

Norme Di Progetto

RAMtastic6

4 aprile 2024



email: ramtastic6@gmail.com

Informazioni sul documento

Versione: 0.8.0
Redattori: Visentin S. Basso L. Tonietto F. Zambon M.
Verificatori: Tonietto F. Brotto D. Zambon M. Basso L. Zaupa R.
Destinatari: T. Vardanega, R. Cardin, *Imola Informatica_G*
Uso: Interno

Registro dei Cambiamenti - Changelog

Versione	Data	Autore	Verificatore	Dettaglio
v.0.8.0	2024-03-28	Zambon M.	Zaupa R.	Modifica della sottosezione 3.1.12 (Glossario tecnico): ridefinita la parte di redazione del glossario e definita la sua verifica; stesura sottosezione 1.1 (scopo del documento) e 1.2 (scopo del prodotto)
v.0.7.2	2024-03-27	Zambon M.	Zaupa R.	Stesura della sottosezione 3.4 (Verifica)
v.0.7.1	2024-03-26	Zambon M.	Zaupa R.	Stesura della sottosezione 3.3 (Gestione della qualità)
v.0.7.0	2024-03-21	Basso L.	Brotto D.	Stesura delle sottosezioni 3.1.2, 3.1.3 (relative ai <i>repository_G</i> per la documentazione) e 3.1.9, 3.1.11, 3.1.12 (relative alle procedure per produrre documentazione)
v.0.6.0	2024-01-18	Basso L.	Tonietto F.	Stesura delle sezione 4 (processi organizzativi) con relative sottosezioni.
v.0.5.0	2024-01-09	Zambon M.	Tonietto F.	Stesura delle sezione 2.4.2 (Analisi dei requisiti)
v.0.4.2	2023-12-12	Visentin S.	Tonietto F.	Prima stesura della sezione 2 <i>Processi primari_G</i>
v.0.4.1	2023-12-09	Visentin S.	Tonietto F.	Piccole modifiche nella sezione 4.
v.0.4.0	2023-11-19	Tonietto F.	Zaupa R.	Stesura della sotto-sottosezione 3.2.4 (relativa al controllo del flusso, con approfondimento sul flusso della documentazione prodotta) e della sottosezione 4.1 (breve analisi e descrizioni dei ruoli di progetto).
v.0.3.0	2023-11-12	Basso L.	Zambon M. Visentin S. Zaupa R.	Stesura della sottosezione 3.2 della sezione relativa alla documentazione e modifiche relative al ciclo di vita di un documento (sottosezione 3.1, parte 3.1.3).
v.0.2.0	2023-11-12	Basso L.	Zaupa R.	Stesura della sezione 3.1 (Documentazione) e delle sottosezioni relative ad essa
v.0.1.0	2023-10-30	Visentin S.	Tonietto F.	Prima versione

Indice

1	Introduzione	5
1.1	Scopo del documento	5
1.2	Scopo del prodotto	5
1.3	Riferimenti	5
1.3.1	Riferimenti normativi	5
1.3.2	Riferimenti informativi	5
2	Processi primari	6
2.1	Descrizione	6
2.2	Acquisizione	6
2.2.1	Descrizione	6
2.2.2	Valutazione capitolati	6
2.2.3	Appalto capitolati	6
2.3	Fornitura	6
2.3.1	Descrizione	6
2.3.2	Piano di progetto	6
2.3.3	Piano di qualifica	7
2.3.4	Rilascio	7
2.4	Sviluppo	7
2.4.1	Descrizione	7
2.4.2	Analisi dei requisiti	7
2.4.3	Progettazione	8
2.4.4	Codifica	8
3	Processi di supporto	9
3.1	Documentazione	9
3.1.1	Obiettivi	9
3.1.2	Scopo e struttura Repository Project 14	9
3.1.3	Scopo e struttura Repository RAMtastic6.github.io	9
3.1.4	Tipologie di documenti	9
3.1.5	Ciclo di vita di un documento	10
3.1.6	Template	10
3.1.7	Struttura di un documento	10
3.1.8	Strumenti	10
3.1.9	Struttura di un progetto in overleaf	10
3.1.10	Versionamento	11
3.1.11	Procedure branch develop	11
3.1.12	Glossario Tecnico	11
3.2	Controllo di configurazione	12
3.2.1	Versionamento	12
3.2.2	Git e Github	12
3.2.3	Struttura del repository	13
3.2.4	Controllo di Flusso	13
3.3	Gestione della qualità	13
3.3.1	Descrizione	13
3.3.2	Obiettivi	14

