  0. Fase di stesura del documento, dove il documento è ancora modificabile;

1. Fase di Verifica del documento, dove il documento non è più modificabile e sta venendo controllato dal Verificatore;
2. Documento ultimato e approvato.

Sarà compito del Responsabile impostare la versione a X.1.0 nel momento in cui assegnerà il *ticket*$_G$ di controllo ai Verificatori e impostare la versione a X.2.0 nel caso in cui il documento risulti approvato dagli stessi.

- Z: indica il numero di modifiche minori apportare al documento. Aumenta al termine di ogni sessione di lavoro sul documento. Non ha un limite massimo.

3.6.4 Ciclo di vita

Ogni documento prodotto attraversa uno specifico percorso di vita, riassunto dai seguenti 4 stati:

1. **Creazione:** viene creata la struttura base del documento senza alcun contenuto;
2. **Lavorazione in corso:** il contenuto del documento viene scritto, ed è soggetto a continue modifiche;
3. **Verifica:** il contenuto del documento viene completato e viene assegnato ai Verificatori che dovranno svolgere la verifica del documento;
4. **Approvazione:** per ogni documento che è stato verificato, il cui esito della Verifica è stato soddisfacente, viene inviato al Responsabile di Progetto, che lo approva. Passato questo ultimo stato, il documento si trova nello stato finale per quella specifica versione.

Ciascun documento può trovarsi in uno stesso stato più volte: quando il documento approvato necessita di una revisione con il Committente, il documento ricomincia il *ciclo di vita*$_G$ che al completamento porterà ad una nuova versione incrementata.

3.7 Glossario

Il documento Glossario contiene le definizioni di tutti i termini di difficile comprensione. La difficoltà di questi termini sta nella loro natura di linguaggi tecnici o dall'uso non comune.
Per ogni temine definito nel Glossario, la sua prima occorrenza in ogni documento è marcata in corsivo e con una G in pedice. Ad esempio: *termine*$_G$.

3.7.1 Inserimento termine

Ogniqualvolta un membro del gruppo incontra un termine di questo tipo, durante la stesura di qualsiasi documento, deve seguire la seguente *procedura*$_G$:

1. Controllare che il termine non sia già presente all'interno del Glossario;
2. Nel caso non fosse presente, interrompere la stesura del documento corrente per inserire il termine nel Glossario;
3. Collocare il termine rispettando l'ordine alfabetico delle definizioni;
4. Dare una definizione chiara e concisa del termine;
5. Nel caso in cui la definizione del termine corrente causi la definizione di altri termini, bisogna tenerne traccia e definirli in ordine;
6. Aggiornare la tabella delle modifiche all'interno del documento Glossario, modificando conseguentemente il numero di versione del documento stesso.

3.7.2 Norme stilistiche

Di seguito vengono elencate le norme che dovranno essere seguite durante la stesura di una definizione:

- Reperire la definizione da una fonte autorevole, oppure ottenerla da più fonti;
- Non superare le 400 lettere per definizione;
- Preferibilmente, iniziare la definizione con un sostantivo;
- Non iniziare mai una definizione con " è un ".

3.7.3 Marcatore termini

Il nostro gruppo si avvale di uno *script*$_G$ automatico in linguaggio *Python*$_G$ per la marcatura dei termini definiti nel Glossario. Questo strumento esegue una scansione di ogni documento basandosi sull'elenco dei termini nel Glossario e marca ogni termine (se presente) come descritto ad inizio paragrafo.

4 Ruoli di progetto

Per la piena riuscita del progetto è indispensabile distinguere i vari ruoli che concorrono alla creazione del prodotto finale e le loro diverse responsabilità e competenze. Ogni ruolo avrà una specifica area di competenza, degli specifici compiti, oneri e particolari autorizzazioni. Ogni componente dovrà limitarsi ai compiti ad esso assegnati e, nel caso qualcosa esuli dal suo campo di pertinenza, lo stesso dovrà rivolgersi al collega occupante il ruolo corrispondente.
Tutti i membri, a rotazione, dovranno occupare come minimo una volta ciascuno dei ruoli descritti sottostante. Per fare in modo che la rotazione dei ruoli non causi problematiche o conflitti, è necessario che ciascuna attività venga pianificata e le persone interessate devono attenersi agli incarichi a loro assegnati.
I ruoli vengono assegnati in base al carico di lavoro che le attività richiedono (per essere svolte entro le scadenze fissate) e anche in base al budget massimo prefissato. A causa di quanto descritto in precedenza il gruppo ha deciso di assegnare più ruoli per persone, garantendo sempre che non vadano in conflitto tra di loro e che nella stessa giornata lavorativa una persona non svolge più ruoli contemporaneamente. Solo dopo aver garantito quanto scritto sopra, il Verificatore dovrà visionare il diario delle modifiche di ciascun documento per scovare incongruenze.
Verrà di seguito fornita una descrizione dei ruoli di progetto, accompagnata dalle responsabilità e le modalità operative alle quali le persone incaricate dovranno attenersi.

4.1 Responsabile di Progetto

Il Responsabile di Progetto incentra su di sé le responsabilità di scelta ed approvazione dei lavori. Ricopre il ruolo di rappresentante del gruppo nei contatti con l'esterno e durante la presentazione dei lavori.
Le sue competenze principali comprendono:

- Pianificazione, coordinamento e controllo delle attività;
- Gestione e controllo delle risorse;
- Approvazione delle analisi di gestione e rischio;
- Approvazione dei documenti;
- Comunicazioni con i Committenti/Proponenti.

Il Responsabile ha il compito di assicurarsi che le attività di Verifica vengano svolte sistematicamente seguendo le *Norme_di_Progetto_v3.2.0*. Deve al contempo accertarsi che vengano rispettati i ruoli e le competenze assegnate nel *Piano_di_Progetto_v3.2.0* e che non vi siano conflitti di interesse tra redattori e Verificatori. A tale proposito, sarà sua responsabilità stilare un *Piano di Progetto* in cui ogni redattore di uno specifico documento o codice potrà solamente verificare prodotti altrui.
Ha inoltre l'onere di gestire la creazione e l'assegnazione dei ticket di pianificazione e di assegnare ad un membro del gruppo il ruolo di Responsabile di quest'ultimo, nel caso riguardi una sotto-attività.
Redige il Piano di Progetto e collabora alla stesura del Piano di Qualifica, per quanto riguarda gli ambiti di pianificazione delle attività.

