Obelix Group
obelixswe@gmail.com

Norme di Progetto

Versione	*v4_0_0*
Data creazione	2017/02/20
Redattori	Emanuele Crespan
	Federica Schifano
Verificatori	Riccardo Saggese
Approvazione	Silvio Meneguzzo
Stato	Approvato
Uso	interno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo Obelix

Sommario

Documento contenente le norme di progetto che il gruppo Obelix seguirà durante tutte le fasi di realizzazione del prodotto Monolith.

Diario delle revisioni

Versione	Modifica	Autore e Ruolo	Data
4.0.0	Approvazione documento	Silvio Meneguzzo Responsabile	2017-07-04
3.1.0	Verifica sezioni "Processi Primari" e "Processi di Supporto"	Riccardo Saggese Verificatore	2017-07-03
3.0.7	Aggiunta sottosezione 3.7 "Processi per la risoluzione dei problemi"	Emanuele Crespan Amministratore	2017-07-03
3.0.6	Aggiunta sottosezione 3.6 "Strumenti per l'analisi dinamica"	Federica Schifano Amministratore	2017-07-02
3.0.5	Modifica sottosezione 3.4 "Verifica"	Emanuele Crespan Amministratore	2017-07-01
3.0.4	Aggiunta sottosezione 3.3 "Processo di garanzia di qualità del prodotto"	Federica Schifano Amministratore	2017-07-01
3.0.3	Aggiunta sottosezione 2.2.5 "Integrazione"	Emanuele Crespan Amministratore	2017-06-29
3.0.2	Modifica sottosezione 2.2.2 "Progettazione"	Federica Schifano Amministratore	2017-06-28
3.0.1	Modifica sezione 2.1 "Fornitura"	Federica Schifano Amministratore	2017-06-27
3.0.0	Approvazione documento	Nicolò Rigato Responsabile	2017-04-28
2.3.0	Verifica sezioni "Processi Organizzativi" e "Processi di Supporto"	Federica Schifano Verificatore	2017-04-27
2.2.3	Aggiunta sottosezione "Gestione della configurazione" alla sezione "Processi di Supporto"	Riccardo Saggese Amministratore	2017-04-26
2.2.2	Aggiunta sottosezione "Procedure di supporto" alla sezione "Processi Organizzativi"	Emanuele Crespan Amministratore	2017-04-25
2.2.1	Aggiunta sottosezione "Ticketing" alla sezione "Processi Organizzativi"	Emanuele Crespan Amministratore	2017-04-25

Versione	Modifica	Autore e Ruolo	Data
2.2.0	Verifica sottosezione "Progettazione"	Silvio Meneguzzo Verificatore	2017-04-25
2.1.1	Modifica sottosezione "Progettazione"	Riccardo Saggese Amministratore	2017-04-24
2.1.0	Verifica lista di controllo	Federica Schifano Verificatore	2017-04-24
2.0.1	Aggiunta lista di controllo all'appendice	Emanuele Crespan Amministratore	2017-04-23
2.0.0	Approvazione del documento	Tomas Mali Responsabile	2017-04-22
1.1.0	Verifica sottosezioni Riferimenti e Sviluppo	Silvio Meneguzzo Verificatore	2017-03-01
1.0.1	Modifica sottosezioni Riferimenti e Sviluppo	Tomas Mali Amministratore	2017-03-01
1.0.0	Approvazione del documento	Federica Schifano Responsabile	2017-02-24
0.1.0	Verifica del documento	Riccardo Saggese Verificatore	2017-02-24
0.0.4	Stesura sezione Processi Organizzativi	Emanuele Crespan Amministratore	2017-02-23
0.0.4	Stesura sezione Processi di supporto	Tomas Mali Amministratore	2017-02-22
0.0.3	Stesura sezione Processi Primari	Emanuele Crespan Amministratore	2017-02-21
0.0.2	Stesura sezione Introduzione	Tomas Mali Amministratore	2017-02-20
0.0.1	Creato template	Tomas Mali Amministratore	2017-02-20

Indice

1 Introduzione

1.1 Scopo del documento

Il documento definisce le norme che i membri del gruppo Obelix dovranno seguire nello svolgimento del progetto Monolith. Tutti i membri del gruppo devono prendere visione del documento. É necessario adottare le norme in esso contenute per ottenere massima coerenza ed efficienza durante lo svolgimento delle attività.

1.2 Scopo del prodotto

Lo scopo del prodotto è quello di permettere la creazione di bolle interattive,che dovranno funzionare nell'ambiente Rocket.chat. Queste bolle permetteranno di aumentare l'interattività tra gli utenti della chat, aggiungendo nuove funzionalità che saranno accessibili direttamente dalla conversazione senza il bisogno di ricorrere all'apertura di applicazioni diverse. Il sistema offrirà agli sviluppatori un set di $API_{|G|}$ per creare e rilasciare nuove bolle e agli utenti finali la possibilità di usufruire di un insieme di bolle predefinite.

1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensibilità dei documenti. I termini che necessitano di essere chiariti saranno marcati con una |G| in pedice alla prima occorrenza e saranno riportati nel *Glossario v3.0.0*.

1.4 Riferimenti

1.4.1 Informativi

- **Javascript 6th edition**:
  `https://exploringjs.com/es6/ch_promises.html`
- **12 Factors app guidelines**:
  `https://12factor.net`
- **SCSS**:
  `http://sass-lang.com`
- **Piano di Progetto**:
  Piano di progetto v4.0.0
- **Piano di Qualifica**:
  Piano di qualifica v4.0.0

1.4.2 Normativi

- **The Airbnb Javascript style guide**:
  `https://github.com/airbnb/javascript`
- **ESLint**:
  `https://github.com/eslint/eslint`

2 Processi primari

2.1 Fornitura

2.1.1 Studio di fattibilità

Il documento inerente allo studio di fattibilità dovrà essere stilato dagli *Analisti* che effettueranno un resoconto delle prime riunioni tra i membri del gruppo. Saranno sviluppati i seguenti punti relativi al capitolato scelto:

- Scopo del progetto
- Dominio applicativo
- Dominio tecnologico
- Criticità e rischi durante lo sviluppo del prodotto

Infine saranno presentate le motivazioni che hanno condotto il gruppo nella scelta del capitolato.

2.1.2 Processi di controllo

É compito dei membri del gruppo eseguire i processi descritti in questo documento e di moritorarne lo stato al fine di individuare, analizzare e risolvere i problemi che potrebbero verificarsi.

2.1.3 Verificazione e Validazione

É compito dei Verificatori attuare le attività di verifica e validazione dei documenti e dei processi per il prodotto. Con i risultati ottenuti si deve valutare lo stato dei processi e nel caso qualche processo non superi la verifica si dovrà discutere con i membri del gruppo Obelix su come intervenire per migliare la qualità del processo in questione.

2.2 Sviluppo

2.2.1 Analisi dei requisiti

Gli *Analisti* si occuperanno,inoltre, della stesura dell'analisi dei requisiti. Saranno analizzati e classificati i requisiti che il prodotto finale deve soddisfare.

Classificazione dei casi d'uso Gli *Analisti* dovranno analizzare i casi d'uso (UC - Use Case) procedendo dal generale al particolare. Ogni caso d'uso sarà identificato nel seguente modo:

UC[codice univoco del padre].[codice progressivo di livello]$-$[sigla identificativa]

Il codice progressivo può includere diversi livelli di gerarchia separati da un punto. L'uso della sigla identificativa è opzionale.