3.3.3	Codifica delle metriche	14
3.4	Verifica	14
3.4.1	Descrizione	14
3.4.2	Obiettivi	14
3.4.3	Analisi statica	15
3.4.4	Analisi dinamica	15
4	Processi organizzativi	16
4.1	Ruoli di progetto	16
4.1.1	Responsabile di Progetto	16
4.1.2	Amministratore di Progetto	16
4.1.3	Analista	17
4.1.4	Progettista	17
4.1.5	Verificatore	17
4.1.6	Programmatore	17
4.2	Gestione di progetto	18
4.2.1	Allineamento organizzativo	18
4.2.2	Comunicazioni interne	18
4.2.3	Comunicazioni esterne	18
4.3	Gestione organizzazione del lavoro	18
4.3.1	Modello di sviluppo	18
4.3.2	Rotazione dei ruoli	19
4.3.3	Gestione delle attività	19
4.4	Infrastruttura	19
4.5	Miglioramento	19
4.6	Formazione	20

1 Introduzione

1.1 Scopo del documento

Il presente documento si pone lo scopo di identificare le *best practices* di progetto e definire il *Way of Working_G* adottato da parte del gruppo, in modo da garantire omogeneità e coesione del lavoro. Per la stesura si utilizza un approccio incrementale e ogni aggiornamento avverrà successivamente a decisioni prese dal gruppo durante l'intera durata del progetto. Ciascun membro del team si impegna a visionare regolarmente tale documento e seguirne le procedure riportate. Eventuali termini tecnici sono definiti all'interno del documento "*Glossario Tecnico*".

1.2 Scopo del prodotto

Il prodotto finale, realizzato tramite un'applicazione web *responsive*, si propone di realizzare un *software_G* innovativo volto a semplificare il *processo_G* di *prenotazione_G* e *ordinazione_G* nei ristoranti, contribuendo a migliorare l'esperienza per clienti e ristoratori. In particolare, *Easy Meal* dovrà consentire agli utenti di personalizzare gli ordini in base alle proprie preferenze, allergie ed esigenze alimentari; interagire direttamente con lo staff del ristorante attraverso una chat integrata e in ultimo, consentire di dividere il conto tra i partecipanti al tavolo.

1.3 Riferimenti

1.3.1 Riferimenti normativi

1. Presentazione del *capitolato_G* d'appalto C3 - Progetto Easy Meal:
<https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C3.pdf>
2. Regolamento del progetto didattico:
<https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD2.pdf>

1.3.2 Riferimenti informativi

1. Lezione "*I processi di ciclo di vita del software_G (T2)*" del corso di Ingegneria del Software:
<https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T2.pdf>

2 Processi primari

2.1 Descrizione

Con la locuzione "processi primari" si intendono "tutti i processi che hanno come clienti soggetti esterni al gruppo".

2.2 Acquisizione

2.2.1 Descrizione

Nel *processo_G* di acquisizione avviene la raccolta e la comprensione dei requisiti con lo scopo di identificare un *capitolato_G* adeguato per il gruppo da proporre per la candidatura.

2.2.2 Valutazione capitolati

Il gruppo ha analizzato la proposta dei proponenti valutandone, in base all'esperienza del gruppo, la loro *complessità*, tale analisi è stata determinante per la scelta definitiva.

2.2.3 Appalto capitolati

Il gruppo si è proposto per il *capitolato_G* C3 dell'azienda IMOLA INFORMATICA. Nonostante un primo riscontro negativo, a seguito di modifiche volte a sistemare lacune presenti nella *candidatura_G* iniziale, il gruppo è riuscito ad aggiudicarsi il *capitolato_G*.

2.3 Fornitura

2.3.1 Descrizione

Il *processo_G* di *fornitura_G* consiste nel chiarire ogni dubbio legato al prodotto finale che il proponente desidera; in modo da evitare incomprensioni durante lo svolgimento del progetto, il gruppo 14 si impegna a comunicare con l'azienda, in modo da raggiungere i seguenti obiettivi:

1. determinare i requisiti da soddisfare nel prodotto finale;
2. ottenere incontri di formazioni sulle tecnologie e strumenti consigliati dall'azienda per realizzare il prodotto;
3. ricevere *feedback_G* in fase di sviluppo su quanto precedentemente svolto.

2.3.2 Piano di progetto

Il documento *Piano di progetto* costituisce uno *strumento_G* di pianificazione per tutte le attività che dovranno essere svolte così da poter rispettare la data di consegna del progetto. Nel dettaglio il *piano di progetto_G* è composto da:

- Analisi dei rischi: analisi delle difficoltà che il gruppo potrebbe riscontrare durante lo svolgimento del progetto, in particolar modo a livello organizzativo e tecnologico;
- Modello di sviluppo;

- Pianificazione;
- Preventivo: che rispecchi quanto comunicato in fase di candidatura;
- Consuntivo: tracciamento dell'andamento del gruppo rispetto al *preventivo_G* fatto.