4.2 Amministratore

L'*Amministratore*$_G$ è il responsabile del controllo, dell'efficienza e dell'operatività dell'ambiente di lavoro e degli strumenti per la condivisone e la sincronizzazione.
Egli deve garantire:

- L'individuazione e la gestione di strumenti per automatizzare quanto più possibile processi o attività;
- L'individuazione e la gestione di strumenti per il controllo dei processi e delle risorse;
- L'individuazione e la gestione di strumenti e strategie per il controllo della qualità;
- Gestione del versionamento.

Redige la sezione delle *Norme di Progetto*, nei pararafi e sezioni relativi agli strumenti utilizzati a suppporto dei processi. Inoltre redige la sezione del *Piano di Qualifica* inerente agli strumenti di Verifica.

4.3 Analista

L'Analista è il responsabile delle attività di Analisi all'interno del progetto. Il suoi compiti comprendono:

- La piena comprensione del problema e il suo dominio;
- La stesura di una specifica precisa, completa e comprensibile dal Proponente, dal Committente e dal Progettista.

L'Analista redige il documento di Studio di Fattibilità, successivamente l' Analisi dei Requisiti e in parte il Piano di Qualifica.

4.4 Progettista

Il pregettista è colui il quale ricerca e delinea la giusta struttura del sistema per soddisfare il problema posto dal Committente e dal Proponente.
I suoi compiti principali sono:

- Disegnare una soluzione fattibile e adatta ad soddisfare i requisiti delineati dagli Analisti;
- Implementare soluzioni di progettazione ottimali, ottimizzate e collaudate;
- Utilizzare tecnologie e standard che rendano facile la realizzazione e la manutenzione del prodotto.

Redige la Specifica Tecnica e la Definizione di Prodotto, oltre a partecipare alla stesura delle metriche di verifica della progettazione del Piano di Qualifica.

4.5 Verificatore

Il Verificatore è responsabile delle attività di Verifica. Ha il compito di assicurare che i documenti e il codice siano conformi alle norme stabilite nel documento *Norme_di_Progetto_v3.2.0*. Inoltre deve controllare che lo stato di ciclo di vita sia corrispondente a quello attuale. Redige la sezione del Piano di Qualifica che illustra gli esiti delle prove e delle verifiche effettuate.

4.6 Programmatore

Il *Programmatore*$_G$ è responsabile delle attività di Codifica e delle componenti di ausilio necessarie per l'esecuzione delle prove di Verifica e Validazione. Le responsabilità di tale ruolo sono:

- Implementare rigorosamente le soluzioni descritte dal Progettista per la Realizzazione del progetto;
- Scrivere codice e la sua relativa documentazione che rispettino gli standard stabiliti per la loro scrittura;
- Implementare i *test*$_G$ sul codice scritto, necessari per prove di Verifica e Validazione.

Redige il Manuale *Utente*$_G$ e tutta la documentazione relativa al codice.

5 Procedure

Verranno elencate le procedure che i ruoli di progetto dovranno attenersi in maniera scrupolosa per raggiungere i fini preposti nel Piano di Qualifica.

5.1 Procedure proattive

Nel *processo*$_G$ documentazione, che appartiene alla categoria dei processi di supporto, sono state adottate le seguenti procedure proattive:

- **Correttore LaTeX:** lo strumento utilizzato, descritto nella sezione delle Norme di Progetto, per la scrittura dei documenti permette di identificare le parole contenenti errori ortografici; in questo modo ci si può accorgere dell'errore e procedere alla correzione mentre si sta scrivendo;
- **Scrittura frasi:** si cerca di scrivere frasi corte e semplici e con termini di uso comune per ottenere un valore accettabile dell'indice di Gulpease;
- **Scrittura paragrafo:** ogni volta che verrà scritto un paragrafo, si procederà alla lettura per più volte per assicurare che quello che è stato scritto abbia senso.

Nell'attività di codifica, appartenente al processo di sviluppo, verranno adottate le seguenti procedure proattive:

- **Ambiente di sviluppo:** l'ambiente di sviluppo, che verrà utilizzato nel prossimo periodo di tempo dedicato alla codifica, permetterà di segnalare, mentre si sta scrivendo, degli errori sintattici; questo permette di correggerli mentre si sta scrivendo del codice.

5.2 Gestione di Progetto

É compito del Responsabile di Progetto gestire questo processo.

5.2.1 Gestione delle attività

5.2.1.1 Pianificazione

All'inizio di ciascuna delle fasi delineate nel *Piano di Progetto v3.2.0*, il Responsabile di Progetto deve realizzare un *diagramma di* Gantt$_{GG}$ su *Redmine*$_G$, come descritto in sezione 6.1.1 di questo documento, per tracciare e gestire le attività dei processi utilizzati dal progetto. Deve inoltre assegnare ogni attività ad un componente o gruppo di componenti.

5.2.1.2 Assegnazione attività

Per assegnare le attività alle persone, il Responsabile di progetto dovrà seguire il sistema di ticketing descritto in sezione 6.1.1.1.

5.2.1.3 Controllo dell'avanzamento

Dopo ogni sessione di lavoro di un componente o gruppo di componenti, gli stessi devono tracciare il loro lavoro modificando l'avanzamento dell'attività e le ore/persona spese mediante il sistema di ticketing scelto, identificato in RedMine. Dovranno inoltre modificare lo stato dell'attività in base al ruolo ricoperto e lo stato di avanzamento della stessa come descritto nel paragrafo successivo. Anche questa operazione verrà eseguita mediante il sistema di ticketing.

5.2.1.4 Stati d'avanzamento

I possibili stati di avanzamento di un'attività sono i seguenti:

- **Nuovo:** l'attività è stata creata ed è in attesa di essere assegnata;
- **In Elaborazione:** l'attività è stata assegnata ed è in lavorazione;
- **Chiuso:** la lavorazione è terminata.

