

25 maggio 2015



Norme di Progetto

Informazioni sul documento

Nome Documento	Norme di Progetto
Versione	2.0.0
Stato	<i>Formale</i>
Uso	<i>Interno</i>
Data Creazione	2 marzo 2015
Data Ultima Modifica	25 maggio 2015
Redazione	Petrucci Mauro
Approvazione	Venturelli Giovanni
Verifica	Tollot Pietro
Lista distribuzione	<i>LateButSafe</i>

Prof. Tullio Vardanega

Prof. Riccardo Cardin

Proponente Zucchetti S.p.a.

Sommario

Il presente documento contiene le norme e le convenzioni che il gruppo LateButSafe intende adottare durante l'intero ciclo di vita del prodotto software Premi.



Registro delle modifiche

Tab 1: Versionamento del documento

Versione	Autore	Data	Descrizione
2.0.0	Venturelli Giovanni	25-05-2015	Approvazione del documento
1.7.0	Petrucci Mauro	24-05-2015	Apportate piccole modifiche
1.6.0	Petrucci Mauro	23-05-2015	Apportate modifiche segnalate dal verificatore Tollot Pietro
1.5.0	Tollot Pietro	22-05-2015	Verifica del documento
1.3.0	Petrucci Mauro	4-05-2015	Inserimento diagrammi attività
1.2.0	Petrucci Mauro	3-05-2015	Definizione ruoli con relative responsabilità e vincoli
1.1.0	Petrucci Mauro	1-05-2015	Correzione struttura e contenuti basata su errori segnalati dal committente
1.0.0	Petrucci Mauro	13-04-2015	Approvazione del documento
0.8.0	Petrucci Mauro	10-04-2015	Apportate le modifiche segnalate dal verificatore Venturelli Giovanni
0.5.0	Busetto Matteo	3-04-2015	Aggiornamento contenuti
0.4.0	Fossa Manuel	30-03-2015	Aggiornamento contenuti
0.3.0	Petrucci Mauro	22-03-2015	Correzione della parte Ingegneria dei Requisiti
0.2.1	Busetto Matteo	19-03-2015	Correzione di errori grammaticali e ortografici
0.2.0	Busetto Matteo	10-03-2015	Inserimento dei contenuti
0.1.0	Busetto Matteo	2-03-2015	Stesura dello scheletro del documento

Storico

pre-RR

Versione 1.0.0	Nominativo
Redazione	Busetto Matteo, Tollot Pietro, Petrucci Mauro, Fossa Manuel
Verifica	Venturelli Giovanni
Approvazione	Tollot Pietro

Tab 2: Storico ruoli pre-RR

$$\mathbf{RR} \rightarrow \mathbf{RP}$$

Versione 2.0.0	Nominativo
Redazione	Petrucci Mauro
Verifica	Tollot Pietro
Approvazione	Venturelli Giovanni

Tab 3: Storico ruoli RR \rightarrow RP



Indice

1	Introduzione	7
1.1	Scopo del documento	7
1.2	Glossario	7
1.3	Riferimenti	7
1.3.1	Informativi	7
2	Ruoli di progetto	8
2.1	Responsabile di Progetto	8
2.2	Amministratore	8
2.3	Analista	9
2.4	Progettista	9
2.5	Verificatore	9
2.6	Programmatore	10
3	Processo di sviluppo	11
3.1	Analisi dei Requisiti	11
3.1.1	Fattibilità	11
3.1.2	Scoperta dei requisiti	11
3.1.3	Interviste	11
3.1.4	Riunioni interne e casi d'uso	11
3.1.5	Classificazione e priorità	12
3.1.6	Specifica	13
3.2	Progettazione architetturale	13
3.2.1	Struttura del documento	13
3.2.2	Task	14
3.2.3	Direttive di progettazione	15
3.2.4	Diagrammi UML	15
3.2.5	Design pattern	16
3.2.6	Tracciamento componenti	16
3.3	Codifica	16
3.3.1	Nomi	17
3.3.2	Documentazione	17
3.4	Procedura di verifica	17
3.4.1	Verifica statica	17
3.4.2	Verifica dinamica	17
3.4.2.1	Definizione dei test	18
3.4.2.2	Codifica dei test di unità e di integrazione	18
3.4.2.3	Rapporto dell'analisi svolta sul codice	18
4	Processi di supporto	20
4.1	Documentazione	20
4.1.1	Template	20
4.1.2	Contenuto e struttura dei documenti	20
4.1.2.1	Verbali	20



4.1.2.2	Lettera di presentazione	21
4.1.3	Norme tipografiche	21
4.1.4	Formati di riferimento e altro	22
4.1.5	Immagini e tabelle	23
4.1.6	Glossario	23
4.1.6.1	Implementazione	24
4.2	Verifica	24
4.2.1	Metriche per gli errori riscontrati e gestione dei cambiamenti	24
4.2.2	Verifica dei processi	26
4.2.3	Verifica dei documenti	26
4.2.3.1	Verifica diagrammi UML	27
4.2.4	Verifica del codice	27
4.2.4.1	Analisi Statica	27
4.2.4.2	Analisi Dinamica	28
4.2.4.3	Test	28
4.2.4.4	Validazione codice	28
4.3	Validazione requisiti	28
4.4	Gestione delle modifiche ai requisiti	28
5	Processi Organizzativi	30
5.1	Applicazione PDCA	30
5.2	Gestione di progetto	31
5.2.1	Pianificazione delle attività	31
5.2.2	Coordinazione e controllo delle attività	31
5.2.3	Gestione e controllo delle risorse	31
5.2.4	Analisi e Gestione dei rischi	31
5.2.5	Elaborazione dati	31
5.2.6	Delega	31
5.2.7	Responsabilità di sotto-progetto	32
5.2.7.1	Assegnazione attività	32
5.2.7.2	Gestione dei cambiamenti	32
5.3	Collaborazione	32
5.3.1	Comunicazioni	32
5.3.1.1	Comunicazioni interne	32
5.3.1.2	Comunicazioni esterne	32
5.3.2	Riunioni	33
5.3.2.1	Interne	33
5.3.2.2	Casi Particolari	33
5.3.2.3	Esterne	33
5.3.2.4	Esito	33
5.3.3	Repository e strumenti per la condivisione di file	33
5.3.3.1	Repository	33
5.3.3.2	Condivisione file	34

6	Ambiente di lavoro	35
6.1	Risorse	35
6.1.1	Risorse _g necessarie:	35
6.1.1.1	Risorse _g umane	35
6.1.1.2	Risorse _g Hardware	35
6.1.1.3	Risorse _g software	35
6.1.2	Risorse _g disponibili	35
6.1.2.1	Risorse _g software	36
6.2	Sistemi Operativi	36
6.3	Coordinamento	36
6.3.1	Software _g di gestione del progetto	36
6.3.2	Versionamento	36
6.3.3	Software _g di Integrazione Continua	37
6.3.4	Condivisione dei file	37
6.3.4.1	Google Drive	37
6.3.5	Google Calendar	38
6.4	Pianificazione	38
6.5	Strumenti per i documenti	38
6.5.1	LATEX	38
6.5.2	Controllo ortografico	38
6.5.3	Grafici UML	38
6.5.4	Fogli di calcolo	38
6.6	Strumenti per la codifica	39
6.6.1	Stesura	39
6.6.2	Verifica	39
6.7	Protocollo per lo sviluppo dell'applicazione	39
6.7.1	Creare un nuovo progetto	39
6.7.2	Creazione ticket	40
6.7.2.1	Ticket _g di pianificazione	40
6.7.2.2	Ticket _g di realizzazione e controllo	41
6.7.2.3	Ticket _g di verifica	41
6.7.2.4	Dipendenze temporali	42
6.7.3	Aggiornamento ticket	42
6.7.3.1	Ticket _g di pianificazione	42
6.7.3.2	Ticket _g di realizzazione e controllo	43
6.7.3.3	Ticket _g di verifica	43
6.7.4	Consigli di utilizzo	46
6.8	LateTack	47
6.8.1	Aggiunta nuovo requisito	47
A	Lista di controllo	49

1 Introduzione

1.1 Scopo del documento

Il seguente documento viene redatto allo scopo di definire insieme di norme che regolamenteranno lo svolgimento del progetto. Le suddette norme verteranno su tutti gli aspetti del progetto:

- **Relazioni interpersonali** : comunicazione fra le varie figure professionali
- **Redazione documenti** : stili di redazione dei vari documenti interni e/o esterni;
- **Codifica**: stili e convenzioni di scrittura del codice sorgente;
- **Procedure di automazione**: strumenti e procedure per l'automazione di attività tecniche;
- **Definizione dell'ambiente di lavoro**: programmi utilizzati dall'intero gruppo di progetto.

Tutti i membri del gruppo di progetto sottoscrivono le norme ivi contenute e vi sottostanno, in modo da migliorare la coerenza fra i vari documenti e incrementare l'efficienza ed efficacia dei vari file prodotti. Qualora si renda necessario, ogni componente del gruppo potrà proporre all'*Amministratore di Progetto* una modifica alle *Norme di Progetto*; egli, sentiti gli altri componenti del gruppo valuterà se effettuare la modifica o meno.

1.2 Glossario

Insieme alla documentazione viene allegato il glossario dei termini (file_g [Glossario_v.2.0.0.pdf](#)), che ha il compito di definire tutti i vocaboli tecnici usati, seguendo convenzione, all'interno dei vari documenti. Ogni occorrenza di vocaboli presenti nel Glossario è marcata da una “g” minuscola in pedice.

1.3 Riferimenti

1.3.1 Informativi

- Piano di Progetto: [PianoDiProgetto_v.2.0.0.pdf](#);
- Piano di Qualifica: [PianoDiQualifica_v.2.0.0.pdf](#);

2 Ruoli di progetto

Durante lo sviluppo del progetto vi saranno diversi ruoli che i membri del gruppo andranno a ricoprire. Tali ruoli rappresentano figure aziendali specializzate, indispensabili per il buon esito del progetto. Ciascun componente del gruppo dovrà ricoprire almeno una volta ogni ruolo(vincolo organigramma). Si deve inoltre certificare che non vi siano conflitti di interesse nello svolgimento delle attività di verifica e di approvazione.

Per garantire che la rotazione dei ruoli non provochi conflitti è necessario che le attività di stesura e verifica vengano pianificate dettagliatamente e che i soggetti interessati rispettino i compiti a loro assegnati. Sarà poi compito del *Verificatore* controllare attentamente il diario delle modifiche di ogni documento per individuare eventuali incongruenze.

Si descrivono ora i diversi ruoli di progetto, con le relative responsabilità e le modalità operative affinché essi possano svolgere i compiti assegnati con l'ausilio dei software scelti per il progetto.

2.1 Responsabile di Progetto

Il *Responsabile di Progetto* rappresenta il progetto, in quanto accentra su di sé le responsabilità di scelta ed approvazione, ed il gruppo, in quanto presenta al committente i risultati del lavoro svolto. Detiene il potere decisionale, quindi la responsabilità su:

- Pianificazione, coordinamento e controllo delle attività;
- Gestione e controllo delle risorse;
- Analisi e gestione dei rischi;
- Approvazione dei documenti;
- Approvazione dell'offerta economica.

Di conseguenza, ha il compito di assicurarsi che le attività di verifica vengano svolte sistematicamente seguendo le *Norme di Progetto*, vengano rispettati i ruoli e le competenze assegnate nel *Piano di Progetto*, non vi siano conflitti di interesse tra redattori e verificatori. Egli è l'unico a poter decidere l'approvazione di un documento e a sancirne la distribuzione. Solo in casi particolari il *Responsabile* può delegare ad un verificatore l'approvazione di un documento come descritto nella sezione §5.2.6.