Per ogni caso d'uso saranno specificati:

- Titolo sintetico e significativo
- Attori principali

- Attori secondari, se presenti
- Precondizione
- Scenario principale e relativo flusso degli eventi
- Scenari secondari,se presenti
- Postcondizione
- Generalizzazioni, se presenti
- Inclusioni, se presenti
- Estensioni, se presenti

Classificazione dei requisiti Dopo aver stilato gli use case gli *Analisti* determineranno i requisiti che saranno organizzati per utilità strategica:

- Ob: Obbligatori
- De: Desiderabili
- Op: Opzionali

Ogni requisito dovrà essere distinto,inoltre ,in base al tipo e quindi al modo in cui dovrà essere verificato:

- Fu: Funzionale
- Qu: Qualitativo
- Di: Dichiarativo

Ogni requisito dovrà rispettare la seguente codifica:

$$R[\text{utilità strategica}][\text{tipo}][\text{codice}]$$

Il codice univoco dovrà essere espresso in forma gerarchica.

Per ogni requisito saranno inseriti una descrizione, il più possibile chiara e priva di ambiguità, e la fonte da cui è emerso:

- Capitolato
- Incontri/Conferenze con il proponente
- Discussioni interne al gruppo
- Casi d'uso

2.2.2 Progettazione

L'attività di progettazione consiste nel tracciamento delle linee guida che dovranno essere seguite per la codifica. Alla fine di questa attività sarà, dunque, definita l'architettura del sistema da realizzare. L'architettura sarà suddivisa in componenti distinte che dipendono il meno possibile le une dalle altre. Ogni componente dovrà soddisfare almeno un requisito e tutti i requisiti dovranno essere soddisfatti.

Il documento derivante da questa attività è Definizione di Prodotto. Alcune sezioni del documento saranno dedicate alla descrizione della progettazione ad alto livello del sistema e dei singoli componenti. Le altre sezioni del documento dovranno riguardare la progettazione di dettaglio del sistema: ogni singola unità di cui è composto il sistema sarà definita in modo dettagliato.

Descrizione progettazione ad alto livello I contenuti da coprire sono i seguenti:

- **Diagrammi UML**:
  - Diagrammi dei package
  - Diagrammi delle classi
  - Diagrammi di sequenza
  - Diagrammi di attività
- **Design Pattern**:

  Devono essere descritti i design patterns utilizzati per realizzare l'architettura. Si deve includere una breve descrizione e un diagramma che esponga il significato e la struttura di ciascuno.
- **Tracciamento dei componenti**: Ogni requisito deve essere tracciato al componente che lo soddisfa. In questo modo è possibile verificare che vengano soddisfatti tutti i requisiti e che ogni componente soddisfi almeno un requisito.

  I *Progettisti* devono definire dei test per verificare la corretta integrazione tra le componenti del prodotto.
- **Definizione test di Sistema**:

  I *Progettisti* devono definire dei test per verificare che tutti i requisiti siano verificati.
- **Definizione test di Validazione**:

  I *Progettisti* devono definire dei test che coprano tutte le funzionalità offerte dal prodotto in modo da mostrarne la correttezza.

Descrizione progettazione di dettaglio I contenuti da coprire sono i seguenti:

- **Diagrammi UML**:
  - Diagrammi delle classi

  - Diagrammi di sequenza
  - Diagrammi di attività

- **Definizioni delle classi**:

  Ogni classe progettata deve essere descritta in modo da spiegarne lo scopo e definirne le funzionalità ad essa associate. Per ogni classe dovranno anche essere definiti i vari metodi ed attributi che la caratterizzano.

- **Tracciamento delle classi**:

  Ogni classe deve essere tracciata ed associata ad almeno un requisito, in questo modo è possibile avere la certezza che tutti i requisiti siano soddisfatti e che ogni classe soddisfi almeno un requisito.

- **Definizione test di Unità**:

  I *Progettisti* devono definire dei test per verificare il corretto funzionamento di una classe. Tali test verranno poi implementati dai *Programmatori*.

2.2.3 Codifica

Lo scopo dell'attività di codifica è quello di realizzare il prodotto software seguendo le linee guida delineate nell'attività di progettazione. É necessario che tutti i *Programmatori* utilizzino lo stesso stile di codifica per garantire la creazione di codice uniforme. Le norme stilistiche riguardano:

- Formattazione
- Intestazione
- Nomi di elementi
- Commenti
- Stile di codifica

Formattazione

- Indentazione: è richiesto l'uso di esattamente quattro spazi
- É necessario rientrare le righe di codice secondo la relativa costruzione logica
- É richiesto di inserire le parentesi di delimitazione dei costrutti di controllo in linea
- É richiesto di inserire le parentesi di delimitazione dei costrutti di dichiarazione di funzioni o classi al di sotto di essi e non in linea

Intestazione L'intestazione di ogni file deve contenere obbligatoriamente le seguenti informazioni:

- Name: nome del file
- Location: percorso di locazione del file
- Author: autore del file
- Creation data: data di creazione del file
- Description: descrizione del file
- History:
  - Version: codice univoco indicante la versione del file
  - Update data: data ultima di modifica
  - Description: descrizione della modifica
  - Author: autore della modifica

```
/*
* Name : { Nome del file }
* Location : {/ path / della / cartella /}
* Author: {Autore del file}
* Creation Data: {Data di creazione del file}
* Description: {Breve descrizione del file}
* History :
*     Version: {Versione del file}
*     Update data: {Data ultima modifica}
*     Description: {descrizione della modifica}
*     Author: {Autore della modifica}
*/
```

Nomi

- Assegnare ad ogni elemento un nome univoco significativo
- I nomi di variabili, metodi e funzioni devono avere la prima lettera minuscola e le successive iniziali delle parole, che ne compongono il nome, in maiuscolo (la cosiddetta notazione a cammello)
- I nomi delle classi devono avere la prima lettera maiuscola

Commenti L'inserimento di commenti all'interno del codice ne agevola la comprensione ed è incoraggiato. É opportuno inserire i commenti per chiarire il comportamento di funzioni e metodi o per precisare tutto ciò che può essere ambiguo nel codice.

Stile di codifica Vanno osservate le regole stilistiche descritte nei riferimenti sopra indicati.

2.2.4 Strumenti usati

Tracciabilità La Tracciabilità dei requisiti e dei casi d'uso avviene tramite l'utilizzo di un database MySQL

Astah Astah è l'editor scelto dal gruppo per la modellazione nel linguaggio UML. *Astah*$_{|G|}$ è un editor gratuito, scaricabile online sia per Microsoft Windows, sia per *Linux*$_{|G|}$ che per Mac OS. É un editor abbastanza semplice e user-friendly. Infatti ad un primo utilizzo, è facile capire come muoversi nell'ambiente di sviluppo, come aggiungere classi, diagrammi e relazioni. Strumento molto utile che l'editor mette a disposizione è quello di poter esportare il diagramma realizzato in un file immagine. Questo software verrà utilizzato nelle attività di analisi e progettazione per la creazione di tutti i diagrammi *UML*$_{|G|}$ necessari.

2.2.5 Integrazione

In questo periodo si cerca di integrare le varie componenti del nostro sdk in modo che si interfaccino correttamente tra di loro. Per verificare ciò vengono utilizzati degli appositi test elencani nel documento "*Piano di qualifica v4.0.0*".

I Progettisti sono responsabili delle attività di integrazione e devono:

- avere piena conoscenza di quello che hanno sviluppato
- saper appostare modifiche del proprio codice o sviluppato da altri per migliorarne l'integrazione
- saper sviluppare le parti di software che permettono l'itegrazione con le varie parti del prodotto software
- saper individuare e correggere bug e requisiti non sodisfatti

L'obiettivo di questo perido è quello di itegrare tutte le parti software sviluppate dai vari progettisti ed correggere eventuali errori che potrebbero apparire durante l'integrazione.

3 Processi di supporto

3.1 Documentazione

3.1.1 Descrizione

Lo scopo del processo di documentazione è quello di illustrare le convenzioni che riguardano la stesura, la verifica e l'approvazione dei documenti. La classificazione di tali documenti è come segue:

- Interni: utilizzo interno al team
- Esterni: distribuzione esterna al gruppo, per il *committente*$_{|G|}$ e/o per il proponente

3.1.2 Strumenti

Come linguaggio di markup per la stesura della documentazione è stato scelto LaTeX. Questa scelta è stata presa per evitare possibili incompatibilità che si possono presentare utilizzando software differenti.