5.2.1.5 Gestione dei rischi

Durante l'avanzamento del progetto il Responsabile deve monitorare costantemente il verificarsi dei rischi descritti nel *Piano di Progetto v3.2.0* così come la possibilità che l'avanzamento comporti nuovi rischi non previsti precedentemente.

5.2.2 Creare un nuovo progetto

La creazione del progetto è onere del Responsabile di Progetto. Il progetto è una macro-attività che verrà suddivisa in molte sotto-attività al fine di semplificare la Progettazione, la Realizzazione e la Verifica delle stesse. Ogni attività sarà a carico di un Responsabile.

5.2.3 Ticket

I ticket sono uno strumento per il *tracciamento*$_G$ di attività, funzionalità e problematiche utile per sincronizzare e mantenere traccia del lavoro del team.

5.2.3.1 Creazione dei ticket

I ticket vengono creati dal:

- **Responsabile di Progetto:** crea i ticket di maggiore importanza per l'avanzamento tra le fasi macroscopiche di progetto;
- **Verificatore:** crea ticket per la segnalazione di problematiche o bug che necessitano di controllo e correzione.

5.2.3.2 Tipologie di Ticket

I ticket rappresentano operazioni e attività che devono essere portate a termine. I ticket avranno specifici stadi di avanzamento:

- **Nuovo**: il ticket è stato creato ed è in attesa di essere assegnato;
- **In Elaborazione**: il ticket è stato assegnato ed è in lavorazione;
- **Chiuso**: la lavorazione è terminata e l'attività è in attesa di verifica;
- **Risolto o Rifiutato**: un Verificatore approva o rigetta il lavoro effettuato. Nel caso di rigetto deve essere creato un nuovo ticket di modifica per risolvere il problema.

Di seguito verranno elencate le tipologie che i ticket possono assumere.

5.2.3.2.1 Ticket di Realizzazione

Assegnano la realizzazione delle attività ai membri del gruppo competenti. Sono organizzati in una gerarchia con vari livelli di priorità.

5.2.3.2.2 Ticket di Verifica

Rappresentano segnalazioni di errori, problematiche o imprecisioni riscontrate dai Verificatori durante l'attività di Verifica. Dovranno essere assegnati ad un membro del team che in quel momento ricopre un ruolo adatto ad intervenire per la correzione e in seguito riverificati da un Verificatore.

5.2.3.2.3 Ticket di Modifica

Rappresentano richieste di modifica fatte da membri del team al Responsabile. Quest'ultimo poi decide se accettare la richiesta di modifica o rifiutarla. Se la richiesta viene accettata, il Responsabile convoca una riunione per la discussione della modifica. Se la modifica viene rettificata dall'assemblea, le viene assegnata una priorità e viene messa nell'elenco di modifiche in attesa di implementazione. Le priorità che possono essere assegnate sono le seguenti:

- **Urgente:** da applicare il prima possibile;
- **Media:** da applicare entro la prossima *milestone*$_G$;
- **Bassa:** l'implementazione non è obbligatoria, non è stata definita una scadenza.

5.2.3.2.4 Ticket di Pianificazione delle attività

Rappresentano le macro-attività di maggiore importanza e sono organizzate in una gerarchia con diversi livelli di importanza; vengono creati dal Responsabile di Progetto, il quale poi individua le sotto-attività da assegnare a chi di dovere.

5.3 Analisi dei requisiti

La stesura del documento denominato Analisi dei Requisiti è compito degli Analisti. In questo documento verranno trovati, analizzati e catalogati tutti i requisiti, impliciti e non, imposti dal progetto.

5.3.1 Scrittura casi d'uso

Ciascun caso d'uso descritto dagli Analisti dovrà rispettare la seguente notazione:

UC[codice progressivo di livello]

Ogni caso d'uso ha un *codice*$_G$ univoco gerarchico.
Il codice progressivo può includere diversi livelli di gerarchia separati da un punto. A ogni descrizione di caso d'uso viene riportato il numero del diagramma associato di cui fa parte, ovvero la rappresentazione del caso d'uso generale a cui esso appartiene. Ogni diagramma associato è scritto nella seguente forma:

Figura[numero progressivo]:[codice univoco e nome del caso d'uso].

5.3.2 Identificazione e classificazione dei requisiti

È compito degli Analisti stilare una lista dei requisiti emersi dal capitolato e da eventuali riunioni con il Proponente. È inoltre loro compito identificare tutti quei requisiti impliciti che per loro natura non vengono specificati ma devono essere individuati e studiati. Tutti questi requisiti dovranno essere classificati per tipo e importanza, utilizzando la seguente codifica:

R[importanza][tipo][codice]

dove:

- Importanza può assumere i seguenti valori:
  - **O:** *requisito*$_G$ obbligatorio;
  - **D:** requisito desiderabile;
  - **F:** requisito facoltativo.
- Tipo può assumere i seguenti valori:
  - **F:** funzionale;
  - **Q:** di qualità;
  - **P:** prestazionale;
  - **V:** vincolo.
- Codice è il codice che identifica univocamente ogni requisito. La sua regolamentazione è descritta nell'*Analisi_dei_Requisiti_v3.2.0*.

5.3.3 Casi d'uso e UML

Successivamente all'individuazione ed al tracciamento dei requisiti si procede all'analisi dei casi d'uso, denominati nelle sezioni seguenti anche come use case o con l'acronimo UC. Per realizzare i diagrammi *UML*$_G$ dei casi d'uso è stato scelto Astah. Il software era stato consigliato dal Professor Cardin e soddisfa tutte le necessità del team. Tutti i diagrammi devono rispettare la specifica UML 2.0.

5.4 Progettazione

5.4.1 Specifica Tecnica

I Progettisti devono descrivere l'*architettura*$_G$ d'alto livello del sistema e dei singoli componenti nella Specifica Tecnica in modo chiaro, formale e non *ambiguo*$_G$; inoltre dovranno essere progettati i corrispondenti test d'integrazione.