Ha inoltre l'incarico di gestire la creazione e l'assegnazione dei ticket delle macro-fasi e di assegnare ad un membro del gruppo il ruolo di responsabile di quest'ultima. Redige il *Piano di Progetto* e collabora alla stesura del *Piano di Qualifica*, in particolare nella sezione relativa alla pianificazione.

2.2 Amministratore

L'*Amministratore* è responsabile del controllo, dell'efficienza e dell'operatività dell'ambiente di lavoro. Le mansioni di primaria importanza che gli competono sono:

- Ricerca di strumenti che possano automatizzare qualsiasi compito che possa essere tolto all'umano;

- Risoluzione dei problemi legati alle difficoltà di gestione e controllo dei processi e delle risorse. La risoluzione di tali problemi richiede l'adozione di strumenti adatti;
- Controllo delle versioni e delle configurazioni del prodotto;
- Gestione dell'archiviazione e del versionamento della documentazione di progetto;
- Fornire procedure e strumenti per il monitoraggio e la segnalazione per il controllo qualità. Redige le *Norme di Progetto*, dove spiega e norma l'utilizzo degli strumenti, redige la sezione del *Piano di Qualifica* dove vengono descritti strumenti e metodi di verifica;

2.3 Analista

L'*Analista* è responsabile delle attività di analisi. Le responsabilità di spicco per tale ruolo sono:

- Produrre una specifica di progetto comprensibile, sia per il Proponente, sia per il Committente che per il *Progettista*, e motivata in ogni suo punto;
- Comprendere appieno la natura e la complessità del problema.

Redige lo *Studio di Fattibilità*, l'*Analisi dei Requisiti* e parte del *Piano di Qualifica*. Partecipa alla redazione del *Piano di Qualifica* in quanto conosce l'ambito del progetto ed ha chiari i livelli di qualità richiesta e le procedure da applicare per ottenerla.

2.4 Progettista

Il *Progettista* è responsabile delle attività di progettazione. Le responsabilità di tale ruolo sono:

- Produrre una soluzione attuabile, comprensibile e motivata;
- Effettuare scelte su aspetti progettuali che applichino al prodotto soluzioni note ed ottimizzate;
- Effettuare scelte su aspetti progettuali e tecnologici che rendano il prodotto facilmente manutenibile.

Redige la *Specifica Tecnica*, la *Definizione di Prodotto* e le sezioni inerenti le metriche di verifica della programmazione del *Piano di Qualifica*.

2.5 Verificatore

Il *Verificatore* è responsabile delle attività di verifica. Ha il compito di effettuare la verifica dei documenti utilizzando gli strumenti e i metodi proposti dal *Piano di Qualifica* e attenendosi a quanto descritto nelle *Norme di Progetto*. Le responsabilità di tale ruolo sono:

- Assicurare che l'attuazione delle attività sia conforme alle norme stabilite;
- Controllare la conformità di ogni stadio del ciclo di vita del prodotto.

Redige la sezione del *Piano di Qualifica* che illustra l'esito e la completezza delle verifiche e delle prove effettuate.

2.6 Programmatore

Il *Programmatore* 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*, da cui seguirà quindi la realizzazione del prodotto;
- Scrivere codice documentato, versionato, manutenibile e che rispetti gli standard stabiliti per la scrittura del codice;
- Implementare i test sul codice scritto, necessari per prove di verifica e validazione. Redige il *Manuale Utente* e produce un'adeguata documentazione del codice.

3 Processo di sviluppo

3.1 Analisi dei Requisiti

3.1.1 Fattibilità

A partire da informazioni preliminari sul capitolato, lo studio di fattibilità dovrà generare un rapporto che indichi la convenienza o meno del gruppo nello sviluppo del sistema. In particolare si dovrà considerare:

1. Sufficienza di risorse umane_g;
2. Rapporto tra i costi ed i benefici;
3. Rischi individuati.

Nello stimare i benefici dovrà essere data molta importanza alle competenze che i membri del gruppo acquisirebbero nello sviluppo del sistema.

3.1.2 Scoperta dei requisiti

3.1.3 Interviste

Al fine di evitare interviste infruttuose verrà preparato un elenco di punti da sottoporre al proponente_g in modo da dare una direzione precisa all'intervista. Potrebbe essere utile discutere con il proponente_g dei casi d'uso_g analizzati internamente al gruppo durante la fase di analisi. Le richieste di interviste al proponente_g avverranno con le modalità descritte in "comunicazioni esterne". Durante ogni intervista dovrà essere scritta una minuta che sarà confermata dal proponente_g, eventualmente con le opportune modifiche. La minuta sarà confermata al termine dell'incontro. Quando non fosse un problema per il proponente_g l'audio dell'intervista dovrà essere registrato per favorire la futura fase di analisi.

3.1.4 Riunioni interne e casi d'uso

Individualmente e durante le riunioni interne gli analisti dovranno analizzare le informazioni raccolte dalle interviste con il proponente_g per individuare problemi e fonti da cui attingere i requisiti_g.

L'individuazione dei requisiti_g funzionali sarà guidata dai casi d'uso. I casi d'uso potranno avere rappresentazione a diagrammi ma ogni caso d'uso dovrà avere anche la rappresentazione testuale. In particolare nella rappresentazione testuale si definirà:

1. Identificativo;
2. Attore primario;
3. Precondizioni;
4. Postcondizioni;
5. Scenario principale;



6. Estensioni_g.

Per la sintassi si rimanda a "Dall'idea al codice_g con UML2.0, Luciano Baresi, Luigi Lavazza, Massimiliano Pianciamore".

3.1.5 Classificazione e priorità

I requisiti_g dovranno essere classificati in:

1. Requisiti_g di processo_g;
2. Requisiti_g di prodotto.

I requisiti_g di prodotto saranno classificati in base a:

1. Tipologia;
2. Importanza;
3. Provenienza.

Dove i gradi di importanza saranno:

- **{Obbligatorio}**: requisito da considerarsi **irrinunciabile** per il cliente. Senza di esso l'applicazione è da considerarsi non soddisfacente per il cliente;
- **{Desiderabile}**: requisiti non strettamente necessari, ma che apportano valore aggiunto importante al prodotto;
- **{Opzionale}**: requisito relativamente utile/importante o che potrebbe essere soggetto di ulteriore contrattazione.

La provenienza può essere:

- **{Capitolato}**: da capitolato;
- **{Interni}**: da analisi interna;
- **{Proponente}**: da incontro con proponente_g.

Mentre le tipologie saranno:

- **{RF}**: requisito funzionale, determina le capacità richieste al sistema;
- **{RQ}**: requisito_g di qualità, requisito volto a portare valore aggiunto al sistema;
- **{RV}**: requisito_g di vincolo, requisiti espressamente indicati nel capitolato d'appalto o nei verbali d'incontro con il Proponente o Committente.

3.1.6 Specifica

Nella specifica dei requisiti_g dovrà essere considerato come riferimento lo standard IEEE 830-1998. In particolare saranno da perseguire le seguenti caratteristiche dei requisiti_g:

1. Non ambigui;
2. Corretti;
3. Completi;
4. Verificabili;
5. Consistenti;
6. Modificabili;
7. Tracciabili;
8. Ordinati per rilevanza.

I requisiti_g dovranno essere specificati in un documento "Analisi dei requisiti" secondo la struttura definita nello standard IEEE 830-1998. La specifica dei requisiti_g dovrà essere documentata in forma tabellare per evitare ambiguità. Per ogni requisito_g dovranno essere definiti un codice_g, una descrizione, un riferimento alla fonte e un riferimento alla verifica. Al fine di rendere meno ambigui i requisiti_g sarà redatto un "Glossario" contenente la definizione di tutti i termini non ovvi usati in fase di analisi.

3.2 Progettazione architettuale

3.2.1 Struttura del documento

Il documento in questione deve rispettare la seguente struttura:

1. Introduzione;
2. Definizione di prodotto;
 - (a) Metodo e formalismo di specifica;
 - (b) Architettura generale;
3. Descrizione dei singoli componenti;
 - Tipo, obiettivo e funzione del componente;
 - Relazioni d'uso di altre componenti;
 - Interfacce con e relazioni di uso da altre componenti;
 - Attività svolte e dati trattati;
 - Logica di progettazione;
 - Descrizione delle classi;

4. Diagramma di attività;
5. Prototipi di interfaccia utente;
6. Stime di fattibilità e bisogno di risorse;
7. Tracciamento.

Nella sezione di descrizione del metodo e formalismo di specifica è necessario indicare quali design pattern verranno utilizzati, quale metodo di progettazione verrà adottato e quali librerie o strumenti sono necessari per lo sviluppo del sistema progettato durante questa fase. La descrizione dei singoli componenti deve operare in modo ricorsivo partendo da una visione ad alto livello, nel caso di un'analisi *top-down*, fino a giungere ad un grado di dettaglio coerente con quello richiesto, specificato nel *Piano di Progetto* per questa fase. Nel caso di un'analisi di tipo *bottom-up* la progettazione procederà nel senso inverso.

3.2.2 Task

Lo scopo di questa attività è quello di realizzare una visione globale di ciò che dovrà essere il sistema a fronte dei requisiti ricavati dall'attività di analisi. Terminata l'attività di progettazione architeturale si deve produrre un documento completo ed esplicativo: la *Specific Tecnica*. Le attività necessarie alla redazione del documento sono:

- Definizione dell'architettura di prodotto a partire dall'Analisi dei Requisiti;
- Analisi delle tecnologie da adottare;
- Individuazione e studio dei design pattern applicabili;
- Individuazione della struttura dei package;
- Individuazione delle classi che compongono il sistema e delle relazioni tra esse;
- Studio di fattibilità;
- Tracciamento componenti-requisiti.

Si devono inoltre definire, in un documento specifico ([PianoDiQualifica_v.2.0.0.pdf](#)), vari test da eseguire sulle parti del sistema per verificarne la corretta interazione:

- **Input:** Analisi dei Requisiti;
- **Output:** Specifica Tecnica, pianificazione test di integrazione e di sistema;
- **Risorse:** Progettisti, documentazione, strumentazione;
- **Misurazioni:** avanzamento dell'elaborazione del documento *Specifica Tecnica* rispetto alla totalità dei requisiti definiti nel documento di Analisi dei Requisiti;
- **Norme:** descritte in seguito.

3.2.3 Direttive di progettazione

Per garantire una buona qualità del software prodotto in fase di codifica è necessario prevedere alcuni accorgimenti durante la progettazione in modo da evitare descrizioni del sistema errate o inutilmente complicate. Durante la fase di progettazione si richiede il soddisfacimento delle seguenti proprietà:

- **Semplicità:** viene ribadita da un principio, noto con il nome di “rasoio di Occam”, il quale afferma che tra tutte le soluzioni che portano al medesimo risultato è utile considerare quella più semplice. È difficile definire criteri precisi per il conseguimento di questa proprietà. Una buona pratica prevede la descrizione, per ogni vista del documento di progetto, delle scelte progettuali effettuate e delle alternative scartate;
- **Incapsulazione:** consiste nel fornire pubblicamente solamente l’interfaccia e nascondere tutti i caratteri implementativi come gli algoritmi e la struttura dati interna di una classe. Questa proprietà garantisce la produzione di codice più facilmente manutenibile e con ridotto numero di dipendenze da altri componenti. La produzione di codice incapsulato deriva da una buona progettazione dei metodi e degli attributi delle classi, la quale verrà affrontata più dettagliatamente nella fase successiva, e un accurato studio delle interfacce e dei package;
- **Coesione:** riguarda il grado di decomposizione di componenti in parti più piccole. La modularità spinge a dividere il più possibile i componenti ma non sempre è una scelta efficace. Nella fase di progettazione questa proprietà si rivela fondamentale in quanto andrà ad influire, in fase di codifica, sulle metriche di qualità del codice. Per garantire la produzione di componenti coesi è necessario chiedersi ogni volta che si struttura una classe se gli attributi e i metodi in essa definiti sono direttamente correlati con la sua entità o se è conveniente raggrupparli in un’ulteriore classe;
- **Accoppiamento:** indica l’utilità di un componente e la dipendenza da altri moduli. L’accoppiamento è caratterizzato da due valori:
 - **Fan-in:** indica il grado di riuso del codice. Maggiore è questo valore più alto è il grado di riuso e quindi maggiore è l’utilità del metodo;
 - **Fan-out:** indica il grado di accoppiamento e di dipendenza da altre porzioni di codice. Un valore elevato denota una maggiore complessità nella fase di esecuzione e di test. Entrambi questi valori vanno a influenzare le metriche di bontà del codice quindi bisognerà prestare attenzione a tracciare e minimizzare le dipendenze tra classi e ad evitare dipendenze circolari a livello di package.