2.3.3 Piano di qualifica

Nel documento *Piano di qualifica* vengono elencate tutte le attività svolte dal verificatore con l'obiettivo di garantire la qualità del prodotto finale. E' formato dai seguenti componenti:

- Qualità di processo
- Qualità di prodotto
- Test: eseguiti sul prodotto che assicurano che i requisiti siano stati rispettati
- Resoconto

2.3.4 Rilascio

Quando il prodotto verrà ultimato, verrà collaudato per garantire il suo corretto funzionamento. Se il prodotto supera il collaudo, verrà consegnato al committente insieme alla documentazione del progetto. Il gruppo non effettuerà manutenzione una volta rilasciato il prodotto.

2.4 Sviluppo

2.4.1 Descrizione

Lo scopo del *processo_G* di sviluppo è quello di dichiarare le attività da svolgere per raggiungere i requisiti necessari del prodotto. A tale scopo, il gruppo si dividerà in diversi ruoli, i quali avranno compiti precisi da svolgere.

2.4.2 Analisi dei requisiti

Analisi dei requisiti è un documento fondamentale per lo sviluppo. Infatti, tale documento deve indicare i requisiti necessari del prodotto finale, che a loro volta andranno a rispecchiare le aspettative del proponente.

Inoltre, questo documento servirà anche come documentazione del prodotto, andandone infatti a contenere tutte le relative funzionalità.

Attori Innanzitutto, verranno definiti gli attori e una panoramica dei vari casi d'uso a loro associati tramite un diagramma

Casi d'uso Successivamente, verranno descritti i vari casi d'uso, i quali hanno il compito di rappresentare le funzionalità che il prodotto finale dovrà rispettare.

I casi d'uso saranno ordinati per attore, ovvero verranno prima descritti tutti i casi d'uso associati a un particolare attore, per poi proseguire con la descrizione di tutti i casi d'uso riguardanti l'attore successivo, e a proseguire.

Ogni *caso d'uso_G* avrà un diagramma ad esso associato in cui verrà inserita la funzionalità che

si vuole descrivere e le eventuali estensioni per le eccezioni. Verranno poi inseriti diagrammi che illustreranno i vari sottocasi d'uso, cercando di arrivare a casi d'uso cosiddetti "atomici".

I casi d'uso verranno inoltre descritti verbalmente, tramite una struttura standard da rispettare:

- Attori
- Precondizioni
- Postcondizioni
- $Scenario_G$ primario
- Scenari alternativi (opzionale)

Gli scenari alternativi saranno presenti solo se il $caso\ d'uso_G$ può generare eccezioni: poiché a più eccezioni corrispondono una singola modalità di gestione delle stesse, per ogni $scenario_G$ alternativo ci potranno essere più primi punti, i quali verranno rappresentati nel formato " $1.xa$ ", nel quale ' x ' rappresenta il punto dello $scenario_G$ principale dal quale viene generata l'eccezione, mentre ' a ' rappresenta il tipo di eccezione. Verrà poi descritto dal punto "2" a seguire lo $scenario_G$ alternativo.

Requisiti Verranno infine descritti i requisiti.

2.4.3 Progettazione

La progettazione, a carico della figura del progettista, definisce la struttura del progetto basandosi sull'analisi dei requisiti. La progettazione avviene su più livelli:

1. Progettazione architetturale: dove viene scelta la struttura del sistema
2. Design: ovvero il $design_G$ dell'interfaccia che deve avere il prodotto
3. Progettazione dettagliata: le specifiche dei componenti del prodotto che comprendono le specifiche architetture, i diagrammi delle classi e i $test_G$ d'unità

2.4.4 Codifica

I programmatori, dopo l'analisi e la progettazione, implementano le funzionalità che deve avere il prodotto finale basandosi sull'analisi dei requisiti e sui documenti di progettazione.

Inoltre, durante questa fase, si devono implementare $test_G$ che assicurano il corretto funzionamento del prodotto.

3 Processi di supporto

3.1 Documentazione

3.1.1 Obiettivi

3.1.2 Scopo e struttura Repository Project 14

Questo *repository_G* ha due funzioni:

- mantenere una versione aggiornata dei sorgenti atti a produrre documentazione (file .tex)
- disporre di un *sistema_G* per produrre automaticamente documentazione (file in formato .pdf) a partire dai sorgenti.