5.4.1.1 Diagrammi UML

Tutti i diagrammi UML rispetteranno lo standard UML 2.0. Devono essere realizzati i seguenti diagrammi UML

- Diagrammi delle attività;
- Diagrammi di sequenza;
- Diagrammi delle classi;
- Diagrammi dei *package*$_G$.

5.4.1.2 Definizione delle componenti

Ogni componente progettato deve essere riportato all'interno della Specifica Tecnica secondo lo schema definito nei formalismi di specifica, nel documento Specifica Tecnica. Il componente verrà modellato secondo la seguente struttura:

- **Nome:** specifica il nome del componente; ogni nome di ciascun componente deve essere scritto secondo la seguente notazione: E1::E2::E3::E4;la struttura è la seguente:
  - **E1:** è il nome del componente generale, ovvero il $namespace_G$ globale;
  - **E2:** è il sotto-componente di E1;
  - **E3:** è il sotto-componente di E2;
  - **E4:** è una singola unità la cui responsabilità è affidata ad un singolo programmatore.
- **Informazioni sul package:** contiene un'immagine che mostra il componente corrente;
- **Descrizione:** descrive in maniera testuale la funzionalità del componente;
- **Sotto-componenti**: specifica se il componente ha altri sotto-componenti;
- **Interazioni con altri componenti:** specifica se il componente ha relazioni con altri componenti.
- **Classi:** specifica se il componente ha eventuali classi; se c'è una $classe_G$ deve essere scritta secondo la struttura definita nella successiva sezione.

5.4.1.3 Definizione delle classi

Ogni classe progettata deve essere riportata all'interno della Specifica tecnica secondo lo schema definito nei formalismi di specifica, nel documento Specifica Tecnica. La classe verrà modellata secondo la seguente struttura:

- **Nome:** specifica il nome della classe, in cui tra parantesi si mette se è astratta oppure no; ogni nome di ciascuna classe deve essere scritto secondo la seguente notazione: E1::E2::E3::E4::E5;la struttura è la seguente:
  - **E1:** è il nome della componente generale, ovvero il namespace globale;
  - **E2:** è il sotto-componente di E1;
  - **E3:** è il sotto-componente di E2;
  - **E4:** è il sotto-componente di E3;
  - **E5:** è il nome della classe.
- **Descrizione:** descrive in maniera testuale la funzionalità della classe;
- **Utilizzo:** descrizione testuale di come viene utilizzata la classe;
- **Relazioni con altre classi:** viene specificato se la classe corrente ha relazioni con delle classi di altri componenti;
- **Classe da cui eredita (superclasse):** specifica se la classe eredita da altre classi; si usa la seguente notazione: E1::E2::E3::E4::E5; la struttura è così formata:
  - **E1:** è il nome del namespace globale;

  - **E2:** package utilizzato;
  - **E3 (eventuale):** componente generico del package E2;
  - **E4 (eventuale):** componente generico del sotto-package E3;
  - **E5:** nome della classe.

- **Classi che ereditano (sottoclassi)**: specifica se la classe viene ereditata da altre classi; si usa la seguente notazione: E1::E2::E3::E4::E5; la struttura è così formata:
  - **E1:** è il nome del namespace globale;
  - **E2:** package utilizzato;
  - **E3 (eventuale):** componente generico del package E2;
  - **E4 (eventuale):** componente generico del sotto-package E3;
  - **E5:** nome della classe.

5.4.1.4 Design Pattern

I Progettisti dovranno fornire una descrizione dei design pattern utilizzati per la progettazione dell'architettura. In essi si deve includere una descrizione ed un diagramma che mostri il funzionamento e la struttura.

5.4.1.5 Tracciamento

Ogni requisito deve essere tracciato al componente che lo soddisfa, ed ogni componente deve essere tracciato al requisito che ha imposto la sua realizzazione.

5.4.1.5.1 Tracciamento requisiti-componenti

Di seguito verrà mostrata una tabella che rappresenta il tracciamento requisiti-componenti.
La tabella avrà la seguente forma:

- **Requisito:** specifica il requisito;
- **Descrizione:** descrizione del requisito;
- **Componenti:** specifica le componenti che soddisfano il requisito.

5.4.1.5.2 Tracciamento componenti-requisiti

Di seguito verrà mostrata una tabella che rappresenta il tracciamento componenti-requisiti.
La tabella avrà la seguente forma:

- **Componenti:** specifica la componente;
- **Requisiti:** specifica i requisiti che vengono soddisfatti dalla componente.

5.4.2 Test

5.4.2.1 Test di integrazione

I Progettisti dovranno definire dei test e delle classi atte a verificare il lavoro svolto.

5.4.2.2 Test di Unità

I Progettisti dovranno definire i test d'unità necessari per verificare che i componenti del sistema funzionino nella modo previsto.

5.4.2.3 Norme per il tracciamento dei test

Nel Piano di Qualifica il tracciamento verrà riportato in forma tabellare e suddiviso per tipologia del test.
La tabella dei test di sistema avrà la seguente forma:

- **Test:** codice univoco che identifica il test;
- **Descrizione:** descrizione testuale del test;
- **Requisito:** codice univoco del requisito a cui si fa riferimento;
- **Stato:** lo stato in questo caso sarà uguale per tutti, ovvero D.E (da eseguire), in quanto i test verranno eseguiti più avanti.

La tabella dei test d'integrazione avrà la seguente forma:

- **Test:** codice univoco che identifica il test;
- **Descrizione:** descrizione testuale del test;
- **Componente:** associa il componente al corrispondente test;
- **Stato:** lo stato in questo caso sarà uguale per tutti, ovvero D.E (da eseguire), in quanto i test verranno eseguiti più avanti.

5.4.2.4 Tracciamento componente-test d'integrazione

La tabella per il tracciamento componente-test d'integrazione avrà la seguente forma:

- **Componente:** il nome del componente associato al corrispondente test;
- **Test:** codice univoco che identifica il test.

5.4.2.5 Tracciamento test d'integrazione-componente

La tabella per il tracciamento test d'integrazione-componente avrà la seguente forma:

- **Test:** codice univoco che identifica il test;
- **Componente:** il componente associato al corrispondente test.