3.2.4 Diagrammi UML

Data la visione a livello medio-alto di dettaglio richiesta per questo documento si dovranno utilizzare schemi UML 2.x in grado di descrivere formalmente i vari componenti del sistema. In particolare si andranno ad utilizzare i seguenti tipi di diagrammi:

- **Diagrammi di package:** saranno utilizzati per raggruppare più elementi UML aventi funzionalità simili. Ogni package dovrà essere identificato da un nome che risulti completamente qualificato e univoco all'interno dello spazio dei nomi. Schemi di questo tipo



sono utili per individuare le dipendenze tra classi e per stimare la complessità strutturale del sistema.

- **Diagrammi di classe:** utilizzati per descrivere i tipi di oggetti che fanno parte di un sistema e le relazioni che vi sono tra di essi. Per garantire una buona leggibilità dello schema si consiglia di valutare l'inserimento degli elementi di una classe in base al loro numero. È possibile omettere gli elementi di una classe anche nel caso in cui lo schema debba riportare un numero elevato di classi. Anche il livello di dettaglio della segnatura dei metodi è a discrezione del progettista con l'indicazione di considerare la seguente lista di priorità:
 - Nome del metodo;
 - Livello di accessibilità;
 - Tipo di ritorno, tipo dei parametri in ingresso ed eccezioni lanciabili;
 - Nome dei parametri in ingresso.
- **Diagrammi di sequenza:** utilizzati per descrivere la collaborazione tra più oggetti che hanno lo scopo di implementare collettivamente un comportamento. Non sono adatti per la modellazione della logica di controllo e vanno preferiti i diagrammi di attività se si intende modellare dei cicli o delle condizioni;
- **Diagrammi di attività:** descrivono la procedura logica con la quale vengono eseguite delle operazioni. Vanno utilizzati quando si vuole descrivere l'esecuzione di flussi paralleli.

3.2.5 Design pattern

I *Progettisti* devono descrivere i design pattern utilizzati per realizzare l'architettura: di essi si deve includere una breve descrizione e un diagramma che ne esemplifichi il funzionamento e la struttura.

3.2.6 Tracciamento componenti

Ogni requisito deve essere tracciato al componente che lo soddisfa. Il software LateTrack genera automaticamente le tabelle di tracciamento come descritto nella sezione §6.8. In questo modo sarà possibile misurare il progresso nell'attività di progettazione e garantire che ogni requisito venga soddisfatto.

3.3 Codifica

Le convenzioni di codifica che tutti i membri del gruppo devono seguire sono quelle specificate alla seguente pagina :

http://www.w3schools.com/js/js_conventions.asp



3.3.1 Nomi

- I nomi di variabili, metodi e funzioni dovranno essere espressi in dromedaryCase;
- I nomi delle classi dovranno essere espressi in CamelCase;
- nomi di variabili globali e costanti dovranno essere in UPPERCASE.

3.3.2 Documentazione

I file contenenti codice dovranno essere provvisti di un'intestazione contenente:

```
1  /*!  
2  * \file Nome del file  
3  * \author Autore (indirizzo email dell'autore)  
4  * \date Data di creazione  
5  * \brief Breve descrizione del file  
6  *  
7  * Descrizione dettagliata del file  
8  */  
9  
10 Qui verranno riportate da ogni programmatore tutte le  
    informazioni necessarie per una semplice comprensione del  
    listato.
```

3.4 Procedura di verifica

Per rendere uniforme la procedura di verifica, vengono qui elencate le norme che regolano la verifica del codice prodotto, si rimanda al [PianoDiQualifica_v.2.0.0.pdf](#) che specifica in maniera dettagliata le tecniche e le modalità con cui verranno condotte le attività di verifica e validazione durante l'intero sviluppo del progetto.

3.4.1 Verifica statica

- **Analisi del flusso di controllo:** si accerta che il codice segua il flusso aspettato, che non si possa entrare in porzioni di codice che possano non terminare, che non esista codice non raggiungibile;
- **Analisi del flusso dei dati:** si accerta che il software non acceda mai a variabili non inizializzate o scriva inutilmente più volte prima di usare una variabile;
- **Analisi del flusso di informazione:** verifica che gli input e gli output di ogni unità di codice o di più unità rientrino nelle specifiche del programma.

3.4.2 Verifica dinamica

- **Test di unità:** test che si effettuano per ogni unità del software con il massimo grado di parallelismo;



- **Test di integrazione:** verifica dei componenti formati dall'integrazione delle varie unità che hanno passato il test di unità;
- **Test di sistema e di collaudo:** verifica che il sistema in cui andrà installato il software rispetti i requisiti richiesti, o che il software riesca ad adattarsi correttamente al contesto dell'azienda proponente. Finito il collaudo del software avverrà il rilascio del prodotto.

3.4.2.1 Definizione dei test

Per agevolare la fase di verifica tramite test è necessario definire una lista di prove da eseguire nelle varie fasi del ciclo di sviluppo del software. La lista dei test da eseguire è riportata nel documento di *Piano di Qualifica*. Ogni test va accompagnato da un identificativo che verrà utilizzato per riferire la singola attività all'interno degli altri documenti. L'identificativo dei test è composto da due caratteri maiuscoli, da un numero e da un carattere minuscolo. Il primo è il carattere T e viene utilizzato per indicare che si tratta di un elemento inerente all'attività di test. Il secondo carattere è uno tra i seguenti:

- **S:** test di sistema;
- **I:** test di integrazione;
- **U:** test di unità.

Il numero posto dopo i due caratteri è incrementale e non gerarchico. L'ultimo carattere identifica i possibili esiti delle operazioni svolte dall'utente, parte dal carattere a e procede in ordine alfabetico crescente. Si consiglia di dare precedenza agli esiti positivi per poi passare a quelli di rifiuto di input non valido o di scatenamento di errori. Nel Piano di Progetto si dovrà inserire una tabella riportante l'identificativo e la descrizione dell'attività di test.

3.4.2.2 Codifica dei test di unità e di integrazione

Per l'automazione dei test di unità e di integrazione il gruppo di sviluppo utilizzeranno gli strumenti elencati nella sezione §4.2.4.3. Questi strumenti permettono di definire delle classi di test e dei metodi appositi per verificare il corretto funzionamento delle procedure sotto esame per poi generare un rapporto sul risultato delle attività. Per ogni unità da testare sarà necessario creare una nuova classe di test avente come nome quello dell'elemento da esaminare seguito dalla parola "Test". Nel caso di test di integrazione il nome dovrà essere indicativo dei componenti coinvolti e del tipo di interazione sotto esame. Il formalismo da adottare per la stesura del codice di test è equivalente a quello indicato per il resto del codice dell'applicativo fatta eccezione per l'inserimento dei commenti che è a discrezione del *Verificatore*. Si consiglia, tuttavia, di inserire una breve descrizione dello scopo dei metodi di test.

3.4.2.3 Rapporto dell'analisi svolta sul codice

Durante le attività di verifica statica e dinamica sul codice i verificatori sono tenuti a compilare un rapporto che descriva i test e le analisi svolte, accompagnate dai rispettivi risultati. Tutte queste informazioni andranno inserite in un documento, in forma tabellare, suddiviso nelle seguenti sezioni:

- **Rapporto dell’analisi statica:** riassunto dei risultati ottenuti dall’analisi del codice con gli strumenti di debugging inseriti nella sezione §4.2.4.1;
- **Valutazione delle metriche:** rapporto dei valori delle metriche del codice misurate con *jsmeter* ed eventuali considerazioni sui risultati;
- **Descrizione dei test:** elenco completo dei test di unità, di integrazione e di sistema. La definizione dei test verrà inserita di pari passo con la loro progettazione. Ogni definizione deve comprendere il nome della classe di test, il nome della classe da testare e l’elenco delle funzioni di verifica accompagnate da una descrizione esplicativa;
- **Rapporto delle sessioni di test:** elenco delle sessioni di test eseguite dai verificatori. Ogni sessione di test deve riportare la data di esecuzione, il nome dei verificatori incaricati e una lista contenente i metodi di test eseguiti e i risultati ottenuti. Per ogni metodo di test eseguito è necessario riportare la dicitura “SUPERATO” o “FALLITO” in base al risultato ottenuto. In caso di fallimento è richiesto l’inserimento di una breve descrizione del problema riscontrato e l’apertura di un ticket di segnalazione. A discrezione dei verificatori sarà possibile inserire un rapporto sul coverage ottenuto durante la sessione.

Se durante la fase di test vengono riscontrati degli errori nel codice è necessario comunicare il problema attraverso lo strumento di ticketing *redmine*. L'aggiunta dei risultati dei test prevede l'inserimento dei seguenti dati nella descrizione:

- Identificativo dell'attività di test;
- Nome dell'elemento testato;
- Descrizione dell'errore;
- Ora del rilevamento;
- Nome del verificatore che ha trovato l'errore;
- Codice di un errore precedentemente rilevato, se in presenza di un duplicato.

Successivamente si procederà alla valutazione della gravità dei vari errori e, se considerati risolvibili, il *Responsabile* assegnerà le correzioni da apportare ai programmatori tramite lo strumento di ticketing.

4 Processi di supporto

4.1 Documentazione

Questo capitolo descrive tutte le convenzioni scelte ed adottate dai LateButSafe riguardo alla stesura, verifica e approvazione della documentazione da produrre.

4.1.1 Template

Tutti i documenti devono essere realizzati utilizzando un template_g L^AT_EX. Onde evitare modifiche manuali che farebbero perdere molto tempo, il nome dei file_g deve rispondere alla seguente formattazione senza spazi: “[nome documento]-[versione]”. La parte della versione deve riportare la dicitura “v.” seguita dal numero di versione (ad es: NormeDiProgetto_v.1.0.0.pdf”). Tale modello si può trovare nel repository in documents/template.

4.1.2 Contenuto e struttura dei documenti

Ogni documento ufficiale deve essere composto dalle seguenti sezioni:

- Prima pagina: deve riportare titolo, logo ed informazioni del documento;
- Breve prefazione;
- Registro delle modifiche;
- Indice del documento;
- Indice di figure e tabelle (se presenti);
- Introduzione;
- Corpo.

Ogni pagina avrà come intestazione e come piè di pagina:

- **Intestazione:** logo del gruppo e nome del documento;
- **Piè di pagina:** versione documento, università e anno accademico, numeri di pagina e licenza.

4.1.2.1 Verballi

Per quanto riguarda i verbali degli incontri, essi devono essere redatti dal Responsabile di Progetto_g ad ogni riunione. Esso deve rispettare la formattazione regolata alla sezione §4.1.3 e successive ma è da considerarsi solo come promemoria per il gruppo.