In questo repository, infatti vi sono due cartelle principali:

- sorgenti: contiene i file sorgente della documentazione (.tex)
- documenti: contiene i risultati della compilazione dei file sorgente (file .pdf)

La cartella documenti deve avere la seguente struttura:

- n_periodo - PERIODO/
 - documenti relativi al periodo
 - verbali/
 - * verbali_interni/
 - * verbali_esterni/

Un *verbale_G* deve essere nominato nel modo seguente: *verbale_G*-AAAA-MM-GG.

3.1.3 Scopo e struttura Repository RAMtastic6.github.io

Questo *repository_G* ha una funzione principale, ovvero quella di permettere una navigazione intuitiva all'interno della documentazione del gruppo; essa non è altro che una copia della cartella documenti del repo Project 14.

3.1.4 Tipologie di documenti

I documenti prodotti possono essere classificati in due classi principali: ad uso interno e ad uso esterno; la prima categoria comprende:

- *Verbali interni_G* (i quali non necessitano di versionamento)
- Norme di progetto

La seconda categoria di documenti comprende:

- Verbali esterni
- Piano di qualifica
- Piano di progetto
- Analisi dei requisiti

3.1.5 Ciclo di vita di un documento

Un documento segue le seguenti fasi di produzione:

- Stesura: uno o più redattori si occupano di redigere il contenuto del documento.
- Verifica: ad uno o più membri del gruppo, diversi da quelli che hanno redatto il documento, viene assegnato il compito di verifica del documento. È importante sottolineare che tutti i documenti sopracitati sono ufficiali e devono essere, quindi, preventivamente approvati da verificatori designati.
- Approvazione: durante questa fase, il responsabile di progetto può decidere se approvare l'inclusione di un particolare documento all'interno del repository. Nel caso in cui il documento non venga approvato, si ritorna alla fase di stesura. Se quest'ultima fase va a buon fine, vengono aggiunte informazioni di *versionamento_G* secondo quanto riportato nell'apposita sezione; infine viene caricato il documento all'interno del *repository_G* nel *branch_G* develop.

3.1.6 Template

Il gruppo ha scelto di utilizzare *template_G* LaTeX per la produzione della documentazione. Per visualizzare la struttura e utilizzare i *template_G*, è sufficiente cercarli su overleaf.

3.1.7 Struttura di un documento

Un documento all'interno del nostro contesto segue una struttura ben definita, le sue sezioni principali includono:

- Prima pagina: contiene il nome del gruppo e informazioni in merito al documento: uso, destinatari, redattori, verificatori, versione
- Indice: elenco strutturato dei contenuti del documento
- Registro dei cambiamenti: una tabella contenente informazioni di *versionamento_G* relative al documento attuale; queste includono: la versione, la data, l'autore, il verificatore e una breve descrizione in merito alle modifiche apportate al documento.
- Intestazione: all'interno di essa vi sono il nome e l'indirizzo email del gruppo.

3.1.8 Strumenti

Per la creazione e la gestione della struttura dei documenti è stato deciso di utilizzare Overleaf, un *editor_G* LaTeX online che permette la stesura collaborativa dei documenti.

3.1.9 Struttura di un progetto in overleaf

Affinché le automazioni per produrre *riferimenti_G* al glossario funzionino, un progetto in *overleaf_G* deve avere la struttura seguente:

- main.tex : file contenente il contenuto principale o, se si vuole, tutto il contenuto del file
- CONTENTS/ : cartella contenente ulteriori file (.tex) che il main.tex usa per poter costruire un documento unico

3.1.10 Versionamento

Il *versionamento_G* scelto per tenere traccia dei documenti è una tripletta di numeri: $x.y.z$.

- x è un numero intero, che fino alla release sarà < 1 , e indica la versione del progetto a cui il documento fa riferimento;
- y è un numero intero positivo, e rappresenta lo stato di verifica del documento;
- z è un numero intero positivo, e rappresenta il singolo cambiamento apportato al file.

3.1.11 Procedure branch develop

Il flusso di lavoro attuale per produrre la documentazione relativa al progetto è il seguente:

1. modificare i documenti in overleaf
2. scaricare i sorgenti dei documenti scritti in *overleaf_G* e inserirli all'interno del *repository_G* locale Project14, nella cartella sorgenti.
3. posizionarsi nella cartella Project 14 ed eseguire le automazioni mediante il comando:

python3 entry_point_automazioni.py

Il risultato di questa esecuzione produrrà dei nuovi sorgenti (.tex): in particolare:

- (a) Una versione aggiornata del glossario tecnico
- (b) Gli stessi file che contenevano *riferimenti_G* al glossario tecnico dove ve ne siano.