5.4.3 Definizione di Prodotto

I Progettisti devono descrivere l'architettura di basso livello del sistema e dei singoli componenti nella Definizione di Prodotto in modo chiaro, formale e non ambiguo, integrando quanto scritto nella Specifica Tecnica.

5.4.3.1 Diagrammi UML

Tutti i diagrammi UML rispetteranno lo standard UML 2.0. Devono essere realizzati i seguenti diagrammi UML:

- Diagrammi delle attività;
- Diagrammi di sequenza;
- Diagrammi delle classi;
- Diagrammi dei package.

5.4.3.2 Definizione delle classi

Ogni classe progettata deve essere riportata all'interno della Definizione di Prodotto secondo lo schema definito nei formalismi di specifica, nel documento Specifica Tecnica. La classe verrà modellata secondo la seguente struttura:

- **Nome:** specifica il nome della classe, in cui tra parantesi si mette se è astratta oppure no; ogni nome di ciascuna classe deve essere scritto secondo la seguente notazione: E1::E2::E3::E4::E5; la struttura è la seguente:
  - **E1:** è il nome della componente generale, ovvero il namespace globale;
  - **E2:** è il sotto-componente di E1;
  - **E3:** è il sotto-componente di E2;
  - **E4:** è il sotto-componente di E3;
  - **E5:** è il nome della classe.
- **Descrizione:** descrive in maniera testuale la funzionalità della classe;
- **Classe da cui eredita (superclasse):** specifica se la classe eredita da altre classi; si usa la seguente notazione: E1::E2::E3::E4::E5; la struttura è così formata:
  - **E1:** è il nome del namespace globale;
  - **E2:** package utilizzato;
  - **E3 (eventuale):** componente generico del package E2;
  - **E4 (eventuale):** componente generico del sotto-package E3;
  - **E5:** nome della classe.
- **Classi che ereditano (sottoclassi)**: specifica se la classe viene ereditata da altre classi; si usa la seguente notazione: E1::E2::E3::E4::E5; la struttura è così formata:
  - **E1:** è il nome del namespace globale;
  - **E2:** package utilizzato;
  - **E3 (eventuale):** componente generico del package E2;
  - **E4 (eventuale):** componente generico del sotto-package E3;
  - **E5:** nome della classe.

5.4.3.3 Definizione dei metodi

Ogni *metodo*$_G$ progettato deve essere riportato all'interno della Definizione di Prodotto. I metodi verranno modellati secondo la seguente struttura:

- **Nome:** identifica il nome del metodo;
- **Accesso:** identifica il livello di visibilità del metodo. L'accesso può essere:
  - **Private:** il metodo è visibile solo all'interno della classe;
  - **Protected:** il metodo è visibile all'interno del package;
  - **Public:** il metodo è visibile ovunque.
- **Descrizione:** è una descrizione testuale del metodo;
- **Tipo di ritorno (eventuale):** specifica il tipo di ritorno del metodo;
- **Parametri in ingresso:** specifica i parametri in ingresso; i parametri hanno la seguente forma:
  - **Nome:** specifica il nome del parametro;
  - **Tipo:** specifica il tipo del parametro.
- **Note (eventuali):** specifica eventuali note del metodo.

5.4.3.4 Definizione degli attributi

Ogni *attributo*$_G$ progettato deve essere riportato all'interno della Definizione di Prodotto. Gli attributi verranno modellati secondo la seguente struttura:

- **Nome:** identifica il nome dell'attributo;
- **Accesso:** identifica il livello di visibilità dell'attributo. L'accesso può essere:
  - **Private:** l'attributo è visibile solo all'interno della classe;
  - **Protected:** l'attributo è visibile all'interno del package;
  - **Public:** l'attributo è visibile ovunque.
- **Tipo:** identifica il tipo dell'attributo;
- **Descrizione:** è una descrizione testuale dell'attributo.

5.4.3.5 Tracciamenti

Di seguito verranno mostrati i seguenti tracciamenti:

5.4.3.5.1 Tracciamento classi-requisiti

Ogni requisito deve essere tracciato alla classe che lo soddisfa, ed ogni classe deve essere tracciata al requisito che ha imposto la sua realizzazione. La tabella del tracciamento avrà la seguente forma:

- **Classe:** specifica il nome della classe;
- **Requisiti:** specifica i requisiti.

5.4.3.5.2 Tracciamento requisiti-classi

La tabella del tracciamento avrà la seguente forma:

- **Requisiti:** specifica i requisiti;
- **Classe:** specifica il nome della classe.

5.4.3.5.3 Tracciamento modulo-test

La tabella del tracciamento avrà la seguente forma:

- **Modulo:** specifica il modulo;
- **Test:** specifica il test.

5.5 Verifica

La verifica è un'attività che compare sempre durante lo svolgimento del Progetto; bisogna eseguirla continuamente su processi, documenti e prodotti che portano alla realizzazione del prodotto finale. Le persone incaricate di svolgere questa attività sono i Verificatori; essi, per cercare di ottenere il miglior risultato possibile, dovranno attenersi a delle metriche specifiche descritte nelle successive sezioni.

5.5.1 Metriche per errori

Sono riportate delle metriche utili alla valutazione degli errori che si potranno riscontrare durante le attività di verifica e ad una loro prima correzione.

Errore	**Gravità**	**Priorità**
Indici Errati	Alta	Urgente
Tracciamento errato	Media	Breve
Errore di progettazione	Alta	Alta
Errore ortografico o di formattazione	Bassa	Entro Milestone
Valore Gulpease fuori norma	Bassa	Entro Milestone
Violazione norme di codifica	Media	Breve

Tabella 3: Tabella errori

La gravità di un errore può essere:

- **Bassa**: l'errore ha impatto minimo e su aspetti marginali;
- **Media**: l'errore ha un impatto significativo e causa un ritardo o un aumento di costo considerevole;
- **Alta**: l'errore rende il prodotto inutilizzabile e porta potenzialmente al fallimento del progetto.

I tempi di risoluzione possono essere:

- **Entro milestone**: l'errore va corretto prima della milestone;
- **Breve**: l'errore va corretto entro qualche giorno dalla sua scoperta;
- **Urgente**: l'errore va corretto il prima possibile.