Il nome di ogni verbale deve rispettare la seguente dicitura: “Verbale_[tipo incontro]-[data]” dove il tipo incontro può essere di due tipi:

- Interno (INT): incontro effettuato tra i membri del gruppo;
- Esterno (EXT): incontro effettuato tra i membri del gruppo e committente_g e/o proponente_g .

La prima pagina di ogni verbale deve obbligatoriamente contenere i seguenti campi, in ordine:

- Data;
- Luogo secondo il formato “[città],[provincia],[sede]”;
- Ora_g secondo il formato “dalle ore [hh]:[mm] alle ore [hh]:[mm]” dove hh indica le ore e mm i minuti i quali vanno espressi nel formato 24 ore secondo lo standard ISO_g 8601:2004;
- Partecipanti interni al gruppo elencandoli rispettando il formato “[nome] [cognome][, [...]]”;
- Partecipanti esterni al gruppo rispettando il formato “[nome] [cognome][ruolo][, [...]]” in cui il ruolo può essere Committente_g oppure Proponente_g;
- Contenuto dell’incontro;
- Firme: devono essere comprese quelle di tutti i partecipanti del gruppo LateButSafea conferma della presa visione del documento.

4.1.2.2 Lettera di presentazione

La lettera di presentazione deve contenere

- Logo del gruppo;
- Intestazione nel seguente formato:
Prof. Tullio Vardanega
Università degli Studi di Padova
Dipartimento di Matematica
Via Trieste 63
35121 Padova (PD)
- Breve introduzione (facoltativa);
- Elenco di tutti i documenti in consegna;
- Varie ed eventuali, osservazioni (facoltative);
- Firma del responsabile nel seguente formato:
Nome Cognome
il Responsabile del gruppo LateButSafe
Firma del responsabile

4.1.3 Norme tipografiche

Per rendere la documentazione organizzata, leggibile e standard abbiamo adottato le forme testuali riportate di seguito.

- **Carattere:** il carattere dovrà avere come dimensione minima 12. Per l’inserimento di linee di codice_g il carattere da utilizzare dovrà essere di tipo Monospace;

- **NP**: Norme di Progetto_g;
- **PQ**: Piano di Qualifica;
- **PP**: Piano di Progetto_g;
- **SF**: Studio di Fattibilità;
- **RR**: Revisione dei Requisiti_g;
- **RP**: Revisione di Progettazione;
- **RQ**: Revisione di Qualifica;
- **RA**: Revisione di Accettazione.

I **Nomi ricorrenti** nei vari documenti devono rispettare le seguenti indicazioni:

- Ruoli di progetto_g e nomi dei documenti: devono essere formattati utilizzando la prima lettera maiuscola di ogni parola che non sia una preposizione (es. Responsabile di Progetto_g);
- Nomi dei file_g: il riferimento deve essere comprensivo dell'estensione_g del file_g e formattato in corsivo;
- Nomi propri: l'utilizzo dei nomi propri deve seguire il formalismo Cognome Nome;
- Nome del gruppo: deve essere sempre espresso nel formato: LateButSafe;
- Nome del progetto_g: deve essere sempre espresso nel formato: Premi.

4.1.5 Immagini e tabelle

Tutte le immagini devono essere in formato JPG, PNG o PDF mentre ogni tabella deve rispettare il formato L^AT_EX.

Ogni figura o tabella inserita deve avere una breve didascalia composta da un identificativo numerico univoco seguito, ove sia ritenuto necessario, da una breve descrizione. La numerazione di immagini e tabelle sarà attribuita da L^AT_EX.

4.1.6 Glossario

Il glossario è unico per tutti i documenti e deve essere organizzato come definito nella sezione Documenti §4.1.1. Tutti i membri del gruppo possono modificarlo.

I termini all'interno del glossario avranno le seguenti caratteristiche:

- Tutti i termini saranno in ordine alfanumerico;
- Tutti i termini devono essere in grassetto e iniziare con la lettera maiuscola , la definizione del termine sarà preceduta dal carattere ”:” ;
- Tutti i termini devono fornire chiarimenti su concetti che possono essere confusi quindi non devono essere inseriti termini il cui significato è già noto.



4.1.6.1 Implementazione

L'inserimento dei termini nel glossario viene eseguito tramite un applicazione interna al team, LateGloss, che funziona nel seguente modo:

- Si inserisce il lemma e la descrizione del lemma negli appositi spazi;
- Si salva il glossario nel formato .tex;
- A tutte le parole presenti nei documenti che hanno una corrispondente definizione nel glossario verrà aggiunto un pedice (g) per indicare che la parola è presente nel glossario.

L'ordine lessicografico non è importante quando si inseriscono nuovi lemmi nel programma_g dato che vengono ordinati automaticamente.

Il file_g relativo al Glossario è il seguente: [Glossario_v.2.0.0.pdf](#)

4.2 Verifica

La verifica di processi, documenti e prodotti è un'attività da eseguire continuamente durante lo sviluppo del Progetto. Di conseguenza, servono modalità operative chiare e dettagliate per i *Verificatori*, in modo da uniformare le attività di verifica svolte ed ottenere il miglior risultato possibile. Si descrivono ora le modalità ordinate e puntuali di verifica di processi, documenti, attività e codice alle quali ci si riferirà in questo documento e alle quali i *Verificatori* dovranno attenersi.

4.2.1 Metriche per gli errori riscontrati e gestione dei cambiamenti

Si definiscono ora delle metriche per gli errori che i *Verificatori* potranno trovare, fornendo criteri per la quantificazione dell'impatto sul prodotto o sul processo e per la definizione delle priorità di intervento. In questo modo si potrà agire prima nella risoluzione di errori a gravità maggiore.

Errore	Gravità	Priorità risoluzione	Modalità operative
Indici fuori range	Alta	Urgente	Ticket
Ritardi superiori a 4-5 giorni nelle attività	Alta	Urgente	Ticket
Errato tracciamento di requisiti e casi d'uso	Alta	Urgente	Ticket
Errore di progettazione	Alta	Urgente	Ticket

Tab 4: Errori nei processi: gravità e procedure di gestione

Errore	Gravità	Priorità risoluzione	Modalità operative
--------	---------	-------------------------	-----------------------



Tab 5: Errori nei documenti e nel codice: gravità e procedure di gestione

- **Bassa** se l'errore ha impatto su aspetti marginali del prodotto o provoca un basso aumento dei costi o dei tempi del processo;
- **Media** se l'errore ha impatto significativo sul prodotto o provoca un aumento percepibile di tempi e costi;
- **Alta** se l'errore rende il prodotto inutilizzabile o provoca un forte aumento dei tempi o dei costi.

Tab 6: Gravità dell'errore e impatto su processi e prodotti

- **Breve:** indica che l'errore deve essere risolto entro 4-5 giorni;



- **Urgente:** indica che l'errore deve essere risolto appena possibile.

Le modalità operative per il *Verificatore* sono le seguenti:

- **Correzione immediata:** è richiesto che il *Verificatore* proceda autonomamente alla correzione dell'errore;
- **Aggiunta alla checklist:** è richiesto che il *Verificatore* aggiunga l'errore riscontrato ad una checklist appropriata che poi verrà assegnata a un correttore che apporterà le modifiche riportate.

4.2.2 Verifica dei processi

Ai Verificatori è richiesto di effettuare quanto segue:

- **Controllo delle metriche:** Alla conclusione di ogni fase del progetto, per ogni macro-attività, definita nel [PianoDiProgetto_v.2.0.0.pdf](#), si calcolano gli indici definiti nella sezione Metriche per i processi del [PianoDiQualifica_v.2.0.0.pdf](#). Al fine di avere un indice complessivo di fase dovrà essere inoltre calcolato il valore medio di tali indici.
- **Grafico PDCA:** Alla conclusione di ogni fase del progetto il *Verificatore* dovrà esportare i dati dal sistema di ticketing utilizzando l'esportazione mediante foglio di calcolo nel formato CSV. I dati esportati devono essere inseriti in un foglio di calcolo ed importati nel template per la generazione del grafico PDCA.
Dopo aver ottenuto il grafico il *Verificatore* con la supervisione del *Responsabile di Progetto* dovrà trarre delle conclusioni generali sulla velocità con cui sono stati portati avanti i processi.

4.2.3 Verifica dei documenti

Il processo di verifica sarà applicato indipendentemente dalla fase di sviluppo del prodotto (Analisi, Progettazione, Validazione) ogni qual volta avvenga un cambiamento sostanziale nello sviluppo del prodotto, ovvero nei seguenti casi:

- Conclusione della prima redazione di un documento;
- Conclusione della prima redazione di un file_g di codice_g;
- Conclusione della modifica sostanziale di un documento: quando il versionamento passa da .x.y.z a .x.y+1.0 oppure a .x+1.0.0.

Per eseguire un'accurata verifica dei documenti redatti è necessario seguire il seguente protocollo:

1. **Controllo sintattico e del periodo:** Utilizzando TeXstudio e GNU Aspell vengono evidenziati e corretti gli errori di grammatica più evidenti. Gli errori di sintassi, di sostituzione di lettere che provocano la creazione di parole grammaticalmente corrette ma sbagliate nel contesto ed i periodi di difficile comprensione necessitano dell'intervento di un verificatore umano. Per questa ragione ciascun documento dovrà essere sottoposto ad un walkthrough da parte dei verificatori per individuare tali errori;

2. **Rispetto delle norme di progetto:** Sono state definite norme tipografiche di carattere generale. Impongono una struttura dei documenti che non può essere verificata in maniera automatica. La verifica delle norme per cui non è stato definito uno strumento automatico richiede che i Verificatori eseguano inspection sul rispetto di quelle norme in ciascun documento;
3. **Lista di controllo:** Il *Verificatore* dovrà utilizzare la lista di controllo per i documenti, descritta nell'appendice §A, e verificare che gli errori tipici non siano presenti;
4. **Verifica del glossario:** Il *Verificatore* si occuperà del controllo dei termini inseriti nel glossario e della corretta pedicizzazione dei termini nei vari documenti, segnalando eventuali errori;
5. **Calcolo dell'indice Gulpease:** Su ogni documento redatto il *Verificatore* deve calcolare l'indice di leggibilità. Nel caso in cui l'indice risultasse troppo basso, sarà necessario eseguire un walkthrough del documento alla ricerca delle frasi troppo lunghe o complesse;
6. **Miglioramento del processo di verifica:** Per avere un miglioramento del processo di verifica, quando i *Verificatori* eseguono walkthrough di un documento, dovranno riportare gli errori più frequentemente trovati. Grazie a tale pratica sarà possibile eseguire inspection su tali errori nelle verifiche future;
7. **Segnalazione degli errori riscontrati:** il *Verificatore* deve generare ticket secondo quanto descritto nella sezione §6.7.2.3.

4.2.3.1 Verifica diagrammi UML

Al *Verificatore* è richiesto il controllo dei diagrammi UML prodotti:

- **Diagrammi di caso d'uso:** Il controllo dei diagrammi di caso d'uso deve avvenire manualmente, controllando il rispetto delle specifiche UML 2.x e il corretto uso delle relazioni di inclusione ed estensione. Il diagramma di caso d'uso deve rappresentare fedelmente quanto descritto dal caso d'uso;
- **Diagrammi delle classi:** Al *Verificatore* è chiesto il controllo del formalismo delle specifiche UML 2.x e di controllare la corrispondenza tra progettazione e diagrammi delle classi.

4.2.4 Verifica del codice

Al *Verificatore* è richiesto l'avvio dei test statici e dinamici e l'analisi dei risultati. Di seguito un elenco degli strumenti da usare per l'analisi.