3.1.12 Glossario Tecnico

Il *Glossario tecnico* è un documento di supporto concepito per evitare ambiguità o incomprensioni riguardanti la terminologia utilizzata in tutta la documentazione per ogni fase del progetto ed è rivolto sia ai componenti del gruppo che a committenti e ai proponenti. Si tratta dell'unico documento da non modificare all'interno di overleaf. Infatti, il glossario tecnico viene costruito a partire da un file *json* contenuto all'interno della cartella *sorgenti/Glossario* del *repository_G Project14*.

Tale file, denominato *glossario.json*, è costituito da un array di oggetti; ogni oggetto è formato da un insieme di due coppie chiave-valore, in particolare vi sono:

- La chiave termine: il termine che necessita di essere definito all'interno del dominio di progetto;
- Il valore definizione: la definizione del termine stesso.

Redazione Per poter inserire o modificare un termine nel glossario tecnico, bisogna seguire i seguenti passaggi:

- Creare un *branch_G* nel *repository_G Project14* associato al *ticket_G* che indica le parole da inserire o modificare all'interno del glossario;
- Aggiungere o modificare le parole e definizioni nel file *glossario.json*;
- modificare il *changelog_G* nel file *build_glossary.py* nella cartella GLOSSARY_AUTOMATIONS;

- Eseguire l'automazione tramite il comando:

python3 entry_point_automazioni.py

- Effettuare il *push_G* in remoto;
- Aprire una *pull request_G* dal *branch_G* creato al *branch_G* develop;
- Spostare il *ticket_G* corrispondente nella sezione "da verificare" all'interno di *Jira*.

Nel momento in cui viene inserita una parola nuova all'interno del glossario bisogna segnalare al responsabile se sono presenti discrepanze tra il modo in cui è stato scritto il termine all'interno del file *glossario.json* e il modo in cui è stato scritto all'interno dei documenti. Viene riportato il seguente esempio: se all'interno del glossario viene riportata la parola "Analisi dei Requisiti" allora all'interno dei documenti tale parola deve essere riportata con le stesse lettere maiuscole e minuscole ad eccezione_G della lettera iniziale (va bene "analisi dei Requisiti" ma non "Analisi dei requisiti").

Verifica Per poter verificare il glossario, si seguono le seguenti azioni:

- Utilizzare il *branch_G* creato dal redattore;
- Verificare i termini e definizioni nel file *glossario.json*;
- Modificare il *changelog_G* nel file *build_glossary.py* nella cartella GLOSSARY_AUTOMATIONS;
- Eseguire l'automazione tramite il comando:

python3 entry_point_automazioni.py

- Effettuare il *push_G* in remoto;
- Spostare il *ticket_G* nella sezione "completato" all'interno di *Jira*.

Infine, il responsabile approva la *pull request_G* associata.

3.2 Controllo di configurazione

3.2.1 Versionamento

Capire come gestire i numeri di versione.

3.2.2 Git e Github

Il gruppo RAMtastic6 ha scelto di utilizzare come *strumento_G* di *versionamento_G* *GitHub* e di utilizzare *Git* come *strumento_G* per collegarsi alla *repository_G* *GitHub*. Inoltre si è scelto di utilizzare *gitflow_G* come flusso di lavoro il quale verrà discusso in modo dettagliato in seguito (Link per il download dell'installer di Git).

Inoltre, a questo link si troverà una breve guida su come utilizzare git. In sintesi si elencano i principali comandi:

- *git_G* clone *link repo*
questo comando copierà la *repository_G* di *github_G* in locale

- git_G add *nome file* (oppure "." per includere tutti i file)
 git add aggiunge le modifiche apportate ai files del $repository_G$, senza eseguire questo comando un file aggiunto, eliminato o modificato non verrà salvato nella $repository_G$ remota tramite il comando git push.
- git_G commit -m "messaggio"
salva le modifiche apportate ai files in locale associando a quello stato un messaggio
- git_G push $origin$ *origine*
salva le modifiche in remoto nel $branch_G$ specificato
- git_G pull
permette di aggiornare la repo in locale e in caso di necessità esegue il merge

3.2.3 Struttura del repository

La struttura della $repository_G$ per i documenti deve essere:

- documenti
 - CANDIDATURA
 - RTB
 - PB
- diari_di_bordo
- documenti_interni

3.2.4 Controllo di Flusso

Il gruppo RAMtastic6 ha deciso di dotarsi di $Gitflow_G$ come $sistema_G$ di controllo del flusso di lavoro, motivato dalla sua facilità d'uso e dalle potenzialità di gestione offerte per il repository. Con una lieve modifica nei comandi per l'esecuzione dei commit, come illustrato in questa guida su $Gitflow_G$, è possibile automatizzare il $processo_G$ di creazione, gestione e chiusura di una $feature_G$. Ulteriori dettagli su come gestire le $feature_G$ sono disponibili a questo link.