Il compito del Verificatore, a seguito del riscontro di un errore, è il seguente:

- **Ticket:** il verificatore apre un ticket per la correzione dell'errore, come descritto in sezione 5.3.1, e la persona responsabile dell'attività procederà nella correzione, chiudendo così il relativo ticket aperto.

5.5.2 Verifica dei documenti

Il processo di verifica viene istanziato nel momento in cui l'output di un processo passa dalla versione X.0.Z alla versione X.1.0. È compito del responsabile assegnare ai Verificatori il ticket di verifica per il documento in questione. Per una corretta verifica del documento è necessario seguire il seguente schema:

- **Controllo sintattico e del periodo:** utilizzando TeXstudio e GNU Aspell vengono evidenziati e corretti gli errori di grammatica più evidenti. Ciascun documento dovrà essere sottoposto ad un Walkthrough da parte dei Verificatori per individuare errori di *sintassi*$_G$ e periodi di difficile comprensione;
- **Rispetto delle norme:** i Verificatori devono effettuare Inspection per verificare che le norme di progetto siano state rispettate;
- **Verifica termini da glossario:** mediante uno script automatizzato vengono marcate tutte le prime occorrenze di un termine del glossario;
- **Calcolo indice Gulpease:** su ogni documento va verificato che l'indice di Gulpease, calcolato con l'apposito script, sia entro i limiti stabiliti del PdQ. Nel caso non fosse così bisognerà effettuare Walkthrough alla ricerca di frasi troppo lunghe o complesse;
- **Miglioramento del processo:** nell'effettuare i processi di verifica, i Verificatori dovranno raccogliere gli errori riscontrati piu frequentemente al fine di poter utilizzare più efficacemente Inspection;
- **Segnalazione errori:** alla fine del processo di verifica, i Verificatori devono creare un ticket per ogni errore da riparare;
- **Lista di controllo:** il Verificatore è tenuto a usare la lista di controllo per la tecnica Inspection, al fine di rimuovere gli errori tipici che essa contiene; la lista è disponibile nell'*appendice*$_G$ A di questo documento.

5.5.3 Verifica diagrammi UML

Il Verificatore dovrà anche controllare i diagrammi UML che sono stati prodotti, in particolare:

- **Diagrammi dei casi d'uso:** il Verificatore dovrà controllare che quanto scritto rispetti lo standard UML 2.0 e controllare che ciascun caso d'uso rispecchi quanto descritto dallo stesso;
- **Diagrammi delle classi:** il Verificatore dovrà controllare che progettazione e i diagrammi delle classi abbiamo corrispondenza tra di loro; inoltre dovrà controllare che quanto scritto rispetti lo standard UML 2.0.

5.5.4 Verifica dei processi

Un altro compito dei Verificatori è quello di calcolare, prima della fine di ogni milestone (ovvero consegna di revisione) prefissata, gli indicatori descritti nella sezione Metriche per i Processi del Piano di Qualifica. Per calcolare tali indici in modo automatico verrà utilizzato il software descritto nella sezione 6.2 di questo documento.

5.5.5 Verifica della Progettazione

Ai Verificatori è richiesto il controllo della qualità della progettazione, sia generale che in dettaglio. È inoltre loro compito assicurarsi che i moduli del sistema abbiano dimensione e contenuto adatti ad essere realizzati da un singolo Programmatore.

5.5.6 Verifica della progettazione architetturale

Il Verificatore dovrà controllare che i valori delle metriche di accoppiamento efferente e afferente siano accettabili.

5.5.7 Verifica della progettazione di dettaglio

Il Verificatore dovrà controllare il documento Definizione di Prodotto, il quale contiene l'architettura in dettaglio, per assicurarsi che i moduli del sistema abbiano dimensione e contenuto adatti ad essere realizzati da un singolo Programmatore. Ciascun modulo dovrà essere associato ad almeno un requisito; un altro compito del Verificatore è quello di controllare tale tracciamento.

5.5.8 Verifica del codice

Il Verificatore dovrà utilizzare i test statici e dinamici per la verifica del codice. Per ciascuna categoria di test verranno utilizzati degli strumenti che il software Jenkins, descritto nella sezione 6.1.4 di questo documento, mette a disposizione.

- **Analisi statica:**
  - **Eclipse Metrics:** plug-in per Eclipse che calcola varie metriche del codice, descritte in sezione 2.9.3 del Piano di Qualifica;
  - **JSHint:** software esterno che può essere aggiunto come plug-in a Jenkins o Eclipse;
  - **Closure Compiler:** tool per ottimizzare codice *JavaScript*$_G$.
- **Analisi dinamica:**
  - **QUnit:** framework per i test d'unità in JavaScript;
  - **Jasmine:** framework per i test d'unità in JavaScript;
  - **Karma:** plug-in per QUnit o per Jenkins, per il *testing*$_G$ di *AngularJS*$_G$;
  - **Protractor:** framework per testing di AngularJS.

5.5.8.1 Documentazione del codice

5.5.8.1.1 Codifica

Nella codifica ci si dovrà attenere alle seguenti convenzioni, al fine di garantire la maggior qualità di processo possibile. Gli sviluppatori dovranno attenersi alle comuni Coding Conventions, in particolare a quelle suggerite dal Prof. Crockford nel suo *sito* web$_{GG}$.[1]

5.5.8.1.2 Nomi

I nomi di classi, metodi, variabili e funzioni dovranno essere in *camel case*$_G$, le parole componenti separate da underscore e scritte in inglese. La prima lettera dovrà essere minuscola, fatta eccezione per le classi che inizieranno sempre con lettera maiuscola.

5.5.8.1.3 Ricorsione

La ricorsione deve essere implementata solo in caso di vera necessità, altrimenti va evitata. Nel caso in cui venga implementata si dovrà fornire una prova di corretta terminazione e un calcolo approssimato delle risorse richieste per la corretta esecuzione.