3.1.3 Ciclo di vita di un documento

I documenti si possono trovare in uno degli stati seguenti:

- Documento in lavorazione
- Documento da verificare
- Documenti approvati

I documenti in lavorazione sono i documenti che si trovano in stesura da parte del redattore. Una volta terminata la loro realizzazione, questi documenti vanno segnati come da verificare. Solo dopo la verifica da parte del relativo *Verificatore* i documenti passano nello stato "approvato". In questo stato i documenti sono stati consegnati al *Responsabile* del Progetto che ha il compito di approvarli in via definitiva.

3.1.4 Documenti finali

Studio di Fattibilità Gli *Analisti* analizzano tutti i capitolati valutandone i fattori positivi e negativi. É un'attività critica perché porterà alla scelta del progetto sul quale il gruppo andrà a lavorare. Questo documento è utilizzato internamente al $team_{|G|}$ e la lista di distribuzione comprende solo il committente.

Norme di Progetto In questo documento vengono riportati gli strumenti, le norme e le convenzioni che il team adotterà nello sviluppo del progetto. Questo documento è utilizzato internamente al team e la lista di distribuzione comprende solo il committente.

Piano di Progetto In questo documento vengono descritte le strategie che il team applicherà per la gestione delle risorse umane e temporali. La lista di distribuzione di questo documento comprende il committente ed il proponente.

Piano di Qualifica Lo scopo di questo documento è quello di illustrare un piano con il quale il team punta a soddisfare gli obiettivi di qualità. La lista di distribuzione di questo documento comprende il committente ed il proponente.

Analisi dei Requisiti In questo documento viene descritta la raccolta dei requisiti e dei casi d'uso nel modo più dettagliato possibile. Questo documento conterrà dunque tutti i casi d'uso ed i diagrammi di attività che rappresentano l'interazione tra utente e sistema. Questo documento è utilizzato sia esternamente che internamente.

Specifica Tecnica I *Progettisti* devono descrivere la progettazione ad alto livello del sistema e dei singoli componenti nella Specifica Tecnica. Devono essere definiti, inoltre, opportuni test di integrazione tra le varie componenti.

Definizione di Prodotto Lo scopo di questo documento è di descrivere i dettagli implementativi del prodotto. Queste informazioni saranno la base per il processo di codifica.

Glossario In questo documento verranno elencati tutti i termini che necessitano disambiguazione provenienti da tutta la documentazione. Ciascuno di questi termini sarà accompagnato dalla spiegazione del significato. Questo documento è utilizzato anche esternamente e la lista di distribuzione comprende committenti e proponenti.

Manuale Utente Con questo documento verrà fornito all'utente una guida dettagliata coprendo tutte le funzionalità che il prodotto offre. Su richiesta del *proponente*$_{|G|}$ verrà stilato in lingua inglese.

Verbali In questo documento vengono descritte formalmente tutte le discussioni e le decisioni prese durante le riunioni. I verbali possono essere interni oppure esterni. Ogni verbale dovrà essere denominato nel seguente modo:

Verbale_Numero del verbale_Tipo di verbale_Data del verbale

dove:

- Numero del verbale: numero identificativo univoco del verbale
- Tipo di verbale: identifica se si tratta di un Verbale Interno oppure Verbale Esterno
- Data del verbale: identifica la data nella quale si è svolta la riunione corrispondente al verbale. Il formato è YYYY-MM-DD

Nella parte introduttiva del verbale dovranno essere specificate le seguenti informazioni:

- Data incontro: data in cui si è svolta la riunione
- Ora inizio incontro: ora di inizio della riunione
- Ora termine incontro: ora di terminazione della riunione
- Luogo incontro: luogo in cui si è svolta la riunione
- Durata: durata della riunione
- Oggetto: argomento della riunione
- Segretario: cognome e nome del membro incaricato a redigere il verbale
- Partecipanti: cognome e nome di tutti i membri partecipanti alla riunione

3.1.5 Struttura del documento

Prima pagina Tutti i documenti dovranno contenere nella prima pagina le seguenti informazioni:

- Nome del gruppo
- Logo del progetto
- Nome del progetto
- Nome del documento
- Versione del documento
- Data di creazione del documento
- Stato del documento
- Nome e cognome del redattore del documento
- Nome e cognome del *Verificatore* del documento
- Nome e cognome del *responsabile* approvatore del documento
- Lista di distribuzione
- Uso del documento
- Email di riferimento del gruppo
- Un sommario contenente una breve descrizione del documento

Diario delle modifiche Il diario delle modifiche sarà presente nella seconda pagina dove verranno inserite tutte le modifiche del documento. Tutte queste informazioni verranno riportate in una tabella che in ogni riga conterrà le seguenti informazioni:

- Versione: versione del documento dopo la modifica
- Descrizione: descrizione della modifica
- Autore e Ruolo: autore della modifica e il suo ruolo
- Data: data della modifica

Indice In ogni documento, dopo il diario delle modifiche, deve essere presente un indice di tutte le sezioni. In presenza di tabelle e/o immagini queste devono essere indicate con i relativi indici.

Formattazione generale delle pagine La formattazione della pagina, oltre al contenuto, prevede un'intestazione e un piè di pagina. L'intestazione della pagina contiene:

- Nome e logo del gruppo
- Nome della sezione corrente

Il piè di pagina contiene:

- Il numero di pagina e il numero totale di pagina

3.1.6 Norme tipografiche

Le seguenti norme tipografiche indicano i criteri riguardanti l'ortografia e la tipografia di tutti i documenti.

Stili di testo

- Corsivo: il testo corsivo verrà utilizzato nelle seguenti situazioni:
  - Ruoli: ogni riferimento a ruoli di progetto va scritto in corsivo
  - Documenti: ogni riferimento ad un documento
  - Glossario: ogni parola presente nel glossario deve essere scritta in corsivo. Inoltre presentano una |G| a pedice
- Font codice: il font per il codice ("font macchina da scrivere") deve essere utilizzato per indicare qualsiasi parte riguardi posizione del codice.

Punteggiatura

- Punteggiatura: ogni simbolo di punteggiatura non può seguire un carattere di spazio
- Lettere maiuscole: le lettere maiuscole vanno utilizzate dopo il punto, il punto interrogativo, il punto esclamativo.

Formati

- Date: le date presenti nei documenti devono seguire lo standard **ISO 8601:2004: YYYY-MM-DD**
  dove:
  - YYYY: rappresenta l'anno
  - MM: rappresenta il mese
  - DD: rappresenta il giorno
- Ore: le ore presenti nei documenti devono seguire lo standard **ISO 8601:2004** con il sistema a 24 ore:
  dove:
  - hh: rappresentano le ore
  - mm: rappresentano i minuti

3.1.7 Composizione email

In questo paragrafo verranno descritte le norme da applicare nella composizione delle email.

Destinatario

- Interno: l'indirizzo da utilizzare è obelixswe@gmail.com
- Esterno: l'indirizzo del destinatario varia a seconda si tratti del Prof. Tullio Vardanega Prof. Riccardo Cardin o i proponenti del progetto

Mittente

- Interno: l'indirizzo è di colui che scrive e spedisce la email
- Esterno: l'indirizzo da utilizzare è obelixswe@gmail.com ed è utilizzabile unicamente dal *Responsabile* di Progetto

Oggetto L'oggetto della mail deve essere chiaro, preciso e conciso in modo da rendere semplice il riconoscimento di una mail tra le altre.

3.1.8 Componenti grafiche

Tabelle Tutte le tabelle in tutti i documenti devono avere una didascalia ed un indice identificativo univoco per il loro tracciamento nel documento stesso.

Immagini Le immagini inserite nel documento sono nel formato PNG. Le immagini hanno una didascalia e compaiono nell'indice dedicato.