Gestione dei Documenti Un' particolare attenzione in tal senso è rivolta alla documentazione. Al fine di mantenere nel $repository_G$ solamente i PDF_G dei documenti prodotti, è stato deciso di adottare la piattaforma Overleaf per la stesura in LaTeX dei documenti e la successiva verifica. Ogni volta che un documento viene redatto o aggiornato, verificato e portato alla versione corretta come precedentemente indicato, può essere comodamente convertito in formato PDF_G tramite Overleaf. Successivamente, il documento può essere caricato nella $repository_G$, con il $push_G$ diretto sul $branch_G$ *develop*, soprattutto quando si parla di documentazione importante e la cui stesura è in itinere.

3.3 Gestione della qualità

3.3.1 Descrizione

La gestione della qualità è un insieme di processi che hanno lo scopo di garantire che il software, gli artefatti e i processi nel ciclo di vita del progetto aderiscano degli standard di qualità rispetto a requisiti specificati al fine di soddisfare le aspettative del proponente e degli utenti finali.

3.3.2 Obiettivi

La gestione della qualità si propone di raggiungere i seguenti obiettivi:

- Realizzare un prodotto di qualità, in linea con le richieste del proponente;
- Ridurre al minimo i *rischi_G* che potrebbero influire sulla qualità del prodotto;
- Rispettare il budget preventivato del progetto.

Gli strumenti utilizzati, per la gestione della qualità dei processi e del prodotto e per valutare il lavoro svolto, sono delle metriche definite nel documento di *Piano di Qualifica*.

3.3.3 Codifica delle metriche

Ogni metrica è identificata dal seguente formato di codice:

M[Tipo][Id]-[Acronimo]

Dove:

- **M** sta per "Metrica"
- **Tipo** può essere PC (per un processo) o PD (per un prodotto)
- **Id** rappresenta un identificativo all'interno di una metrica di un certo tipo
- **Acronimo** indica l'acronimo del nome della metrica utilizzata

Per ciascuna metrica vengono fornite delle descrizioni; inoltre per ogni tipo di *processo_G* viene fornita una tabella avente: il codice della metrica, il nome della metrica, valori accettabili e valori preferibili.

3.4 Verifica

3.4.1 Descrizione

La verifica del *software_G* è un *processo_G* che valuta il prodotto durante le varie fasi del progetto, dalla progettazione alla manutenzione. Essa mira a garantire che il *software_G* sia conforme alle aspettative e ai requisiti specificati fondandosi su criteri come coerenza, completezza e correttezza dei risultati.

3.4.2 Obiettivi

Il *processo_G* di verifica si propone di raggiungere i seguenti obiettivi:

- Assicurarci che il prodotto mantenga una buona qualità nel corso del suo sviluppo;
- Individuare errori e anomalie prima di proseguire con lo sviluppo del progetto.

Nel documento "*Piano di Qualifica*" vengono definiti gli obiettivi da raggiungere e i criteri di accettazione che saranno impiegati per condurre il *processo_G* di verifica in modo accurato ed efficiente.

3.4.3 Analisi statica

L'analisi statica è una metodologia di verifica che prescinde dall'esecuzione del prodotto e che si basa su una revisione del codice e della documentazione. Lo scopo principale di questa analisi è quello di verificare l'assenza di difetti e la conformità ai requisiti e alle specifiche richieste.

L'analisi statica adotta comunemente due metodi di lettura:

- **Walkthrough:** si tratta di una tecnica collaborativa che coinvolge il verificatore e l'autore del prodotto e che consiste nel revisionare nel suo complesso il codice e la documentazione forniti, con una successiva discussione degli eventuali problemi trovati;
- **Inspection:** si tratta di una tecnica che consiste nel revisionare parti specifiche del codice e della documentazione attraverso liste di controllo (*checklist*) nel momento in cui si ha già un'idea di dove possano esserci possibili problemi in modo da intervenire tempestivamente e sistematicamente.

Nel documento "*Piano di Qualifica*" vengono definite delle liste di controllo in modo da applicare la tecnica dell'*inspection*, preferibile a quella del *walkthrough*.

3.4.4 Analisi dinamica

L'analisi dinamica è una metodologia di verifica che si basa sull'esecuzione del codice. Le tecniche principali utilizzate in questa fase sono i $test_G$ (definiti nel documento di "*Piano di Qualifica*") finalizzati per individuare e verificare il comportamento del prodotto software.

4 Processi organizzativi

4.1 Ruoli di progetto

In questa sezione viene riportata una breve descrizione dei ruoli e delle responsabilità dei membri di un gruppo dedicato allo sviluppo di un qualsiasi tipo di *project*.