[1] http://javascript.crockford.com/code.html

5.5.8.1.4 Concorrenza

La concorrenza dovrà essere per quanto possibile evitata. Nel caso questo non sia possibile si dovrà cercare di evitare la condivisione di risorse e serializzare il più possibile l'accesso alle risorse.

5.5.8.1.5 Header dei file

Ogni file dovrà contenere un header strutturato come segue:

- **File:** nome del file;
- **Module:** modulo di appartenenza;
- **Author:** autore (indirizzo email dell'autore);
- **Created:** data di creazione;
- **Version:** versione corrente;
- **Description:** descrizione dettagliata del file;
- **Modification History:** tabella dei cambiamenti effettuati sul file.

5.5.8.1.6 Documentazione dei metodi

Ogni metodo/funzione di codice dovrà essere preceduta da un commento contenente le seguenti informazioni:

- **Name:** nome del metodo;
- **Param:** lista del tipo dei parametri;
- **Descr:** breve descrizione del comportamento del metodo;
- **Return:** cosa ritorna la funzione.

5.5.8.1.7 Documentazione delle classi

Ogni classe deve essere preceduta da un commento contenente le seguenti informazioni:

- **Name:** nome della classe;
- **Descr:** breve descrizione della classe.

5.6 Calcolo indice Gulpease

Il calcolo dell'indice di Gulpease dovrà essere applicato dopo l'attività di verifica dello stesso documento, ma prima dell'approvazione. Il risultato di questo calcolo sarà un indice della qualità del documento. Se l'indice si scosterà in negativo dagli obiettivi del gruppo, si dovrà informare il Responsabile. Il Responsabile, successivamente, potrà decidere di chiedere una revisione dell'intero documento al fine di aumentarne la leggibilità.

6 Ambiente e strumenti di lavoro

6.1 Coordinamento

6.1.1 Redmine

Come piattaforma per la gestione del progetto è stato scelto Redmine. Le principali funzioni che esso fornisce sono:

- Creazione di un nuovo progetto;
- Un sistema di gestione dei ticket;
- Il grafico Gantt delle attività;
- Un calendario per organizzare i compiti e le attività;
- La visualizzazione del *repository*$_G$ associato al progetto;
- Un sistema di gestione del tempo e delle milestone.

Per quanto riguarda la creazione di un progetto, se ne occuperà il Responsabile di Progetto, il quale lo scompone in macro attività, assegnando ciascuna di esse al ruolo corrispondente. Su Redmine per creare un nuovo progetto bisogna seguire la seguente procedura:

- Cliccare su Progetti, in alto a sinistra nella barra;
- Cliccare su Nuovo Progetto;
- Assegnare un nome al progetto che si desidera creare;
- Specificare un identificativo.

6.1.1.1 Ticket

Per quanto riguarda i ticket, si possono aprire le seguenti tipologie di ticket:

- Ticket di realizzazione e controllo;
- Ticket per la verifica;
- Ticket per *pianificare*$_G$ attività;
- Ticket per di modifica.

6.1.1.1.1 Ticket di realizzazione e controllo

- Entrare nel progetto;
- Cliccare, nel menù in alto, sulla voce corrispondente a Nuova segnalazione;
- Nel menù a tendina con la voce Tracker selezionare Realizzazione;
- Specificare un oggetto per il ticket, che corrisponde ad una breve sintesi;
- Specificare una descrizione, che corrisponde allo scopo specifico di quel ticket;
- Specificare uno stato, ovvero all'inizio sarà nuovo;

- Specificare una priorità;
- Specificare a chi assegnare il ticket;
- Specificare la data di inizio;
- Specificare la data di scadenza;
- Specificare il tempo stimato;
- Specificare la percentuale che rappresenta quanto è stato completato.

6.1.1.1.2 Ticket per la verifica

Un verificatore può creare un ticket di verifica, e deve attenersi alla seguente linea guida:

- Entrare nel progetto;
- Cliccare, nel menù in alto, sulla voce corrispondente a Nuova segnalazione;
- Nel menù a tendina con la voce Tracker selezionare Verifica;
- Specificare un oggetto per il ticket, che corrisponde ad una breve sintesi dell'errore;
- Specificare una descrizione, che corrisponde ad una stesura dettagliata dell'errore, e specificare anche la posizione dell'errore;
- Specificare uno stato, ovvero all'inizio sarà nuovo;
- specificare una priorità;
- Specificare a chi assegnare il ticket;
- Specificare la data di inizio;
- Specificare la data di scadenza;
- Specificare il tempo stimato;
- Specificare la percentuale che rappresenta quanto è stato completato.

6.1.1.1.3 Ticket per pianificare attività

- Entrare nel progetto;
- Cliccare, nel menù in alto, sulla voce corrispondente a Nuova segnalazione;
- Nel menù a tendina con la voce Tracker selezionare Attività;
- Specificare un oggetto per il ticket, che corrisponde ad una breve sintesi dell'attività;
- Specificare una descrizione, che corrisponde ad una stesura dettagliata dell'attività;
- Specificare uno stato, ovvero all'inizio sarà nuovo;
- Specificare una priorità;
- Specificare a chi assegnare il ticket;

- Specificare la data di inizio;
- Specificare la data di scadenza;
- Specificare il tempo stimato;
- Specificare la percentuale che rappresenta quanto è stato completato.

6.1.1.1.4 Ticket di modifica

- Entrare nel progetto;
- Cliccare, nel menù in alto, sulla voce corrispondente a Nuova segnalazione;
- Nel menù a tendina con la voce Tracker selezionare Modifica;
- Specificare un oggetto per il ticket, che corrisponde ad una breve sintesi della modifica;
- Specificare una descrizione, che corrisponde ad una stesura dettagliata della modifica;
- Specificare uno stato, ovvero all'inizio sarà nuovo;
- Specificare una priorità;
- Specificare a chi assegnare il ticket;
- Specificare la data di inizio;
- Specificare la data di scadenza;
- Specificare il tempo stimato;
- Specificare la percentuale che rappresenta quanto è stato completato.

6.1.2 Condivisione dei file

É stato usato uno strumento per la condivisione dei file online, per velocizzare e semplificare l'organizzazione di alcune attività.