3.1.9 Versionamento

Ogni documento prodotto deve essere identificato dal nome e dal numero di versione nel seguente modo:

$$\_vX.Y.Z$$

dove:

- X: indica il numero di uscite formali del documento e viene incrementato in seguito all'approvazione finale da parte del *Responsabile* di Progetto. L'incremento dell'indice X comporta l'azzeramento degli indici Y e Z
- Y: indica il numero crescente delle verifiche. L'incremento viene eseguito dal *Verificatore* e comporta l'azzeramento dell'indice Z
- Z: indica il numero di modifiche minori apportate al documento prima della sua verifica. Viene aumentato in modo incrementale

3.2 Gestione della configurazione

La gestione della configurazione deve comprendere tutte le attività necessarie a rendere affidabile l'evoluzione del progetto, tenendo traccia delle sue versioni.

3.2.1 Controllo di versione

Per il versionamento ed il salvataggio dei file prodotti durante l'attività di progetto è stato deciso di affidarsi ad alcuni *repository*$_{|G|}$ su GitHub. L'*Amministratore* di Progetto si occupa della creazione dei repository e, successivamente, gli altri membri del gruppo vi accederanno tramite il loro account personale.

3.2.2 Repository

I file, che saranno prodotti dal team, saranno suddivisi in due repository distinti: uno contenente i file relativi ai documenti e un'altro contenente i file necessari alla creazione del prodotto software.

Struttura del repository dei documenti I file all'interno del repository dei documenti verranno organizzati secondo questa struttura:

- RR
  - Interni
  - Esterni
- RP
  - Interni
  - Esterni
- RQ
  - Interni
  - Esterni
- RA
  - Interni
  - Esterni
- tools
- file_comuni

Per la visualizzazione dei documenti nel formato pdf sarà necessario scaricare e compilare i file.tex relativi.

Struttura del repository del codice sorgente I file all'interno del repository del codice sorgente verranno organizzati secondo questa struttura:

- SDK
- Bolle predefinite
- Demo

Nomi dei file I nomi dei file interni al repository devono sottostare alle seguenti norme:

- devono avere lunghezza minima di tre caratteri
- devono identificare in modo non ambiguo i file
- devono riportare le informazioni dal generale al particolare
- devono, nel caso contengano date, rispettare il formato YYYY-MM-DD

Norme sui commit Ogni volta che vengono effettuate delle modifiche ai file del repository bisogna specificarne le motivazioni. Questo avviene utilizzando il comando `"git commit"` accompagnato da un messaggio riassuntivo e una descrizione in cui va specificato:

- lista dei file coinvolti
- liste delle modifiche effettuate, ordinate per ogni singolo file

Prima di eseguire tale procedura, va aggiornato il diario delle modifiche, secondo le regole viste in §3.1.5.

3.3 Processo di garanzia di qualità del prodotto

Per garantire la qualità del prodotto vengono adottato varie metodologie tra le quali ritroviamo i processi di verifica e di validazione successivamente illustrati. Nel caso in cui i task assegnati al team non siano soddisfatti per eventuali problematiche, questi devono essere riportati nella lista di problemi menzionata nel *processo di soluzione di problemi*.
Un ulteriore garanzia di qualità è data dall'assicurarsi che i requisiti principali del progetto siano soddisfatti alla fine dello sviluppo del software. Ci si può assicurare di questo attraverso una documentazione valida ed un incontro con i committenti ed i proponenti che dovranno accertarsi della validità del prodotto finale.
Infine per stabilire in maniera quantitativa la qualità ottenuta abbiamo individuato delle metriche che andremo ora a esplicitare, indicando i range accettabili e quelli ottimali. Il loro valore effettivo è invece indicato nel *Piano di qualifica v4.0.0*. Le metriche individuate sono state suddivise in

- **Qualità funzionale**
- **Qualità di affidabilità**
- **Qualità di usabilità**
- **Qualità di efficienza**
- **Qualità di manutenibilità**

3.3.1 Qualità funzionale

Completezza dell'implementazione funzionale Indica la percentuale di requisiti funzionali coperti dall'implementazione.

- **Misurazione**:
$$C = \left(1 - \frac{N_{FM}}{N_{FI}}\right) \cdot 100$$
dove N_{FM} è il numero di funzionalità mancanti nell'implementazione e N_{FI} è il numero di funzionalità individuate nell'attività di analisi.

Accuratezza rispetto alle attese Indica la percentuale di risultati concordi alle attese.

- **Misurazione**:
$$A = \left(1 - \frac{N_{RD}}{N_{TE}}\right) \cdot 100$$
dove N_{RD} è il numero di test che producono risultati discordanti rispetto alle attese e N_{TE} è il numero di test-case eseguiti.

3.3.2 Qualità di affidabilità

Densità di *failure* Indica la percentuale di operazioni di testing che si sono concluse in fallimenti.

- **Misurazione**:
$$F = \frac{N_{FR}}{N_{TE}} \cdot 100$$
dove N_{FR} è il numero di fallimenti rilevati durante l'attività di testing e N_{TE} è il numero di test-case eseguiti.

Blocco di operazioni non corrette Indica la percentuale di funzionalità in grado di gestire correttamente i *fault* che potrebbero verificarsi.

- **Misurazione**:
$$B = \frac{N_{FE}}{N_{ON}} \cdot 100$$
dove N_{FE} è il numero di *failure* evitati durante i test effettuati e N_{ON} è il numero di test-case eseguiti che prevedono l'esecuzione di operazioni non corrette, causa di possibili *failure*.

3.3.3 Qualità di efficienza

Tempo di risposta Indica il periodo temporale medio che intercorre fra la richiesta al software di una determinata funzionalità e la restituzione del risultato all'utente.

- **Misurazione**:
$$T_{RISP} = \frac{\sum_{i=1}^{n} T_i}{n}$$
con T_{RISP} misurato in secondi, e dove T_i è il tempo intercorso fra la richiesta i di una funzionalità ed il completamento delle operazioni necessarie a restituire un risultato a tale richiesta.

3.3.4 Qualità di manutenibilità

Capacità di analisi di *failure* Indica la percentuale di *failure* registrate delle quali sono state individuate le cause.

- **Misurazione**:
$$I = \frac{N_{FI}}{N_{FR}} \cdot 100$$
dove N_{FI} è il numero di *failure* delle quali sono state individuate le cause e N_{FR} è il numero di *failure* rilevate.

Impatto delle modifiche Indica la percentuale di modifiche effettuate in risposta a *failure* che hanno portato all'introduzione di nuove *failure* in altre componenti del sistema.

- **Misurazione**:
$$I = \frac{N_{FRF}}{N_{FR}} \cdot 100$$
dove N_{FRF} è il numero di *failure* risolte con l'introduzione di nuove *failure* e N_{FR} è il numero di *failure* risolte.

3.4 Verifica

3.4.1 Descrizione

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, in modo da uniformare le attività di verifica svolte ed ottenere il miglior risultato possibile.

Metriche per la pianificazione Qualsiasi eventuale valore negativo ottenuto dalle metriche: Schedule Variance o/e Budget Variance dovrà essere compensato entro la fine dell'attività di progetto modificando e revisionando in modo coerente il Piano di Progetto, in quanto non è possibile eccedere le ore di lavoro finali e il budget finale indicato nella pianificazione.

Schedule Variance Indica se si è in linea, in anticipo o in ritardo rispetto la pianificazione temporale prevista dal *Piano di progetto v4.0.0*. Per calcolare questa metrica utilizziamo la seguente formuala: SV = BCWP −BCWS, dove BCWP sono le attività completate ad un certo momento e BCWS le attività che, secondo la pianificazione, dovrebbero essere state completate a quel momento. Il range di accettazione per questa metrica è un valore superiore o uguale a zero in questo modo ci si assicura di non avere accumulato eventuali ritardi per la consegna.