4.2.4.1 Analisa Statika

- **jSHint**: tool che permette di rilevare potenziali errori nel codice_g javascript_g;
- **QUnit**: framework_g per i test d'unità del codice_g javascript_g;
- **jsmeter**: strumento per il calcolo di alcune metriche_g del codice_g javascript_g.



4.2.4.2 Analisi Dinamica

Verranno utilizzati strumenti e plugin interni al browser_g *Chrome* quali **SpeedTracer** per verificare la velocità dell'applicazione web_g;

4.2.4.3 Test

- **Jasmine**: framework per behavior-driven per il test sul codice javascript;
- **Mocha**: framework per eseguire test sul codice javascript;
- **Protractor**: framework per eseguire test end to end su angular.js;
- **Karma**: tool per l'automatizzazione dei test javascript;
- **Selenium**: tool per l'automatizzazione dei test sui browser.

4.2.4.4 Validazione codice

La validazione_g del codice_g HTML e CSS_g dell'applicazione da noi sviluppata verrà fatta tramite il servizio W3C_g Validator32 del W3C_g.

4.3 Validazione requisiti

L'attività di validazione consiste nei seguenti attività:

- Preparare i test dei singoli requisiti e la specifica dei test per l'analisi dei risultati;
- Assicurarsi che i requisiti testati riflettano uno specifico uso dell'applicazione;
- Esecuzione dei test la quale a sua volta include:
 - Stress test e casi limite;
 - Testare il prodotto software per la sua abilità di isolare e minimizzare l'effetto degli errori; questo per verificare la robustezza e l'affidabilità del software anche negli stress test e casi limite;
 - Testare che il prodotto software sia in grado di far svolgere all'utente tutti i suoi task.
- Assicurarsi che il software Premi soddisfi gli scopi per cui è stato creato.

4.4 Gestione delle modifiche ai requisiti

A tutte le proposte di modifica dei requisiti_g dovrà essere applicata la seguente procedura:

1. Deduzione, analisi e specifica dei cambiamenti;
2. Stima dei costi del cambiamento considerando quante modifiche dovranno essere fatte ai requisiti_g e al progetto_g del sistema;

3. Decisione ed eventuale implementazione del cambiamento nei requisiti_g e nel progetto_g di sistema.

Per gestire i cambiamenti e per facilitare il tracciamento dei requisiti_g verrà usato un software_g appositamente creato dal gruppo. L'amministratore avrà il compito di gestire il server_g e amministrare i diritti di accesso degli utenti alle funzionalità fornite. In particolare gli analisti dovranno usare i modelli definiti all'inizio della fase di analisi. Per evitare problemi dovuti a modifiche concorrenti alla base dati l'amministratore dovrà garantire che ad ogni istante solo un analista possa modificare un certo sotto albero della foresta dei requisiti_g e dei test.

5 Processi Organizzativi

5.1 Applicazione PDCA

Per poter garantire un costante miglioramento dei processi il gruppo LateButSafeadotterà il PDCA con le seguenti modalità:

- Plan:

- *Determinare gli obiettivi e i destinatari:* Gli Analisti con la supervisione del Responsabile di progetto hanno formulato correttamente tutti gli obiettivi che il gruppo dovrà ottenere. Gli obiettivi devono essere indicati in modo concreto e dettagliato e occorre fornire a tutti i membri del gruppo le informazioni necessarie. Gli obiettivi devono essere quantificati e devono riguardare problemi che il gruppo può risolvere con la collaborazione di tutti i ruoli. Gli obiettivi devono essere presentati al gruppo senza limitazioni di livelli gerarchici. Quanto più il gruppo è orizzontale e privo di frontiere, tanto più sarà facile coinvolgere i componenti nel raggiungimento degli obiettivi;
- *Determinare i metodi per raggiungere gli obiettivi:* Per raggiungere gli obiettivi occorre mettere a punto procedure razionali e facili da seguire. Determinare un metodo significa standardizzarlo e renderlo utile e accessibile. Un metodo e una procedura, però, non possono essere perfetti e solo l'esperienza e l'abilità dei singoli componenti può supplire all'inadeguatezza di standard e regole.

- Do:

- *Svolgere il lavoro:* Nessuna procedura basata su standard, ritenuti erroneamente perfetti, può garantire un'esecuzione priva di difetti. Il singolo componente del gruppo applica quanto sa e ha appreso, tenendo presenti gli standard, ma utilizzando la propria esperienza e abilità. Il singolo componente può però applicare anche solo nel proprio ambito un ciclo PDCA contribuendo in modo determinante al miglioramento continuo del gruppo;
 - *Formazione e istruzione:* La formazione dei componenti è indispensabile per la comprensione, applicazione e miglioramento degli standard di lavoro. La distribuzione dei compiti e la delega di responsabilità, fattore insostituibile per la realizzazione di un sistema qualità, risulta possibile solo con componenti formati;
- **Check:** Lo scopo del controllo è scoprire ciò che viene realizzato in modo non accettabile e contrario ai risultati attesi. Il problema, in questo caso, diventa come scoprire le non conformità. Per far questo vengono in aiuto tutti gli strumenti per l'analisi statica, l'analisi dinamica e i test;
 - **Act:** L'essenziale non è trovare le cause delle criticità, quanto prendere le iniziative adeguate per eliminarle. Non è sufficiente apportare modifiche ai fattori casuali individuati, occorre eliminarli. Correggere e prevenire sono due azioni diverse e separate. Per eliminare le cause delle criticità è necessario risalire fino alla fonte stessa del problema e prendere le misure adeguate.

5.2 Gestione di progetto

Le responsabilità di gestione dell'intero progetto, dalla nascita alla conclusione, sono da attribuire al *Responsabile di Progetto*. Quest'ultimo dovrà garantire un corretto sviluppo delle attività utilizzando, qualora sia possibile, degli strumenti che gli consentano di:

- Pianificare, coordinare e controllare le attività;
- Gestire e controllare le risorse;
- Analizzare e gestire i rischi;
- Elaborare i dati.

5.2.1 Pianificazione delle attività

Per pianificare le attività il *Responsabile di Progetto* deve realizzare un diagramma di Gantt per ciascuna fase indicata nella sezione Pianificazione del [PianoDiProgetto_v.2.0.0.pdf](#) , utilizzando Redmine come descritto nella sezione §6.4.

5.2.2 Coordinazione e controllo delle attività

Per coordinare e controllare le attività il *Responsabile* di Progetto deve riportare la struttura creata con Redmine sfruttando il suo sistema di ticketing come descritto nella sezione §6.7. In questo modo ciascun componente del gruppo sarà avvisato delle attività ad esso assegnate e potrà inserire lo stato delle stesse permettendo al *Responsabile* di verificare immediatamente l'avanzamento del progetto.

5.2.3 Gestione e controllo delle risorse

Per gestire e controllare le risorse il *Responsabile* di Progetto deve utilizzare Redmine come indicato nella sezione §6.7 che gli consente anche di verificare l'avanzamento di ogni processo come riportato nel [PianoDiProgetto v.2.0.0.pdf](#).

5.2.4 Analisi e Gestione dei rischi

Durante l'avanzamento del progetto il *Responsabile di Progetto* deve monitorare costantemente il verificarsi dei rischi descritti nel [PianoDiProgetto_v.2.0.0.pdf](#) ed eventuali nuovi rischi, attuando le contromisure descritte e riportando gli effettivi riscontri.

5.2.5 Elaborazione dati

Il *Responsabile di Progetto* deve sfruttare i fogli di calcolo, come descritto nella sezione §6.5.4, per elaborare i dati raccolti durante lo sviluppo del progetto e riportarli nel *Piano di Progetto*.

5.2.6 Delega

Il *Responsabile di Progetto*, nel caso in cui abbia redatto una parte di un documento, può delegare l'approvazione di tale documento ad un *Verificatore*.

5.2.7 Responsabilità di sotto-progetto

Ogni macroattività può essere assegnata dal *Responsabile* ad un responsabile di sottoprogetto, i cui compiti saranno l'assegnazione delle singole attività alle risorse rese disponibili e la gestione dei cambiamenti.

5.2.7.1 Assegnazione attività

Per assegnare attività alle risorse disponibili, il responsabile di sotto-progetto dovrà seguire le procedure di ticketing descritte in §6.7.2.2.

5.2.7.2 Gestione dei cambiamenti

In caso di errori, in seguito alla notifica da parte del *Verificatore* tramite ticket, il responsabile di sotto-progetto dovrà assegnare la correzione mediante la procedura di ticketing descritta in §6.7.2.2. Al termine della correzione, sarà compito del responsabile di sotto-progetto accettare o respingere la modifica, e richiederne eventualmente il rifacimento. Nel caso la correzione riguardi un'attività di codifica, sarà compito del responsabile di sotto-progetto programmare una nuova esecuzione dei test di unità e di integrazione correlati al modulo modificato.

5.3 Collaborazione

5.3.1 Comunicazioni

5.3.1.1 Comunicazioni interne

Per le comunicazioni interne è stato aperto un gruppo privato su Facebook accessibile ai singoli membri del team.

<https://www.facebook.com/groups/1709354699290988>

Inoltre ogni membro del team dovrà annotare i propri impegni sullo strumento Google Calendar, il quale verrà utilizzato per segnare qualsiasi tipo di impegno: di gruppo e individuale.

In caso di comunicazioni vocali o videoconferenze verrà utilizzato Skype.

5.3.1.2 Comunicazioni esterne

Per quanto riguarda le comunicazioni esterne (verso Committente_g e/o Proponente_g) è stata creata una casella di posta elettronica dedicata, gestita dal *Responsabile di progetto_g*:

latebutsafe@gmail.com

è compito del *Responsabile* gestire le informazioni in entrata e in uscita avvisando il proprio gruppo e il Committente_g/Proponente_g di eventuali comunicazioni rispettivamente in entrata e in uscita.

5.3.2 Riunioni

5.3.2.1 Interne

- Ogni membro del gruppo può richiedere una riunione interna tramite un post all'interno del gruppo su Facebook (tramite l'uso del tag_g [Richiesta Riunione Interna x] con x numero incrementato di 1 rispetto alla richiesta precedente). Questa richiesta in base alle risposte degli altri componenti verrà presa in esame dal *Responsabile*;
- Una volta valutate le motivazioni della richiesta il *Responsabile* controlla sul calendario del gruppo le disponibilità dei vari componenti;
- Il *Responsabile*, entro 1 giorno lavorativo pubblica una nuova discussione con tag_g [Esito Richiesta Interna x], dove annuncia in caso positivo l'orario ed il luogo della riunione, altrimenti annulla la riunione oppure rimanda la richiesta al successivo incontro;
- Nel caso in cui, per diversi motivi, alla riunione non potessero presenziare più di due membri, si procede a fissare una nuova riunione (vedi punto 2 e seguenti).

5.3.2.2 Casi Particolari

Per le richieste di riunioni interne vicine (5 giorni lavorativi) ad una milestone_g, se approvate dal *Responsabile*, verranno indette il giorno stesso o il seguente.

5.3.2.3 Esterne

Per le riunioni esterne (quindi gli incontri con il Proponente/Committente_g) la prassi è la medesima delle riunioni interne; può essere avanzata da qualsiasi membro del gruppo con il tag_g [Richiesta Riunione esterna *x*]. In questo caso il *Responsabile* avrà il duplice compito di valutare la richiesta dopo aver consultato il calendario e di contattare il committente_g, per accordarsi su tempi e luogo dell'incontro, che verranno poi riferiti sulla piattaforma di comunicazioni interne tramite il tag_g [Esito Richiesta Riunione Esterna *x*].

5.3.2.4 Esito

Ad ogni riunione (sia interna che esterna) il *Responsabile* ha il dovere di assicurarsi che venga redatto un verbale che riassume gli argomenti trattati durante l'incontro e tutte le eventuali decisioni prese; i membri del gruppo hanno l'obbligo di applicare le eventuali modifiche o correzioni decise durante la riunione ed è del *Responsabile* il dovere che i problemi emersi durante il verbale siano stati risolti.