4.1.1 Responsabile di Progetto

Il Responsabile di Progetto è la figura professionale, punto di *riferimento_G* sia per il committente sia per il fornitore, con lo scopo di mediare tra le due parti. Assume la responsabilità delle decisioni del gruppo dopo averle approvate.

Le sue responsabilità includono:

- Approvare l'emissione della documentazione;
- Approvare l'offerta economica sottoposta al committente;
- Pianificare e coordinare le attività di progetto;
- Gestire le risorse umane;
- Studiare e gestire i rischi.
- Chiedere l'approvazione dei verbali alle persone esterne che hanno partecipato
- Assegnare le attività, tramite il *sistema_G* di tracking issues fornito da Jira, ai membri che le dovranno svolgere
- Gestire le *milestones* e fissarne di nuove, o modificare quelle attuali in base all'andamento del team

4.1.2 Amministratore di Progetto

L'Amministratore di Progetto è responsabile delle procedure di controllo e amministrazione dell'ambiente di lavoro, con piena responsabilità sulla capacità operativa e sull'efficienza.

In particolare, si occupa di:

- Ricercare, studiare e mettere in opera risorse per migliorare l'ambiente di lavoro, automatizzando quando possibile;
- Risolvere problemi legati alla gestione dei processi;
- Salvaguardare la documentazione di progetto;
- Effettuare il controllo di versioni e configurazioni del prodotto software;
- Redigere e attuare i piani e le procedure per la gestione della qualità.

4.1.3 Analista

L'Analista è una figura con maggiori competenze riguardo al dominio applicativo del problema. Le sue responsabilità includono:

- Studiare il problema e il relativo contesto applicativo;
- Comprendere il problema e definire la complessità e i requisiti;
- Redigere il documento "Analisi dei Requisiti";
- Studiare i casi d'uso e redigere il loro relativo schema UML.

4.1.4 Progettista

Il Progettista gestisce gli aspetti tecnologici e tecnici del progetto. In particolare, si occupa di:

- Effettuare scelte riguardanti gli aspetti tecnici e tecnologici del progetto, favorendone l'efficienza e l'efficienza;
- Definire un'architettura del prodotto da sviluppare che miri all'economicità e alla manutenibilità a partire dal lavoro svolto dall'Analista;
- Redigere la "Specifica Tecnica" e la parte pragmatica del "Piano di Qualifica".

4.1.5 Verificatore

Il Verificatore è responsabile della sorveglianza sul lavoro svolto dagli altri componenti del gruppo, sulla base delle proprie competenze tecniche, esperienza e conoscenza delle norme. In particolare, si occupa di:

- Esaminare i prodotti in fase di revisione, con l'ausilio delle tecniche e degli strumenti definiti nel presente documento;
- Verificare la conformità dei prodotti ai requisiti funzionali e di qualità;
- Verificare i documenti segnalando eventuali errori.
- Caricare i verbali, dopo averli verificati, all'interno della *repository*_G GitHub nel ramo *develop*.

4.1.6 Programmatore

Il Programmatore è incaricato della *codifica*_G del progetto e delle componenti di supporto che verranno utilizzate per eseguire prove di verifica e di validazione del prodotto. In particolare, si occupa di:

- Implementare la "Specifica Tecnica" redatta dal Progettista;
- Scrivere un codice pulito e facilmente mantenibile che rispetti le norme definite nel presente documento;
- Realizzare gli strumenti per la verifica e la validazione del software;
- Redigere il "Manuale Utente" relativo alla propria codifica;
- Redigere i verbali delle riunioni interne ed esterne del team.

4.2 Gestione di progetto

4.2.1 Allineamento organizzativo

Per l'allineamento e il coordinamento delle attività sono stati scelti due tipi di comunicazione:

- Interne: coinvolge tutti i membri del gruppo;
- Esterne: coinvolge proponenti e committenti.

4.2.2 Comunicazioni interne

Le comunicazioni interne avvengono tramite le applicazioni:

- *Telegram*: permette una comunicazione veloce tra i membri del gruppo e viene utilizzato principalmente per organizzare incontri interni o discutere di eventuali quesiti.
- *Discord*: viene utilizzato dai membri del gruppo per tenere incontri interni, in quanto dispone di un canale vocale; inoltre, permette la condivisione di contenuti multimediali in streaming video.

4.2.3 Comunicazioni esterne

Le comunicazioni esterne vengono gestite dal Responsabile di Progetto tramite l'utilizzo dei seguenti canali:

- Posta Elettronica: tramite l'indirizzo e-mail del gruppo *ramtastic6@gmail.com*.
- *Telegram*: permette di instaurare un canale veloce di comunicazione con la proponente.