Budget Variance Indica se alla data corrente si è speso di più o di meno rispetto a quanto pianificato. Per calcolare questa metrica utilizziamo la seguente forumal: BV = BCWS − ACWP , dove BCWS è il costo pianificato per realizzare le attività di progetto alla data corrente e ACWP è il costo effettivamente sostenuto alla data corrente. Il range di accettazione per questa metrica è un valore superiore o ugale a zero, segnale che i costi sono stati preventivati in maniera corretta e non si esce dal budget prefeissato all'inizio

Ottimalità delle misurazioni Questa metrica misura quante metriche misurate sono ottimali e assicura che almeno una buona quantita lo siano. **Misurazione:**
$$\frac{\text{Misurazioni Ottimali}}{\text{Misurazioni Ottimali} + \text{Misurazioni Accettabili}}$$
Il valore di accettazione è pari a 0.3 mentre il valore ottimale è 0.4 questo per garantire l'ottimo range per almeno un numero cospicuo di metriche così da aumentare la qualità del software.

Requisiti obbligatori soddisfatti Indica il numero di requisiti obbligatori ottenuti dall'attività di analisi.

- **Misurazione**:

$$C = \left(1 - \frac{N_{FM}}{N_{FI}}\right) \cdot 100$$

dove N_{FM} è il numero di funzionalità mancanti nell'implementazione e N_{FI} è il numero di funzionalità individuate nell'attività di analisi.

Metriche per i documenti

indice di Gulpease L'indice Gulpease rispetto ad altri indici di leggibilità possiede il vantaggio di utilizzare la lunghezza delle parole in lettere anziché in sillabe, semplificando il calcolo dell'indice stesso. Esso considera due variabili linguistiche:

- La lunghezza della parola.
- La lunghezza della frase rispetto al numero delle lettere.

I valori dell'indice sono compresi tra 0 e 100, dove il valore 100 indica la leggibilità più alta e 0 la leggibilità più bassa.

Metriche per la progettazione dell'architettura di sistema

Livello di instabilità Per comprendere questa metrica è necessario dare una semplice spiegazione di accoppiamento afferente e di accoppiamento efferente.

- **Accoppiamento afferente**: indica il numero di classi esterne ad un package che dipendono da classi interne ad esso. Un alto valore implica un alto grado di dipendenza del resto del software dal package. Un valore eccessivamente basso, invece, potrebbe evidenziare che un package fornisce poche funzionalità.
- **Accoppiamento efferente**: indica il numero di classi interne al package che dipendono da classi esterne ad esso. Mantenendo il valore di tale indice basso è possibile garantire funzionalità di base indipendentemente dal resto del sistema.

La stabilità di un package indica la possibilità di effettuare modifiche a tale package senza influenzarne altri all'interno dell'applicazione. Tale indice è strettamente legato all'accoppiamento efferente ed afferente e viene calcolato dalla seguente formula:

$$\frac{\text{Accoppiamento Efferente}}{\text{Accoppiamento Afferente} + \text{Accoppiamento Efferente}}$$

Astrattezza Questa metrica misura l'astrattezza di ogni package tramite il seguente rapporto

$$\frac{\text{Numero classi astratte e interfacce}}{\text{Numero totale classi}}$$

In questa metrica non esiste un valore accettabile ma solo un valore che si vorrà ottenere. esso sarà indicato nel *Piano di qualifica v4.0.0.*

Distanza dalla sequenza principale Questa metrica misura il bilanciamento tra l'astrattezza e la stabilità del package da noi sviluppato. I package idealmente perfetti sono infatti quelli o compleatamente concreti e instabili(I=1, A=0) o quelli astratti e stabili (I=0, A=1). Per misurare questa metrica si utilizza quindi la seguente formula matematica:

$$|\text{Astratezza} + \text{Livello di stabilità} - 1|$$

La sequenza principale menzionata nel titolo è infatti la linea retta A+I=1 e la formula appena descritta misura la distanza del nostro package da esso.

Metriche per la progettazione dell'architettura in dettaglio

Numero di metodi per classe Rappresenta il numero di metodi per ogni classe. Un numero molto elevato potrebbe evidenziare la necessità di spezzettare le varie componenti della gerarchia in più classi distinte. Per contro un se ogni classe ha un indice abbastanza basso, questa metrica potrebbe indicare un analogo errore di progettazione, in quanto abbiamo classi che potrebbero essere raggruppate in classi più complesse e funzionali.

Numero di attributi per classe Questa metrica prevede di valutare la qualità del software in base al numero di attributi presenti in una classe. Un numero elevato di attributi potrebbe evidenziare un possibile errore di progettazione con conseguente necessità di suddividere la classe in più classi in relazione tra loro seguendo il principio dell'incapsulamento.

Numero di parametri per metodo Rappresenta il numero di parametri da passare per la chiamata di un metodo. Un numero elevato per un dato metodo potrebbe evidenziare la necessità di ridurne le funzionalità associate e/o suddividerle in altri metodi ausiliari. Un alto valore di questo indice potrebbe evidenziare pertanto un possibile errore di progettazione.

Metriche per la codifica del Software

Complessità ciclomatica Questo indice viene utilizzato per misurare la complessità di un programma. Esso misura direttamente il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso. I nodi del grafo corrispondono a gruppi indivisibili di istruzioni, mentre gli archi connettono due nodi se il secondo gruppo di istruzioni può essere eseguito immediatamente dopo il primo gruppo. Alti valori di complessità ciclomatica implicano una ridotta manutenibilità del codice. Valori bassi potrebbero però determinare una scarsa

efficienza dei metodi. McCabe, ideatore di questa metrica, raccomandava che i programmatori contassero la complessità dei moduli in sviluppo, e li dividessero in moduli più piccoli, qualora tale complessità superi 10 ma osservando che in certe circostanze può essere appropriato rilassare tale restrizione e permettere moduli con una complessità anche di 15.

Linee di commento per linee di codice Questo indice indica il rapporto tra linee di commento e linee di codice ed è utile per stimare la manutenibilità e la comprensibilità del codice.

Numero di livelli di annidamento per metodo Rappresenta il numero di strutture di controllo, annidate tra loro internamente ad un metodo. Un valore elevato per questa metrica potrebbe essere indice di una complessità troppo elevata del metodo stesso, e di un basso livello di astrazione del codice.

Metriche per la validazione

Test di Unità eseguiti Indica la percentuale di test di unità eseguiti.

- **Misurazione**: $UE = \frac{N_{TUE}}{N_{TUP}} \cdot 100$

dove N_{TUE} è il numero di test di unità eseguiti e N_{TUP} è il numero di test di unità pianificati. Il valore di accettazione richiesto è un intrevallo tra 90-100, il valore ottimale è 100 per garantire che tutti o la maggior parte dei test di Unità siano eseguiti effettivamente.

Test di Integrazione eseguiti Indica la percentuale di test di integrazione eseguiti.

- **Misurazione**: $IE = \frac{N_{TIE}}{N_{TIP}} \cdot 100$

dove N_{TIE} è il numero di test di integrazione eseguiti e N_{TIP} è il numero di test di Integrazione pianificati. Il valore di accettazione richiesto è 60-100 mentre quello ottimale è un range di 70-100 per garantire che tutti o la maggior parte dei test di Integrazione siano eseguiti effettivamente.

Test di Validazione eseguiti Indica la percentuale di test di validazione eseguiti manualmente.

- **Misurazione**: $VE = \frac{N_{TVE}}{N_{TVP}} \cdot 100$

dove N_{TVE} è il numero di test di validazione eseguiti e N_{TVP} è il numero di test di Validazione pianificati. Il Valore di accettazione richiesto è 100, come il valore ottimale . questo per garantire che tutti o la maggior parte dei test di Validazione siano eseguiti effettivamente.

Test di Sistema eseguiti Indica la percentuale di test di sistema eseguiti in modo automatico.