5.3.3 Repository e strumenti per la condivisione di file

5.3.3.1 Repository

Sono stati creati due repository Git:

- documents.git disponibile all'indirizzo:



<https://github.com/PetrucchiMauro/documents>

conterrà i sorgenti \LaTeX e gli script necessari alla stesura dei documenti;

- source.git disponibile all'indirizzo:

<https://github.com/PetrucchiMauro/source>

conterrà i sorgenti dell'applicazione.

Una volta terminata la fase di lavorazione di un documento, verrà creato un branch di verifica. In questo modo i Verificatori potranno lavorare parallelamente al resto del gruppo ed effettuare il merge delle loro modifiche, una volta terminato il lavoro di verifica. Il meccanismo di verifica e approvazione è descritto in dettaglio nella sezione §4.2.3.

5.3.3.2 Condivisione file

Per la condivisione informale di file e per il lavoro collaborativo su documenti di supporto, si usa la piattaforma di condivisione file online Google Drive. Trattandosi di strumenti informali, non si definiscono procedure rigorose d'uso e se ne lascia la descrizione alle sezioni §6.3.4.

6 Ambiente di lavoro

6.1 Risorse

6.1.1 Risorse_g necessarie:

6.1.1.1 Risorse_g umane

I ruoli necessari a garantire la qualità del prodotto sono:

- Responsabile di Progetto;
- Amministratore;
- Verificatore;
- Programmatore.

6.1.1.2 Risorse_g Hardware

Saranno necessari:

- Computer con installato software_g necessario allo sviluppo del progetto_g in tutte le sue fasi;
- Luoghi in cui svolgere riunioni, preferibilmente dotati di connessione ad Internet.

6.1.1.3 Risorse_g software

Saranno necessari:

- Strumenti per automatizzare i test;
- Framework_g per eseguire test di unità;
- Piattaforma di versionamento per la creazione e gestione di ticket_g;
- Debugger per i linguaggi di programmazione scelti;
- Browser_g come piattaforma di testing dell'applicazione da sviluppare;
- Strumenti per effettuare l'analisi statica_g del codice_g per misurare le metriche_g.

6.1.2 Risorse_g disponibili

Sono disponibili:

- Computer personali dei membri del gruppo;
- Computer presenti nelle aule informatiche del Dipartimento di Matematica;
- Aule disponibili per incontri nel Dipartimento di Matematica;
- Un dispositivo Raspberry Pi 2 Model B, utilizzato come server_g per programmi_g organizzativi e di testing.

6.1.2.1 Risorse_g software

- Strumenti per il coordinamento §6.3;
- Strumenti per i documenti §6.5;
- Strumenti per la codifica §6.6;
- Strumenti verifica §4.2.3.1.

6.2 Sistemi Operativi

L'intero sviluppo del progetto_g viene svolto in ambienti Unix-Like e Windows_g, nello specifico, Ubuntu_g, Mac, Windows_g. Tale scelta è maturata dopo aver appurato che le tecnologie utilizzate per lo sviluppo del progetto_g sono indipendenti dall'ambiente di sviluppo e di impiego.

6.3 Coordinamento

è stato predisposto un server_g dedicato sul quale sono installate alcune applicazioni web_g che facilitano la gestione del progetto_g. Per connettersi al server_g, l'indirizzo_g è il seguente:

<http://gioberry.no-ip.org/>

6.3.1 Software_g di gestione del progetto

Come piattaforma di gestione del progetto, è stato scelto **Redmine**, che fornisce:

- Un sistema flessibile di gestione dei ticket_g;
- Il grafico Gantt delle attività;
- Un calendario per organizzare i compiti;
- La visualizzazione del repository_g associato al progetto_g;
- Un sistema di rendicontazione del tempo.

6.3.2 Versionamento

Come strumento di versionamento si è deciso di utilizzare **Git**. Git è uno strumento di versionamento veloce e di facile apprendimento che rappresenta uno dei migliori strumenti attualmente esistenti.

Per lo sviluppo collaborativo abbiamo deciso di appoggiarci al servizio **Github** che fornisce non solo un repository `git`, ma anche strumenti utili alla collaborazione fra più persone, come il servizio di **Ticket**, **Wiki** e **Milestone**.

Per quanto riguarda l'uso di Git sui computer di sviluppo, si è deciso l'uso della versione ufficiale rilasciata dal team di sviluppo di Git(2.3.3).

Per interfacciarsi con il repository_g viene utilizzato **SmartGit**, un client multi piattaforma che permette di utilizzare Git in maniera rapida.

Si descrive ora la procedura di corretto utilizzo del programma, SmartGit.

- **Clonare il repository:** è possibile clonare il repository_g remoto in locale attraverso la seguente procedura:
 - Premere nel menu in alto il pulsante Repository_g e successivamente Clone;
 - Nel riquadro comparso, inserire il link_g del repository
<https://github.com/PetrucciMauro/documents.git>
oppure
<https://github.com/PetrucciMauro/source.git>
successivamente premere il pulsante **next**;
 - Nella schermata successiva lasciare spuntati entrambi i box e premere **next**;
 - Selezionare la posizione in cui verrà salvata la versione locale del repository_g.
- **Sincronizzare il repository:** Dalla schermata principale premere il pulsante pull;
- **Salvare una modifica in locale:** Dalla schermata principale premendo il pulsante commit_g e inserendo nell'apposita textbox un Messaggio di commit, si salvano le modifiche effettuate ai file_g;
- **Inviare le modifiche al repository_g remoto:** Dalla schermata principale premere il pulsante Push e, successivamente alla comparsa del nuovo riquadro, ancora push, ciò comporterà l'invio delle modifiche ai file_g al repository_g remoto.

6.3.3 Software_g di Integrazione Continua

Si è scelto di adottare **Travis** per applicare l'integrazione continua allo sviluppo del progetto_g. Tale software_g permette di pianificare ed eseguire dei compiti da eseguire sui file_g sorgente. Mette inoltre a disposizione un cruscotto su cui è possibile visualizzare lo stato del codice_g prodotto. Tale software_g è infatti in grado di interagire con il software_g di versionamento, e se disponibile con software_g per l'esecuzione di test sul codice_g prodotto.

6.3.4 Condivisione dei file

Si è inoltre scelto di utilizzare degli strumenti online che permettono di condividere file_g in modo semplice e veloce e che consentono di organizzare gli appuntamenti personali dei singoli componenti del gruppo.

6.3.4.1 Google Drive

In questa piattaforma di condivisione file_g verranno salvati i documenti che:

- Non necessitano di controllo di versione;
- Hanno bisogno di grande interattività tra i componenti del gruppo;
- Possono essere acceduti tramite l'uso di un semplice browser_g.

Questo strumento permetterà a 2 o più componenti del gruppo di interagire lavorando sugli stessi documenti contemporaneamente. Google Drive viene utilizzato come strumento di supporto allo sviluppo della documentazione e del software, presente su Git .

6.3.5 Google Calendar

Google Calendar viene utilizzato all'interno del gruppo per gestire le risorse umane_g. In particolare tale strumento viene utilizzato per notificare in quali giorni un determinato membro non può essere disponibile e per segnalare date rilevanti per il gruppo, come ad esempio le date delle riunioni.

6.4 Pianificazione

Per pianificare le attività legate allo sviluppo del progetto e la gestione delle risorse si è scelto di utilizzare *Redmine*. *Redmine* è un programma per il project management. I motivi per cui è stato scelto questo software vengono descritti nella sezione §6.3.1

6.5 Strumenti per i documenti

6.5.1 LATEX

Per la stesura dei documenti è stato scelto di utilizzare il sistema \LaTeX .

Il motivo principale dietro a questa scelta è la facilità nel separare il contenuto dalla formattazione: con \LaTeX è possibile definire l'aspetto delle pagine in un `file_g template_g` condiviso da tutti i documenti. Altre soluzioni come Microsoft Office, LibreOffice o Google Docs non avrebbero consentito questa separazione, duplicando il lavoro di formattazione del testo e non garantendo un risultato uniforme.

Il grande numero di pacchetti_g esistenti consente di implementare funzionalità comuni in maniera semplice. L'estensibilità_g di L^AT_EX può essere sfruttata per creare funzioni_g e variabili globali che rendono la scrittura del contenuto più corretta da un punto di vista semantico. Un esempio è dato dal comando `/role{ruolo}` che identifica ogni ruolo all'interno del progetto_g.

Per la scrittura di documenti L^AT_EX l'editor_g consigliato è **TeXstudio**.

6.5.2 Controllo ortografico

Il software_g per il controllo ortografico è **Aspell** . Il programma_g viene richiamato da linea di comando.

6.5.3 Grafici UML

Per la stesura dei grafici UML viene utilizzato il programma_g **Visual Paradigm**. Il programma_g viene utilizzato in licenza Community Edition che ne permette l'uso gratuito per fini non commerciali.

6.5.4 Fogli di calcolo

L'utilizzo di fogli di calcolo_g quali Calc, Excel e Numbers è a discrezione del singolo componente. I fogli di calcolo vengono usati per:

- Grafici a torta per l'utilizzo delle risorse;
- Grafici a torta per il costo dedicato a ciascuna risorsa;

- Istogrammi per le ore assegnate ad ogni componente del gruppo;
- Tabelle per il confronto tra preventivo e consuntivo;
- Istogrammi per il confronto tra ore preventivate e ore realmente impiegate da ciascuna risorsa.

6.6 Strumenti per la codifica

6.6.1 Stesura

Per al stesura del codice HTML e javascript verrà usata l'IDE *Aptana*, si è scelto d'utilizzarla perché fornisce un ambiente di sviluppo per applicazioni web completo, gratuito e multiplatforma. *Aptana* presenta molti strumenti già integrati, come code-assistant e debugger.

6.6.2 Verifica

L'analisi statica e dinamica integrate in Travis vengono lanciate automaticamente ad ogni push di codice nel repository. Al verificatore è richiesto quindi soltanto di analizzare l'output del tool, visualizzabile dalla dashboard di Travis.

6.7 Protocollo per lo sviluppo dell'applicazione

Per procedere con uno sviluppo controllato dei documenti e del codice_g si è scelto di adottare il sistema di ticketing_g **Redmine**.

Le motivazioni alla base della scelta di tale software_g sono descritte nella sezione §6.3.1.

6.7.1 Creare un nuovo progetto

La creazione di un progetto_g è compito del *Responsabile di Progetto*.

Un nuovo progetto_g rappresenta una macro-attività caratterizzata da molte sotto-attività supervisionate da un responsabile.

Per creare un nuovo progetto_g:

- Aprire **Progetti**;
- Selezionare **Nuovo progetto**;
- Assegnare un **Nome** breve ma significativo;
- Nel caso in cui si voglia creare un sotto-progetto_g indicare il nome del progetto_g padre dall'omonimo campo;
- **Identificativo**: scrivere in minuscolo ed indicare codice_g della fase a cui si riferisce;
- Lasciare inalterati gli altri campi.



6.7.2 Creazione ticket

I ticket_g vengono creati da:

- **Responsabile di Progetto:** crea i ticket_g più importanti che rappresentano le macro fasi evidenziate dalla pianificazione;
- **Responsabile di Sotto-progetto:** crea i ticket_g per i processi_g non pianificati inizialmente, che si evidenziano necessari per l'avanzamento del sotto-progetto_g assegnato;
- **Verificatore:** crea i ticket_g per segnalare errori ed imprecisioni trovate durante il processo_g di verifica.