4.3 Gestione organizzazione del lavoro

4.3.1 Modello di sviluppo

Il gruppo, al fine di minimizzare ritardi e massimizzare lo svolgimento delle proprie attività, ha deciso di implementare il *framework_G Scrum*, che permette di dividere il tempo di lavoro in intervalli piccoli, frammentati all'interno di un periodo di circa due settimane, definito come "*sprint*". In particolare, all'interno di questo, vengono svolte un numero di attività commisurate che devono essere completate entro i suoi termini. Individuiamo dunque una serie di fasi:

- **sprint planning**: in questa fase, si discutono gli obiettivi da raggiungere all'interno dello *sprint*; vengono quindi pianificate le attività da svolgere durante lo *sprint* in funzione di questi, facendo attenzione alle disponibilità di ogni membro e preparando un preventivo per il periodo; è in questa fase, inoltre, che viene svolto il cambio dei ruoli. Lo *sprint planning* consiste in una riunione tenuta tramite *Discord* che si svolge all'inizio dello *sprint* e richiede la partecipazione di tutti i membri del gruppo.
- **sprint review**: in questa fase, si discutono gli obiettivi da raggiunti nello *sprint*. Alla fine, dovrebbe essere prodotto almeno un incremento, cioè un software utilizzabile. In particolare, in questa fase vengono definiti a consuntivo le risorse impiegate commisurate agli obiettivi, questi ultimi suddivisi in raggiunti e non raggiunti, capendo cosa va migliorato e cosa non è stato fatto per poter aggiungere ulteriori obiettivi per lo *sprint* successivo;

- **sprint retrospective:** fase che conclude definitivamente lo *sprint* appena svolto, avente l'obiettivo di valutarne l'andamento generale e cercando di capire cosa è stato fatto bene e cosa può essere migliorato. In questo modo, è meglio definire come ripianificare le attività, decidendo come iniziare/continuare/concludere attività presenti o future da realizzare.

4.3.2 Rotazione dei ruoli

É prevista una rotazione dei ruoli all'interno del gruppo a cadenza periodica. L'attribuzione dei ruoli viene svolta secondo i seguenti criteri:

- Equità;
- Disponibilità;
- Assenza di conflitti.

4.3.3 Gestione delle attività

Per gestire le attività da istanziare all'interno del *framework_G Scrum*, è stato deciso di utilizzare *Jira*. Un'attività, definita come *ticket* all'interno di *Jira*, ha un ciclo di vita; esso si compone di quattro stati:

- **Da completare:** l'attività è stata istanziata e assegnata ad un membro del gruppo; è stata inoltre definita una stima in termini temporali associata ad essa.
- **In corso:** l'attività è entrata nella fase di svolgimento; all'interno di questa fase, il membro al quale è stata assegnata ha il compito di dover inserire le informazioni di tracciamento temporale associate al *ticket*; per fare ciò, basta selezionare un *ticket* e andare a modificare il campo *Tracciamento temporale*, inserendo il tempo impiegato fino ad ora per svolgere l'attività.
- **In fase di verifica:** durante questa fase, il Verificatore si assicura che l'attività sia stata svolta correttamente e secondo dei criteri di qualità attesa (al momento, ancora da specificare).
- **Completata:** l'attività è stata verificata e approvata dal Responsabile di Progetto.

Il passaggio di un *ticket* da uno stato all'altro è responsabilità del membro al quale è stato assegnato, eccetto per lo stato di completamento; infatti, spetta al Responsabile di Progetto decidere quando un'attività risulti o meno completata.

4.4 Infrastruttura

Strumenti per la gestione delle attività di progetto e procedure legate ad essi.

4.5 Miglioramento

Durante lo svolgimento delle attività e successiva stesura dei documenti, il gruppo si impegna ad operare secondo il *principio di miglioramento continuo*, al fine di individuare facilmente attività, ruoli e possibili miglioramenti, cercando nuove o diverse soluzioni alle problematiche insorte.

4.6 Formazione

Al fine di operare continuativamente un miglioramento sulle attività svolte e proseguire nel mantenimento corretto delle attività in modo asincrono, occorre da parte di tutti i membri del gruppo lo studio in autonomia delle *tecnologie_G* e delle modalità operative presenti, al fine di velocizzare il *processo_G* di formazione e di conoscenza degli strumenti utilizzati. Si listano, al fine di completezza, alcune documentazioni utilizzate durante lo sviluppo, sia a livello documentale che organizzativo:

- *GitHub*: <https://docs.github.com/>
- *Jira*: <https://confluence.atlassian.com/jira>
- *Git*: <https://docs.github.com/en/get-started/using-git/about-git>
- *Scrum*: <https://scrumguides.org/scrum-guide.html>