- **Misurazione**: $SE = \frac{N_{TSE}}{N_{TSP}} \cdot 100$

dove N_{TSE} è il numero di test di sistema eseguiti e N_{TSP} è il numero di test di sistema pianificati. Il valore di accettazion è un range tra 70-100 mentre il valore ottimale è 80-100 per garantire che tutti o la maggior parte dei test di Sistema siano eseguiti effettivamente.

Test superati Indica la percentuale di test superati.

- **Misurazione**: $S = \frac{N_{TS}}{N_{TE}} \cdot 100$

dove N_{TS} è il numero di test superati e N_{TE} è il numero di test eseguiti. Il valore di accettazione è un range tra 90-100, il valore ottimale è 100. La motivazione è garantire che il numero di test totali eseguiti sia uguale o quasi a quelli preventivati per garantire tutte le funzionalità previste.

Copertura dei test Questo indice indica la percentuale di istruzioni del prodotto che vengono eseguite durante i test. Un valore percentuale alto indica una maggiore copertura dei test e quindi una maggiore probabilità che le componenti abbiano una ridotta quantità di errori. Tale indice può però essere abbassato da metodi molto semplici che non richiedono testing come ad esempio metodi `getter` e/o `setter`.

Branch coverage Il *Branch converage* indica, in percentuale, la quantità di decisioni coperte durante l'esecuzione dei test. Una decisione avviene ogni qualvolta vi è una possibilità di variazione di esecuzione, ad esempio una condizione `if-then-else` o qualsiasi cosa che potrebbe dare TRUE o FALSE a seconda dei dati inseriti.

- **Misurazione**:

$$\frac{\text{Condizioni eseguite}}{\text{Condizioni totali}}$$

Statement coverage Lo *Statement coverage* indica, in percentuale, la quantità di codice che viene eseguita quando vengono eseguiti i test. Un programma con un alto livello di copertura ha minori possibilità di contenere malfunzionamenti o comportamenti inaspettati.

- **Misurazione**:

$$\frac{\text{Codice Testato}}{\text{Codice totale}}$$

Metriche per l'integrazione

Componenti integrate **Misurazione**: $I = \frac{N_{CI}}{N_{CP}} \cdot 100$, dove N_{CI} è il numero di componenti attualmente integrate nel sistema e N_{CP} è il numero di componenti delineate nell'attività di progettazione. Per avere un buon soddisfacimento dei requisiti abbiamo deciso di avere un valore di accettazione pari a quello ottimale cioè di 100.

Metriche per la gestione dei rischi

Rischi non preventivati Indicatore che evidenzia i rischi non preventivati. Per questa metrica l'indice numerico viene incrementato nel momento in cui si manifesta un rischio non individuato nell'attività di analisi dei rischi; il valore di accettazione è tra 0 e 5.

3.4.2 Analisi

Analisi statica L'analisi statica è il processo di valutazione di un sistema senza che esso venga eseguito. Ci sono diverse tecniche di analisi statica. Le due più applicate sono Walkthrough e Inspection.

- Walkthrough: riguarda un analisi informale del codice svolta da vari partecipanti umani che 'operano come il computer': in pratica, si scelgono alcuni casi di test e si simula l'esecuzione del codice a mano (si attraversa (walkthrough) il codice). Si presta attenzione sulla ricerca dei difetti, piuttosto che sulla correzione.
- Inspection: questa tecnica di ispezione di codice può rilevare ed eliminare anomalie fastidiose e rendere più precisi i risultati. Inspection è una tecnica completamente manuale per trovare e correggere errori. Spesso l'analisi dei difetti effettuata viene discussa e vengono decise le eventuali azioni da intraprendere. Il codice è analizzato usando checklist dei tipici errori di programmazione.

Analisi dinamica L'analisi dinamica consiste nell'esaminare il software quando è in esecuzione ed è tipicamente effettuata dopo aver compiuto l'analisi statica di base.

3.4.3 Test

Di seguito verranno elencate tutte le tipologie di Test che il gruppo ha deciso di implementare

Test di unità Lo scopo del test di unità è quello di verificare che tutte le componenti del software funzionino correttamente. Questo test aiuta a ridurre il più possibile la presenza di errori nelle componenti. I test di unità sono identificati dalla seguente sintassi:

$$TU[\text{Codice Test}]$$

Test di Integrazione Questo test permette di verificare che ciascuna componente che è stata validata singolarmente, funzioni correttamente anche dopo averla integrata con l'intero sistema. I test di integrazione sono identificati dalla seguente sintassi:

$$TI[\text{Codice Test}]$$

Test di sistema I test di sistema hanno lo scopo di verificare che tutti i requisiti siano soddisfatti. Questo test viene effettuato quando il prodotto è giunto alla versione definitiva. I test di sistema sono identificati dalla seguente sintassi:

$$TS[\text{Codice Requisito}]$$

Test di regressione Il test di regressione consiste nell'effettuare i test per tutte le componenti che sono state modificate. I test di regressione sono identificati dalla seguente sintassi:

$$TR[\text{Codice Test}]$$

3.4.4 Verifica dei documenti

É compito del Responsabile di Progetto assegnare ai Verificatori i ticket per l'attività di verifica. Attraverso il diario delle modifiche è possibile focalizzare l'attenzione maggiormente sulle sezioni che hanno subito dei cambiamenti dall'ultima verifica, riducendo il tempo necessario al controllo. Per eseguire un'accurata verifica dei documenti redatti è necessario seguire i seguenti punti:

1. **Controllo sintattico del periodo**: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**: in questo documento sono state definite norme tipografiche di carattere generale. Queste regole impongono una struttura dei documenti che non può essere verificata in maniera automatica, di conseguenza è necessario che i Verificatori eseguano inspection sul rispetto di tali norme

3. **Verifica delle proprietà di glossario**: il Verificatore dovrà controllare che i termini che hanno bisogno di disambiguazione siano presenti nel Glossario con la relativa definizione

4. **Miglioramento del processo di verifica**: per avere un miglioramento del processo di verifica, quando il Verificatore utilizza la tecnica walkthrough su un documento, dovrà riportare gli errori più frequenti, per consentire l'utilizzo di inspection su tali errori nelle verifiche future

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 walkthrough del documento alla ricerca delle frasi troppo lunghe o complesse

3.4.5 Verifica del codice

Per la verifica del codice al Verificatore è richiesto l'avvio dei test statici e dinamici e l'analisi dei risultati ottenuti.

3.5 Validazione

3.5.1 Test di validazione

Il test di validazione coincide con il collaudo del software in presenza del proponente. Ha l'obbiettivo di verificare che il prodotto finale sia conforme a quanto è stato pianificato. In caso di esito positivo si procede con il rilascio del software. I test di validazione sono identificati dalla seguente sintassi:

$$TV[\text{Codice requisito}]$$

3.6 Strumenti per l'analisi dinamica

- **Mocha**: framework che permette di eseguire test di unità e di integrazione andando a creare dei mock che simulino il comportamento delle varie parti, con lo scopo di testare ogni dettaglio in modo indipendente dal resto
- **Chai**: libreria per formulare asserzioni sull'output atteso
- **Jest**: framework sviluppato ed utilizzato da facebook per il testing di codice JavaScript che non ha bisogno di configurazione
- **Enzyme**: utility per React per il testing di codice JavaScript

3.7 Processi per la risoluzione dei problemi

Per ogni problema trovato all'interno del progetto (sia esso di qualunque natura e importanza), esso viene inserito all'interno di un documento non ufficiale presente nella repository di documenti.
All'interno di esso sarà presente uno schema concettuale per dividere i problemi a seconda della loro priorità e natura. Il problema dovrà essere descritto in maniera precisa e chiara, inoltre sarà presente un campo dove verrà inserito se il problema in questione è in uno dei seguenti tre stati

- *Identificato*: il problema è stato trovato ma non è ancora stato preso in carica da nessun componente del team.
- *In lavorazione*: il problema è stato già analizzato ed è stato preso in carica da uno più componenti del team per la sua risoluzione. una volta che entra in questo stato devono essere indicati anche i componenti del team che stanno lavorando su di esso.
- *Risolto*: il problema è già stato risolto.