I ticket_g possono essere di tre tipologie:

- **Ticket_g di pianificazione:** rappresentano le macro-attività di maggiore importanza. Sono organizzate in una gerarchia con vari livelli di priorità. Tali attività vengono create da:
 - *Responsabile di Progetto:* durante la pianificazione identifica le attività più importanti e generali;
 - *Responsabile di Sotto-progetto:* durante lo svolgimento delle attività può scomporre in sotto-problemi l'attività indicata dal Responsabile di Progetto_g.
- **Ticket_g di realizzazione e controllo:** tutti i documenti redatti, durante la stesura attraversano due stadi:
 - **Realizzazione:** un redattore del documento effettua una prima stesura;
 - **Controllo:** un redattore, diverso da quello della precedente fase, esegue un primo controllo sui contenuti della parte scritta.
- **Ticket_g di verifica:** rappresentano gli errori identificati dai Verificatori durante il controllo che la realizzazione dell'attività sia conforme a quanto richiesto e che rispetti tutte le norme.

6.7.2.1 Ticket_g di pianificazione

- Selezionare **Nuova segnalazione** da menù principale;
- **Tracker:** indicare la natura del ticket_g:
 - **Documento:** stesura di un documento. Il tipo di attività svolta dal redattore del documento viene definito durante la rendicontazione;
 - **Codifica:** stesura di codice_g;
 - **Verifica:** macro-attività di verifica sul prodotto dei sotto-processi_g.
- **Oggetto:** descrizione breve e significativa;
- **Descrizione:** descrizione comprensibile e con riferimenti esterni mediante link_g se necessario;

- **Stato:** Plan;
- **Attività principale:** se si vuole creare una **sotto-attività** indicare l'id del ticket_g padre;
- **Categoria:** PDCA, solo se il ticket_g viene creato dal *Responsabile di Progetto*;
- **Assegnato a:** inserire il nome del responsabile;
- **Osservatori:** aggiungere eventuali collaboratori.

6.7.2.2 Ticket_g di realizzazione e controllo

- Selezionare **Nuova segnalazione** da menù principale;
- **Tracker**: indicare la natura del ticket_g:
 - **Documento**: stesura di un documento. Il tipo di attività svolta dal redattore del documento viene definito durante la rendicontazione;
 - **Codifica**: stesura di codice_g;
 - **Verifica**: attività di verifica sui prodotti dei processi_g.
- **Oggetto**: descrizione breve e significativa secondo il principio: nome ticket_g padre attività da svolgere (realizzazione o controllo);
- **Descrizione**: descrizione comprensibile e con riferimenti esterni mediante link_g se necessario;
- **Stato**: New;
- **Attività principale**: se si vuole creare una **sotto-attività** indicare l'id del ticket_g padre;
- **Inizio**: dare una data di inizio presunta;
- **Scadenza**: dare una data di fine presunta;
- **Assegnato a**: inserire il nome del responsabile;
- **Osservatori**: aggiungere eventuali collaboratori.

6.7.2.3 Ticket_g di verifica

Un *Verificatore* per creare un *ticket_g di verifica* deve:

1. Assicurarsi che esista all'interno del progetto_g l'attività *Verifica*. Su tale attività vi devono essere due sotto-attività: "Verifica - realizzazione", "Verifica - approvazione".
Tutti i ticket_g creati devono essere sotto-attività di: "Verifica - realizzazione";
2. Creare quindi il ticket_g secondo le seguenti direttive:
 - Selezionare **Nuova segnalazione** da menù principale;
 - **Tracker**: Bug;

- **Oggetto:** descrizione breve e significativa dell'errore incontrato;
- **Descrizione:** descrivere in modo dettagliato e chiaro: la natura e la posizione dell'errore;
- **Stato:** New;
- **Attività principale:** tutti i ticket_g devono essere figli del ticket_g "Verifica - realizzazione" del progetto_g su cui si sta eseguendo la verifica;
- **Assegnato a:** inserire il nome del responsabile del progetto_g padre (es. responsabile delle *Norme di Progetto*).

Tutti i campi non segnalati sono da lasciare vuoti. Sarà poi compito del responsabile del progetto_g padre decidere a chi assegnare la correzione dell'errore. Nel caso in cui l'errore segnalato non sia considerato valido dal *Responsabile del sotto-progetto* verrà confermato il rifiuto dal *Responsabile di Progetto*.

6.7.2.4 Dipendenze temporali

Dopo la creazione del ticket_g , per aggiungere **dipendenze temporali** tra i ticket_g :

- Andare su **segnalazioni**;
- Aprire il link_g alla segnalazione a cui aggiungere la dipendenza;
- Nella sezione **segnalazioni correlate** premere **aggiungi**;
- Scegliere **segue** e indicare il numero della segnalazione che lo blocca ed eventuali giorni di slack.

Tutti i campi non segnalati sono da lasciare vuoti.

6.7.3 Aggiornamento ticket

Esistendo due tipologie di ticket_g , viene qui definito la procedura per effettuare l'aggiornamento di entrambe.

6.7.3.1 Ticket_g di pianificazione

- Andare sul menù **Segnalazioni**;
- Selezionare il ticket_g di interesse;
- Cliccare il link_g **Aggiorna**;
- Commentare ciò che si è fatto sulla form **Note**;
- Cambiare lo stato del ticket_g secondo la seguente logica:
 - **Do**: quando un ticket_g è in questo stato indica che una o più persone stanno lavorando su tale attività;
 - **Check**: quando un ticket_g è in questo stato indica che una o più persone stanno lavorando sulla verifica di tale attività;

- **Act:** l'attività è stata conclusa e verificata, e ne sono state tratte le conclusioni adeguate.

- Se viene concluso, aggiornare lo stato del ticket_g di pianificazione padre.

6.7.3.2 Ticket_g di realizzazione e controllo

- Andare sul menù **Segnalazioni**;
- Selezionare il ticket_g di interesse;
- Cliccare il link_g **Aggiorna**;
- Indicare il tempo impiegato in ore;
- Indicare il tipo di attività svolta;
- Commentare ciò che si è fatto sulla form **Note**;
- Cambiare lo stato del ticket_g secondo la seguente logica:
 - **In Progress**: quando un ticket_g è in questo stato indica che una o più persone stanno lavorando su tale attività. La percentuale di completamento deve essere impostata tra lo 0% ed il 90%;
 - **Closed**: l'attività è stata conclusa. La percentuale di completamento dell'attività è al 100%.
- Aggiornare lo stato del ticket_g di pianificazione padre secondo tali principi:
 - Ticket_g figlio passa da New a In Progress: il ticket_g padre passa da Plan a Do, o da Do a Check;
 - Ticket_g figlio passa a Closed: il ticket_g padre deve essere in Do o Check;
 - Tutti i ticket_σ figli vengono chiusi: il ticket_σ padre passa ad Act.

6.7.3.3 Ticket_g di verifica

- Andare sul menù **Segnalazioni**;
- Selezionare il ticket_g di interesse;
- Cliccare il link_g **Aggiorna**;
- Indicare il tempo impiegato in ore;
- Indicare Verifica come tipo di attività svolta;
- Commentare le correzione nella form **Note**;
- Cambiare lo stato del ticket_g secondo la seguente logica:

- **In Progress:** quando un ticket_g è in questo stato indica che una o più persone stanno lavorando su tale attività. La percentuale di completamento deve essere impostata tra lo 0% ed il 90%;
 - **Closed:** l'attività è stata conclusa. La percentuale di completamento dell'attività è al 100%;
 - **Rejected:** l'attività di verifica è stata rifiutata dal *Responsabile del sotto progetto* in accordo con il *Responsabile di Progetto*.
- Aggiornare lo stato del ticket_g di pianificazione padre secondo tali principi:
 - Ticket_g figlio passa da New a In Progress: il ticket_g padre passa da Plan a Do, o da Do a Check;
 - Ticket_g figlio passa a Closed: il ticket_g padre deve essere in Do o Check;
 - Tutti i ticket_g figli vengono chiusi: il ticket_g padre passa ad Act.