6.1.2.1 *Google Drive*$_G$

Questo strumento permette di condividere tra i membri del gruppo alcuni documenti, che possiedono queste caratteristiche:

- Non è richiesta la versione del documento;
- Richiedono interattività e cooperazione tra i membri del gruppo;
- Possono essere consultati da tutti, come ad esempio guide o libri utili.

6.1.3 Google Calendar

Viene messo a disposizione del team un calendario per coordinare temporalmente le attività e gli impegni di ciascuno dei componenti. Tutti i membri del team useranno il calendario messo a disposizione su Google Calendar per segnalare i giorni in cui per loro non sarà possibile lavorare al progetto o partecipare a riunioni e sessioni di lavoro di gruppo. I componenti si impegnano a segnare sul calendario quanto prima qualsiasi impegno o impedimento dovesse sorgere e a controllare periodicamente il suddetto per essere aggiornati su eventuali impegni altrui e/o di gruppo. Il calendario verrà inoltre usato dal Responsabile per la segnalazione di riunioni o sessioni di lavoro di gruppo.

6.1.4 Jenkins

Questo software fornisce dei servizi di integrazione continua per lo sviluppo del software. Rende più facile per gli sviluppatori integrare le modifiche al progetto; inoltre permette di effettuare test automatici, delle misurazioni o dei test statici sul codice. Jenkins al momento non è utilizzato, ma verrà preso in considerazione più avanti quando comincerà l'attività di codifica.

6.1.5 Git

Viene messo a disposizione un repository *Git*$_G$ su *GitHub*$_G$ per la gestione e il versionamento di codice e documenti tra i vari membri del gruppo.
Il repository pubblico è disponibile all'indirizzo:

https://github.com/ApertureSoftware/AperturePublic.git.

6.1.5.1 *Branch*$_G$

Nel repository saranno disponibili vari branch per favorire lo sviluppo del codice da parte degli sviluppatori.

6.1.5.1.1 Master

Il branch principale sarà chiamato master e conterrà l'ultima versione del software stabile rilasciata. Affinché una nuova versione possa essere caricata nel branch master, quest'ultima deve compilare senza errori o warning e deve aver superato tutti i test disegnati per verificarne la qualità.

6.1.5.1.2 Secondari

Gli sviluppatori possono richiedere la creazione di un branch secondario, provando nuove strategie di sviluppo senza alterare il branch master. Il Responsabile di Progetto è il responsabile dell'approvazione della richiesta.

6.2 Strumenti per la pianificazione

6.2.1 OpenProj

Si è scelto di utilizzare il software OpenProj per gestire e pianificare le attività e le risorse inerenti al progetto. Questo software è gratuito e open-source, sostituisce i più costosi software di project management, inoltre è multipiattaforma, ovvero è disponibile per più sistemi operativi. Questo software dispone delle seguenti caratteristiche:

- Open-source;
- Fornisce diagrammi di Gantt;
- Fornisce diagrammi di PERT;
- Generazione automatica della Work Breakdown Structure(WBS), partendo dal Gantt;
- Permette di calcolare le metriche Schedule Variance (SV) e Budget Variance (BV).

6.3 Strumenti per la documentazione

6.3.1 LaTeX

Per la stesura dei documenti verrà utilizzato il *linguaggio di markup*$_G$ LaTeX. Esso rende possibile definire template di documenti standardizzati da poter applicare a qualsiasi contenuto, separando così formattazione da contenuto e facilitando il lavoro di stesura. LaTeX inoltre dispone di qualsiasi strumento di formattazione di cui si possa avere bisogno, eliminando il bisogno di strumenti ausiliari alla stesura dei documenti.

6.3.2 Strumenti di Verifica

Le risorse software che si utilizzeranno durante i processi di Verifica sono:

- **Correttore automatico TeXstudio:** questo strumento integra i dizionari di OpenOffice.org e segnala i potenziali errori ortografici presenti nel testo;
- **Aspell:** strumento per la correzione tipografica dei documenti redatti in LaTeX;
- **Glossario-script:** script, scritto dai componenti del gruppo, che marca i termini nel glossario con la simbologia sopracitata;
- **Script Gulpease:** script per il calcolo automatico dell'indice di Gulpease.

6.3.3 Diagrammi UML

Come strumento per rappresentare i diagrammi delle classi, diagrammi di sequenza e diagrammi di attività, è stato utilizzato il software di modellazione Astah. Questo strumento è stato consigliato dai docenti del corso di Ingegneria del Software, in quanto è gratuito, semplice da usare e adeguato alle nostre esigenze. Tutti i diagrammi seguono gli standard UML 2.0.

6.4 Tracciamento dei requisiti

Per semplificare e automatizzare il tracciamento dei requisiti è stato realizzato un *database*$_G$ *relazionale*$_G$ che associa ad ogni requisito la sua sorgente e gli eventuali moduli che lo implementano. È inoltre possibile effettuare la ricerca inversa e, partendo da un componente od un use case, vedere i requisiti associati.

A Lista di Controllo

Durante le attività di Walkthrough sono stati riscontrati principalmente i seguenti errori:

- **Stili di testo:**
  - Elenco puntato non termina con ; nel caso in cui la riga non sia l'ultima, o non termina con . nel caso lo sia;
  - Elenco puntato non inizia con lettera maiuscola;
  - Le definizioni del glossario iniziano con È un;
  - È viene scritta incorrettamente come E';
  - Alcuni riferimenti ai documenti non vengono scritti in corsivo;

  - I vari ruoli non cominciano con la lettera maiuscola;
  - Le parole Proponente e Committente non vengono scritte con la lettera iniziale maiuscola;
  - Alcune tabelle non rispettavano lo standard stabilito.

- **Lingua italiana:**
  - Alcune parole accentate vengono scritte con l'apostrofo e non con il relativo accento;
  - In alcune frasi era omesso lo spazio tra la virgola e la parola successiva.
- **Sintassi UML:**
  - Nome di alcuni casi d'uso non rispecchiavano la descrizione dello stesso.
- **LaTeX**
  - La parola LaTeX non viene scritta con il comando giusto;
  - Per le parole accentate non viene usato il comando apposito.