4 Processi Organizzativi

4.1 Scopo

Lo scopo di questi processi è quello di migliorare e facilitare l'organizzazione interna al gruppo stabilendo le norme per la pianificazione e gestione dei ruoli e per le comunicazioni interne ed esterne.

4.2 Aspettative del Processo

Questo processo produce:

- Definizione dei ruoli del team
- Identificazione delle modalità di comunicazione interne al gruppo
- Documentazione sulle modalità di ticketing

4.3 Ruoli di Progetto

Durante l'intero sviluppo del progetto ogni membro del team dovrà ricoprire, a turno, ognuno dei ruoli elencati di seguito. Inoltre non potrà mai accadere che un membro del gruppo risulti redattore e verificatore di un medesimo documento. In questo modo si tende ad evitare il conflitto di interessi che potrebbe sorgere se la responsabilità della stesura e della verifica di un documento fosse affidata ad un'unica persona. Un membro può inoltre ricoprire più ruoli contemporaneamente.

4.3.1 Amministratore di Progetto

L' *Amministratore* di Progetto ha il compito di controllare ed amministrare l'ambiente di lavoro assumendosi le seguenti responsabilità:

- studio e ricerca di strumenti per migliorare l'ambiente lavorativo
- minimizzare il carico di lavoro umano automatizzando dove possibile
- garantire un controllo della qualità del prodotto fornendo procedure e strumenti di segnalazione e monitoraggio
- risolvere i problemi legati alla gestione dei processi e alle risorse disponibili
- gestire il versionamento e l'archiviazione della documentazione di progetto

4.3.2 Responsabile di Progetto

Il *Responsabile* di Progetto funge da punto di riferimento per il gruppo, il committente ed il proponente.
Approva le scelte prese dal gruppo e si assume le seguenti responsabilità:

- approvazione della documentazione
- approvazione dell'offerta economica
- gestione delle risorse umane
- coordinamento e pianificazione delle attività di progetto
- studio e gestione dei rischi

4.3.3 Analista

L' *Analista* deve effettuare ricerche e studi approfonditi sul dominio del problema. Non è richiesta la sua presenza per l'intera durata del progetto e si assume le seguenti responsabilità:

- comprensione della natura e della complessità del problema
- produzione dello Studio di Fattibilità e dell'Analisi dei Requisiti delineando specifiche comprensibili ad ogni figura coinvolta (proponente e committente)

4.3.4 Progettista

Il *Progettista* deve avere profonde conoscenze delle tecnologie utilizzate e competenze tecniche aggiornate, in modo tale da poter interagire con il committente per arrivare ad un chiarimento progressivo delle varie caratteristiche del sistema finale.
Il *Progettista* si assume le seguenti responsabilità:

- effettuare scelte efficienti ed ottimizzate su aspetti tecnici del progetto
- effettuare scelte che rendano il prodotto più facilmente mantenibile

4.3.5 Verificatore

Il *Verificatore* deve avere ampia conoscenza delle normative di progetto.
Il suo compito è controllare che le attività di progetto siano svolte secondo le norme stabilite.

4.3.6 Programmatore

Il *Programmatore* è responsabile delle attività di codifica e di creazione delle componenti di supporto, utili a effettuare le prove di verifica e validazione sul prodotto software.
Si assume le seguenti responsabilità:

- implementare le soluzioni previste dal *Progettista*
- scrivere codice pulito, mantenibile e conforme alle norme di progetto
- versionare il codice prodotto
- realizzare gli strumenti utili per poter compiere le prove di verifica e validazione

4.4 Comunicazioni

4.4.1 Comunicazioni interne

Per le comunicazioni interne è stato deciso di utilizzare l'applicazione *Slack*$_{|G|}$ dove è stato creato un gruppo di lavoro "Obelix" all'interno del quale i membri del gruppo possono comunicare attraverso la chat principale.

Per quanto riguarda la comunicazione su argomenti specifici è possibile la creazione di alcune chat "interne" per mandare messaggi solo ai membri interessati. Mel caso fosse necessaria una video-chiamata verrà utilizzata l'applicazione *Skype*.

4.4.2 Comunicazioni esterne

Per le comunicazioni esterne è stato creato un indirizzo email apposito: obelix-swe@gmail.com.
Il Responsabile *del Progetto* avrà l'onere di gestire le comunicazioni esterne riservandosi di condividere le mail che lo richiedono con il resto del team attraverso Slack o una mailing list.

4.5 Riunioni

Il Responsabile *di progetto* ha il compito di indire le riunioni sia interne che esterne.

4.5.1 Riunioni interne

Per ogni incontro interno il Responsabile *di progetto* dovrà accordarsi con il team inviando un messaggio sull'adeguata chat di Slack almeno 2 giorni prima della data scelta.
Una volta che si sarà accordato coi membri del gruppo potrà creare l'evento dell'incontro sul calendario di *Slack* indicando:

- data
- ora
- luogo
- argomento di discussione

Vi possono essere casi eccezionali, come una milestone, in cui il vincolo sull'avviso (2 giorni prima) può essere ignorato.
Gli altri componenti del gruppo possono richiedere una riunione interna straordinaria, presentando al Responsabile *di progetto* le motivazioni per le quali si ritiene necessaria una riunione. In questi casi il Responsabile *di progetto* può:

- autorizzare lo svolgimento della riunione
- negare lo svolgimento della riunione, nel caso in cui non ritenga le motivazioni presentate valide abbastanza da richiedere una riunione
- suggerire mezzi di comunicazione differenti.

4.5.2 Riunioni esterne

Il Responsabile di progetto ha il compito di fissare le riunioni esterne con i proponenti o con i committenti, contattandoli tramite la casella di posta elettronica del gruppo. Il Responsabile di progetto ha, inoltre, il compito di accordarsi con i proponenti o committenti riguardo data, orario e luogo dell'incontro, tenendo conto anche della disponibilità degli elementi del gruppo e cercando, per quanto

possibile, di far partecipare tutti alla riunione.
Per le comunicazioni con il proponente è possibile utilizzare anche *Slack* e *Skype*.

4.6 Ticketing

Per suddividere i task all'interno del gruppo verrà utilizzato Asana, un sistema che permette di ottimizzare e facilitare il lavoro di squadra. Con Asana è possibile:

- Creare task
- Assegnare task
- Seguire task
- Commentare task

Essendo un applicativo web sarà possibile visionare in ogni istante i compiti completati, in corso e assegnati a ciascun membro del gruppo.

4.6.1 Struttura del ticket

Ogni ticket creato dovrà avere le seguenti caratteristiche:

- Titolo: deve riassumere in modo conciso il task svolto o da svolgere. Se si tratta di un documento bisogna specificarne il nome
- Commento: breve commento per chiarire, se necessario, il task qualora non fosse abbastanza chiaro
- Lista dei tags: dovranno essere inserite le giuste etichette per indicare lo stato e l'ambito del task
- Subtask: è possibile inserire dei compiti secondari se necessario

Il nome dell'assegnatario sarà impostato del Responsabile di Progetto, le date della creazione e terminazione del ticket vengono inserite e memorizzate dall'applicativo Asana.