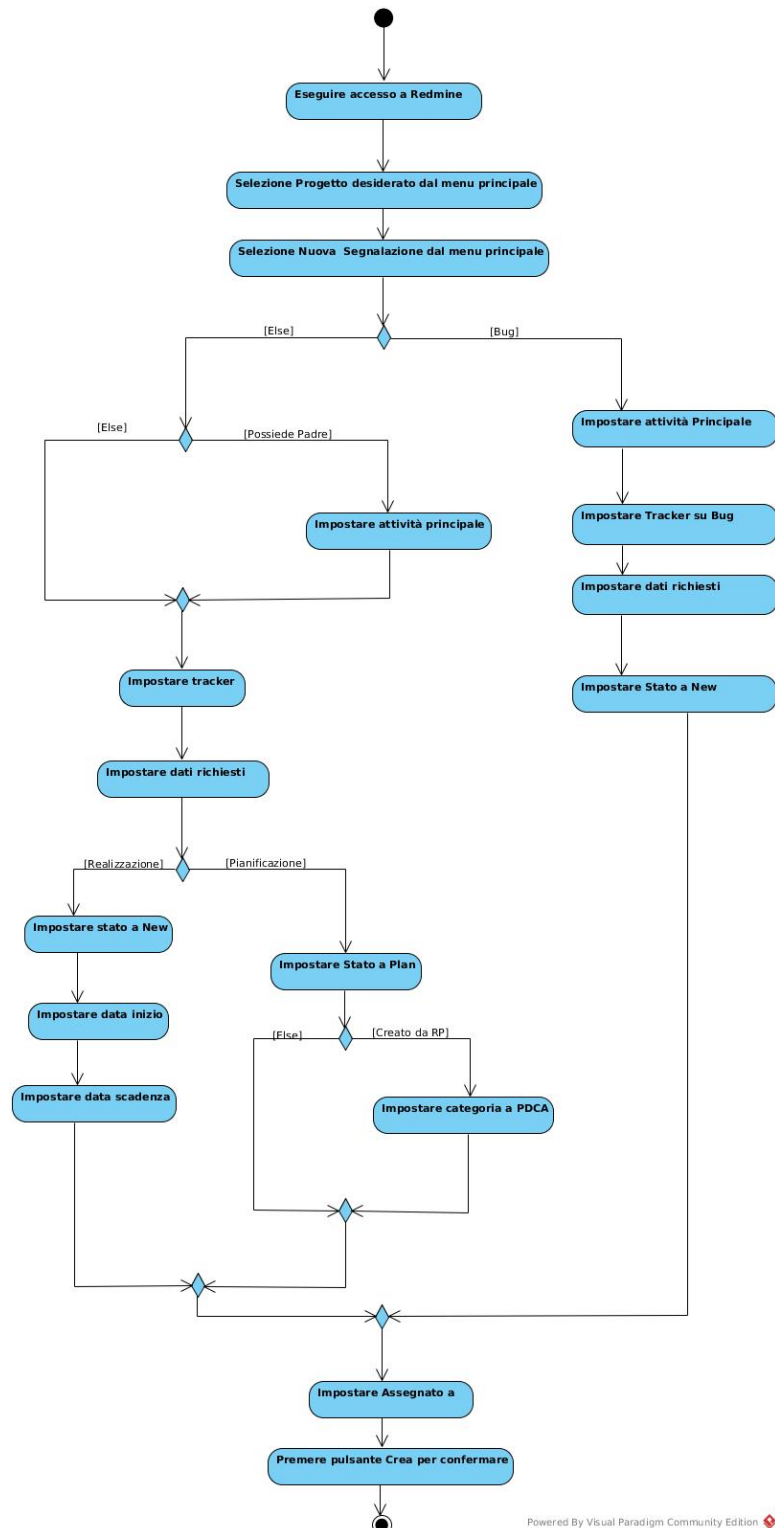


Fig 1: Diagramma attività - Creazione nuovo ticket

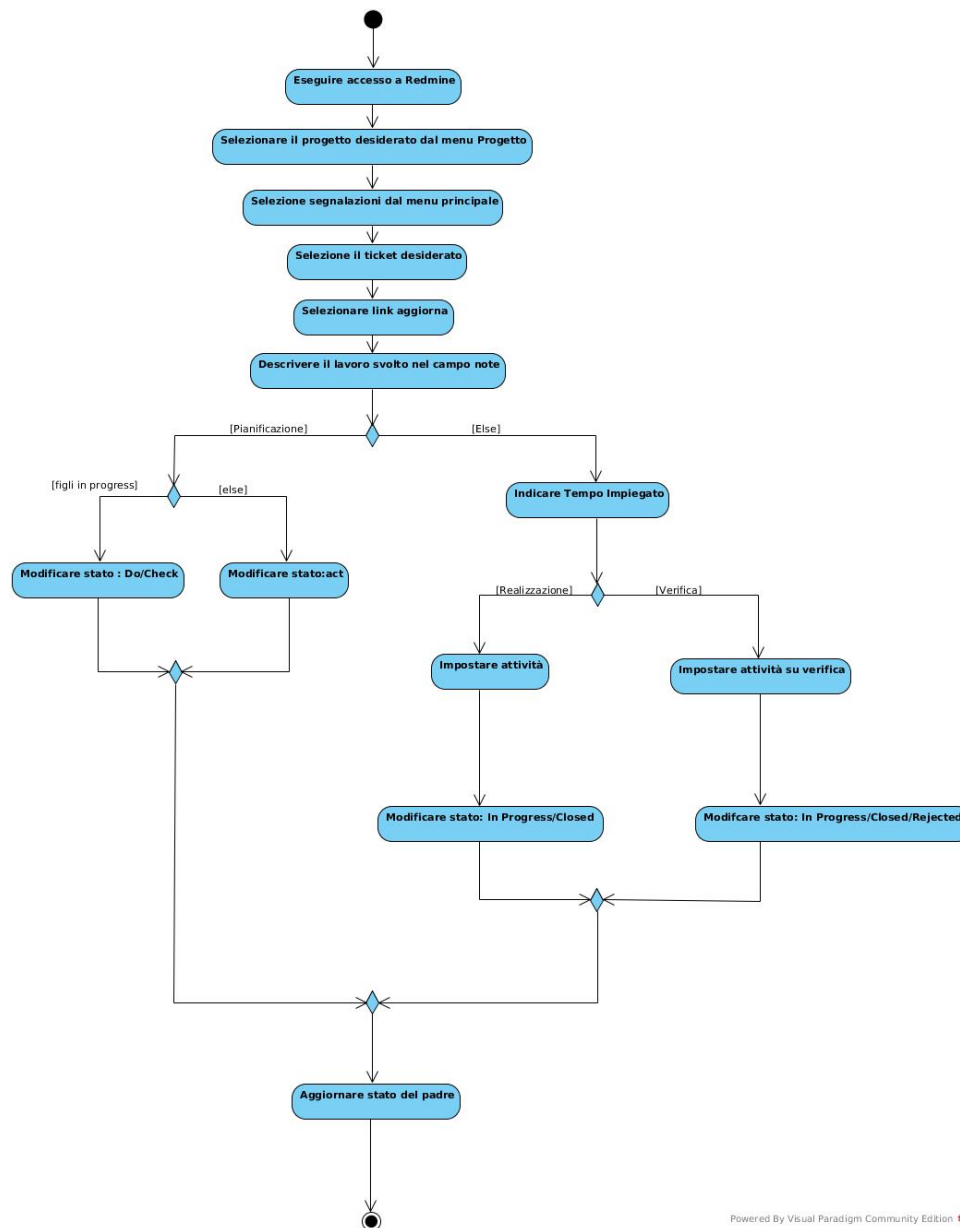


Fig 2: Diagramma attività -Aggiornamento ticket

6.7.4 Consigli di utilizzo


Per avere una immediata visualizzazione dei ticket_g assegnati, è consigliato personalizzare la pagina personale:

- Andare alla **Pagina personale**;
- Cliccare il link_g **Personalizza la pagina**;
- Dal menù a tendina **La mia pagina di blocco**, selezionare **Le mie segnalazioni** e premere il pulsante verde +;
- Ripetere il punto precedente per aggiungere **Segnalazioni osservate**.

Per avere una visualizzazione più chiara delle segnalazioni si consiglia di ordinarle per oggetto. Tale risultato può essere ottenuto premendo **Oggetto** dalla pagina **Segnalazioni**.

6.8 LateTack

Per semplificare il tracciamento dei requisiti è stato realizzato internamente il software *Late-Track*, un'applicazione web accessibile tramite browser ed ospitata sul server dedicato. Quest'applicazione consente di gestire il tracciamento dei requisiti.



LateButSafe

Analisi dei Requisiti

Tutti

Funzionali

Qualità

Vincoli

ADD NEW	Nome	Importanza	Descrizione	Fonti
EDIT	RF 1	Obbligatorio	L'utente deve essere in grado di registrarsi al sistema	UC 0.8
EDIT	-> RF 1.1	Obbligatorio	L'utente deve essere in grado di immettere uno username che lo identifichi univocamente	UC 0.8.1
EDIT	-> RF 1.2	Obbligatorio	L'utente deve essere in grado di immettere una password	UC 0.8.2
EDIT	RF 3	Obbligatorio	L'utente deve essere in grado di autenticarsi al sistema con un account valido inserendo le proprie credenziali	UC 0.10
EDIT	-> RF 3.1	Obbligatorio	L'utente deve essere in grado di inserire il proprio username	UC 0.10.1
EDIT	-> RF 3.2	Obbligatorio	L'utente deve essere in grado di inserire la propria password	UC 0.10.2
EDIT	RF 4	Obbligatorio	L'utente deve essere in grado di creare una nuova presentazione vuota	UC 1.2
EDIT	RF 7	Obbligatorio	L'utente deve essere in grado di entrare in modalità modifica	UC 1.3
EDIT	-> RF 7.1	Obbligatorio	L'utente deve essere in grado di modificare una presentazione da dispositivo desktop	UC 1.3
EDIT	--> RF 7.1.1	Obbligatorio	L'utente deve essere in grado di inserire un nuovo frame sul piano della presentazione	UC 1.3.1.1
EDIT	---> RF 7.1.1.1	Obbligatorio	L'utente deve essere in grado di selezionare il tipo di frame/ped(g) da inserire	UC 1.3.1

Fig 3: ScreenShot del programma LateTrack

6.8.1 Aggiunta nuovo requisito

Per aggiungere un nuovo requisito al progetto è necessario eseguire i passi descritti nel diagramma in figura §4.

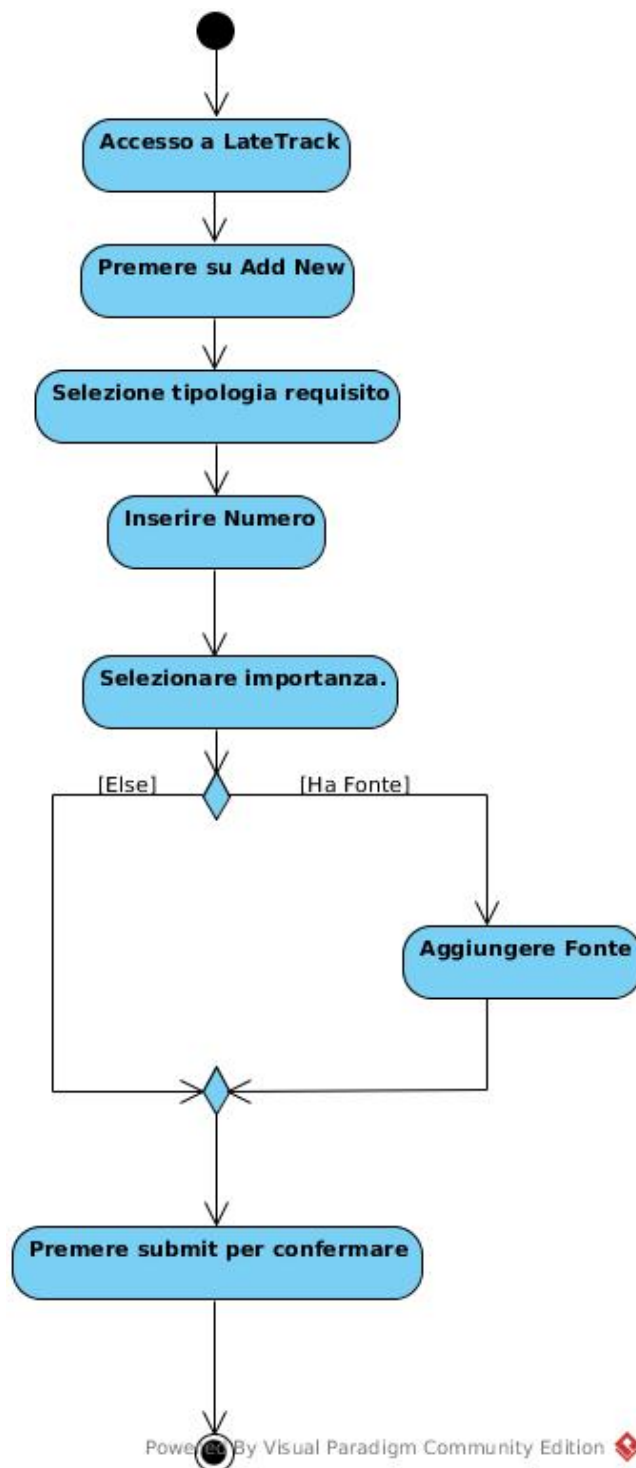


Fig 4: Creazione nuovo requisito

A Lista di controllo

Durante l'applicazione del walkthrough ai documenti, sono state riportate le tipologie di errori più frequenti. La lista di controllo risultante è la seguente:

- **Norme stilistiche:**

- Elenco puntato: non inizia con la lettera maiuscola;
- Elenco puntato: non termina con il punto e virgola oppure con il punto se è l'ultimo elemento;
- Elenco numerato: non termina con il punto e virgola oppure con il punto se è l'ultimo elemento;
- Nome proprio di persona: non rispetta la norma Cognome Nome;
- Parole Proponente e Committente: non vengono scritte con la maiuscola iniziale.

- Italiano:

- Periodi: frasi troppo lunghe rendono i concetti di difficile comprensione;
- Doppie negazioni: evitare l'utilizzo di doppie negazioni perché complicano la comprensione della frase;
- Punto e virgola: evitare l'uso del punto e virgola quando è necessario usare il punto;
- Proponente e Committente: non si deve confondere il loro significato.

- L^AT_EX:

- Lettere accentate nelle variabili: non viene utilizzato il comando apposito;
- Carattere di spaziatura: non deve essere utilizzato all'interno dei tag;
- Macro `\LaTeX`: non viene scritta usando l'apposito comando.

- UML:

- Il sistema non deve mai essere un attore;
- Controllo ortografico: deve essere effettuato in modo dettagliato a causa dell'impossibilità di automatizzare i controlli sui diagrammi;
- Direzione delle frecce non corrette;
- Consistenza della nomenclatura tra i diagrammi e le descrizioni testuali nei documenti.

La seguente lista di controllo vuole riassumere invece gli errori più frequenti rilevati durante il walkthrough del tracciamento requisiti :

- Tracciamento requisiti:

- Ad ogni caso d’uso deve corrispondere almeno un requisito;
- Ad ogni requisito deve corrispondere almeno una fonte;
- La fonte “Capitolato” non deve comparire nei requisiti interni;
- Deve esserci copertura totale del capitolato nei requisiti.