4.6.2 Etichette

Ogni ticket avrà una o più etichette, ciò aiuterà a supervisionare le attività e il loro stato. I tags che devono essere utilizzati sono:

- I ticket non assegnati saranno identificati dal tag di stato Aperto
- I ticket di cui si avrà presa visione saranno identificati dal tag di stato Accettato
- I ticket su cui si sta lavorando saranno identificati dal tag di stato In corso
- I ticket interrotti per mancanza di risorse, input o problematiche, saranno identificati dal tag di stato In attesa
- I ticket completati e che sono in attesa di verifica saranno identificati dal tag di stato Completato
- I ticket verificati, quindi conclusi, saranno identificati dal tag di stato Verificato

4.6.3 Ciclo di vita del ticket

Ogni ticket rappresenterà un compito da effettuare, e dovrà rispettare i seguenti passi:

1. Creazione di un nuovo task
2. Il Responsabile di Progetto assegna il task ad un membro del team (se non è ancora assegnato verrà inserita nella lista dei tags l'etichetta Aperto)
3. L'assegnatario può accettare oppure no:
   - Se non accetta, l'assegnatario deve inserire un commento con le giuste motivazioni, e verrà rieseguito il punto 2
   - Se accetta, l'assegnatario inserisce nella lista dei tags l'etichetta Accettato
4. Se il task richiede una dipendenza, andrà inserita l'etichetta In attesa
5. Esecuzione del task
6. Se il task è completo, l'assegnatario inserisce il tag Completato

4.6.4 Verifica Ticket

Ogni ticket con etichetta di stato Completato dovrà essere verificato. Il procedimento per fare ciò è il seguente:

1. Al completamento di ogni task viene automaticamente creato un nuovo task di verifica associato
2. Il Responsabile di Progetto assegna il task ad un Verificatore
3. Il Verificatore individua eventuali anomalie e crea i task necessari per la correzione. In mancanza di errori chiude il task inserendo l'etichetta di stato Verificato
4. Il Responsabile di Progetto assegna gli eventuali task di correzione prestando attenzione ad evitare conflitto di interessi

4.7 Procedure di supporto

Di seguito vengono descritte le procedure di supporto che i componenti del gruppo sono tenuti a seguire per convergere agli obiettivi definiti nel *Piano di qualifica v4.0.0*. Il Responsabile di Progetto ha la responsabilità di gestione dell'intero progetto. Dovrà garantire un corretto sviluppo delle attività utilizzando gli strumenti che gli permetteranno di:

- Coordinare e pianificare le attività
- Analizzare i rischi
- Elaborare i dati di avanzamento

4.7.1 Pianificazione delle attività

Per ogni periodo individuato nel documento Piano di Progetto, il Responsabile di Progetto dovrà realizzare un diagramma di Gantt. Esso dovrà anche eseguire le seguenti attività:

- Creare un calendario lavorativo per il progetto
- Inserire le attività da svolgere
- Inserire eventuali dipendenze tra le attività
- Inserire le milestone indicanti il termine previsto delle attività

4.7.2 Coordinamento delle attività

Una volta pianificate le attività il Responsabile di Progetto ha il compito di assegnarle, attraverso il sistema di ticketing offerto da Asana, ai singoli componenti del gruppo. Ogni membro del team potrà vedere le attività che gli sono state assegnate e modificarne lo stato, permettendo al Responsabile di Progetto di monitorare lo stato di avanzamento.

4.7.3 Gestione dei rischi

Il Responsabile *di Progetto* ha il compito di individuare i rischi indicati nel *Piano di Progetto*. Quando questi vengono individuati, il *Responsabile* ha il compito di estendere l'analisi dei rischi ai nuovi casi rilevati.
Complessivamente la procedura prevede i seguenti passi:

1. rilevazione ed individuazione attiva dei rischi previsti e dei problemi non calcolati
2. registrazione del riscontro effettivo dei rischi nel *Piano di Progetto*
3. trattazione e pianificazione per la gestione dei nuovi rischi individuati
4. ridefinizione e aggiornamento delle strategie in base alle necessità e alle previsioni.

4.8 Strumenti

4.8.1 Sistema operativo

Il gruppo di progetto opera sui seguenti sistemi operativi:

- MacOS 10.12
- MacOS 10.9
- Ubuntu x64 16.04 LTS
- Windows 10 Pro
- Linux Mint 17.1
- Windows 7 Pro

4.8.2 Git

Git è un sistema software di controllo di versione distribuito, creato da *Linus Torvalds*$_{|G|}$ nel 2005. La versione utilizzata è maggiore o uguale alla 2.10.1.

4.8.3 GitHub

GitHub è un servizio web di hosting per lo sviluppo di progetti software, che usa il sistema di controllo di versione Git. Può essere utilizzato anche per la condivisione e la modifica di file di testo e documenti revisionabili (sfruttando il sistema di versionamento dei file di Git). *GitHub*$_{|G|}$ ha diversi piani per repository privati sia a pagamento, sia gratuiti, molto utilizzati per lo sviluppo di progetti open-source.

4.8.4 Asana

Asana è uno strumento di project management utilizzato per la gestione e assegnazione dei compiti e delle attività. É strutturato in aree di lavoro, progetti, task e subtask.

4.8.5 Microsoft Project

Microsoft Project è un software di pianificazione sviluppato e venduto da Microsoft. É uno strumento per assistere i responsabili di progetto nella pianificazione, nell'assegnazione delle risorse, nella verifica del rispetto dei tempi, nella gestione dei budget e nell'analisi dei carichi di lavoro. Questo software verrà utilizzato per la creazione di tutti i diagrammi di Gantt necessari.

4.8.6 Slack

Slack è una piattaforma per la comunicazione tra gruppi di lavoro. É organizzato in canali e permette di condividere file in modo facile e veloce.

4.8.7 Telegram

Telegram è un servizio di messaggistica istantanea erogato senza fini di lucro dalla società *Telegram*$_{|G|}$ LLC. I *client* ufficiali di Telegram sono distribuiti come software libero per diverse piattaforme.
Viene utilizzato dai membri del gruppo per comunicare liberamente sul progetto e su dettagli organizzativi.

4.8.8 Google Drive

Il sevizio *cloud* based *Google Drive* viene utilizzato per un rapido scambio di documenti e file non soggetti a versionamento che non hanno necessità di risiedere nel *repository*.

4.8.9 Google Calendar

Google Calendar viene utilizzato dai membri del team per segnare le date di importanti scadenze, come *milestone*, incontri o riunioni.

4.8.10 Skype

Skype è un software proprietario freeware di messaggistica istantanea e VoIP. Esso unisce caratteristiche presenti nei client più comuni (chat, salvataggio delle conversazioni, trasferimento di file) ad un sistema di telefonate basato su un network Peer-to-peer.

A Lista di controllo

- **Norme stilistiche**
  - norme documento: non viene indicata la versione del documento relativo dove necessaria
  - mancanza di punto alla fine della frase
  - inizio del paragrafo senza la lettera maiuscola
  - mancanza di due punti per introdurre un elenco
  - parole troppo distanti in una frase: utilizzo di due spazi consecutivi anziché uno
- **Italiano**
  - frasi troppo lunghe rendono i concetti di difficile comprensione
  - utilizzo ridondante delle virgole lì dove non sarebbe necessario
  - utilizzo erroneo degli articoli determinativi
  - utilizzo erroneo del termine “fase”
  - carattere ‘É’: non viene scritto correttamente utilizzando il comando apposito
  - vengono utilizzate frasi troppo lunghe creando difficoltà nel comprendere il significato di quest’ultime
  - caratteri accentati erroneamente scritti con l’accento acuto anziché grave
- **UML**
  - mancanza del nome del caso d’uso di riferimento nelle figure chiamandolo UC anziché con il proprio nome
  - didascalia dei diagrammi ambigua
- **LaTex**
  - mancanza di utilizzo del comando "empf" per le parole in corsivo
  - mancanza di utilizzo del comando per andare a capo
  - indirizzi URL non “cliccabili”
  - utilizzo erroneo del comando “item”
  - calcolo erroneo dei sottoparagrafi
  - numero elevato di commenti lasciati nei documenti .tex inquinando l’ambiente di lavoro
- **Tracciamento dei requisiti**
  - ad ogni requisito deve corrispondere almeno una fonte
  - ad ogni caso d’uso deve corrispondere almeno un elemento
- **Glossario**
  - termini presenti nei documenti mancanti nel *Glossario v3.0.0*